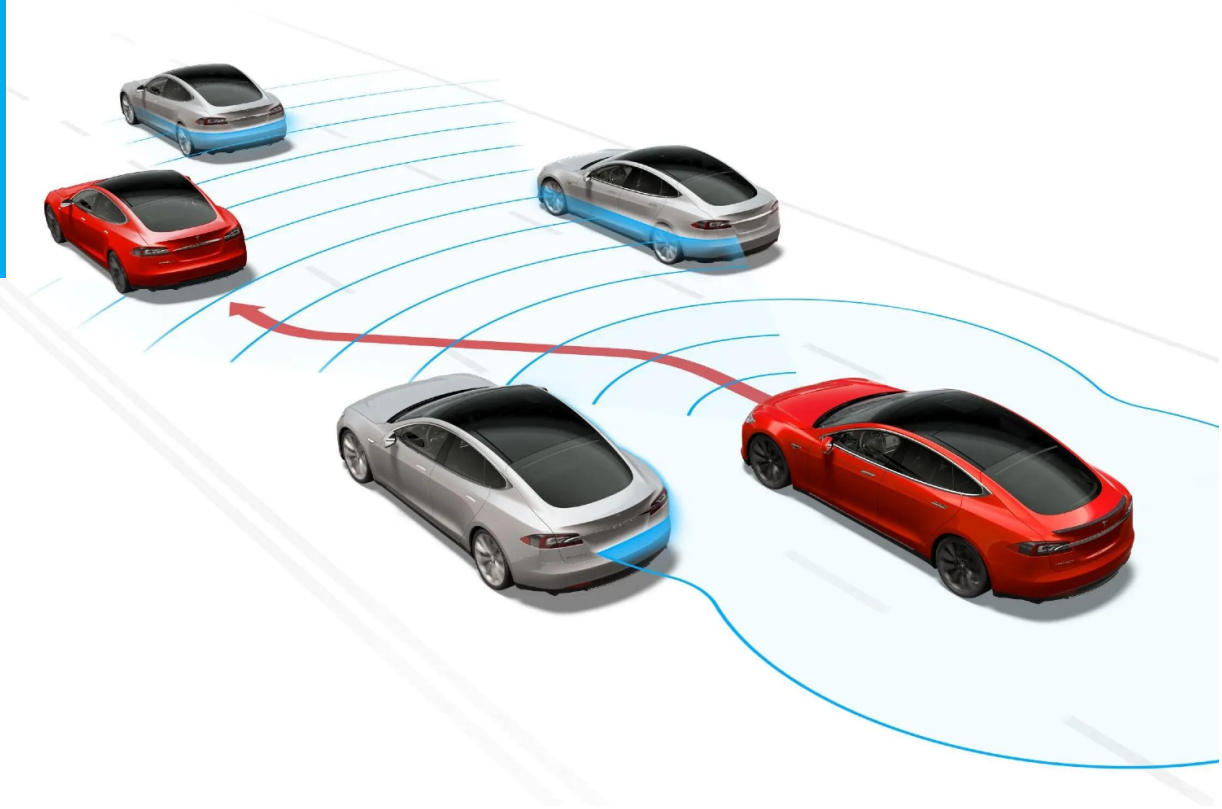
Interaction-aware Planning for Automated Vehicles in the Forced Lane Merging Scenario

Shaohang Han

Master of Science Thesis



TU Delft Delft University of Technology

Cognitive Robotics (CoR)

Interaction-aware Planning for Automated Vehicles in the Forced Lane Merging Scenario

Master of Science Thesis

For the degree of Master of Science in Robotics at Delft University of Technology

by

Shaohang Han

August 23, 2023

Supervisor: Prof. M. Wisse
L. Zhang, MSc.
Reader: Dr.ing. S. Grammatico
Dr. J. Alonso-Mora

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Automated vehicles represent an exciting advancement in transportation, offering a range of benefits that have the potential to revolutionize how we travel. They can improve safety, efficiency, accessibility, and sustainability, holding promise for transforming our cities and communities. However, generating safe, comfortable, and efficient motion plans, especially in interactive scenarios like forced lane merging, remains a significant challenge in the field.

Lane merging is a pivotal skill for automated vehicles, as it frequently involves changing lanes to reach a destination. For instance, when approaching an intersection, a vehicle might need to merge into a specific lane beforehand to execute a turn later. Traditional pipelines in automated driving typically decouple prediction and planning. They assume perfect upstream prediction and generate robust motion plans to avoid collisions with multi-modal predictions. However, in dense traffic conditions, conservative planning might hinder the ego vehicle from merging effectively, resulting in it becoming stuck. This underscores the necessity of combining prediction and planning, a concept we term interaction-aware planning algorithms.

The first major contribution of this thesis is an efficient game-theoretic behavior planner that captures interactions under different behavior modes. In this approach, we represent the behaviors of the vehicles as actions in a matrix game and select the Nash equilibrium to capture their mutual influence. To generate the cost of the action pairs, we model the merging process as a gap selection process and evaluate the trajectories generated by interactive models. The effectiveness of the proposed planner is validated in the high-fidelity CARLA simulator.

In the real world, human drivers may not always adhere rationally to the equilibrium of a game model. They could choose a behavior mode different from the game theory solution. Therefore, it might be more beneficial to consider different motion modes simultaneously, rather than favoring the "most likely" one while neglecting the others.

For this purpose, we also explore the usage of Branch Model Predictive Control (B-MPC) in this thesis. By predicting the motion of the surrounding vehicle as a scenario tree, the B-MPC approach can generate a trajectory tree as a motion plan. By executing only the root node, the ego vehicle can consider different future scenarios simultaneously and plan contingency motions. We further extend the B-MPC approach by incorporating interactive policies, using different solving schemes, and including collision avoidance constraints that

consider the orientations of the vehicles. The effectiveness of these proposed methods was validated through experiments conducted in a handcrafted lightweight simulator.

Overall, this thesis focuses on developing interaction-aware planning methods to facilitate safe, successful, and comfortable lane merging scenarios. Nevertheless, the major limitations lie in the modeling error and the potential long-tail issues. To address these challenges and further improve motion planning, future work could explore data-driven (learning-based) approaches that leverage real-world driver behavior to generate more informed and adaptable motion plans.

Acknowledgements

The past two years at TU Delft have been an incredible journey. The master's study indeed leads me to the fantastic field of robotics. Beyond academics, these two years away from my home country have fostered my personal growth, making me more resilient and prepared for the challenges that lie ahead. Here, I want to thank all the people who have been a part of this journey.

First and foremost, I would like to thank my thesis supervisor, Prof. Martijn Wisse, whose unwavering support and expert guidance have been invaluable throughout my thesis project. Your teachings about impactful research: addressing critical problems, conducting thorough analysis and experiments, and telling an appealing story, will always resonate with me. I also want to thank my daily thesis advisor, Luyao Zhang sincerely. Your guidance extends beyond offering insights and answering my questions: you actually serve as a role model of a dedicated and curious researcher.

I am equally grateful to Dr. Wei Pan for leading me into the research world. Your patience, encouragement, and insightful guidance have been essential in shaping my early steps in this academic journey. Your passion and brilliance will keep encouraging me in the future. I'd also like to extend my appreciation to Dr. Desong Du for always being kind and considerate to me. Our exciting discussions are precious and memorable.

I want to express my sincere gratitude to my amazing friends at TU Delft. Your friendship has been a source of strength for me, especially during the hard times. I've cherished our time together and the valuable lessons from each of you. Your diligence and remarkable aspirations have consistently driven me forward during my master's studies.

Lastly, my gratitude goes to my dear parents and my girlfriend. Pursuing an education in a foreign land has always been a dream of mine, and your unwavering support and understanding have made it possible. Your encouragement and love have been my pillars of strength, without which I could not have achieved this milestone.

Delft, University of Technology
August 23, 2023

Shaohang Han

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

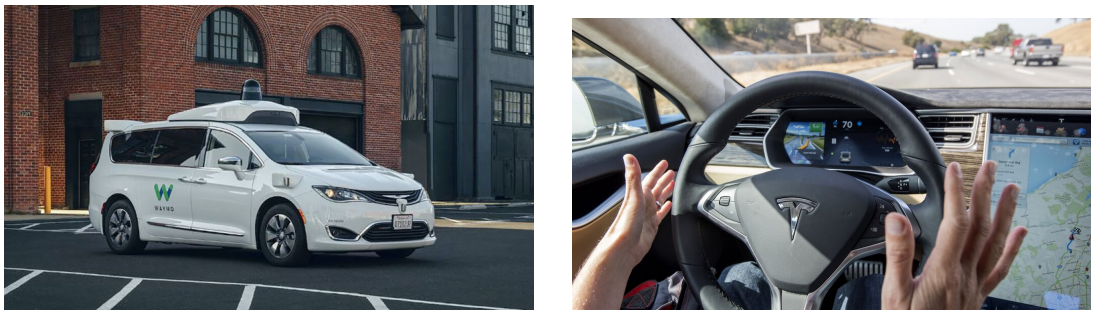
In this chapter, we begin by laying out the motivation for this thesis through an introduction to the lane merging challenge for automated vehicles (Section 1-1). Afterward, we outline the contributions of this thesis (Section 1-2) and offer an overview of the thesis's structure (Section 1-3).

1-1 Motivation

Automated vehicles have the potential to revolutionize transportation by enhancing traffic safety, reducing congestion, and providing mobility to non-drivers. For the first time, companies have deployed fully automated taxis on public roads (see Figure 1-1(a)) [1] , allowing passengers to experience the convenience and efficiency of self-driving vehicles. Moreover, automated driving technologies are being developed to assist drivers in navigating various scenarios (see Figure 1-1(b)) [2], offering features like lane-keeping assistance and adaptive cruise control. These advancements highlight the significant progress in achieving safer and more efficient transportation.

Despite the rapid development of automated vehicles, they still encounter challenges when navigating complex traffic scenarios. One challenging scenario is lane merging in urban sites. In this situation, the ego vehicle changes lanes by the routing requirements. It must identify a suitable gap in the target lane and potentially force the upstream traffic to slow down, ensuring a safe merging process [3]. Figure 1-2(a) visually depicts this challenging scenario, which can pose difficulties for both human drivers and automated vehicles alike. Real-world demonstrations have shown instances where automated vehicles struggle to merge into the desired lane or endure prolonged waiting times [4], as illustrated in Figure 1-2(b).

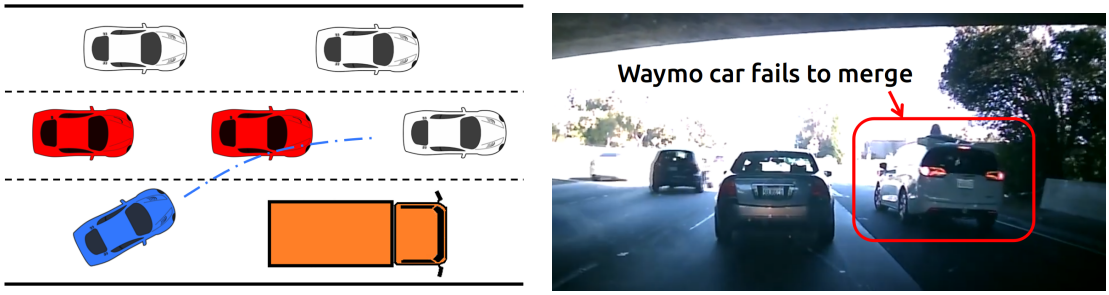
Within this context, automated vehicles must effectively coordinate with other road users. While communication is possible between robotic systems, exchanging messages with human drivers directly is not always feasible. Consequently, automated vehicles must be capable of planning the negotiations with surrounding vehicles, considering the potential variations in



(a) Automated taxi from Waymo (b) Autopilot system from Tesla

Figure 1-1: Illustration of the real-world applications of automated driving. (a) shows a fully automated taxi from Waymo. (b) shows the autopilot system from Tesla.

their behavior. This necessitates the ability to plan safe and efficient trajectories, avoiding being overly conservative or aggressive.



(a) Illustration of a forced lane merging scenario (b) A real-world failure case of merging

Figure 1-2: The forced lane merging scenario. In (a), the ego vehicle (shown in blue) needs to change lanes to the upper lane due to an obstacle in the current lane, such as a slow-moving truck (shown in orange). To safely merge into the target lane, the ego vehicle may need to negotiate with other vehicles (shown in red), potentially forcing them to slow down. (b) shows a failure case where a Waymo automated vehicle froze in front of the target lane and failed to merge.

The challenges in the forced lane merging scenario can be summarized as follows:

- **Interaction with surrounding vehicles:** The ego vehicle not only needs to predict the motion of the surrounding vehicles during planning, but also needs to consider the impact of its own motion on the surrounding vehicles.
- **Multi-modal behavior of surrounding vehicles:** In the lane merging scenario, the surrounding vehicles may have different behaviors. For example, some vehicles may slow down to let the ego vehicle merge, while others may not. This multi-modal behavior poses a significant challenge for the planning algorithms.

1-2 Contribution

The goal of this master's thesis is to develop planning algorithms for automated lane merging. Specifically, this thesis makes the following contributions:

- **An Efficient Game Theoretic Behavior Planner:** We propose a novel framework that integrates game theory and interactive trajectory generation. Leveraging domain knowledge of the lane merging scenario, we model the interaction as a gap selection process. The multi-modal behavior of surrounding vehicles is formulated as actions in a matrix game, and a socially compliant trajectory is determined by finding the Nash equilibrium. We thoroughly validate the effectiveness of this framework using the high-fidelity simulator CARLA [5], demonstrating its capability to generate efficient and safe trajectories.
- **A Branch Model Predictive Control Motion Planner:** To address the challenges posed by the multi-modal behavior of surrounding vehicles, we explore the use of branch model predictive control (B-MPC) for motion planning. This approach constructs a trajectory tree that captures the various potential behaviors of surrounding vehicles. By optimizing the expected cost of the trajectory tree, B-MPC plans a trajectory with branches. Operating in a receding horizon fashion, it executes the root branch, which is shared by all scenarios. In this thesis, we implement a baseline B-MPC method and compare several variants. Our results illustrate the capability of B-MPC in the lane merging scenario: it can effectively accommodate the multi-modal behavior of surrounding vehicles.

1-3 Thesis Outline

The remaining sections of this thesis are structured as follows. Chapter 2 comprehensively reviews the existing literature on planning techniques for highly interactive scenarios. Chapter 3 introduces the preliminary concepts that form the basis for the subsequent chapters. Chapter 4 presents our proposed behavior planning algorithm and the scientific findings. Chapter 5 shifts the focus to motion planning. We begin by introducing the B-MPC method and detailing the baseline implementation. Following this, we explore several variants of the B-MPC method and compare their performance in the lane merging scenario. Lastly, Chapter 6 serves as the conclusion of this thesis, summarizing the key findings and insights. Additionally, it discusses potential avenues for future research and development.

Chapter 2

Literature Review

In this chapter, we conduct a literature review of algorithms for automated lane merging. Firstly, we introduce classical methods that involve decoupling prediction and planning into separate modules (Section 2-1). While these methods serve as the foundation for understanding the solution pipeline, we also explore more advanced interaction-aware planning methods that consider prediction and planning jointly (Section 2-2). Finally, we summarize the advantages and disadvantages associated with these methods, providing a comprehensive overview of the existing approaches in the field of automated lane merging (Section 2-3).

2-1 Decoupled Prediction and Planning

Traditionally, the decision-making pipeline in automated driving systems separates motion prediction and motion planning into distinct modules. The prediction module generates motion predictions for non-ego road users based on their motion history and the surrounding environment. Since the motion prediction methods are out of the scope of this thesis, the interested readers are referred to [6], [7] for further details in this area. Subsequently, the planning module assumes flawless predictions of surrounding non-ego vehicle movements over the planning horizon.

The trajectory planning problem can be challenging due to various constraints, including collision avoidance and adherence to traffic laws, which render the problem constrained but non-convex [8]. Consequently, the problem may exhibit multiple local minima, each constrained in different aspects [9]. To address this issue, modern automated driving frameworks typically employ a two-layer approach consisting of behavior and motion planning layers. In the behavior planning layer, algorithms select maneuvers globally, considering factors such as lane changes, merging, and overtaking. The subsequent motion planning layer then generates optimal trajectories that are not only feasible but also efficient and safe, taking into account the specific constraints and objectives of the driving scenario.

2-1-1 Maneuver Selection

One representative approach involves constructing spatial-temporal graphs through sampling. A uniformly sampled dense graph is commonly referred to as a lattice. In the state lattice approach, states are connected using spirals [10] or polynomials [11]–[13]. Dynamic programming [10], [13] is generally employed to find the optimal path. While the state lattice approach is intuitive, it can be computationally expensive due to *the curse of dimensionality*. To mitigate this issue, states can be randomly and sparsely sampled when constructing graphs [14], [15]. Recent work by De Groot et al. [16] utilizes a Visibility-PRM [17] to build a graph from the start to the goal, enabling the determination of an optimal path using algorithms such as A* search. However, this approach requires a pre-determined single target, which is unsuitable for lane merging scenarios where the target is generally unknown.

Another approach to maneuver selection focuses on identifying patterns of maneuvers. This method leverages domain knowledge of driving scenarios to classify maneuvers and impose constraints [9], [14]. Recent work by Lim et al. [18] validates the effectiveness and computational efficiency of this approach through real-world experiments. By relying on domain knowledge, this method is also more interpretable. However, generalizing this approach to other scenarios, such as open spaces, is difficult.

2-1-2 Trajectory Generation

Once a maneuver decision has been made, the next crucial step is to generate a trajectory that ensures both safety and dynamic feasibility. Trajectories can be represented either as parameterized splines or as sequences of discrete states. The former, being more computationally efficient due to fewer decision variables involved [19], lacks the ability to represent the dynamics as constraints explicitly. On the other hand, the latter allows for dynamic constraints in the optimization problem but comes with a higher computational cost.

In the literature, researchers have explored the use of splines, such as piece-wise Bezier curves [20] or piece-wise polynomials [21], [22], in conjunction with driving corridors or signed distance for collision avoidance constraints. Alternatively, discrete state representations have been employed in studies such as [23]–[25]. Nonlinear dynamics constraints are sometimes transformed into soft constraints [23], [25] or linearized iteratively [24].

2-2 Interaction-aware Planning

In the real world, the decoupled scheme may be overly conservative as it overlooks the potential reactions of other vehicles to the motion plans of the ego vehicle. This conservative approach can lead to hesitant behavior, commonly called the *frozen robot problem* [26]. To address this issue, interaction-aware planning is a more natural and effective approach, which involves jointly considering both planning and prediction. By incorporating prediction into the planning process, the ego vehicle can anticipate how other vehicles might respond to its motion plans, resulting in more proactive and confident behavior. Next, we will introduce two behavior planning methods, namely Partially Observable Markov Decision Process (POMDP) and Game Theory, along with a motion planning approach called trajectory tree.

2-2-1 POMDP

One promising interaction-aware method is the Partially Observable Markov Decision Process (POMDP) [27]. POMDP provides a rigorous mathematical framework for handling incomplete information, such as the unknown intentions of the surrounding vehicles. However, solving a POMDP could be computationally demanding when the problem size is large.

Prior research has focused on applying POMDP to address the issue of unknown intentions in lane-merging scenarios. Online solvers, such as POMCP [28], POMCPOW [29], and DESOPT [30], employ sampling techniques to estimate the action-value function. Other approximations of POMDP, such as QMDP [31], and heuristics [32], [33], have also been proposed.

2-2-2 Game Theory

Game theory can capture the complex interaction between multiple agents [34]. Previous research has extensively explored equilibrium solutions in the context of automated lane merging. Some studies have focused on seeking a Nash equilibrium by jointly planning trajectories for all vehicles [35], [36]. However, these methods are limited to finding local equilibria. Therefore, the solutions rely heavily on the initial guess.

In contrast, alternative approaches employ semantic-level actions as strategies. Among them, a leader-follower structure within a Stackelberg game has been proposed [3], [37], [38]. High-level actions, such as motion primitives [3] or waiting time before merging [37], [38], represent the vehicle strategies. However, determining the leader's and follower's relative roles can pose challenges [3]. While most methods assume the ego vehicle as the leader, the rationale behind this assumption is not always clear. Rather than the leader-follower structure of the Stackelberg game, a Nash game treats all agents equally. A representative method based on a Nash game is proposed in [39], although it lacks validation in a high-fidelity simulator. Another approach utilizes level-k reasoning to model human drivers' behavior [40]. However, this framework incurs a substantial computational burden due to the need for modeling the depth of human thinking [41].

2-2-3 Trajectory Tree

Another promising approach to interaction-aware planning is the concept of a trajectory tree, also known as contingency planning [42], [43] or scenario tree planning [44], [45]. The trajectory tree approach involves generating a trajectory with multiple branches, each representing a potential future scenario capturing different behavior modes of the surrounding vehicles. The ego vehicle then optimizes the tree by minimizing the expected cost. Only the shared root node is executed to satisfy the safety requirements across all possible scenarios.

In comparison to the POMDP [27], which needs to plan in a discrete state and action space due to computational constraints, the trajectory tree approach assumes only the unobservable intentions of the surrounding vehicles to be discrete. This makes the trajectory tree more scalable for real-world scenarios while actively gathering information as in POMDP [43], [45].

To address the nonlinear optimization problem over a tree structure, the authors in [43], [46] employ the differential dynamic programming (DDP) method. Alternatively, the authors in

[44], [45], [47] utilize the nonlinear model predictive control (N-MPC) method. Additionally, the authors in [47] adopt a real-time iteration scheme similar to sequential quadratic programming (SQP) to achieve high computational efficiency.

Enforcing the constraints of all branches as hard constraints can lead to excessive conservatism and potential infeasibility. To address this challenge, the authors in [44] propose a novel approach by introducing dynamic chance constraints that adapt based on the prediction outputs. This adaptive mechanism allows for a more balanced treatment of uncertainty and promotes feasible solutions.

2-3 Discussion

The decoupled planning approach offers simplicity and efficiency but can sometimes result in conservative motion plans. Conversely, interaction-aware methods explicitly model the behavior of surrounding vehicles, enabling a more proactive approach. In the long run, interaction-aware methods show greater promise as they have the potential to tackle highly interactive scenarios, including highly congested traffic. Currently, there is a lack of comparative studies evaluating the performance of these two approaches. This thesis addresses this gap through a small-scale comparative study in Section 4-5-2.

The POMDP approach, although mathematically appealing, can suffer from computational intractability due to its exponential complexity growth. On the other hand, game theory utilizes equilibrium concepts to plan the trajectories of interacting vehicles jointly. Existing studies primarily focus on modeling and formulating the game, leveraging the extensive research in game theory. However, game-theoretic planners typically select a single equilibrium solution, which may not align perfectly with real-world situations as human drivers may not always act rationally. To mitigate this issue, the trajectory tree approach can be employed to consider multiple scenarios and generate a contingency solution simultaneously. This thesis explores both the game theory and trajectory tree approaches in the context of automated lane merging.

2-4 Summary

The highlights of this chapter are summarized as follows:

- In this section, we thoroughly review the existing literature about planning in the context of automated lane merging. Typically, these planning frameworks consist of two layers: behavior planning and motion planning. We introduced some popular algorithms for each layer.
- Our investigation reveals that interaction-aware planners show greater promise in highly dynamic environments. This is mainly because they can account for the influence of the ego vehicle's motion plans on the predictions of surrounding vehicles.

Chapter 3

Preliminaries

In this chapter, we discuss the preliminaries of the upcoming chapters. We start by presenting the problem formulations in this thesis, which involve the matrix game (Section 3-1) and optimal control problem (Section 3-2). Then, we describe the modeling choice in this thesis, including a dynamics model (Section 3-3) and the knowledge-based behavior models (Section 3-4).

3-1 Matrix Game

A matrix game, also known as a normal form game [48], involves a set of players selecting actions from a finite action set to minimize their individual costs. Formally, a matrix game can be defined as a tuple $(\mathcal{N}, \Pi, J)$, where $\mathcal{N}$ is the set of players, $\Pi = \times_{\sigma\in\mathcal{N}}\Pi_\sigma$ represents the joint action space, and $J = \times_{\sigma\in\mathcal{N}}J_\sigma$ denotes the joint cost function. Here the sign $\times$ denotes the Cartesian product. J_σ is a function that maps the joint action space Π to the real number space $\mathbb{R}$ for player σ.

A two player matrix game is commonly visualized as a table, where rows represent the actions of player one and columns represent the actions of player two. The entries in the table correspond to the costs associated with the joint actions. When there are more than two players, the table extends into a multi-dimensional array.

Instead of focusing on optimality, the matrix game employs the concept of equilibrium to characterize its solution. One recognized type of equilibrium concept is the pure-strategy Nash equilibrium, which refers to a joint strategy where no player can unilaterally deviate to achieve a better cost [49].

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

$$J_\sigma(\pi^*_\sigma, \pi^*_{-\sigma}) \leq \min_{s_\sigma\in\Pi_\sigma} J_\sigma(s_\sigma, \pi^*_{-\sigma}),$$

where $\pi_{-\sigma}$ represents the set of actions taken by all players except player σ.

In contrast to the simultaneous decision-making in the Nash equilibrium, a Stackelberg equilibrium represents a sequential solution where one player makes decisions before the other players. This player, known as the leader, determines their strategy first, and the remaining players, called followers, play the best response action [49]. The technical definition is provided as follows.

Definition 2. *(Stackelberg equilibrium). A Stackelberg equilibrium is a set of two players' actions,* $\{\pi_L^*, \pi_F^*(\cdot)\}$ *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 subscripts, L and F, represent the leader and the follower, respectively.

3-2 Optimal Control Problem

The task of trajectory planning is often formulated as an optimal control problem (OCP) [50]. In this case, the trajectory can be represented as a sequence of discrete states $\boldsymbol{x}(0), \ldots, \boldsymbol{x}(N)$. The objective is to find a sequence of control inputs that minimize a cost function while satisfying various constraints. In discrete time, the OCP can be formulated as follows:

$$\begin{aligned}
&\underset{\substack{\boldsymbol{x}(0),\ldots,\boldsymbol{x}(N),\\ \boldsymbol{u}(0),\ldots,\boldsymbol{u}(N-1)}}{\text{minimize}} && \sum_{t=0}^{N-1} l(\boldsymbol{x}(t), \boldsymbol{u}(t)) + l_N(\boldsymbol{x}(N)) && \text{(3-1a)}\\
&\text{subject to} && \boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \forall t \in [0, N-1] && \text{(3-1b)}\\
& && \boldsymbol{u}(t) \in \mathcal{U}, \quad \forall t \in [0, N-1] && \text{(3-1c)}\\
& && \boldsymbol{x}(t) \in \mathcal{X}, \quad \forall t \in [0, N] && \text{(3-1d)}\\
& && \boldsymbol{x}(0) = \boldsymbol{x}^{\text{init}} && \text{(3-1e)}\\
& && \boldsymbol{x}(N) \in \mathcal{X}_f && \text{(3-1f)}
\end{aligned}$$

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ represent the state and input, respectively, at time step t. The planning horizon is denoted by N, l represents the stage cost, l_N is the terminal cost, $\mathcal{U}$ represents the control input constraint set, $\mathcal{X}$ denotes the state constraint set, and $\mathcal{X}_f$ represents the state terminal constraint set. The OCP is solved by computing a sequence of control inputs and states that minimize the cost function.

When only the first control input $\boldsymbol{u}(0)$ is applied to the system, and the optimization problem is solved in a receding horizon fashion, the method is commonly referred to as model predictive control (MPC). MPC is a powerful control technique that utilizes prediction to compute control inputs over a finite horizon. If the system's dynamic model is nonlinear, or the constraints are non-convex, or the cost is non-convex, the optimization problem becomes a nonlinear program (NLP). To address this NLP, one can implement algorithms like sequential quadratic programming (SQP) [51] or utilize existing solvers like IPOPT [52].

3-3 Kinamic Bicycle Model

In this thesis, we employ the kinematic bicycle model to represent the dynamics of the vehicles. The kinematic bicycle model is a simplified yet widely used model in the relevant literature [3], [20], [23], [25]. Although the kinematic bicycle model may introduce errors at high speeds, it remains a computationally efficient model that provides sufficient accuracy within certain velocity and acceleration constraints [53]. Figure 3-1 illustrates the kinematic bicycle model.

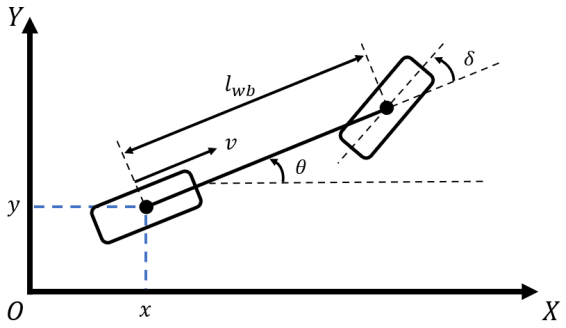


Figure 3-1: An illustration of the kinematic bicycle model.

By assuming: (i) We only steer the front wheels; (ii) The tires are infinitely stiff; (iii) There is no side slip. The kinematic bicycle model is described by the following continuous-time differential equations:

$$\begin{aligned}\dot{x} &= v\cos(\theta)\\ \dot{y} &= v\sin(\theta)\\ \dot{v} &= a\\ \dot{\theta} &= \frac{v}{l_{\text{wb}}}\tan\delta\end{aligned}$$

where x and y are the Cartesian positions of the vehicle; v represents the speed of the vehicle; a is the acceleration with the same direction as v; θ is the heading angle; δ is the front wheel steering angle; l_{wb} describes the length of the wheelbase. Thus, the state $\boldsymbol{x} = [x, y, \theta, v]^\top$. In this model, the control inputs are the acceleration and the front-wheel steering, $\boldsymbol{u} = [a, \delta]^\top$, which can be executed by an Ackermann steering vehicle.

3-4 Knowledge-based Behavior Model

In many literature discussed in Chapter 2, the human drivers are commonly modeled using knowledge-based physical models. This thesis adopts a similar approach, and the design choices related to this modeling strategy are discussed in the subsequent sections.

3-4-1 Lane Keeping Model

The Intelligent Driver Model (IDM) [54] and its variants are widely employed in the literature as the predominant lane-keeping models. The IDM is a car-following model that characterizes

the acceleration of a following vehicle based on factors such as the distance to the leading vehicle, the speed difference with the leading vehicle, and the speed of the following vehicle. The acceleration is computed as follows:

$$\dot{v} = a\left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right)$$
$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right)$$

where v is the speed of the following vehicle, a is the maximum acceleration, v_0 is the desired speed, δ is the acceleration exponent, s is the distance to the leading vehicle, s_0 is the minimum gap, T is the time gap, Δv is the speed difference between the following vehicle and the leading vehicle, b is the comfortable deceleration. The IDM is a deterministic model, which means that the acceleration is a function of the state of the following vehicle and the leading vehicle.

To account for the reactive behavior of vehicles on the adjacent lane, the authors in [55] introduce the Predictive Intelligent Driver Model (P-IDM). The P-IDM incorporates the cooperative nature of following vehicles by predicting the future motion of preceding vehicles on the adjacent lane. The acceleration is calculated using the same equation as in Equation 3-4-1, but the vehicles on the adjacent lane can also be regarded as leaders if their estimated lateral position is in close proximity to the following vehicle.

3-4-2 Behavior Modes

In this context, we characterize the behavior of a vehicle through a collection of policies $\pi^i, i = 1, \ldots, M$, where each π_i represents a policy corresponding to a specific behavior mode. Here we assume there are M behavior modes in total. Each policy π_i is a function that maps the joint state of the vehicles to a control input. Inspired by works such as [45], [47], we adopt the "noisily rational" Boltzmann model from cognitive science [56]. The probability of a vehicle selecting a particular policy π^i is determined using a softmax function:

$$P(\pi^i \mid \boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}}) = \frac{\exp\left(Q^i\left(\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}}\right)\right)}{\sum_{i=1}^{M} \exp\left(Q^i\left(\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}}\right)\right)}$$

where Q^i is the expected reward of policy π^i. $\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}}$ stand for the state of the ego vehicle and the surrounding vehicles, respectively. The model assumes humans are exponentially more inclined to choose a policy that offers a higher expected reward. This expected reward Q^i can be determined either through rule-based heuristics [47] or learned from data [57]. In this thesis, we adopt the former approach due to its interpretability and ease of implementation.

3-5 Summary

The highlights of this chapter are summarized as follows:

- We introduce the tools we will use in this thesis, namely the matrix game and the OCP.
- We present the modeling choices in this thesis, including a kinematic bicycle model for vehicle dynamics, a model for lane keeping, and a model for behavior selection.

Chapter 4

Behavior Planning: An Efficient Game-Theoretic Planner

In this chapter, we present a novel behavior planner that merges game theory with interactive trajectory generation to address the challenge of automated lane merging. We initiate by highlighting our contribution by contrasting the proposed method with the current state-of-the-art approaches (Section 4-1). Subsequently, we delve into the problem formulation through the utilization of a matrix game in the context of lane merging (Section 4-2). The introduction of the game-theoretic planner follows (Section 4-3). Further insights into implementation specifics are covered (Section 4-4). Finally, our proposed method is validated via numerical simulations (Section 4-5).

The contents of this chapter have been incorporated into a scientific paper, which has successfully been accepted by the IEEE Intelligent Transportation Systems Conference (ITSC) 2023[1]. For the sake of completeness, the scientific paper version is included in Appendix A. The author acknowledges the possibility of repetition resulting from this inclusion and apologizes for any inconvenience.

4-1 Introduction

Automated vehicles are facing a significant challenge when navigating in highly interactive environments, such as the lane-merging scenario shown in Figure 4-1. Traditional methods typically adopt a hierarchical structure where motion prediction [58], [59] and planning [20], [25] are decoupled. Consequently, these methods might be overly conservative since they often overlook the mutual interaction between the ego vehicle and the surrounding ones. Although newly developed learning-based approaches [60]–[62] consider such interaction, they typically require large amounts of data and might lack interpretability. Another popular interaction-aware method is the Partially Observable Markov Decision Process (POMDP) [27], which

[1] https://2023.ieee-itsc.org/

provides a rigorous mathematical framework for handling incomplete information, such as the unknown intentions of the surrounding vehicles. However, as the problem size grows, solving a POMDP becomes computationally intractable.

As an approximation of the POMDP framework, the multiple policy decision making (MPDM) method [63] and its extension EPSILON [64] have demonstrated promising results in generating practically reasonable trajectories while remaining computationally efficient. The approach involves conducting multi-vehicle forward simulations based on semantic-level policies, followed by a trajectory evaluation step to select the best trajectory using handcrafted criteria. However, the rule-based trajectory evaluation in EPSILON might be overly aggressive or conservative when the surrounding vehicles have multiple behavior modes. Additionally, although the open-loop planning strategy is computationally efficient, it sacrifices the advantages of active information gathering in the original POMDP approach.

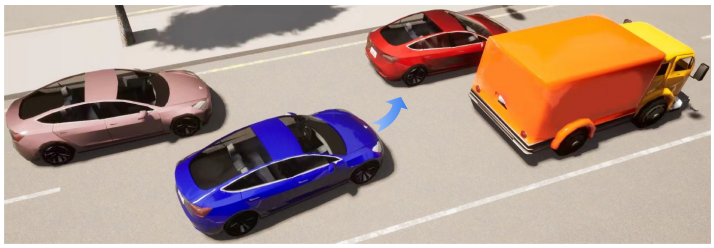


Figure 4-1: Lane-merging scenario in the CARLA simulator. The ego vehicle (blue) is merging onto a lane in dense traffic.

To systematically evaluate the trajectories, we use game theory, which is a powerful mathematical framework that captures the mutual influence between multiple agents [34]. Previous research has explored equilibrium solutions extensively for automated lane merging. Some studies have focused on jointly planning trajectories for all vehicles by seeking a Nash equilibrium [35], [36]. However, these methods can only find a local equilibrium, and the quality of the solution might heavily depend on the initial guess. In contrast, other approaches use semantic-level actions as strategies. Among them, some studies propose a Stackelberg game with a leader-follower structure [3], [37], [38]. However, determining the relative role of the leader or follower might be difficult [3]. In contrast to the leader-follower structure in a Stackelberg game, a Nash game treats all agents equally. A representative method based on a Nash game is proposed in [39], but it lacks validation in a high-fidelity simulator.

Contribution: We propose a novel approach that combines game theory with the idea of interactive trajectory generation. To make the algorithm practical and efficient, we leverage the semantic-level actions and model vehicle interaction as a gap selection process (Section 4-3-1). Additionally, to tackle the issue of multi-modality, we represent the behavior of the surrounding vehicles as actions in a matrix game, and then select the Nash equilibrium with the lowest social cost [65] (Section 4-3-3). We also investigate the existence of Nash equilibria and the relationship between Nash and Stackelberg equilibria through both theoretical analyses (Section 4-3-4) and numerous numerical simulations (Section 4-5-1). Moreover, we validate the effectiveness of the proposed planner in the high-fidelity CARLA simulator [5] (Section 4-5-2).

4-2 Problem Formulation

In this chapter, we consider a mixed-traffic scenario where an automated ego vehicle interacts with the surrounding vehicles, as shown in Figure 4-1. Specifically, the ego vehicle aims at driving efficiently but a low-speed vehicle (SV0) travels in front of it. To avoid being blocked, the decision-making system of the ego vehicle needs to consider the diverse driving behaviors of the surrounding vehicles, select a suitable merging gap, and determine if/when to change the lane. For example, in Figure 4-1, the ego vehicle can merge ahead of or after the pink vehicle (SV2). In fact, if SV2 yields, then the gap enlarges and the ego vehicle merges ahead of SV2. Otherwise, if the gap is not sufficiently wide, then the ego vehicle might slow down and then merge after SV2.

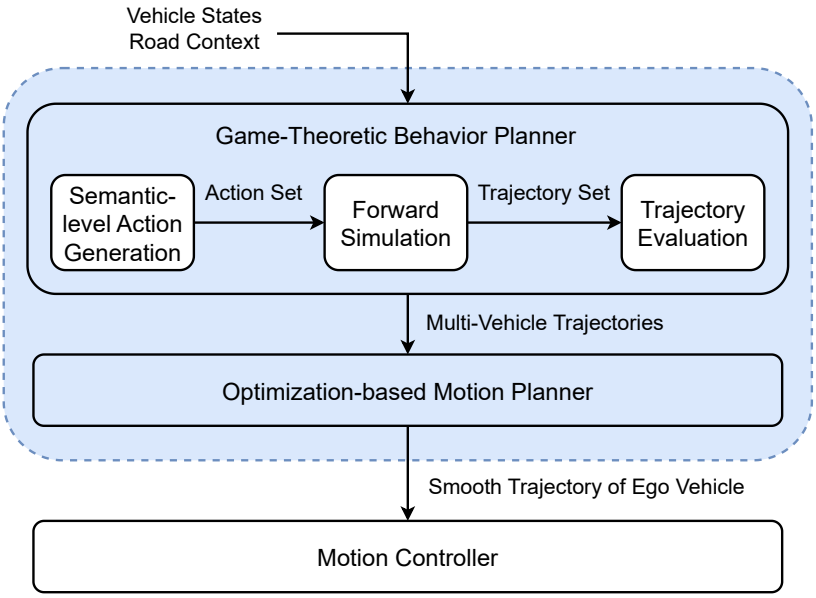


Figure 4-2: Structure of the proposed behavior and motion planner. The behavior planner, motion planner, and controller will be presented in Section 4-3 and Section 4-4, respectively.

4-2-1 Structure of the Planner

We propose a game-theoretic planner as illustrated in Figure 4-2. Unlike other conventional behavior planners that require a motion predictor as an upstream module, in our approach, we combine motion prediction and behavior planning. Our proposed game-theoretic behavior planner consists of three modules: semantic-level action generation, forward simulation, and trajectory evaluation. First, we enumerate the possible semantic-level decision sequences of the traffic participants over the decision horizon. For the ego vehicle, a semantic-level decision can be making a lane change, accelerating or decelerating. Then, we form the action tuples by combining the decision sequences of the ego vehicle and surrounding vehicles. For each action tuple, the forward simulator generates the motion of the relevant vehicles. Subsequently, the trajectory evaluator determines the costs of the trajectories for each action tuple. Next, we construct a matrix game and seek an equilibrium. Finally, the trajectory evaluator selects

the action tuple associated with the equilibrium and outputs the corresponding multi-vehicle trajectories. The formulation using the matrix game is presented in Section 4-3.

Although we apply the kinematic bicycle model (see Section 3-3) to simulate the motion of the ego vehicle, the trajectory generated by the behavior planner might not be sufficiently smooth due to the coarse discretization step. Therefore, we employ an additional local motion planner [25] to produce a kinematically feasible trajectory. To ensure safety, the simulated trajectories of the surrounding vehicles are used to impose dynamic collision avoidance constraints. More details of the motion planner and the subsequent controller will be presented in Section 4-4.

4-2-2 Lane Merging as a Matrix Game

We model the decision-making process as a matrix game with two players. In the game, each player selects an action from its finite action set to optimize its individual cost. A two-player matrix game is defined by a tuple $(\mathcal{N}, \Pi, J)$, where $\mathcal{N}$ is the set of two players, $\Pi = \times_{\sigma\in\mathcal{N}}\Pi_\sigma$ is the joint action space, and $J = \times_{\sigma\in\mathcal{N}}J_\sigma$ is the joint cost function. Other details, including the equilibrium concepts, have already been introduced in Chapter 3. Next, let us introduce the three ingredients of the matrix game for the lane-merging problem: players, actions and cost functions.

Players

We consider the ego vehicle (EV) and the group of the surrounding vehicles (SV) as two players, $\mathcal{N} := \{\mathrm{EV}, \mathrm{SV}\}$. A set of assumptions is made to treat all surrounding vehicles as a single group, which is elaborated on in Section 4-3-1. In later sections, we use σ to denote one of the players, where $\sigma \in \mathcal{N}$.

Actions

Human drivers typically make semantic-level decisions to make lane changes safely and efficiently [64]. Inspired by human drivers, we represent the action of player σ by a semantic-level decision sequence, denoted as $\pi_\sigma = \{\pi_{\sigma,0}, \ldots, \pi_{\sigma,k}, \ldots, \pi_{\sigma,H-1}\}$, where H is the decision horizon. We provide more design details on the decision sets and the method for enumerating all possible decision sequences later in Section 4-3-1.

Cost functions

Before computing the costs, the forward simulator, which is introduced in Section 4-3-2, generates multi-vehicle trajectories. The cost function J_σ of vehicle σ evaluates the corresponding trajectory based on user-defined metrics, such as safety, efficiency, comfort, and navigation. We consider the surrounding vehicles as a whole by calculating the total cost as $J_{\mathrm{SV}} := \sum_{\sigma=2}^{N} J_\sigma$, where N is the number of vehicles. Technical details are provided in Section 4-3-3.

A matrix game is represented in Table 4-1, where each entry represents a cost tuple $(J_{\mathrm{SV}}^{ij}, J_{\mathrm{EV}}^{ij})$ received by the group (SV) and the ego vehicle (EV) after performing their respective actions, π_{SV}^{i} and π_{EV}^{j}. Next, we look for an equilibrium of the matrix game.

Table 4-1: Game in normal form.

	π_{EV}^{1}	$\cdots$	$\pi_{\text{EV}}^{M_{\text{EV}}}$
π_{SV}^{1}	$(J_{\text{SV}}^{1,1}, J_{\text{EV}}^{1,1})$	$\cdots$	$(J_{\text{SV}}^{1,M_{\text{EV}}}, J_{\text{EV}}^{1,M_{\text{EV}}})$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\pi_{\text{SV}}^{M_{\text{SV}}}$	$(J_{\text{SV}}^{M_{\text{SV}},1}, J_{\text{EV}}^{M_{\text{SV}},1})$	$\cdots$	$(J_{\text{SV}}^{M_{\text{SV}},M_{\text{EV}}}, J_{\text{EV}}^{M_{\text{SV}},M_{\text{EV}}})$

As previously discussed in Section 3-1, two distinct types of equilibria have been identified: the pure-strategy Nash equilibrium and the Stackelberg equilibrium. This thesis takes into account both types. Furthermore, we should note that the roles of leader and follower are not static on the road [3]. In other words, the ego vehicle can alternate between these roles. Consequently, two variations of the Stackelberg equilibrium are considered: one with the ego vehicle as the leader and another with the ego vehicle as the follower.

4-3 Game-Theoretic Behavior Planner

In this section, we present the details of the game-theoretic behavior planner.

4-3-1 Semantic-Level Action Generation

Actions of Ego Vehicle

In the lane-merging problem, the semantic-level decision involves selecting a gap and determining the desired lateral position. As shown in Figure 4-3, the ego vehicle in blue has three potential gaps to choose from. To reach the target gap, the ego vehicle needs to perform a sequence of lateral decisions. The common lateral decisions are lane changing and lane keeping. Furthermore, we introduce one additional intermediate lane, represented by the dashed blue line in Figure 4-3, to enable a probing decision. This allows the ego vehicle to gather information and negotiate with the surrounding vehicles. Overall, the complete lateral decision set can be defined as:

$$D^{\text{lat}} := \{\texttt{LaneKeep}, \texttt{LeftLaneChange}, \texttt{LeftLaneProbe}\}.$$

The semantic-level decision at decision step k is denoted by an action tuple $\pi_{\text{EV},k} := (g_k, d_k^{\text{lat}})$, 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 tuples. For example, if the ego vehicle chooses `Gap0`, then the only available lateral action is keeping the current lane.

Next, we construct a decision tree to enumerate all the possible decision sequences. Each node in the tree represents a decision tuple. 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. By employing human-understandable

semantic-level decisions, we can establish rules for tree pruning. To begin, we introduce a constraint on the number of decision changes across the planning horizon. This choice aligns with the observed tendency of human drivers to maintain their chosen actions for extended periods. In our implementation, we have imposed a restriction that within each planning cycle, the action sequence can only encompass a single change of action. Given the brief time span typically associated with each semantic-level action (around 1 second), we have further assumed that the action sequence will remain unaltered when the tree's depth exceeds a certain value. This assumption is justifiable, as the distant future has less impact on the current moment, and our planner operates in a receding horizon manner. Additionally, we have excluded specific transitions that do not conform to the behaviors of ordinary human drivers. For instance, we have disregarded transitions like moving from $(\texttt{Gap1}, \texttt{LeftLaneChange})$ to $(\texttt{Gap2}, \texttt{LeftLaneChange})$. This action space refinement aids in minimizing the computational cost associated with the planner.

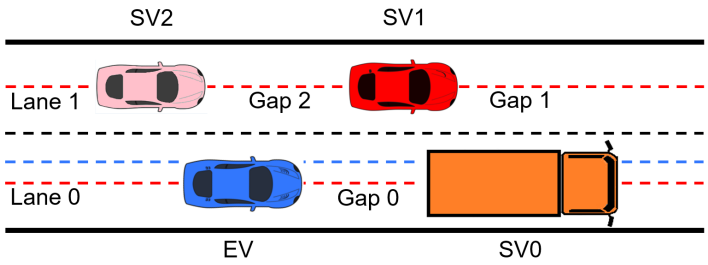


Figure 4-3: Semantic-level decisions. Three gaps are available for the ego vehicle to choose from: `Gap0`, `Gap1` and `Gap2`. The red dashed lines represent the centerlines of the lanes, and the blue dashed line represents the probing line. EV stands for the ego vehicle, while SV0, SV1, and SV2 represent the surrounding vehicles. Specifically, SV0 represents a slow-moving trunk that obstructs the path of the ego vehicle.

Actions of the Surrounding Vehicles

We make the following assumptions on the surrounding vehicles: (i) The surrounding vehicles maintain their lanes and have only longitudinal motion - a common assumption in prior work [3], [37]–[40]; therefore, we define their longitudinal decision set as $\{\texttt{Assert}, \texttt{Yield}\}$. (ii) The surrounding vehicles maintain their decisions throughout each forward simulation. This assumption is reasonable in practice since the planner runs in a receding horizon fashion. (iii) The ego vehicle only directly interacts with at most one surrounding vehicle throughout each forward simulation. For instance, as shown in Figure 4-3, if the ego vehicle selects `Gap2`, its motion affects SV2 but not the vehicles ahead (SV0 and SV1). Therefore, we treat the group of the surrounding vehicles as one single player in the matrix game so that we can sum up the trajectory cost of every surrounding vehicle. The action set of the player is $\Pi_{\text{SV}} := \{\texttt{Assert}, \texttt{Yield}\}$.

4-3-2 Multi-vehicle Forward Simulation

Vehicle Dynamics

Next, we intend to generate the trajectories by simulating the motion of the vehicles from the initial states. We represent the dynamics of vehicle σ as a kinematic bicycle:

$$\dot{x}_\sigma = v_\sigma \cos(\theta_\sigma),\ \dot{y}_\sigma = v_\sigma \sin(\theta_\sigma),\ \dot{\theta}_\sigma = \frac{v_\sigma}{l_{\text{wb}}} \tan(\delta_\sigma),\ \dot{v}_\sigma = a_\sigma,$$

where (x_σ, y_σ), θ_σ and v_σ are the position, the heading angle, and the speed, respectively; a_σ and δ_σ are the acceleration and the steering angle; l_{wb} represents the length of the wheelbase. A detailed description with illustration has been done in Section 3-3. The state vector is denoted as $\boldsymbol{x}_\sigma = [x_\sigma, y_\sigma, \theta_\sigma, v_\sigma]^\top$. Since we assume that the surrounding vehicles do not make lane changes, their heading and steering angles are equal to zero during the forward simulation ($\theta_\sigma = 0$, $\delta_\sigma = 0$). We discretize the dynamics via the Runge-Kutta 3 method.

Motion of the Ego Vehicle

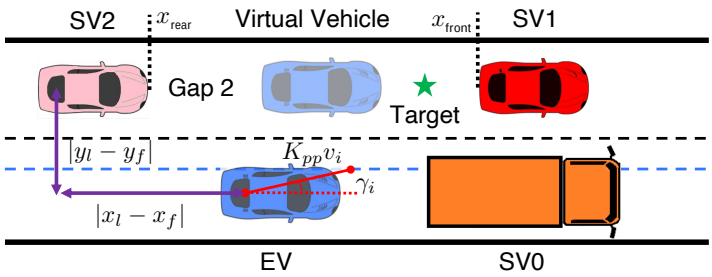


Figure 4-4: Ingredients for controller designs. $|x_l - x_f|$ and $|y_l - y_f|$ represent the longitudinal and lateral distances between the lane-changing vehicle (leader) and the interactive vehicle (follower), respectively.

We use two separate controllers to simulate the longitudinal and lateral motion of the vehicle. For the longitudinal motion, we track the target longitudinal position and the desired speed via a PD controller. The target longitudinal position within the desired gap is illustrated in Figure 4-4. The desired gap, `Gap2`, is defined based on the positions of the front and rear vehicles, denoted as x_{front} and x_{rear}, respectively. Similar to [64], we establish the desired longitudinal position and speed through a rule-based approach. For instance, a straightforward technique involves incorporating a safe distance to the interactive vehicle's current longitudinal position to derive the target longitudinal position.

As for the lateral motion, we adopt a pure pursuit controller [66] that requires the current vehicle speed and the target line as inputs. The steering angle is computed by $\delta_\sigma^{\text{ctrl}} = \tan^{-1}\left(\frac{2L\sin(\gamma_\sigma)}{K_{pp}v_\sigma}\right)$, where γ_σ represents the angle between the heading direction and lookahead direction, K_{pp} is the feedback gain, and $K_{pp}v_\sigma$ is the lookahead distance.

Motion of the Surrounding Vehicles

To model the behavior of the surrounding vehicles, we propose a modified intelligent driver model (IDM). Unlike the original IDM [54], which focuses solely on car following and disregards vehicles on adjacent lanes, our modified model considers lane-changing vehicles by projecting them onto their target lanes, resulting in virtual vehicles as shown in Figure 4-4. Subsequently, we calculate the distance between the virtual leader and the follower using the following approach:

$$d_{\text{idm}} = |x_l - x_f| e^{\kappa |y_l - y_f|},\ \kappa = 2\log(\beta)/w_{\text{lane}},$$

where w_{lane} is the lane width, and β is a parameter characterizing the level of willingness to yield. In fact, by adjusting the value of β, we can model different actions performed by the group of surrounding vehicles. 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 ($|y_l - y_f| = 0$), the virtual distance between them is equivalent to the true distance.

4-3-3 Trajectory Evaluation

After generating multi-vehicle trajectories for each action tuple, we proceed to select a specific action tuple by solving a matrix game. For constructing the cost matrix, we first introduce the cost function J_σ of vehicle σ, where $\sigma \in \mathcal{N}$. The cost function is typically a combination of several user-defined metrics, including safety, efficiency, comfort, navigation, and information cost: $J_\sigma = J_\sigma^{\text{saf}} + J_\sigma^{\text{eff}} + J_\sigma^{\text{com}} + J_\sigma^{\text{nav}} + J_\sigma^{\text{inf}}$. The value of the cost function J_σ depends on the trajectories generated by the forward simulator, which are influenced by the semantic-level decision sequences of the ego vehicle (π_{EV}) and the surrounding vehicles (π_{SV}). Given our existing assumption of solely longitudinal motion, we can ignore the information cost for the surrounding vehicles, as they are no longer capable of producing "probing" actions.

We calculate the safety cost by examining vehicle collisions. Here, the footprint of vehicle σ is modeled as a rectangle $\mathcal{R}_\sigma(\boldsymbol{x}_\sigma)$. If the distance between two rectangles is less than a small value $\underline{d}$, indicating a potential collision, we assign a very large penalty to the corresponding trajectory. With this in mind, we compute the safety cost as follows:

$$J_\sigma^{\text{saf}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := \sum_{t=0}^{T} \sum_{\tau=1,\, \tau\neq\sigma}^{N} P(\boldsymbol{x}_\sigma(t), \boldsymbol{x}_\tau(t))$$

where T is the planning horizon and N is the number of vehicles. We design P as follows:

$$P(\boldsymbol{x}_\sigma(t), \boldsymbol{x}_\tau(t)) := \begin{cases} w_1^{\text{saf}} & \text{if } 0 \le d_{\sigma\tau}(t) < \underline{d} \\ w_2^{\text{saf}} & \text{if } \underline{d} \le d_{\sigma\tau}(t) \le \bar{d} \\ 0 & \text{else} \end{cases}$$

where $d_{\sigma\tau}(t)$ represents the distance between $\mathcal{R}_\sigma(\boldsymbol{x}_\sigma(t))$ and $\mathcal{R}_\tau(\boldsymbol{x}_\tau(t))$, and w_1^{saf} is large than w_2^{saf}. By giving a relatively small penalty w_2^{saf} when $d_{\sigma\tau}(t)$ falls within the range of $[\underline{d}, \bar{d}]$, we encourage the vehicle to keep a suitable distance from the surrounding vehicles. This can be understood as slightly expanding the vehicle's dimensions, as depicted in Figure 4-5.

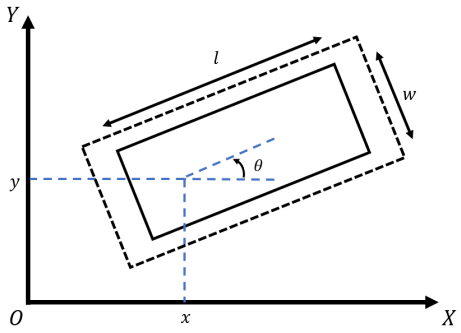


Figure 4-5: The footprint of a vehicle. The dashed edges represent the inflated boundary of the vehicle.

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:

$$J_\sigma^{\text{eff}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{eff}} \sum_{t=0}^{T} (v_\sigma(t) - v_\sigma^{\text{des}})^2.$$

For the comfort cost, we consider the acceleration change, known as jerk. We use the finite difference to approximate the jerk, and subsequently define the comfort cost as follows:

$$J_\sigma^{\text{com}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{com}} \sum_{t=1}^{T} \left(\frac{a_\sigma(t) - a_\sigma(t-1)}{\Delta t} \right)^2.$$

Next, we penalize the differences between the vehicle's lateral position and its desired lateral position to encourage the lane-changing maneuver:

$$J_\sigma^{\text{nav}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{nav}} \sum_{t=0}^{T} (y_\sigma(t) - y_\sigma^{\text{des}})^2.$$

In addition, inspired by [57], we introduce an information cost $J_{\text{EV}}^{\text{inf}}$, which can motivate the ego vehicle to identify the intentions of the surrounding vehicles actively. The behavior of the surrounding vehicles group is determined by their intention, denoted as π_{SV}, which is not directly observable. Therefore, we consider π_{SV} as a latent variable and introduce a belief state $b(\pi_{\text{SV}}^i)$ to capture the uncertainty associated with the intention i. Following Bayesian inference principles, we update the belief state using the following simplified equation:

$$b\left(\pi_{\text{SV}}^i\left(t+1\right)\right) \propto P\left(\pi_{\text{SV}}^i\left(t+1\right) \mid \boldsymbol{x}_{\text{SV}}(t), \boldsymbol{x}_{\text{SV}}(t)\right) \cdot b\left(\pi_{\text{SV}}^i(t)\right)$$

We assume that surrounding vehicles exhibit an exponential preference for selecting intentions with lower costs. This modeling choice has been motivated in Section 3-4-2 of this thesis. Thus, given that EV follows action j, we have:

$$P\left(\pi_{\text{SV}}^i\left(t+1\right) \mid \boldsymbol{x}_{\text{SV}}(t), \boldsymbol{x}_{\text{SV}}(t)\right) = \frac{\exp\left(-J_{\text{SV}}^{ij}\right)}{\sum_{i=1}^{M_{\text{SV}}} \exp\left(-J_{\text{SV}}^{ij}\right)}$$

where $i \in \{1, , \ldots, , M_{\mathrm{SV}}\}$ and $j \in \{1, , \ldots, , M_{\mathrm{EV}}\}$, the cost $J_{\mathrm{SV}} = J_{\mathrm{SV}}^{\mathrm{saf}} + J_{\mathrm{SV}}^{\mathrm{eff}} + J_{\mathrm{SV}}^{\mathrm{com}} + J_{\mathrm{SV}}^{\mathrm{nav}}$. To quantify the information cost $J_{\mathrm{EV}}^{\mathrm{inf}}$ considering all the behavior modes of SV, we utilize the entropy measure, denoted as $H(\cdot)$, similar to [40]:

$$J_{\mathrm{EV}}^{\mathrm{inf}} = H\left(b\left(\pi_{\mathrm{SV}}^{1:M_{\mathrm{SV}}}(t+1)\right)\right) - H\left(b\left(\pi_{\mathrm{SV}}^{1:M_{\mathrm{SV}}}(t)\right)\right)$$

where $b\left(\pi_{\mathrm{SV}}^{1:M_{\mathrm{SV}}}\right)$ stands for $b\left(\pi_{\mathrm{SV}}^{1}\right) \ldots b\left(\pi_{\mathrm{SV}}^{M_{\mathrm{SV}}}\right)$.

In fact, the information cost motivates the ego vehicle to create motion plans that yield distinct trajectory costs among the surrounding vehicles. This objective is accomplished by maximizing the entropy of the belief state.

In practice, the ego vehicle needs to estimate the cost functions of other vehicles by observing their trajectories since it cannot directly access these costs. Similar to POMDP, we account for the uncertainty in the aggregate cost of the surrounding vehicles by integrating the probabilities into the cost function. The modified aggregate cost is computed as follows:

$$\bar{J}_{\mathrm{SV}}^{ij} := (1 - p(\pi_{\mathrm{SV}}^{i}))J_{\mathrm{SV}}^{ij}, \quad \sum_{i=1}^{M_{\mathrm{SV}}} p(\pi_{\mathrm{SV}}^{i}) = 1,$$

where p represents the probability associated with the action of the surrounding vehicles. This design can be understood as incorporating prior knowledge about the behavior of the group of surrounding vehicles into the aggregate cost. For example, if we have prior knowledge suggesting that the group is likely to yield, then we can set the corresponding probability close to 1, which reduces the modified aggregate cost. We use a similar estimation algorithm as in [3] to recursively estimate p at the beginning of each planning cycle. Specifically, p can be updated using the following equation:

$$p\left(\pi_{\mathrm{SV}}^{i} \mid \boldsymbol{x}_{\mathrm{SV}}(t)\right) = \frac{\mathcal{N}\left(r^{i}(t), 0, W\right) p\left(\pi_{\mathrm{SV}}^{i} \mid \boldsymbol{x}_{\mathrm{SV}}(t-1)\right)}{\sum_{k}^{M_{\mathrm{SV}}} \mathcal{N}\left(r^{k}(t), 0, W\right) p\left(\pi_{\mathrm{SV}}^{k} \mid \boldsymbol{x}_{\mathrm{SV}}(t-1)\right)}$$

where $\mathcal{N}\left(r, 0, W\right)$ is the probability density function of the multivariate normal distribution with mean 0 and covariance W, evaluated at r. The residual r is given by the difference between the observed position of SV and the predicted position of SV.

After constructing the cost matrix, we compute a Nash equilibrium for the matrix game by enumerating all possible combinations of semantic-level actions. If multiple Nash equilibria exist, we select the equilibrium with the lowest social cost. If a Nash equilibrium does not exist, we choose the Stackelberg equilibrium with the ego vehicle as the follower as a backup solution.

4-3-4 On Nash and Stackelberg Equilibria

In this section, we examine the conditions for the existence of a pure-strategy Nash equilibrium and explore the relationship between the Nash and Stackelberg equilibrium. We consider a specific cost matrix where $\pi_{\mathrm{SV}}^{1} :=$ `Assert` and $\pi_{\mathrm{SV}}^{2} :=$ `Yield` as mentioned in Section 4-3-1. We call an action tuple $(\pi_{\mathrm{SV}}, \pi_{\mathrm{EV}})$ feasible if the corresponding multi-vehicle trajectories are free from collisions, and we assume that there always exists at least one feasible action tuple.

Next, we show the existence of a Nash equilibrium for the matrix game. To obtain the results of this section, we assume that for the group of the surrounding vehicles (SV), behaving politely incurs a higher cost, while the ego vehicle (EV) achieves a lower cost if SV shows polite behavior. This assumption is required by Propositions 1 and 2.

Proposition 1. *Assume that the inequalities* $0 \leq J_{SV}^{1m} \leq J_{SV}^{2m}$ *and* $J_{EV}^{1m} \geq J_{EV}^{2m} \geq 0$ *hold for all feasible action tuples and* $m \in \{1, \ldots, M_{\mathrm{EV}}\}$*. If* $p(\pi_{SV}^{1}) \geq 0.5$*, then there exists* $n \in \{1, \ldots, M_{EV}\}$ *such that the action tuple* $(\pi_{SV}^{1}, \pi_{EV}^{n})$ *is a pure-strategy Nash equilibrium.*

Proof. We select n such that $J_{\mathrm{EV}}^{1n} \leq J_{\mathrm{EV}}^{1m}$ for all $m \in \{1, , \ldots, , M_{\mathrm{EV}}\}$. Then, π_{EV}^{n} is the best response to π_{SV}^{1}. Using the assumptions in the proposition, we can conclude that $0 \leq J_{\mathrm{SV}}^{1n} \leq J_{\mathrm{SV}}^{2n}$. Furthermore, the inequality $(1-p(\pi_{\mathrm{SV}}^{1}))J_{\mathrm{SV}}^{1n} \leq p(\pi_{\mathrm{SV}}^{1})J_{\mathrm{SV}}^{2n}$ holds for $p(\pi_{\mathrm{SV}}^{1}) \geq 0.5$. Therefore, π_{SV}^{1} is the best response to π_{EV}^{n}, and $(\pi_{\mathrm{SV}}^{1}, \pi_{\mathrm{EV}}^{n})$ is a Nash equilibrium. □

In the following, we establish a connection between a Nash equilibrium and a Stackelberg equilibrium.

Proposition 2. *If the action tuple* $(\pi_{SV}^{2}, \pi_{EV}^{n})$ *is a Nash equilibrium, then it is also a Stackelberg equilibrium with EV as the leader.*

Proof. As $(\pi_{\mathrm{SV}}^{2}, \pi_{\mathrm{EV}}^{n})$ is a Nash equilibrium, π_{SV}^{2} is the best response to π_{EV}^{n}, and the inequality $J_{\mathrm{EV}}^{2n} \leq J_{\mathrm{EV}}^{2m}$ holds for all $m \in \{1, , \ldots, , M_{\mathrm{EV}}\}$. Based on the inequality $J_{\mathrm{EV}}^{1m} \geq J_{\mathrm{EV}}^{2m} \geq 0$, we can conclude that $J_{\mathrm{EV}}^{2n} \leq J_{\mathrm{EV}}^{2m} \leq J_{\mathrm{EV}}^{1m}$. Therefore, $(\pi_{\mathrm{SV}}^{2}, \pi_{\mathrm{EV}}^{n})$ is a Stackelberg equilibrium with EV as the leader. □

4-4 Implementation Details

As illustrated in Figure 4-2, the proposed planning framework is structured into three subsequent layers: the behavior planning layer, the motion planning layer, and the control layer. A coarse reference trajectory is generated in the behavior planning layer, then refined in the motion planning layer to ensure safety and dynamic feasibility. While Chapter 4 extensively discusses the interaction-aware behavior planning layer, we now provide detailed information about the motion planning layer and the control layer.

4-4-1 Motion Planning Layer

We adopt the motion planner proposed in [25] and utilize their official codebase. This planner is an optimization-based approach in the Cartesian frame, capable of explicitly enforcing dynamic constraints. It employs an OCP formulation (see 3-2) and the kinematic bicycle model (see Section 3-3) to generate a sequence of control commands. The state at time step t is represented by $\boldsymbol{x}(t) = [x(t), y(t), \theta(t), v(t)]^T$, where $x(t)$ and $y(t)$ denote the vehicle's position in the Cartesian frame, $\theta(t)$ is the heading angle, and $v(t)$ is the velocity. The state input is given by $\boldsymbol{u}(t) = [a(t), \delta(t)]^T$, where $a(t)$ stands for the acceleration, and $\delta(t)$ is the

steering angle. The overall OCP can be written as:

$$\begin{aligned}
&\underset{\substack{\boldsymbol{x}(0),\ldots,\boldsymbol{x}(N),\\ \boldsymbol{u}(0),\ldots,\boldsymbol{u}(N-1)}}{\text{minimize}} && J && \text{(4-1a)}\\
&\text{subject to} && \boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \forall t \in [0, N-1] && \text{(4-1b)}\\
& && \boldsymbol{u}(t) \in \mathcal{U}, \quad \forall t \in [0, N-1] && \text{(4-1c)}\\
& && \boldsymbol{x}(t) \in \mathcal{X}, \quad \forall t \in [0, N] && \text{(4-1d)}\\
& && \boldsymbol{x}(0) = \boldsymbol{x}^{\text{init}} && \text{(4-1e)}
\end{aligned}$$

Cost Function

The cost function of the OCP (Equation 4-1a) is quadratic and defined as:

$$J = \sum_{t=0}^{N} \left\| \boldsymbol{x}(t) - \boldsymbol{x}^{\text{ref}}(t) \right\|^2 + w_u \cdot \|\boldsymbol{u}(t)\|^2 \tag{4-2}$$

where $\boldsymbol{x}^{\text{ref}}(t)$ represents the reference trajectory generated by the behavior planning layer and w_u is a non-negative weighting parameter. This cost function penalizes deviations from the reference trajectory and control effort, which is a common choice for an OCP [50]. Originally in [25], the reference trajectory was generated by a state lattice planner, as described in Section 2-1-1. In this thesis, the reference trajectory is given by the output of the game-theoretic behavior planner. We also use the reference trajectory as the warm-starting initial guess at each planning cycle.

Constraints

The constraints of this OCP include simple bound constraints on both the states and the control commands, collision avoidance constraints, and nonlinear dynamics constraints.

Bound Constraints: Here, we assume that both states and control inputs are subject to limits, as commonly practiced in control applications. The bound constraints (Equation 4-1c, 4-1d) can be mathematically represented as $\boldsymbol{x}^{\min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}^{\max}$ and $\boldsymbol{u}^{\min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^{\max}$. We also use boundary constraint to depict the initial state constraint (Equation 4-1e) of the vehicle: $\boldsymbol{x}(0) = \boldsymbol{x}^{\text{init}}$. In our implementation, we disregard the terminal constraint since the vehicle is driving on a continuous road.

Collision avoidance constraints: The safety of the ego vehicle must be ensured concerning both surrounding vehicles and road edges, achieved through collision avoidance constraints (Equation 4-1d). To deal with the non-convex and potentially non-differentiable collision avoidance constraints [67], we first represent the ego vehicle as a set of circles, then construct the spatial-temporal corridors to form linear collision avoidance constraints.

The ego vehicle can be approximated as a union of circles, similar to [44]. As illustrated in Figure 4-6, we use $n_c = 3$ circles to represent the vehicle's footprint. Here, we have the radius $r = \frac{1}{2}\sqrt{(l/n_{\text{c}})^2 + w^2}$, where l and w are length and width of the vehicle, respectively. The

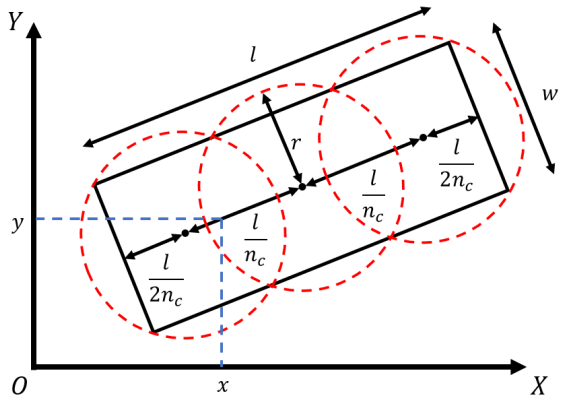


Figure 4-6: An illustration of collision avoidance constraints using circle approximation for vehicles.

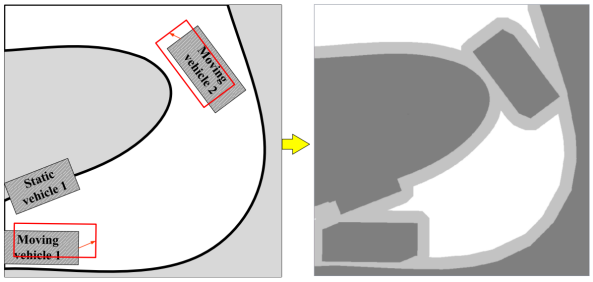
centers c can be expressed as

$$\begin{aligned} c^j(t) &= [x^j(t),\, y^j(t)]^T, \\ x^j(t) &= x(t) + \frac{l}{2n_c}(2j - n_c - 1)\cos(\theta(t)), \\ y^j(t) &= y(t) + \frac{l}{2n_c}(2j - n_c - 1)\sin(\theta(t)), \\ j &= 1, \cdots, n_c \end{aligned}$$

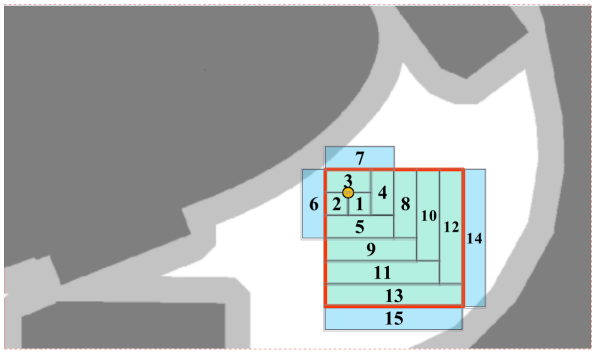
Next, we illustrate the process of creating a corridor by Figure 4-7(a) and 4-7(b). Initially, in Figure 4-7(a), we generate an occupancy grid map by considering the road edges' footprint, the configuration of static obstacles, and the predicted positions of moving obstacles. To account for the shape of the ego vehicle, the map is expanded by the radius r. This strategy enables us to model the ego vehicle using the centers of circles. The predicted positions of the surrounding vehicles are derived from the game-theoretic behavior planner. Then in Figure 4-7(b), the creation of local boxes involves iteratively evaluating the feasibility of incremental expansions in the four axis-aligned directions. The local corridor at the time step t can be represented as $x^{\text{lb}}(t), x^{\text{ub}}(t)$ and $y^{\text{lb}}(t), y^{\text{ub}}(t)$. A sufficient condition for collision avoidance is then $x^{\text{lb}}(t) \leq x^j_{\text{EV}}(t) \leq x^{\text{ub}}(t)$ and $y^{\text{lb}}(t) \leq y^j_{\text{EV}}(t) \leq y^{\text{ub}}(t)$, with $j = 1, \cdots, n_c$.

Dynamics constraints: To uphold the dynamics constraints (Equation 4-1b) , we utilize an iterative optimization framework. The core concept is to convert the dynamics constraints into soft constraints and then iteratively minimize the associated cost. This iterative solving scheme is demonstrated in Algorithm 1. During each iteration, the dynamics constraints are translated into an extra weighted penalty term and integrated into the cost function defined by Equation 4-2. This leads to creating an intermediate OCP (Line 6). The penalty term is given by:

$$l^{\text{penalty}} = \sum_{t=0}^{N-1} \|\boldsymbol{x}(t+1) - f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)\|^2 \tag{4-3}$$



(a) Occupancy grid map that considers the road edges and the obstacles.



(b) Formation of a local corridor.

Figure 4-7: Corridor creation in [25]: (a) depicts the generation of the occupancy grid map. (b) demonstrates the formation of a local corridor through the expansion of local boxes.

In the intermediate OCP, the total cost function is:

$$\sum_{t=0}^{N} \left\| \boldsymbol{x}(t) - \boldsymbol{x}^{\text{ref}}(t) \right\|^2 + w_u \cdot \|\boldsymbol{u}(t)\|^2 + w^{\text{penalty}} \cdot l^{\text{penalty}}$$

while the constraints include Equation 4-1c, 4-1d and 4-1e. Then, we can solve the intermediate OCP using non-linear solvers such as IPOPT [52] (Line 7). The solution obtained in one iteration serves as a warm start for the next iteration. Then, we can assess infeasibility by computing l^{penalty} based on this solution (Line 8). This iterative process continues until the penalty term becomes sufficiently small, ensuring improved feasibility of the OCP with each iteration (Line 9). In each iteration, w^{penalty} is enlarged by a factor α. If the OCP is feasible, the algorithm will end in advance. However, if the feasibility is not achieved, the algorithm will provide the best-optimized trajectory that it has been able to compute.

The iterative optimization framework we have employed stands out for its efficiency and ease of tuning. With the flexibility to adjust the parameter ϵ, we can effectively strike a balance between ensuring feasibility and optimizing efficiency. In comparison to ensuring dynamics feasibility through the concept of differential flatness [13], the explicit formulation of dynamics constraints can provide better guarantees, particularly when the reference trajectories exhibit curvature [25]. In contrast to methods that address the dynamics constraints by iterative linearization [24], incorporating dynamics constraints within the cost function holds a larger feasible region in each iteration. These characteristics make it a preferred method for integrating dynamics constraints into the optimization process.

Algorithm 1 An Iterative Trajectory Optimization Approach

procedure SOLVE($\bar{\boldsymbol{x}}^{\text{ref}}, \bar{\boldsymbol{x}}^{\text{opt}}$) ▷ $\bar{\boldsymbol{x}}^{\text{ref}}$: reference trajectory, $\bar{\boldsymbol{x}}^{\text{opt}}$: optimized trajectory
  $\bar{\boldsymbol{x}}^{\text{opt}} \leftarrow \bar{\boldsymbol{x}}^{\text{ref}}$
  $w^{\text{penalty}} \leftarrow w^{\text{penalty},0}$ ▷ $w^{\text{penalty},0}$ is the initial value of w^{penalty}
  $i \leftarrow 0$
  while $i < i^{\max}$ **do**
    $OCP \leftarrow$ Form the Intermediate OCP
    $\bar{\boldsymbol{x}}^{\text{opt}} \leftarrow$ Solve OCP by IPOPT
    $l^{\text{penalty}} \leftarrow$ Calculate by Equation 4-3
    if ($l^{\text{penalty}} < \epsilon$) **then** ▷ ϵ is a very small positive number
      break
    else
      $w^{\text{penalty}} \leftarrow w^{\text{penalty}} \cdot \alpha$ ▷ w^{penalty} is enlarged by α
      $i \leftarrow i + 1$
    end if
  end while
  return $\bar{\boldsymbol{x}}^{\text{opt}}$
end procedure

4-4-2 Control Layer

To control a vehicle within the high-fidelity simulator CARLA, generating low-level control commands such as throttle, steering, and brake is necessary. For this purpose, a PID controller was formulated to track a reference trajectory produced by the motion planning layer. Consequently, the inputs for the controller encompass the reference states and control inputs, while the controller's output includes the low-level control commands for the subsequent time step.

Specifically, the control input produced by the motion planning layer is utilized to establish a feedforward component through a table lookup. Subsequently, a feedback term is computed by the PID controller. The longitudinal and lateral control commands are computed separately within the PID controller. To be precise, the throttle is determined based on the longitudinal acceleration error, while the steering is calculated using the heading angle error.

4-4-3 Re-planning Scheme

When the vehicle drives on the road, it keeps taking new information from the perception module. For example, it might detect new obstacles or road boundaries. Therefore, it is necessary to conduct re-planning and run the planning layer in a receding horizon fashion. At the beginning of each planning cycle, a straightforward approach is to use the current state $\boldsymbol{x}^{\text{real}}$ of the ego vehicle as the starting point for re-planning. However, this approach can lead to discontinuities, as depicted in Figure 4-8, and result in poor tracking performance in the control layer due to a sudden reset of accumulated error. To overcome this, we initiate re-planning from the corresponding state $\boldsymbol{x}^{\text{planned}}$ of the last planned trajectory's time.

In our implementation, we assume a synchronous operation of the entire planning stack. This implies that the processing time of each planning layer is disregarded, making the subsequent

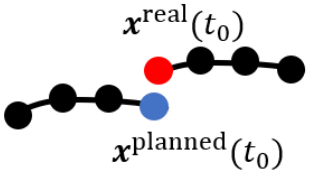


Figure 4-8: An illustration of the straightforward re-planning scheme. The red point represents the actual state at the start of the planning cycle, while the blue point signifies the planned state. Direct re-planning from the x^{real} can result in discontinuities in the trajectory output from the planning layer.

layer run only after the completion of the previous one. However, in real-world scenarios, the execution time of these planners cannot be neglected. To ensure trajectory smoothness more effectively, an alternative approach might be planning from a future pose instead of the current pose along the original trajectory, as suggested in [68]. In other words, the trajectory should remain consistent within the processing duration.

Moreover, the authors of [68] propose a trajectory stitching technique. This method involves binding the initial three states of the newly planned trajectory to the existing one, ensuring a third-level continuity in the trajectory. The mathematical foundation for this approach, involving variational analysis, is presented in [68].

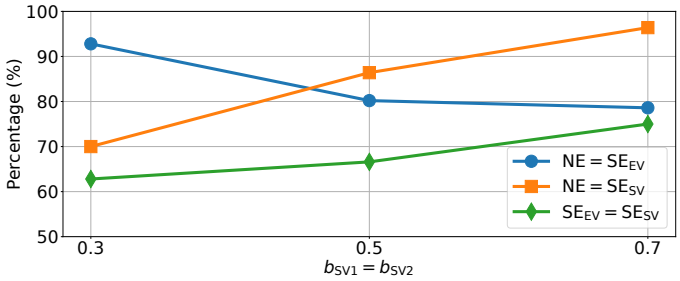
4-5 Numerical Simulations

We consider a lane-merging scenario as introduced in Section 4-2. The red (SV1) and pink (SV2) vehicles are considered interactive vehicles since the ego vehicle can influence their motion, while the orange vehicle (SV0) is a non-interactive dynamic obstacle. We set the initial beliefs on the interactive vehicles' decisions to $b_{\text{SV1}}(\pi^1_{\text{SV}}) = b_{\text{SV2}}(\pi^1_{\text{SV}}) = 0.5$. To simplify the notation, we use b^1_{SV1} and b^1_{SV2} to represent $b_{\text{SV1}}(\pi^1_{\text{SV}})$ and $b_{\text{SV2}}(\pi^1_{\text{SV}})$, respectively. We use a planning horizon of $T = 25$, a discretization step of $\Delta t = 0.2\,\text{s}$, a decision time period of $\Delta h = 1\,\text{s}$ and a decision horizon of $H = 5$.

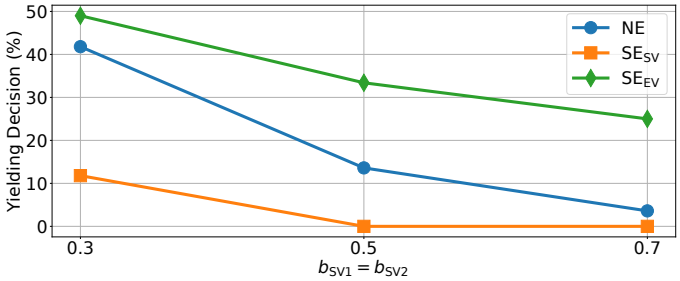
4-5-1 Open-loop Simlation

We conduct open-loop Monte Carlo simulations to empirically verify the existence of a Nash equilibrium and investigate its relationship with the Stackelberg equilibrium. We first specify nominal initial multi-vehicle states that are collision-free. Then, the state of the ego vehicle is perturbed by $\pm 10\,\text{m}$ for the initial position, $\pm 5\,\text{m/s}$ for the initial longitudinal speed. We run 500 simulations for each belief and compute three equilibria: Nash (NE), Stackelberg with the ego vehicle as the leader (SE_{EV}), and Stackelberg with the group of the surrounding vehicles as the leader (SE_{SV}). The results are presented in Figure 4-9(a). A Nash equilibrium is found in all simulations despite no theoretical guarantee in general. As mentioned before, if there are multiple Nash equilibria, we select the one with the lowest social cost. We observe that the selected Nash equilibrium coincides with one of the Stackelberg equilibria.

Figure 4-9(b) illustrates the statistical results on SV's yielding decisions in different equilibria over 500 simulations. Specifically, SV is more likely to yield when both players employ a SE_{EV}. In contrast, SV is less inclined to yield when both players select a SE_{SV}. This implies that SV seems to behave cooperatively if the ego vehicle is the leader. On the contrary, if the ego vehicle is the follower, then it tends to show conservative behavior because the other vehicles are likely to assert. Overall, we conclude that adopting a Nash equilibrium exhibits less interactive behavior than adopting a SE_{EV}, but it is less conservative compared to adopting a SE_{SV}. In other words, it seems that a vehicle can switch between interactive and conservative behavior automatically by selecting a Nash equilibrium.



(a) Relationship between Nash and Stackelberg equilibria



(b) Comparison of decisions in different equilibria

Figure 4-9: Results of Monte Carlo simulations. (a) illustrates the relationship between Nash and Stackelberg equilibria. (b) shows the percentage of yielding decisions made by SV in various equilibria over 500 simulations. The notations NE, SE_{SV} and SE_{EV} represent the Nash equilibrium, Stackelberg equilibrium with SV as the leader and Stackelberg equilibrium with EV as the leader, respectively.

4-5-2 Closed-loop Simulation in CARLA

We conduct closed-loop simulations in the CARLA simulator to evaluate our proposed approach. CARLA is an open-source simulator for automated driving research, which provides a realistic urban environment and a variety of vehicles. The experiments are conducted on a laptop with a Ryzen 7 5800H CPU and an NVIDIA RTX 3060 GPU. The proposed planner is implemented in Python, while the motion planner and tracking controller are implemented in C++. Communication between these components is facilitated through the ROS framework [69]. To focus on planning performance, we assume perfect state estimation and simulate

synchronous mode.

The scenario depicted in Figure 4-3 requires the ego vehicle to merge onto the target lane as quickly as possible within a limited lane length of 100 m. To simulate a real-world scenario, we add an additional surrounding vehicle behind the pink vehicle (SV2), resulting in three surrounding vehicles on the target lane. A successful lane merge is defined as the ego vehicle reaching a position within 0.5 m from the center of the target lane, with a heading parallel to the target lane.

We choose to control the behavior of the surrounding vehicles by the P-IDM model introduced in Section 3-4-1. Under this model, the surrounding vehicles use a constant velocity assumption to predict the motion of the ego vehicle and make informed decisions based on the prediction. Table 4-2 displays the P-IDM parameters used in the simulation. The cooperation coefficient is sampled from a uniform distribution, denoted by $U(\cdot, \cdot)$. While the slow-moving trunk is programmed to behave in a polite manner, the other surrounding vehicles on the target lanes are programmed to behave in a selfish manner to simulate a traffic flow with higher speeds. The initial positions of the surrounding vehicles are randomly generated while maintaining a relative distance of 10 m to 15 m.

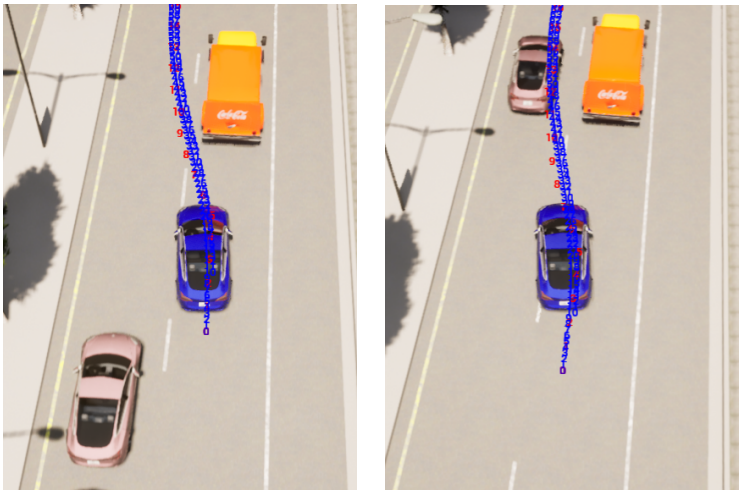
Table 4-2: IDM Parameters

IDM Parameter		Polite	Selfish
Desired velocity ($\mathrm{m\,s^{-1}}$)	v_0	5.0	10.0
Desired time gap (s)	T	1.5	1.0
Jam distance (m)	d_0	3.0	2.0
Max acceleration ($\mathrm{m\,s^{-2}}$)	a	2.0	2.0
Max deceleration ($\mathrm{m\,s^{-2}}$)	b	2.0	3.0
Velocity exponent (1)	δ	4.0	4.0
Cooperation coefficient (m)	c	$U(0, 3.5)$	$U(0, 3.5)$

We compare the proposed planner with two interaction-aware planners and one classical behavior planer that decouples planning and prediction. The two interaction-aware planners are the planner proposed in EPSILON, which selects a trajectory with the lowest cost, and a planner that selects a trajectory by seeking a Stackelberg equilibrium with the ego vehicle as the leader. The classical behavior planner is based on the spatial-temporal lattice planner adopted from the codebase of [25], which employs dynamic programming to determine the optimal reference trajectory from the spatial-temporal lattice. We utilized a straightforward yet practical constant velocity model for motion prediction of the surrounding vehicles, as motivated by [70]. We conduct experiments on the planners using two initial traffic speeds. For each initial condition, we run 200 simulations to ensure statistical significance. Screenshots during different simulation scenarios can be seen in Figure 4-10(a) and 4-10(b).

Comparison to Interaction-aware Baselines

In Table 4-3, we observe that all the planners performed well at a low speed. However, as the traffic speed increases, our proposed game-theoretic method achieves a higher success rate



(a) Merging when the surrounding vehicle (in pink) is polite.

(b) Merging when the surrounding vehicle (in pink) is selfish.

Figure 4-10: Screenshots of simulation in CARLA. The red numbers represent the output of the behavior planning layer, while the blue numbers stand for the trajectory generated by the motion planning layer. In scenario (a), where the surrounding vehicle (in pink) is polite, the ego vehicle can proceed to overtake. Conversely, in scenario (b), the surrounding vehicle (in pink) acts selfishly, compelling the ego vehicle to wait until the former has cleared the way.

than the baselines while requiring approximately the same merging time.

Comparison to the Classical Baseline

From the obtained results, it is evident that the proposed behavior planner demonstrates superior performance compared to the classical planner, particularly regarding success rate. However, it is noteworthy that the classical planner is still capable of accomplishing successful merges even in highly congested traffic scenarios. In fact, it may even require less time to achieve a successful merge due to its gradual convergence towards the target trajectory. Essentially, the classical planner selects a trajectory that closely aligns with the target lane, thus exhibiting an "unconscious" probing behavior that can induce surrounding vehicles to yield. Nevertheless, the classical planner often encounters difficulties in finding feasible solutions despite extensive tuning. This limited feasibility region makes it impractical for use in dense traffic conditions and real-world scenarios. These findings indicate that the classical approach, utilizing the spatial-temporal lattice, poses challenges in terms of fine-tuning and adaptability.

4-6 Discussion

While the proposed planner demonstrates the ability to capture interactions and generate informed motion plans, it does have certain limitations.

Table 4-3: Comparison of Planners in Closed-loop Simulations

Initial Speed	Metric	Nash	Stackelberg	EPSILON	Classical
Low ($\approx$ 5 m/s)	success rate	99 %	94 %	98 %	71 %
	collision rate	1 %	6 %	2 %	0 %
	time to merge (s)	8.75	8.17	8.28	5.12
	no solution rate	0 %	0 %	0 %	28 %
High ($\approx$ 10 m/s)	success rate	95 %	89 %	70 %	39 %
	collision rate	5 %	11 %	30 %	1 %
	time to merge (s)	6.92	6.90‘	6.78	3.98
	no solution rate	0 %	0 %	0 %	57 %

- **Computation time:** The planner's real-time performance is currently constrained due to the computational demands of multi-vehicle forward simulation. However, this limitation can potentially be mitigated by leveraging parallel computing acceleration, which has the potential to reduce the computational cost significantly. Additionally, future improvements to efficiency are anticipated through implementing the planner in C++.
- **Assumptions of the game theory:** The proposed game-theoretic planner assumes the rationality of surrounding vehicles by selecting a Nash equilibrium as the solution. However, real-world human drivers may exhibit different behaviors influenced by cognitive limitations, as highlighted in studies by [40], [57]. This motivates us to develop a motion planning approach that is robust to a wide range of potential behaviors exhibited by surrounding vehicles.
- **Limited scalability:** Despite strong interpretability, the proposed planner based on gap selection can only be applied to lane merging scenarios. Therefore, we intend to explore an interaction-aware but more general framework that can be applied to various traffic scenarios. A potential solution framework will be discussed in Chapter 5.

4-7 Summary

The highlights of this chapter are summarized as follows:

- We propose a novel game-theoretic behavior planner that integrates planning and prediction to generate informed motion plans.
- We validate the proposed planner through a benchmark evaluation in CARLA, demonstrating its superior performance compared to both the state-of-the-art interaction-aware behavior planners and a classical behavior planner.
- We discuss the limitations of the proposed planner and outline potential future research directions.

Chapter 5

Motion Planning: A Branch Model Predictive Control Approach

In this chapter, we unveil a novel branch model predictive control (B-MPC) motion planner that holds the promise of addressing the limitations presented by the behavior planner in Chapter 4. To start, we provide motivation for the proposed approach by highlighting the theoretical merits of the B-MPC approach (Section 5-1). Following this, we delve into the details of B-MPC in Section 5-2. The chapter culminates with a series of numerical simulations designed to assess the effectiveness of the proposed approach (Section 5-3).

5-1 Introduction

Despite the promising results, the game-theoretic behavior planning framework in Chapter 4 faces limitations when dealing with multi-modal behavior. The planner typically selects a single trajectory based on the Nash equilibrium assumption, assuming fully rational behavior from surrounding vehicles. However, in reality, the behavior of surrounding vehicles may exhibit noise and rationality, introducing ambiguity and multiple behavior modes [40], [71]. This inherent stochastic behavior poses challenges to accurate motion prediction. While efforts can be made to enhance prediction using game models, it is also intriguing to explore design approaches on the planning side, aiming to address these uncertainties and improve the overall effectiveness of the planning process.

One straightforward way to address this issue is through robust planning [72], which generates trajectories that avoid all the possible motions of the surroundings. This method can be overly conservative, which has been discussed in the literature review (see Sections 2-1-1 and 4-5-2). Another approach is to generate a trajectory tree (also called contingency planning), the idea of which is to create a branched motion plan that covers all the possible behaviors of the surrounding vehicles. The relevant literature has already been reviewed in Chapter 2. In this chapter, we will focus on the branch model predictive control (B-MPC) approach, which will be shown as a promising solution for the problem above.

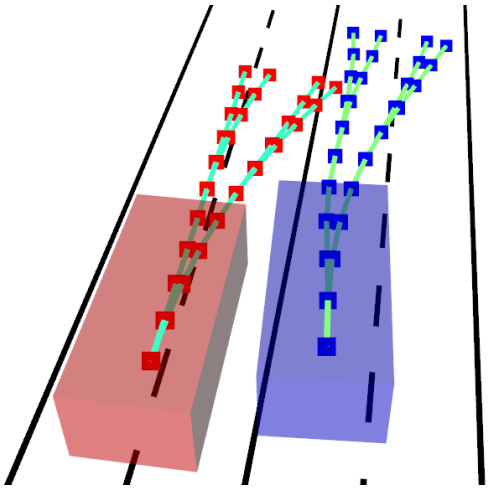


Figure 5-1: Illustration of trajectories using B-MPC in the lane merging scenario. The red box represents the surrounding vehicle (SV), while the blue box signifies the ego vehicle (EV). The red and blue points correspond to the trajectory trees of SV and EV, respectively.

5-2 Branch Model Predictive Control (B-MPC)

The Branch Model Predictive Control (B-MPC) approach is proposed in [47] as a way to generate a trajectory tree considering the multi-modal behavior of the surrounding vehicles. For simplicity, we assume that the ego vehicle (EV) interacts with a single surrounding vehicle (SV), as illustrated in Figure 5-1. This assumption is commonly adopted in the previous literature [3], [37]–[39]. In this section, our initial step involves generating what we refer to as the scenario tree, which comprises the motion predictions of the surrounding vehicle (SV). Following this, we proceed to formulate an optimal control problem (OCP) aimed at deriving an optimally structured trajectory tree for the ego vehicle (EV) that retains a similar topology as the scenario tree.

5-2-1 Motion Prediction as a Scenario Tree

To understand the behavior of a traffic participant, leveraging domain knowledge has proven to be an efficient approach [64]. In the context of lane merging, we assume that SV only makes longitudinal semantic-level actions characterized by a discrete set of choices:

$$\mathcal{D} = \{\texttt{LaneKeep}, \texttt{Yield}\}.$$

Each semantic-level action corresponds to a distinct behavior mode. This discretization is a reasonable approach considering the multi-modal nature of SV's future motion, as discussed in previous research studies [58], [59], [61], [62]. What is more, it's noteworthy that constraining SV's behavior to only longitudinal actions is a common assumption in the existing literature [3], [37]–[40].

To further predict the motion of SV in the trajectory level, we associate each semantic-level action with a pre-designed policy $\pi^i_{\mathrm{SV}}(\hat{\boldsymbol{x}}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}})$, where $i \in \{1, \cdots, M_{\mathrm{SV}}\}$ stands for the index of the policy. Here, we have $M_{\mathrm{SV}} = 2$ when given the semantic-level actions specified

above. The state $\boldsymbol{x} = [x, y, \theta, v]^\top$ is given by the kinematic bicycle model as introduced in Section 3-3. Here, x, y, θ, and v are the longitudinal position, lateral position, heading angle, and speed, respectively. Instead of purely depending on the current state of SV [47], in this thesis, we let the policy π^i_{SV} maps the states of EV and SV to an appropriate control input $\boldsymbol{u} = [a, \delta]^\top$. Here, a is acceleration, while δ stands for the steering angle. These policies can either be manually crafted [64] or acquired through data-driven learning [62]. In this research, we embrace a rule-based methodology for policy design. Further elaboration on this approach can be found in Section 5-3-4.

It is crucial to recognize that these policies of SV depend not only on the simulated state of SV but also on the states of EV. Consequently, the predicted motion of SV can be influenced by the motion plans of EV, possibly leading to forced merging maneuvers. Although the current motion plan of EV only affects the predicted motion of SV for the next time step, the negotiation between the two exists due to the small time step duration (e.g., 0.1 s). An experimental comparison can be found in Section 5-3-4.

Next, we present the structure of the scenario tree, which is illustrated in Figure 5-2. In this design, each node, represented as a circle, corresponds to a predicted state pair of the vehicles at time step t, denoted by $(\hat{\boldsymbol{x}}_{\mathrm{EV}}(t), \boldsymbol{x}_{\mathrm{SV}}(t))$. A branch, depicted as a dashed rectangle, encompasses multiple nodes generated by simulating SV's behavior using a policy introduced before. Therefore, a branch contains the states for a brief time interval. However, the root branch comprises only one node, capturing the current state of the vehicles.

The child branches are established by iteratively cycling through SV's policies at regular intervals of N time steps. In a trajectory tree with depth d, we represent a branch as b^k, where the index $k \in \mathbf{K} = \{0, \cdots, \frac{M_{\mathrm{SV}}(M_{\mathrm{SV}}^{d-1}-1)}{M_{\mathrm{SV}}-1}\}$. In the example depicted in Figure 5-2, the tree has a depth of $d = 3$ and consists of branches $b^0, \cdots, b^6$. We use the notation $\mathrm{anc}(k)$ to indicate the index of a branch's ancestor and $\mathrm{ch}(k)$ to signify the index of its children. For example, we have the index $\mathrm{anc}(1) = \{0\}$ and $\mathrm{ch}(1) = \{3, 4\}$. We further denote the state pairs in a branch as $\left(\hat{\boldsymbol{x}}_{\mathrm{EV}}(t_0^k), \boldsymbol{x}_{\mathrm{SV}}(t_0^k)\right), \ldots, \left(\hat{\boldsymbol{x}}_{\mathrm{EV}}(t_f^k), \boldsymbol{x}_{\mathrm{SV}}(t_f^k)\right)$, where t_0^k and t_f^k denotes the time step of the first and last node of branch k.

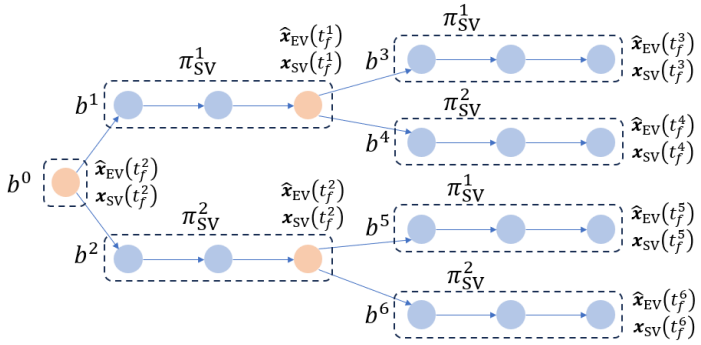


Figure 5-2: An example of the scenario tree. In this example, we assume SV has two semantic-level actions. The depth of the tree is denoted as $d = 3$, indicating the number of levels or stages in the planning process. The tree branches at $N = 3$ time steps. The branching nodes, depicted in orange, signify nodes with multiple predecessors, while the blue nodes represent nodes with only one predecessor. The nodes inside one dashed rectangle use the same policy to generate control inputs.

Here, we first predict the future states of EV, marked as $\hat{\boldsymbol{x}}_{\mathrm{EV}}(t_0^k), \ldots, \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_f^k)$, through forward simulation using control inputs shifted from the previous time step's B-MPC solution. Subsequently, the predicted states of SV, represented by $\boldsymbol{x}_{\mathrm{SV}}(t_0^k), \cdots, \boldsymbol{x}_{\mathrm{SV}}(t_f^k)$ are simulated based on $\boldsymbol{x}_{\mathrm{SV}}(t_f^{\mathrm{anc}(k)}), \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_0^k), \cdots, \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_f^k)$ using the corresponding policy π_{SV}^i of branch b^k.

To compute the probability of each behavior mode of SV, we can assess the cost $J_{\mathrm{SV}}^{k,i}$ for SV within the branch using the same rule-based approach as explained in Section 4-3-3. Here, i stands for the index of the policy used in branch k. Furthermore, we leverage the behavior model introduced in Section 3-4-2 to compute the probability of branch b^k as:

$$P(\pi_{\mathrm{SV}}^i \mid \boldsymbol{x}_{\mathrm{SV}}(t_f^{\mathrm{anc}(k)}), \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_0^k), \cdots, \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_f^k)) = \frac{\exp\left(-J_{\mathrm{SV}}^{k,i}\right)}{\sum_{i=1}^{M_{\mathrm{SV}}} \exp\left(-J_{\mathrm{SV}}^{k,i}\right)} \tag{5-1}$$

5-2-2 B-MPC Formulation

In this section, we will proceed with the computation of EV's trajectory tree. As depicted in Figure 5-3, the structure of EV's trajectory tree closely resembles that of the scenario tree. However, there's a significant distinction: a branching node (highlighted in orange) only has a single child. This one-step delay facilitates a contingency control input for EV, allowing it to effectively handle all possible motions of SV.

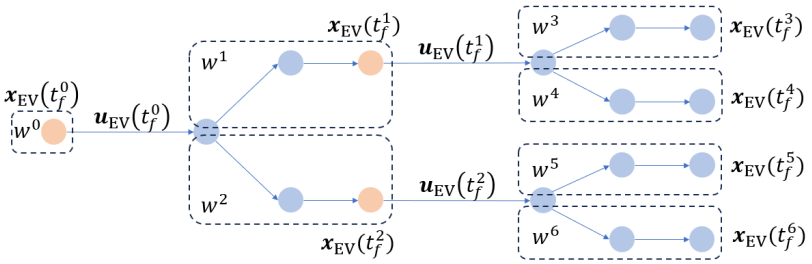


Figure 5-3: An example of the trajectory tree of EV. The orange nodes correspond to the branching nodes in the scenario tree.

To incorporate the probability of the multi-modal prediction of SV, we associate a weight w^k to each branch. While the root weight $w^0 = 1$, the subsequent weights are calculated by multiplying the conditional probabilities, i.e.,

$$w^k = w^{\mathrm{anc}(k)} P\left(\pi_{\mathrm{SV}}^i \mid \boldsymbol{x}_{\mathrm{SV}}(t_f^{\mathrm{anc}(k)}), \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_0^k), \cdots, \hat{\boldsymbol{x}}_{\mathrm{EV}}(t_f^k)\right)$$

We design a classical quadratic cost for each branch:

$$J_{\mathrm{EV}}^0 = (\boldsymbol{x}_{\mathrm{EV}}(t_0^0) - \boldsymbol{x}^{\mathrm{ref}}(t_0^0))^T Q(\boldsymbol{x}_{\mathrm{EV}}(t_0^0) - \boldsymbol{x}^{\mathrm{ref}}(t_0^0)) + \boldsymbol{u}_{\mathrm{EV}}(t_0^0)^T R \boldsymbol{u}_{\mathrm{EV}}(t_0^0)$$

$$J_{\mathrm{EV}}^k = \sum_{t=t_0^k}^{t_f^k} (\boldsymbol{x}_{\mathrm{EV}}(t) - \boldsymbol{x}^{\mathrm{ref}}(t))^T Q(\boldsymbol{x}_{\mathrm{EV}}(t) - \boldsymbol{x}^{\mathrm{ref}}(t)) + \boldsymbol{u}_{\mathrm{EV}}(t)^T R \boldsymbol{u}_{\mathrm{EV}}(t), \forall k \in \mathbf{K} \setminus \{0\}$$

where Q and R are the state cost matrix and the control cost matrix, respectively. $\boldsymbol{x}^{\mathrm{ref}}$ stands for the reference state. When a branch has no children, $\boldsymbol{u}_{\mathrm{EV}}(t_f^k)$ is no longer a decision variable

and the term $\boldsymbol{u}_{\mathrm{EV}}(t)^T R \boldsymbol{u}_{\mathrm{EV}}(t)$ is set to zero. The overall OCP for the B-MPC approach is formulated as follows:

$$
\begin{aligned}
\underset{\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{u}_{\mathrm{EV}}}{\text{minimize}} \quad & \sum_{k \in \mathbf{K}} w^k J_{\mathrm{EV}}^k && \text{(5-2a)}\\
\text{subject to} \quad & \boldsymbol{x}_{\mathrm{EV}}(t) = f\left(\boldsymbol{x}_{\mathrm{EV}}(t-1), \boldsymbol{u}_{\mathrm{EV}}(t-1)\right), \forall k \in \mathbf{K}, \forall t = t_0^k + 1, \ldots, t_f^k && \text{(5-2b)}\\
& \boldsymbol{x}_{\mathrm{EV}}(t_0^k) = f\left(\boldsymbol{x}_{\mathrm{EV}}(t_f^{\mathrm{anc}(k)}), \boldsymbol{u}_{\mathrm{EV}}(t_f^{\mathrm{anc}(k)})\right), \forall k \in \mathbf{K} \backslash \{0\} && \text{(5-2c)}\\
& w^0 = 1 && \text{(5-2d)}\\
& w^k = w^{\mathrm{anc}(k)} P\left(\pi_{\mathrm{SV}}^i \mid \boldsymbol{x}_{\mathrm{SV}}(t_f^{\mathrm{anc}(k)}), \boldsymbol{x}_{\mathrm{EV}}(t_0^k), \cdots, \boldsymbol{x}_{\mathrm{EV}}(t_f^k)\right) && \text{(5-2e)}\\
& \boldsymbol{u}_{\mathrm{EV}} \in \mathcal{U} && \text{(5-2f)}\\
& \boldsymbol{x}_{\mathrm{EV}} \in \mathcal{X} && \text{(5-2g)}\\
& \boldsymbol{x}_{\mathrm{EV}}(t_0^0) = \boldsymbol{x}_{\mathrm{EV}}^{\mathrm{init}} && \text{(5-2h)}\\
& \boldsymbol{x}_{\mathrm{SV}}(t_0^0) = \boldsymbol{x}_{\mathrm{SV}}^{\mathrm{init}} && \text{(5-2i)}
\end{aligned}
$$

where $(\boldsymbol{x}_{\mathrm{EV}}^{\mathrm{init}}, \boldsymbol{x}_{\mathrm{SV}}^{\mathrm{init}})$ stands for the current state pair of EV and SV. We use $\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{u}_{\mathrm{EV}}$ to represent all the states and control inputs within the trajectory tree of EV.

The optimization problem depicted by Equation 5-2 encompasses nonlinear dynamics constraints (Equations 5-2b and 5-2c) and potentially non-convex collision avoidance constraints (Equation 5-2g), making it a nonlinear programming problem (NLP). In this thesis, we perform a comparative analysis of various solving schemes for the NLP in Section 5-3-2, and we investigate diverse formulations of collision avoidance constraints in Section 5-3-3. What is more, we also add the bound constraints:

$$\boldsymbol{x}_{\mathrm{EV}}^{\min} \leq \boldsymbol{x}_{\mathrm{EV}}(t) \leq \boldsymbol{x}_{\mathrm{EV}}^{\max}, \boldsymbol{u}_{\mathrm{EV}}^{\min} \leq \boldsymbol{u}_{\mathrm{EV}}(t) \leq \boldsymbol{u}_{\mathrm{EV}}^{\max}, t \in \{t_0^k, \ldots, t_f^k\}, \forall k \in \mathbf{K}$$

as typically done in control applications. To further ensure motion comfort during driving, we consider the slew rate constraints on the change of control inputs:

$$\|\boldsymbol{u}_{\mathrm{EV}}(t) - \boldsymbol{u}_{\mathrm{EV}}(t-1)\| < \epsilon_u$$

where ϵ_u is a positive number.

By solving the aforementioned OCP problem, we can derive the states for the EV's trajectory tree. The B-MPC runs in a receding horizon manner. Only the control input of the root node is executed. As the root node is the ancestor of all other branches, the resulting control input can be viewed as a contingency plan that accounts for all potential future behaviors of the SV. Here we summarize the connection between the scenario tree and EV's trajectory tree:

- The scenario tree provides the motion prediction for SV. To create this prediction, we perform forward simulations using the policies. As the policies also depend on EV's future state, we predict EV's motion by shifting the B-MPC's solution from the previous time step.
- After constructing the scenario tree, we proceed to build a trajectory tree for EV with a similar topology. Specifically, the branch probabilities derived from the scenario tree

are used to adjust the corresponding weights in the trajectory tree. The motion prediction of SV helps in formulating collision avoidance constraints. Moreover, EV's motion prediction from the scenario tree serves as an initial guess for the trajectory tree optimization.

The complete B-MPC approach is summarized in Algorithm 2.

Algorithm 2 B-MPC Approach

procedure BMPC($\boldsymbol{x}_{\text{EV}}^{\text{ref}}, \boldsymbol{u}'_{\text{EV}}$) ▷ $\boldsymbol{x}^{\text{ref}}$: reference trajectory, $\boldsymbol{u}'_{\text{EV}}$: last inputs
  $w^0 \leftarrow 1$
  $(\boldsymbol{x}_{\text{EV}}(t_0^0), \boldsymbol{x}_{\text{SV}}(t_0^0)) \leftarrow (\boldsymbol{x}_{\text{EV}}^{\text{init}}, \boldsymbol{x}_{\text{SV}}^{\text{init}})$
  $\hat{\boldsymbol{x}}_{\text{EV}}(t_0^k), \cdots, \hat{\boldsymbol{x}}_{\text{EV}}(t_f^k), \forall k \in \mathbf{K} \leftarrow$ Simulate EV's motion using $\boldsymbol{u}'_{\text{EV}}$
  $\boldsymbol{x}_{\text{SV}}(t_0^k), \cdots, \boldsymbol{x}_{\text{SV}}(t_f^k), \forall k \in \mathbf{K} \leftarrow$ Generate SV's prediction
  $OCP \leftarrow$ Form the OCP as Equation 5-2 using $\boldsymbol{x}_{\text{EV}}^{\text{ref}}$ and the scenario tree
  $\boldsymbol{x}_{\text{EV}}, \boldsymbol{u}_{\text{EV}} \leftarrow$ Solve the OCP
  return $\boldsymbol{u}_{\text{EV}}(t_0^0)$
end procedure

5-2-3 Real-time Iteration Scheme

Equation 5-2 represents an NLP, which can be computationally expensive to solve in general. To mitigate this computational burden, we employ a real-time iteration (RTI) scheme [73] to solve the OCP. The RTI scheme leverages the fact that the solution of the OCP of the current time step is close to the solution of the previous time step. This approximation is reasonable since the time step is typically small when the algorithm runs in a receding horizon manner.

The RTI scheme can be summarized as follows: In the first phase, we linearize the OCP around the previous solution and formulate a Quadratic Programming (QP) problem. In the second phase, we solve the QP problem to obtain the solution of the OCP. The overall RTI scheme is outlined in Algorithm 3.

Algorithm 3 RTI Scheme for Solving B-MPC

Phase 1:
procedure PREPARATION($\boldsymbol{x}'_{\text{EV}}, \boldsymbol{u}'_{\text{EV}}$) ▷ Using previous solution as input
  $(\hat{\boldsymbol{x}}_{\text{EV}}, \hat{\boldsymbol{u}}_{\text{EV}}) \leftarrow$ Shift $(\boldsymbol{x}'_{\text{EV}}, \boldsymbol{u}'_{\text{EV}})$
  Linearize the dynamics and constraints at $(\hat{\boldsymbol{x}}_{\text{EV}}, \hat{\boldsymbol{u}}_{\text{EV}})$
  $QP \leftarrow$ Formulate the QP problem
  return QP
end procedure
Phase 2:
procedure FEEDBACK(QP)
  $(\delta\boldsymbol{x}_{\text{EV}}, \delta\boldsymbol{u}_{\text{EV}}) \leftarrow$ Solve QP by OSQP
  $(\boldsymbol{x}_{\text{EV}}, \boldsymbol{u}_{\text{EV}}) \leftarrow (\hat{\boldsymbol{x}}_{\text{EV}}, \hat{\boldsymbol{u}}_{\text{EV}}) + (\delta\boldsymbol{x}_{\text{EV}}, \delta\boldsymbol{u}_{\text{EV}})$ ▷ Apply the full Newton step
  return $(\boldsymbol{x}_{\text{EV}}, \boldsymbol{u}_{\text{EV}})$
end procedure

5-3 Numerical Simulations

5-3-1 Simulation Setup

In Chapter 4, we utilized the high-fidelity simulator CARLA to conduct our experiments. CARLA offers robust physical simulations, thereby necessitating the generation of low-level control inputs (throttle, brake, steering) for the ego vehicle. Consequently, we needed to develop a comprehensive planning and control stack, resulting in a complex system. While we successfully implemented this system in Chapter 4, we observed that it exhibited strong coupling, making it challenging to evaluate the performance of individual planners.

Therefore, we apply a simplified handcrafted simulator in this chapter. The simulator, implemented in Python, consists of two vehicles driving on a city road, as illustrated in Figure 5-1. The vehicles are simulated using the kinematic bicycle model introduced in Section 3-3. We represent the vehicles as rectangles and detect collision via the separating axis theorem. By ignoring the low-level control, we can evaluate the performance of the B-MPC approach more objectively. Additionally, we simulate the behavior of SV by the P-IDM model, which has been introduced in Section 3-4-1. The model parameters are the same as in Table 4-2. In each simulation, the driving style of SV is randomly selected. The initial speeds of both vehicles are set to be $10\,\mathrm{m\,s^{-1}}$. EV is initialized to be ahead of SV by a positive longitudinal distance, chosen uniformly from 1 m to 5 m. The reference state $\boldsymbol{x}_{\mathrm{EV}}^{\mathrm{ref}}$ is defined by adding 10 m to the SV's longitudinal state. In other words, EV is supposed to overtake SV.

In contrast to the simple unicycle model in [47], this thesis utilizes the more practical kinematic bicycle model and the B-MPC algorithm. The implementation of the algorithm is in C++ to ensure real-time performance. For algorithmic differentiation and solving optimization problems, we utilize the CasADi framework [74]. To facilitate message passing between different programs, we rely on ROS [69]. To ensure statistical significance, we conduct 200 runs for each experiment setting. For simplicity, we set the depth of the tree to one. The behavior mode `LaneKeep` is represented by a constant velocity policy, while `Yield` is a constant deceleration braking policy. The main experiment results are summarized in Table 5-1.

Table 5-1: Experiment results of B-MPC

Metric	IPOPT + Clearance	IPOPT + Circle	RTI + Clearance	RTI + Circle
success rate	41 %	59 %	51 %	97 %
collision rate	15 %	41 %	1 %	3 %
time to merge	4.72 s	1.78 s	3.88 s	2.10 s
computation time	0.23 s	0.47 s	0.02 s	0.02 s

5-3-2 Comparison of Solving Schemes

The results presented in Table 5-1 clearly demonstrate that the RTI scheme demands less computational time compared to the nonlinear IPOPT solver while still achieving superior performance. Nevertheless, it is worthwhile to mention that in our experiments, the RTI scheme can encounter difficulties finding a solution when the tree depth increases. This behavior may be attributed to its strong dependence on the initial guess.

It's noticeable that using the nonlinear solver results in a lower success rate and a higher collision rate, which is contrary to expectations. Typically, a nonlinear solver should provide a more precise solution to the OCP at the expense of increased computation time. This unexpected experimental outcome might be attributed to the predefined $\boldsymbol{x}_{\mathrm{EV}}^{\mathrm{ref}}$. In our experiment, we utilize a single reference state with a constantly larger longitudinal position than SV. This choice is consistent with the approach in [47]. However, this might not be the most reasonable selection since SV's continual movement renders the preset state invalid over time. This strategy performs reasonably in the RTI solving scheme due to linearization errors, but it could undermine performance when employing more accurate solving schemes like the nonlinear solver IPOPT. In the future, our plan is to address this issue by employing the output of the behavior planner (discussed in Chapter 4) as the reference states. This approach aims to provide a more accurate and contextually relevant reference for the trajectory optimization process.

5-3-3 Comparison of Collision Avoidance Constraints

To prevent collisions, the original method in [47] represents vehicles as rectangles and enforces clearance in both longitudinal and lateral directions. An illustration can be seen in Figure 5-4. The collision avoidance constraint is written as follows:

$$\frac{\Delta x e^{\kappa \Delta x} + \Delta y e^{\kappa \Delta y}}{e^{\kappa \Delta x} + e^{\kappa \Delta y}} \geq 1 \tag{5-3}$$

where κ is a positive constant, $\Delta x = \frac{|x_{\mathrm{EV}} - x_{\mathrm{SV}}|}{\Delta x^{max}}$ and $\Delta y = \frac{|y_{\mathrm{EV}} - y_{\mathrm{SV}}|}{\Delta y^{max}}$. Here, Δx^{max} and Δy^{max} represent the predefined maximum longitudinal and lateral clearances, respectively. To satisfy the constraint specified in Equation 5-3, it's necessary for either the longitudinal or the lateral clearance to be equal to or greater than the maximum clearance.

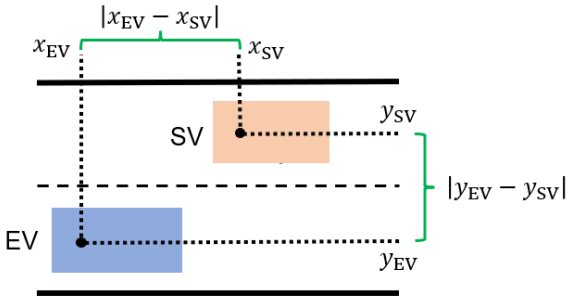


Figure 5-4: An illustration of the vehicle clearance. The blue and orange rectangles correspond to the ego vehicle and the interactive surrounding vehicle, respectively. The disparity between their longitudinal and lateral positions signifies the longitudinal and lateral clearance.

Equation 5-3 is computational efficiency and easy to implement. However, this approach using clearance tends to be conservative since it overlooks the orientation of the vehicles. In contrast, this thesis represents the EV as circles, as already demonstrated in Section 4-4-1. Instead of applying the spatial-temporal corridors, we also represent the SV as circles, leading to the collision avoidance constraints as:

$$\left\| c_{\mathrm{EV}}^{i} - c_{\mathrm{SV}}^{j} \right\|_2 - 2r \geq 0, \quad \forall i, j \in [n_{\mathrm{c}}] \tag{5-4}$$

where c_{EV} and c_{SV} denote the centers of EV and SV, respectively. The parameter n_c signifies the total number of circles used for each vehicle, while r stands for the radius of these circles, as explained in Section 4-4-1. Though the introduction of additional dimensions escalates computational demands, the constraints in Equation 5-4 account for the vehicle's orientation, leading to less overly conservative motion plans. In this chapter, We ignore the constraints of the road edges for simplicity.

In Table 5-1, we observe that the collision avoidance constraints proposed in this thesis result in a significantly higher success rate using both solving schemes. However, there is also a slight increase in the collision rate, indicating that the relaxation of collision avoidance constraints can lead to some risky situations.

5-3-4 Performance of Interactive Policy

In this thesis, we utilize the policy $\pi^i_{\mathrm{SV}}(\boldsymbol{x}_{\mathrm{EV}}, \boldsymbol{x}_{\mathrm{SV}})$ for SV. Unlike the original policy $\pi^i_{\mathrm{SV}}(\boldsymbol{x}_{\mathrm{SV}})$ described in [47], the proposed policy considers the states of both EV and SV. This approach allows the motion prediction of SV to be influenced by the motion plans of EV, resulting in less conservative motion plans. To validate the effectiveness of applying interactive prediction for SV, we utilize a P-IDM model in `LaneKeep` mode while using a constant velocity model as a baseline. To keep the evaluation simple, we limit the motion mode of SV to only `LaneKeep`. Furthermore, we employ the RTI solving scheme in conjunction with the newly proposed collision avoidance constraints, as they have demonstrated superior performance in our experiments.

The results are shown in Table 5-2. It is evident that the rate of successful lane merging is significantly higher when using interactive prediction, compared to employing the constant velocity model. Meanwhile, the time cost of using interactive prediction is smaller. These findings indicate that interactive prediction can leverage the nature of SV to react to EV's motions, resulting in the potential yield behavior of SV during lane merging.

Table 5-2: Performance of Interactive Prediction

Metric	Interactive Prediction	Constant Velocity Prediction
success rate	93 %	78 %
collision rate	7 %	2 %
time to merge (s)	1.97 s	2.72 s
computation time (s)	0.02 s	0.02 s

5-4 Discussion

In this chapter, we discuss the B-MPC approach, which is initially proposed in [47]. Nevertheless, the work in this thesis still holds some limitations:

- **Branching Strategy:** It is important to mention that there are different strategies for approximating a trajectory tree, as discussed in [44]. However, the authors also report

in [44] that not much improvement could be observed when using different strategies. Therefore, in this thesis, we utilize a straightforward approach of creating branches at regular intervals for simplicity.

- **Influence of Behavior Planning:** The motion planner proposed in this chapter still relies on a reference line and an initial guess, which can significantly influence the solution due to the nonlinear nature of the optimization problem. Consequently, a behavior planning module is still necessary before applying the B-MPC motion planner. As a part of future work, we aim to integrate the game theoretic behavior planner proposed in Chapter 4 with the motion planner in this chapter to create a comprehensive planning stack.
- **Multiple Surrounding Vehicles:** Although the B-MPC approach can perform well when dealing with a single vehicle, directly applying it to multiple vehicles becomes challenging due to the exponential growth of the tree size, rendering the problem intractable. However, a potential solution to simplify the problem is to assume that the ego vehicle interacts with the surrounding vehicles in pairwise [3]. Under this assumption, we can create branches without considering a second surrounding vehicle, making the problem computationally cheaper. The adaptation of B-MPC to multiple surrounding vehicles is regarded as another area for future work.

5-5 Summary

Our contribution to the B-MPC approach can be summarized as follows:

- We utilize the more complex kinematic bicycle model instead of the simple unicycle model to ensure dynamics feasibility.
- We predict the trajectory tree of SV via policies depending on the states of SV and EV, enabling the prediction of SV to be influenced by the motion plans of EV.
- We compare the RTI solving approach with solving by IPOPT, a general-purpose NLP solver. The RTI solver can achieve faster computation and better performance with parameter tuning, but it might encounter difficulties finding a solution when the horizon becomes longer.
- We propose new collision avoidance constraints that consider the orientation of the vehicles to make motion plans less conservative.
- We discuss the current limitations of the B-MPC approach and explore possible future directions.

Chapter 6

Conclusion

6-1 Summary

The highlights of this thesis can be summarized as follows:

- We present a novel approach combining game theory with interactive behavior planning. To ensure practicality and efficiency, we leverage semantic-level actions and model vehicle interactions as a gap selection process. Addressing the multi-modality challenge, we represent the surrounding vehicles' behavior as actions in a matrix game and then select the Nash equilibrium with the lowest social cost. Our simulation study in CARLA demonstrates the superior performance of our proposed method compared to the state-of-the-art approaches.
- We have developed an implementation of the Branch Model Predictive Control (B-MPC) planner that generates contingency motion plans by considering the multi-modal behavior of the surrounding vehicles. In addition to reproducing the original planner proposed in [47], we reproduced the algorithm in C++ and used a kinematic bicycle model instead of a unicycle model. Apart from different solving schemes, we also explored more commonly used collision avoidance constraints and incorporated an interactive policy while predicting the motion of the surrounding vehicles. The experiments on a handcrafted lightweight simulator show the findings of our exploration.

6-2 Future Work

In the future, our first goal is to integrate the behavior planner and motion planner into a comprehensive planning stack. To assess the effectiveness of this integrated system, we plan to use datasets collected from real human drivers. Additionally, to evaluate real-time performance, we intend to implement the algorithm stack on actual automated vehicles.

While classical model-based approaches are appealing due to their determinism and explainability, they may lack generalizability. For example, algorithms designed for straight roads may not directly apply to curved roads. Moreover, model-based approaches could become non-optimal when underlying modeling assumptions are violated. For instance, in previous chapters, we assumed that the ego vehicle only directly interacts with a single surrounding vehicle, which is commonly adopted but not necessarily valid in real-world scenarios. Hence, model-based approaches may inherently be limited as "expert systems".

In contrast, learning-based approaches are gradually showing promise despite not yet reaching the same level of performance as classical approaches. One potential direction involves using reinforcement learning (RL), allowing the ego vehicle to learn an optimal policy on its own. A policy learned through RL is expected to capture all the relevant decision-making factors. With the introduction of certificate functions, such as a control Lyapunov barrier function, the safety guarantee of an RL policy becomes less challenging [75]. Notably, the paper [75] is also a work by the thesis author during his master's study, and a full copy can be found in Appendix B. As another area of future work, the author intends to explore the usage of RL to generate policies for lane merging of automated vehicles.

Appendix A

ITSC Paper

This appendix presents the paper version of Chapter 4. The paper titled "An Efficient Game-Theoretic Planner for Automated Lane Merging with Multi-Modal Behavior Understanding" was accepted on July 13, 2023, by the IEEE Intelligent Transportation Systems Conference (ITSC).

An Efficient Game-Theoretic Planner for Automated Lane Merging with Multi-Modal Behavior Understanding

Luyao Zhang[1,†], Shaohang Han[2,†] and Sergio Grammatico[1]

***Abstract*— In this paper, we propose a novel behavior planner that combines game theory with search-based planning for automated lane merging. Specifically, inspired by human drivers, we model the interaction between vehicles as a gap selection process. To overcome the challenge of multi-modal behavior exhibited by the surrounding vehicles, we formulate the trajectory selection as a matrix game and compute an equilibrium. Next, we validate our proposed planner in the high-fidelity simulator CARLA and demonstrate its effectiveness in handling interactions in dense traffic scenarios.**

I. INTRODUCTION

Automated vehicles are facing a significant challenge when navigating in highly interactive environments, such as the lane-merging scenario shown in Figure 1. Traditional methods typically adopt a hierarchical structure where motion prediction and planning are decoupled. Consequently, these methods might be overly conservative since they often overlook the mutual interaction between the ego vehicle and the surrounding ones. Although newly developed learning-based approaches [1], [2] consider such interaction, they typically require large amounts of data and might lack interpretability. Another popular interaction-aware method is the Partially Observable Markov Decision Process (POMDP) [3], which provides a rigorous mathematical framework for handling incomplete information, such as the unknown intentions of the surrounding vehicles. However, solving a large-scale POMDP is computationally intractable.

As an approximation of the POMDP framework, the multiple policy decision-making (MPDM) [4] and its extension EPSILON [5] have demonstrated promising results in generating practically reasonable trajectories while remaining computationally efficient. The approach involves conducting multi-vehicle forward simulations based on semantic-level policies, followed by a trajectory evaluation step to select the best trajectory using handcrafted criteria. However, the rule-based trajectory evaluation in EPSILON might be overly aggressive or conservative when the surrounding vehicles have multiple behavior modes. Additionally, although the open-loop planning strategy is computationally efficient, it sacrifices the advantages of active information gathering in the original POMDP approach.

To systematically evaluate the trajectories, we use game theory, which is a powerful mathematical framework that captures the mutual influence between multiple agents. Previous research has explored equilibrium solutions extensively for automated lane merging. Some studies have focused on jointly planning trajectories for all vehicles by seeking a Nash equilibrium [6], [7]. However, these methods can only find a local equilibrium, and the quality of the solution might heavily depend on the initial guess. In contrast, other approaches use semantic-level actions as strategies. Among them, some studies propose a Stackelberg game with a leader-follower structure [8], [9]. However, determining the relative role of the leader or follower might be difficult [8]. In contrast to the leader-follower structure in a Stackelberg game, a Nash game treats all agents equally. A representative method based on a Nash game is proposed in [10], but it lacks validation in a high-fidelity simulator.

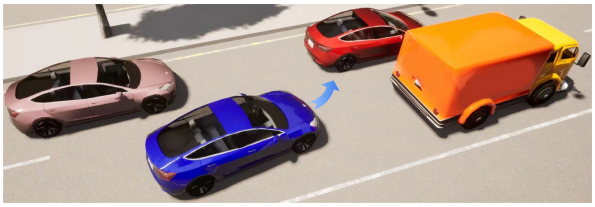


Fig. 1: Lane-merging scenario in the CARLA simulator. The ego vehicle (blue) is merging onto a lane in dense traffic.

Contribution: In this paper, we propose a novel approach that combines game theory with interactive trajectory generation. To make the algorithm practical and efficient, we leverage the semantic-level actions and model the interaction between vehicles as a gap selection process (Section IV-A). Additionally, to tackle the issue of multi-modality, we represent the behavior of the surrounding vehicles as actions in a matrix game, and then select the Nash equilibrium with the lowest social cost [11] (Section IV-C). We also investigate the existence of Nash equilibria and the relationship between Nash and Stackelberg equilibria through both theoretical analysis and numerous numerical simulations (Section V and VI-A). Moreover, we validate the effectiveness of the proposed planner in the high-fidelity CARLA simulator [12] (Section VI-B).

II. RELATED WORK

POMDP for lane merging. Prior research has explored POMDP for addressing the issue of unknown intentions in lane-merging scenarios. Online solvers, such as POMCPOW [13] and DESOPT [14], estimate the action-value function through sampling. Other POMDP approximations, such as QMDP [15] and heuristics [16], have been proposed as well.

*This work is partially supported by NWO under project AMADeuS.

† Equal contribution. Luyao Zhang and Sergio Grammatico are with the Delft Center for Systems and Control, TU Delft, The Netherlands. {l.zhang-7, s.grammatico}@tudelft.nl

[2]Shaohang Han is with the Department of Cognitive Robotics, TU Delft, The Netherlands. s.han-5@student.tudelft.nl

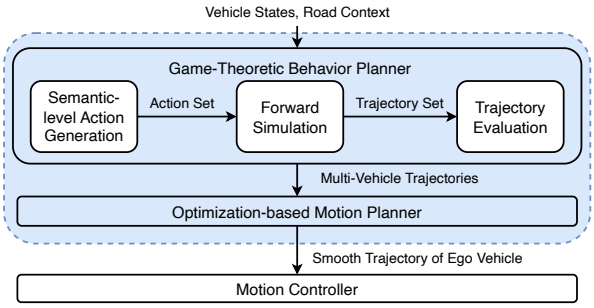


Fig. 2: Structure of the proposed behavior and motion planner. The focus of this paper is on the game-theoretic behavior planner.

Game-theoretic planning for lane merging. In methods that use Stackelberg games, vehicle strategies are represented by semantic-level actions such as motion primitives [8] or waiting time before merging [9]. While most of these methods assume the ego vehicle to be the leader, the rationale behind this is not always clear. Another approach utilizes level-k reasoning to model human drivers' behavior [17]. However, the computational burden of this framework is substantial due to the necessity of modeling the depth of human thinking [18].

III. PROBLEM FORMULATION

In this paper, we consider a mixed-traffic scenario where an automated ego vehicle interacts with the surrounding vehicles, as shown in Figure 1. Specifically, the ego vehicle aims at driving efficiently but a low-speed vehicle (SV0) travels in front of it. To avoid being blocked, the decision-making system of the ego vehicle needs to consider the diverse driving behaviors of the surrounding vehicles, select a suitable merging gap, and determine if/when to change lane. For example, in Figure 1, the ego vehicle can merge ahead of or after the pink vehicle (SV2). In fact, if SV2 yields, then the gap enlarges and the ego vehicle merges ahead of SV2. Otherwise, if the gap is not sufficiently wide, then the ego vehicle might slow down and then merge after SV2.

A. Structure of the Planner

We propose a game-theoretic planner as illustrated in Figure 2. Unlike other conventional behavior planners that require a motion predictor as an upstream module, in our approach, we combine motion prediction and behavior planning. Our proposed game-theoretic behavior planner consists of three modules: semantic-level action generation, forward simulation, and trajectory evaluation. First, we enumerate the possible semantic-level decision sequences of the traffic participants over the decision horizon. For the ego vehicle, a semantic-level decision can be making a lane change, accelerating or decelerating. Then, we form the action tuples by combining the decision sequences of the ego vehicle and surrounding vehicles. For each action tuple, the forward simulator generates the trajectories of the relevant vehicles. Subsequently, the trajectory evaluator determines the costs of the trajectories for each action tuple. Next, we construct a matrix game and seek an equilibrium. Finally, the trajectory evaluator selects the action tuple associated with the equilibrium, and outputs the multi-vehicle trajectories.

Although we apply the kinematic bicycle model to simulate the motion of the ego vehicle, the trajectory generated by the behavior planner might not be sufficiently smooth due to the coarse discretization step. Therefore, we employ an additional local motion planner [19] to produce a kinematically feasible trajectory. To ensure safety, the simulated trajectories of the surrounding vehicles are used to impose dynamic collision avoidance constraints.

B. Matrix Game

We model the decision-making process as a matrix game with two players. In the game, each player selects an action from its finite action set to optimize its individual cost. A two-player matrix game is defined by a tuple $(\mathcal{N}, \Pi, J)$, where $\mathcal{N}$ is the set of two players, $\Pi = \times_{i\in\mathcal{N}}\Pi_i$ is the joint action space, and $J = \times_{i\in\mathcal{N}} J_i$ is the joint cost function. Next, we introduce the three ingredients of the matrix game: players, actions and cost functions.

1) Players: We consider the ego vehicle (EV) and the group of the surrounding vehicles (SV) as two players, $\mathcal{N} := \{\mathrm{EV}, \mathrm{SV}\}$.

2) Actions: Human drivers typically make semantic-level decisions to make lane changes safely and efficiently. Inspired by human drivers, we represent the action of player $i \in \mathcal{N}$ by a semantic-level decision sequence, denoted as $\pi_i = \{\pi_{i,0}, \ldots, \pi_{i,k}, \ldots, \pi_{i,H-1}\}$, where H is the decision horizon. We provide more design details on the decision sets and the method for enumerating all possible decision sequences later in Section IV-A.

3) Cost functions: Before computing the costs, the forward simulator (Section IV-B) generates multi-vehicle trajectories. The cost function J_i of vehicle i evaluates the corresponding trajectory based on user-defined metrics, such as safety, efficiency, comfort and navigation. We consider the surrounding vehicles as a whole by calculating the total cost as $J_{\mathrm{SV}} := \sum_{i=2}^{N} J_i$, where N is the number of vehicles. Technical details are provided in Section IV-C.

A matrix game is depicted in Table I, where each entry represents a cost tuple $(J_{\mathrm{SV}}^{ij}, J_{\mathrm{EV}}^{ij})$ received by the group (SV) and the ego vehicle (EV) after performing their respective actions, π_{SV}^{i} and π_{EV}^{j}. Next, we look for an equilibrium

TABLE I: Game in normal form.

	π_{EV}^{1}	$\cdots$	$\pi_{\mathrm{EV}}^{M_{\mathrm{EV}}}$
π_{SV}^{1}	$(J_{\mathrm{SV}}^{1,1}, J_{\mathrm{EV}}^{1,1})$	$\cdots$	$(J_{\mathrm{SV}}^{1,M_{\mathrm{EV}}}, J_{\mathrm{EV}}^{1,M_{\mathrm{EV}}})$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\pi_{\mathrm{SV}}^{M_{\mathrm{SV}}}$	$(J_{\mathrm{SV}}^{M_{\mathrm{SV}},1}, J_{\mathrm{EV}}^{M_{\mathrm{SV}},1})$	$\cdots$	$(J_{\mathrm{SV}}^{M_{\mathrm{SV}},M_{\mathrm{EV}}}, J_{\mathrm{EV}}^{M_{\mathrm{SV}},M_{\mathrm{EV}}})$

of the matrix game. As previously mentioned in Section I, there are two recognized types of equilibria in the context of autonomous driving.

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*

$$J_i(\pi_i^*, \pi_{-i}^*) \leq \min_{s_i\in\Pi_i} J_i(s_i, \pi_{-i}^*),$$

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

The definition above indicates that no player can reduce its cost by unilaterally changing its strategy [20].

In a Stackelberg game, the idea is that the leader can take the action first, and then the follower plays the best response action [20]. The technical definition is provided as follows.

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 subscripts, L and F, represent the leader and the follower, respectively.

The leader and follower roles are not always fixed on the road. In other words, the ego vehicle can switch between being the leader and the follower. Thus, we consider two Stackelberg equilibria: one with the ego vehicle as the leader and the other with the ego vehicle as the follower.

IV. GAME-THEORETIC BEHAVIOR PLANNER

A. Semantic-Level Action Generation

1) Actions of Ego Vehicle: In the lane-merging problem, the semantic-level decision involves selecting a gap and determining the desired lateral position. As shown in Figure 3, the ego vehicle in blue has three potential gaps to choose from. To reach the target gap, the ego vehicle needs to perform a sequence of lateral decisions. The common lateral decisions are lane changing and lane keeping. Furthermore, we introduce one additional intermediate lane, represented by the dashed blue line in Figure 3, to enable a probing decision. This allows the ego vehicle to gather information and negotiate with the surrounding vehicles. 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 tuple $\pi_{\text{EV},k} := (g_k, d_k^{\text{lat}})$, 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 tuples. For example, if the ego vehicle chooses `Gap0`, then the only available lateral action is to keep the current lane. Next, we construct a decision tree to enumerate all the possible decision sequences. Each node in the tree represents a decision tuple. 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. By using semantic-level decisions that can be easily understood by humans, 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 of time. In addition, we can rule out certain transitions that would not be considered by normal human drivers, such as the transition from $(\texttt{Gap1}, \texttt{LeftChange})$ to $(\texttt{Gap2}, \texttt{LeftChange})$.

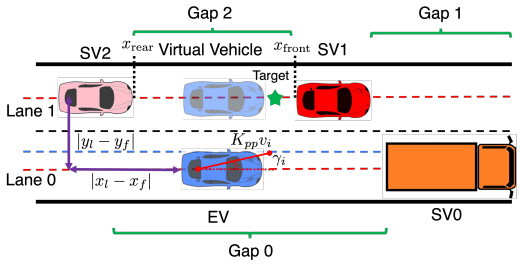


Fig. 3: Semantic-level decisions. Three gaps are available for the ego vehicle to choose from: `Gap0`, `Gap1` and `Gap2`. The dashed red lines represent the centerlines of the lanes, and the dashed blue line represents the probing line. EV stands for the ego vehicle, while SV0, SV1, and SV2 represent the surrounding vehicles. $|x_l - x_f|$ and $|y_l - y_f|$ represent the longitudinal and lateral distances between the lane-changing vehicle (leader) and the interactive vehicle (follower), respectively.

2) Actions of the Surrounding Vehicles: We make the following assumptions on the surrounding vehicles: (i) The surrounding vehicles maintain their lanes and have only longitudinal motion - a common assumption in prior work [8]–[10], [17], [21]; therefore, we define their longitudinal decision set as $\{\texttt{Assert}, \texttt{Yield}\}$. (ii) The surrounding vehicles maintain their decisions throughout each forward simulation. This assumption is reasonable in practice since the planner runs in a receding horizon fashion. (iii) The ego vehicle only directly interacts with at most one surrounding vehicle throughout each forward simulation. For instance, as shown in Figure 3, if the ego vehicle selects `Gap2`, its motion affects SV2, but not the vehicles ahead (SV0 and SV1). Therefore, we can treat all surrounding vehicles as a group, and the action set of the group is $\Pi_{\text{SV}} := \{\texttt{Assert}, \texttt{Yield}\}$.

B. Multi-vehicle Forward Simulation

1) Vehicle Dynamics: Next, we intend to generate the trajectories by simulating the motion of the vehicles from the initial states. We represent the dynamics of vehicle i as a kinematic bicycle:

$$\dot{x}_i = v_i \cos(\theta_i),\ \dot{y}_i = v_i \sin(\theta_i),\ \dot{\theta}_i = \frac{v_i}{l}\tan(\delta_i),\ \dot{v}_i = a_i,$$

where (x_i, y_i), θ_i and v_i are the position, the heading angle, and the speed, respectively; a_i and δ_i are the acceleration and the steering angle; l represents the inter-axle distance. The configuration vector is denoted as $q_i := [x_i, y_i, \theta_i]^\top$. Since we assume that the surrounding vehicles do not make lane changes, their heading and steering angles are equal to zero during the forward simulation ($\theta_i = 0$, $\delta_i = 0$). We discretize the dynamics via the Runge-Kutta 3 method.

2) Motion of the Ego Vehicle: We use two separate controllers to generate the longitudinal and lateral motion for the ego vehicle. For the longitudinal motion, we track the target longitudinal position and the desired speed via a PD

controller. One example of the target longitudinal position within the desired gap is illustrated in Figure 3. The desired gap, `Gap2`, is defined based on the positions of the front and rear vehicles, denoted as x_{front} and x_{rear}, respectively. Similar to [5], we determine the target longitudinal position and speed using a rule-based method.

As for the lateral motion, we adopt a pure pursuit controller that requires the current vehicle speed and the target line as inputs. The steering angle is computed by $\delta_i^{\text{ctrl}} = \tan^{-1}\left(\frac{2L\sin(\gamma_i)}{K_{pp}v_i}\right)$, where γ_i represents the angle between the heading direction and lookahead direction, K_{pp} is the feedback gain, and $K_{pp}v_i$ is the lookahead distance.

3) Motion of the Surrounding Vehicles: To model the behavior of the surrounding vehicles, we propose a modified intelligent driver model (IDM). Unlike the original IDM [22], which focuses solely on car following and disregards vehicles on adjacent lanes, our modified model considers lane-changing vehicles by projecting them onto their target lanes, resulting in virtual vehicles as shown in Figure 3 Subsequently, we calculate the distance between the virtual leader and the follower using the following approach:

$$d_{\text{idm}} = |x_l - x_f|e^{\kappa|y_l - y_f|},\ \kappa = 2\log(\beta)/w_{\text{lane}},$$

where w_{lane} is the lane width, and β is a parameter characterizing the level of willingness to yield. In fact, by adjusting the value of β, we can model different actions performed by the group of surrounding vehicles. 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 $\big(|y_l - y_f| = 0\big)$, the virtual distance between them is equivalent to the true distance.

C. Trajectory Evaluation

After generating multi-vehicle trajectories for each action tuple, we proceed to select a specific action tuple by solving a matrix game. For constructing the cost matrix, we first introduce the cost function J_i of vehicle i, which is typically a combination of several user-defined metrics, including safety, efficiency, comfort and navigation cost: $J_i = J_i^{\text{saf}} + J_i^{\text{eff}} + J_i^{\text{com}} + J_i^{\text{nav}}$. The value of the cost function J_i depends on the trajectories generated by the forward simulator, which are influenced by the semantic-level decision sequences of the ego vehicle (π_{EV}) and the surrounding vehicles (π_{SV}).

We calculate the safety cost by examining vehicle collisions. Here, the footprint of vehicle i is modeled as a rectangle $\mathcal{R}_i(q_i)$. If the distance between two rectangles is less than a small value $\underline{d}$, indicating a potential collision, we assign a very large penalty to the corresponding trajectory. With this in mind, we compute the safety cost as follows: $J_i^{\text{saf}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := \sum_{t=0}^{T}\sum_{j=1,\, j\neq i}^{N} P(q_i(t), q_j(t))$, where T is the planning horizon and N is the number of vehicles. We design P as follows:

$$P(q_i(t), q_j(t)) := \begin{cases} w_1^{\text{saf}} & \text{if } 0 \le d_{ij}(t) < \underline{d} \\ w_2^{\text{saf}} & \text{if } \underline{d} \le d_{ij}(t) \le \bar{d} \\ 0 & \text{else} \end{cases}$$

where $d_{ij}(t)$ represents the distance between $\mathcal{R}_i(q_i(t))$ and $\mathcal{R}_j(q_j(t))$, and w_1^{saf} is large than w_2^{saf}. By giving a relatively small penalty w_2^{saf} when $d_{ij}(t)$ falls within the range of $[\underline{d}, \bar{d}]$, we encourage the vehicle to keep a suitable distance from the surrounding vehicles. 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: $J_i^{\text{eff}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{eff}}\sum_{t=0}^{T}(v_i(t) - v_i^{\text{des}})^2$. For the comfort cost, we consider the change in acceleration, which is known as jerk. We use the finite difference to approximate the jerk, and subsequently define the comfort cost as follows: $J_i^{\text{com}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{com}}\sum_{t=1}^{T}(a_i(t) - a_i(t-1))^2/\Delta t^2$. Next, we penalize the differences between the vehicle lateral position and its desired lateral position to encourage the lane-changing maneuver: $J_i^{\text{nav}}(\pi_{\text{EV}}, \pi_{\text{SV}}) := w^{\text{nav}}\sum_{t=0}^{T}(y_i(t) - y_i^{\text{des}})^2$. Additionally, we introduce an information gain metric in the cost function, inspired by [23], to motivate the ego vehicle to actively identify the surrounding vehicles' intentions.

In practice, the ego vehicle needs to estimate the cost functions of other vehicles by observing their trajectories since it cannot directly access these costs. Similar to POMDP, we account for the uncertainty in the aggregate cost of the surrounding vehicles by integrating the beliefs into the cost function. The modified aggregate cost is computed as follows: $\bar{J}_{\text{SV}}^{ij} := (1 - b(\pi_{\text{SV}}^i))J_{\text{SV}}^{ij},\ \ \sum_{i=1}^{M_{\text{SV}}} b(\pi_{\text{SV}}^i) = 1$, where b represents the belief associated with the action of the surrounding vehicles. This design can be understood as incorporating prior knowledge about the behavior of the group of surrounding vehicles into the aggregate cost. For example, if we have prior knowledge suggesting that the group is likely to yield, then we can set the corresponding belief close to 1, which reduces the modified aggregate cost. We use a Bayesian estimation algorithm [24] to recursively estimate the beliefs at the beginning of each planning cycle.

After constructing the cost matrix, we compute a Nash equilibrium for the matrix game by enumerating all possible combinations of semantic-level actions. If multiple Nash equilibria exist, we select the equilibrium with the lowest social cost. If a Nash equilibrium does not exist, we choose the Stackelberg equilibrium with the ego vehicle as the follower as a backup solution.

V. ON NASH AND STACKELBERG EQUILIBRIA

In this section, we examine the conditions for the existence of a pure-strategy Nash equilibrium and explore the relationship between the Nash and Stackelberg equilibrium. We consider a specific cost matrix where $\pi_{\text{SV}}^1 :=$ `Assert` and $\pi_{\text{SV}}^2 :=$ `Yield` as mentioned in Section IV-A. We call an action tuple $(\pi_{\text{SV}}, \pi_{\text{EV}})$ feasible if the corresponding multi-vehicle trajectories are free from collisions, and we assume that there always exists at least one feasible action tuple.

Next, we show the existence of a Nash equilibrium for the matrix game. To obtain the results of this section, we assume that for the group of the surrounding vehicles (SV), behaving politely incurs a higher cost, while the ego vehicle (EV) achieves a lower cost if SV shows polite behavior. This assumption is required by Propositions 1 and 2.

Proposition 1. *Assume that the inequalities* $0 \leq J_{SV}^{1m} \leq J_{SV}^{2m}$ *and* $J_{EV}^{1m} \geq J_{EV}^{2m} \geq 0$ *hold for all feasible action tuples and* $m \in \{1, \ldots, M_{EV}\}$. *If* $b(\pi_{SV}^1) \geq 0.5$, *then there exists* $p \in \{1, \ldots, M_{EV}\}$ *such that the action tuple* (π_{SV}^1, π_{EV}^p) *is a pure-strategy Nash equilibrium.*

Proof. We select p such that $J_{\mathrm{EV}}^{1p} \leq J_{\mathrm{EV}}^{1m}$ for all $m \in \{1, , \ldots, , M_{\mathrm{EV}}\}$. Then, π_{EV}^p is the best response to π_{SV}^1. Using the assumptions in the proposition, we can conclude that $0 \leq J_{\mathrm{SV}}^{1p} \leq J_{\mathrm{SV}}^{2p}$. Furthermore, the inequality $(1 - b(\pi_{\mathrm{SV}}^1))J_{\mathrm{SV}}^{1p} \leq b(\pi_{\mathrm{SV}}^1)J_{\mathrm{SV}}^{2p}$ holds for $b(\pi_{\mathrm{SV}}^1) \geq 0.5$. Therefore, π_{SV}^1 is the best response to π_{EV}^p, and $(\pi_{\mathrm{SV}}^1, \pi_{\mathrm{EV}}^p)$ is a Nash equilibrium. □

In the following, we establish a connection between a Nash equilibrium and a Stackelberg equilibrium.

Proposition 2. *If the action tuple* (π_{SV}^2, π_{EV}^p) *is a Nash equilibrium, then it is also a Stackelberg equilibrium with EV as the leader.*

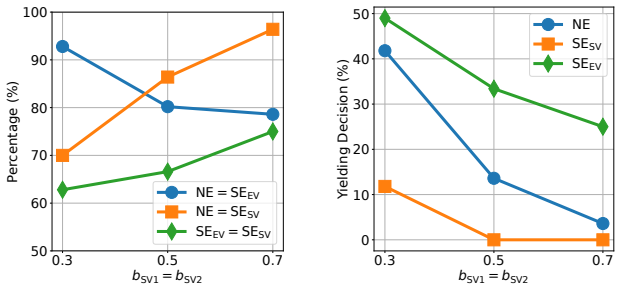
Proof. As $(\pi_{\mathrm{SV}}^2, \pi_{\mathrm{EV}}^p)$ is a Nash equilibrium, π_{SV}^2 is the best response to π_{EV}^p, and the inequality $J_{\mathrm{EV}}^{2p} \leq J_{\mathrm{EV}}^{2m}$ holds for all $m \in \{1, , \ldots, , M_{\mathrm{EV}}\}$. Based on the inequality $J_{\mathrm{EV}}^{1m} \geq J_{\mathrm{EV}}^{2m} \geq 0$, we can conclude that $J_{\mathrm{EV}}^{2p} \leq J_{\mathrm{EV}}^{2m} \leq J_{\mathrm{EV}}^{1m}$. Therefore, $(\pi_{\mathrm{SV}}^2, \pi_{\mathrm{EV}}^p)$ is a Stackelberg equilibrium with EV as the leader. □

VI. NUMERICAL SIMULATIONS

We consider a lane-merging scenario as introduced in Section III. The red (SV1) and pink (SV2) vehicles are considered potential interactive vehicles since the ego vehicle can influence their motion, while the orange vehicle (SV0) is a non-interactive dynamic obstacle. We set the initial beliefs on the interactive vehicles' decisions to $b_{\mathrm{SV1}}(\pi_{\mathrm{SV}}^1) = b_{\mathrm{SV2}}(\pi_{\mathrm{SV}}^1) = 0.5$. To simplify the notation, we use b_{SV1}^1 and b_{SV2}^1 to represent $b_{\mathrm{SV1}}(\pi_{\mathrm{SV}}^1)$ and $b_{\mathrm{SV2}}(\pi_{\mathrm{SV}}^1)$, respectively. We use a planning horizon of $T = 25$, a discretization step of $\Delta t = 0.2\,\mathrm{s}$, a decision time period of $\Delta h = 1\,\mathrm{s}$ and a decision horizon of $H = 5$.

A. Monte Carlo simulations

We conduct open-loop Monte Carlo simulations to empirically verify the existence of a Nash equilibrium and investigate its relationship with the Stackelberg equilibrium. We first specify collision-free initial multi-vehicle states. Then, the state of the ego vehicle is perturbed by $\pm 10\,\mathrm{m}$ for the initial position, $\pm 5\,\mathrm{m/s}$ for the initial longitudinal speed. We run 500 simulations for each belief and compute three equilibria: Nash (NE), Stackelberg with the ego vehicle as the leader ($\mathrm{SE_{EV}}$), and Stackelberg with the group of the surrounding vehicles as the leader ($\mathrm{SE_{SV}}$). The results are presented in Figure 4a. A Nash equilibrium is found in all simulations despite no theoretical guarantee in general. As mentioned before, if there are multiple Nash equilibria, we select the one with the lowest social cost. We observe that the selected Nash equilibrium coincides with one of the Stackelberg equilibria.



(a) Relationship between Nash and Stackelberg equilibria.

(b) Comparison of decisions in different equilibria.

Fig. 4: Results of Monte Carlo simulations. (a) illustrates the relationship between Nash and Stackelberg equilibria. (b) shows the percentage of yielding decisions made by SV in various equilibria over 500 simulations. The notations NE, $\mathrm{SE_{SV}}$ and $\mathrm{SE_{EV}}$ represent the Nash equilibrium, Stackelberg equilibrium with SV as the leader and Stackelberg equilibrium with EV as the leader, respectively.

Figure 4b illustrates the statistical results on SV's yielding decisions in different equilibria over 500 simulations. Specifically, SV is more likely to yield when both players employ a $\mathrm{SE_{EV}}$. In contrast, SV is less inclined to yield when both players select a $\mathrm{SE_{SV}}$. This implies that SV seems to behave cooperatively if the ego vehicle is the leader. On the contrary, if the ego vehicle is the follower, then it tends to show conservative behavior because the other vehicles are likely to assert. Overall, we conclude that adopting a Nash equilibrium exhibits less interactive behavior than adopting a $\mathrm{SE_{EV}}$, but it is less conservative compared to adopting a $\mathrm{SE_{SV}}$. In other words, it seems that a vehicle can switch between interactive and conservative behavior automatically by selecting a Nash equilibrium.

B. CARLA simulations

We conduct closed-loop simulations in the CARLA simulator to evaluate our proposed approach. The scenario depicted in Figure 3 requires the ego vehicle to merge onto the target lane as quickly as possible within a limited lane length of $100\,\mathrm{m}$. To simulate a real-world scenario, we add an additional surrounding vehicle behind the pink vehicle (SV2), resulting in three surrounding vehicles on the target lane. A successful lane merging means the ego vehicle stays within $0.5\,\mathrm{m}$ from the center of the target lane, with a heading parallel to the lane.

We choose to control the behavior of the surrounding vehicles by the Predictive Intelligent Driver Model [25]. In this model, surrounding vehicles anticipate the ego vehicle's motion using a constant velocity model and respond when the ego vehicle comes within a certain lateral distance, which is determined by the behavior mode. The slow-moving trunk follows polite behavior, while other surrounding vehicles on the target lane exhibit selfish behavior, simulating a traffic flow with higher speeds.

We compare the proposed planner with two other baseline planners: the planner proposed in EPSILON, which selects a trajectory with the lowest cost; and a planner that selects a trajectory by seeking a Stackelberg equilibrium with the ego

vehicle as the leader. We track the trajectories generated by each planner using a lower-level PID controller.

We conduct experiments on three different planners using two initial traffic speeds. For each initial condition, we run 200 simulations to ensure statistical significance. In Table II, we observe that all the planners performed well at a low speed. However, as the traffic speed increases, our proposed game-theoretic method achieves a higher success rate compared to the baselines while requiring approximately the same merging time.

TABLE II: Evaluation in closed-loop simulation

Initial Speed	Metric	Nash	Stackelberg	EPSILON
Low ($\approx 5\,\mathrm{m/s}$)	success rate	99%	94%	98%
	collision rate	1%	6%	2%
	time to merge (s)	8.75	8.17	8.28
High ($\approx 10\,\mathrm{m/s}$)	success rate	95%	89%	70%
	collision rate	5%	11%	30%
	time to merge (s)	6.92	6.90	6.78

VII. CONCLUSION

This paper focuses on developing a game-theoretic planning algorithm for automated vehicles to perform lane changes in interactive environments. Our simulation study in CARLA demonstrates the superior performance of our proposed method compared to the state-of-the-art approaches. By comparing different equilibria via numerical simulations, we observe that selecting a Nash equilibrium allows for automatically switching between interactive and conservative driving behavior. Although forward simulations are time-consuming, they can be easily performed in parallel to reduce the computation time. In future work, we plan to implement our method in C++ and validate it on a hardware platform.

REFERENCES

[1] W. Zeng, W. Luo, S. Suo, *et al.*, "End-To-End Interpretable Neural Motion Planner," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 8652–8661.

[2] D. Chen and P. Krahenbuhl, "Learning from All Vehicles," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 17 201–17 210.

[3] M. Lauri, D. Hsu, and J. Pajarinen, "Partially Observable Markov Decision Processes in Robotics: A Survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, Feb. 2023.

[4] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA: IEEE, May 2015, pp. 1670–1677.

[5] W. Ding, L. Zhang, J. Chen, and S. Shen, "EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1118–1138, Apr. 2022.

[6] S. L. Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A Fast Solver for Constrained Dynamic Games," in *Robotics: Science and Systems XVI*, Jul. 2020. arXiv: 1910.09713 [cs].

[7] X. Liu, L. Peters, and J. Alonso-Mora, *Learning to Play Trajectory Games Against Opponents with Unknown Objectives*, Dec. 2022. arXiv: 2211.13779 [cs].

[8] 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 Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 474–488, Jan. 2023.

[9] C. Wei, Y. He, H. Tian, and Y. Lv, "Game Theoretic Merging Behavior Control for Autonomous Vehicle at Highway On-Ramp," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 127–21 136, Nov. 2022.

[10] 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 Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3609–3620, Apr. 2022.

[11] 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 Robotics and Automation Letters*, vol. 6, no. 3, pp. 4978–4985, Jul. 2021.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, Oct. 2017, pp. 1–16.

[13] Z. Sunberg and M. J. Kochenderfer, "Improving Automated Driving Through POMDP Planning With Human Internal States," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 073–20 083, Nov. 2022.

[14] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 558–573, Feb. 2021.

[15] M. Naghshvar, A. K. Sadek, and A. J. Wiggers, "Risk-averse Behavior Planning for Autonomous Driving under Uncertainty," Dec. 2018.

[16] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 1617–1624.

[17] 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 *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 4509–4515.

[18] K. Ji, N. Li, M. Orsag, and K. Han, "Hierarchical and game-theoretic decision-making for connected and automated vehicles in overtaking scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 150, p. 104 109, May 2023.

[19] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 729–15 741, Sep. 2022.

[20] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics, Jan. 1998.

[21] 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 Transactions on Intelligent Vehicles*, vol. 5, no. 1, pp. 75–89, Mar. 2020.

[22] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000, arXiv:cond-mat/0002177, ISSN: 1063-651X, 1095-3787.

[23] 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," *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, Oct. 2018.

[24] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, Mar. 2002.

[25] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 808–18 821, Oct. 2022.

Appendix B

ICRA Paper

This appendix presents the submitted version of a scientific paper focusing on RL for robotics planning. As mentioned in Section 6-2, RL methods with safety and reachability guarantees hold promise for autonomous driving and will be listed as future work for this thesis. The paper, titled "Reinforcement Learning for Safe Robot Control using Control Lyapunov Barrier Functions", has been published at the IEEE International Conference on Robotics and Automation (ICRA) 2023[1]. This research was conducted as the author's research assignment during the master's study. The inclusion of this paper in the thesis aims to offer a comprehensive overview of the author's master's work.

In this research, we adapted the soft actor-critic algorithm [76] to create a learning method for a control Lyapunov barrier critic and an optimal control policy. This combination ensures both safety and reachability of a system. In the context of automated lane merging, the objective is to smoothly change lanes without colliding with other vehicles or road edges. We might apply the proposed RL framework in the lane merging scenario in the future.

[1] https://www.icra2023.org/

Reinforcement Learning for Safe Robot Control using Control Lyapunov Barrier Functions

Desong Du*[1,2], Shaohang Han*[2], Naiming Qi[1], Haitham Bou Ammar[3,4], Jun Wang[4] and Wei Pan[5,2]

***Abstract*— Reinforcement learning (RL) exhibits impressive performance when managing complicated control tasks for robots. However, its wide application to physical robots is limited by the absence of strong safety guarantees. To overcome this challenge, this paper explores the control Lyapunov barrier function (CLBF) to analyze the safety and reachability solely based on data without explicitly employing a dynamic model. We also proposed the Lyapunov barrier actor-critic (LBAC), a model-free RL algorithm, to search for a controller that satisfies the data-based approximation of the safety and reachability conditions. The proposed approach is demonstrated through simulation and real-world robot control experiments, i.e., a 2D quadrotor navigation task. The experimental findings reveal this approach's effectiveness in reachability and safety, surpassing other model-free RL methods.**

I. Introduction

Reinforcement learning (RL) has achieved impressive and promising results in robotics, such as manipulation [1], unmanned vehicle navigation [2], drone flight [3], [4], etc., thanks to its ability of handling intricate models and adapting to diverse problem scenarios with ease. Meanwhile, a safe control policy is imperative for a robot in the real world, as dangerous behaviors can cause irreparable damage or costly losses. Therefore, the RL methods that can provide a safety guarantee for robot control have received considerable interest and progress [5], [6], [7], [8], [9], [10].

A recent line of work focuses on designing novel RL algorithms, e.g., actor-critic, for constrained Markov Decision Process (CMDP). In these methods, the system encourages the satisfaction of the constraints by adding a constant penalty to the objective function [6] or constructing safety critics while doing policy optimization in a multi-objective manner [5], [7], [11], [12]. Although these approaches are attractive for their generality and simplicity, they either need model [6], or only encourage the safety constraints to be satisfied probabilistically.

An alternative type of methods focuses on reachability and safety guarantee (sufficient conditions) by constructing/learning control Lyapunov functions (CLF) and control barrier functions (CBF) that can respectively certify the reachability and safety [8], [10], [13], [14], [15], [16], [17], [18]. The relevant safe controllers are normally designed by adding a safety filter to a reference controller, such as a RL controller [8], [10], [13], a model predictive control (MPC) controller [14], etc. Unfortunately, these approaches have two disadvantages: (1) there might be conflicts between CLFs and CBFs as separate certificates [19], [20] (see Figure 2 in Section V-A); (2) the CLFs and CBFs are generally non-trivial to find [19], especially for nonlinear systems. Even though there are learning methods to find CLFs and CBFs, knowledge of dynamic models has to be explicitly used [21].

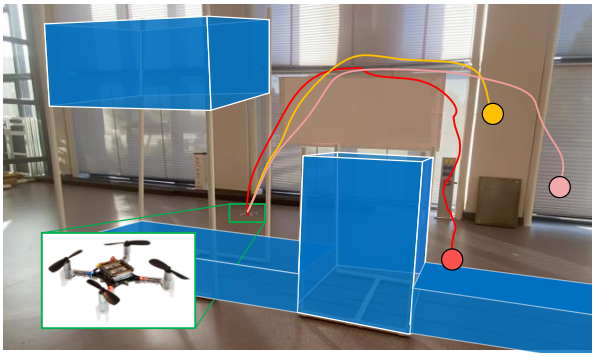


Fig. 1. The 2D quadrotor navigation task. Lines stand for trajectories. The circles are the initial position. The blue regions represent obstacles. Video is available at `https://youtu.be/_8Yr_QRRYik`.

In this paper, we propose a data-based reachability and safety theorem without explicitly using the knowledge of a dynamic system model. The contribution of this paper can be summarized as follows: (1) we used samples to approximate the critic as a control Lyapunov barrier function (CLBF), a single unified certificate, which is parameterized by deep neural networks, so as to guarantee both reachability and safety. The corresponding actor is a controller that satisfies both the reachability and safety guarantees. (2) we deploy the learned controller to a real-world robot, i.e., a Crazyflie 2.0 quadrotor, for a 2D quadrotor navigation task. The 2D quadrotor navigation task is shown as in Figure 1. The experiments show our approach has better performance than other model-free RL methods. Our approach, by using CLBFs, can avoid conflicts between the CLFs and CBFs certificates. Compared to the model-based approaches that learn CLBFs using supervised learning [19] or handcraft CLBFs [22], our method does not need the knowledge of models explicitly.

II. Related Works

Prior work has studied safety in RL in several ways, including imposing constraints on expected return [5], [7], risk measures such as Conditional Value at Risk and percentile estimates [12], [23], [24], and avoiding regions where

*indicates equal contribution

The work is supported by Huawei and China Scholarship Council No.202006120130.

[1]School of Astronautics, Harbin Institute of Technology, China. [2]Department of Cognitive Robotics, Delft University of Technology, Netherlands. [3]Huawei Technologies, United Kingdom. [4]Department of Computer Science, University College London, United Kingdom. [5]Department of Computer Science, University of Manchester, United Kingdom.

constraints are violated [25], [26], [27]. This paper focuses on the reach-avoid problem that belongs to the last situation.

To solve the reach-avoid problem, a popular strategy involves modifying the policy optimization procedure of standard RL algorithms to reason about task rewards and constraints simultaneously. One method is constrained policy optimization (CPO), which adds a constraint-related cost to the policy objective [5]. Another type of method tries to optimize a Lagrangian relaxation [7], [11], [23], [27], [28]. They normally use a safety critic to ensure safety, but this separate critic can only evaluate risk in a probabilistic way. Other methods involve constructing Lyapunov functions for the unsafe region [29], [30]. However, these approaches require a baseline policy that already satisfies the constraints.

III. Preliminaries and Background

In RL for safe control, the dynamical system is typically characterized by CMDP $\hat{M} = (\mathcal{S}, \mathcal{A}, P, c, \gamma, \mathcal{I})$ [31]. $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$ is the state vector at time t, S denotes the state space. The agent then takes an action $a_t \in \mathcal{A} \subseteq \mathbb{R}^m$ according to a stochastic policy/controller $\pi\left(a_t \mid s_t\right)$. The transition of the state is dominated by the transition probability density function $P\left(s_{t+1} \mid s_t, a_t\right)$, which denotes the probability density of the next state s_{t+1}. A cost function $c(s_t, a_t)$ is used to measure the immediate performance of a state-action pair (s_t, a_t), and $\mathcal{I}(s_t)$ indicates whether the state violates the safety constraints or not. The goal is to find π^* that can minimize the objective function return the expected return $J(\pi) \triangleq \sum_{t=1}^{\infty} \mathbb{E}_{s_t, a_t} \gamma^t c\left(s_t, a_t\right)$ with the discount factor $\gamma \in [0, 1)$, and $\forall t \in \mathbb{Z}_+, \mathcal{I}(s_t) = 0$. Moreover, some notations are to be defined. The closed-loop state distribution at a certain instant t as $p(s \mid \rho, \pi, t)$, which can be defined iteratively: $p\left(s' \mid \rho, \pi, t+1\right) = \int_{\mathcal{S}} P(s'|s, \pi(s)) p(s \mid \rho, \pi, t) \mathrm{d}s, \forall t \in \mathbb{Z}_+$ and $p(s \mid \rho, \pi, 0) = \rho(s)$.

In this paper, we focus on the reach-avoid problems, in which the agent reaches the goal condition and avoids certain unsafe conditions. It is defined as follows:

Definition 1. *(Reach-Avoid Problem). In a CMDP setting with a goal configuration s_{goal} and a set of unsafe states $\mathcal{S}_{unsafe} \subseteq \mathcal{S}$, find a controller $\pi^*(a|s)$ such that all trajectories s_t under $P\left(s_{t+1} \mid s_t, a_t\right)$, and $s_0 \in \mathcal{S}_{initial} \subseteq \mathcal{S}$ have the following properties:* ***Reachability****: given a tolerance δ, $\exists T \geq 0$, such that $\mathbb{E}_{s_t}\left\|s_t - s_{goal}\right\| \leq \delta$, $\forall t \geq t_0 + T$;* ***Safety****: $\mathbb{P}(s_t \notin \mathcal{S}_{unsafe} \mid s_0, \pi, t) = \int_{\mathcal{S} \backslash \mathcal{S}_{unsafe}} p(s \mid s_0, \pi, t) \mathrm{d}s = 0$, $\forall t \geq t_0$.*

The state $s_{\text{irrecoverable}} \in \mathcal{S}_{\text{irrecoverable}} \not\subset \mathcal{S}_{\text{unsafe}}$ are not themselves unsafe, but inevitably lead to unsafe states under the controller π. Thus, we also consider $s_{\text{irrecoverable}}$ to be unsafe for the given controller π.

Definition 2. *A state is said to be* ***irrecoverable*** *if $s \notin \mathcal{S}_{unsafe}$ under the controller $a \sim \pi(a|s)$, the trajectory defined by $s_0 = s$ and $s_{t+1} \sim P(s_{t+1}|s_t, \pi(s_t))$ satisfies $\mathbb{P}(s_t \in \mathcal{S}_{unsafe} \mid s_0, \pi, t) = \int_{\mathcal{S}_{unsafe}} p(s \mid s_0, \pi, t) \mathrm{d}s \neq 0$, $\exists \hat{t} > t_0$.*

Therefore, the safety and unsafety of a certain state can be described as: the state $s \in \mathcal{S}_{\overline{\text{unsafe}}} = \mathcal{S}_{\text{irrecoverable}} \cup \mathcal{S}_{\text{unsafe}}$ is unsafe, while the state $s \in \mathcal{S}_{\overline{\text{safe}}} = \mathcal{S} \backslash \mathcal{S}_{\overline{\text{unsafe}}}$ is safe.

In reach-avoid problems, CLFs and CBFs are widely used to ensure reachability and safety of the system [21], respectively. To avoid the conflicts between separate certificates, we rely on the CLBF, a single unifying certificate for both reachability and safety [22]. In this paper, the definition of the CLBF is related to [19]. We extend it from a continuous-time system to CMDP (similar to the definition of CBF in discrete-time system [32]). In CMDP, the definition of CLBF is given as follows.

Definition 3. *(CLBF). A function $V: \mathcal{S} \to \mathbb{R}$ is a CLBF, for some constant $\hat{c}$, $\lambda > 0$, ① $V\left(s_{goal}\right) = 0$, ② $V(s) > 0, \forall s \in \mathcal{S} \backslash \mathcal{S}_{goal}$, ③ $V(s) \geq \hat{c}, \forall s \in \mathcal{S}_{\overline{unsafe}}$, ④ $V(s) < \hat{c}, \forall s \in \mathcal{S}_{\overline{safe}}$. ⑤ there exists a controller π, such that $\mathbb{E}_{s'}[V\left(s'\right) - V(s) + \lambda V(s)] \leq 0, \forall s \in \mathcal{S} \backslash s_{goal}$, where $s' \sim P(s'|s, \pi(s))$.*

Thus, any controller $\pi \in \{\pi \mid \mathbb{E}_{s'}[V(s') - V(s) + \lambda V(s)] \leq 0, s' \sim P(s'|s, \pi(s))\}$ can satisfy reachability and safety [19]. In this definition, the transition $P(s'|s, \pi(s))$ requires the knowledge of a dynamic system model, but modeling error can hardly be avoided in reality. Next, we will show how we can use model-free RL to learn CLBFs and controllers with reachability and safety guarantee.

IV. Reinforcement Learning Algorithm with Safety Guarantee

In an actor-critc framework, the high-level plan is as follows. We first choose the value function $V(s)$ to be the CLBF, similar to those done in approximate/adaptive dynamic programming [33] on choosing the Lyapunov function. Then we expect to impose some properties of CLBF as constraints in the Bellman recursion to find the value function (i.e., CLBF) and hope to search the corresponding policy, similar to what is done in [25], [29], [34]. Conceptually, we are interested in the following conceptual problem formulation:

Repeat

- Find: V. Subject to: CLBF constraints
- Find: π using V

Untill V, π convergence.

A. CLBF as Critic

To enable the actor-critic learning, the control Lyapunov barrier critic Q_{LB} is designed to be dependent on s and a, while $V(s) = Q_{\text{LB}}(s, \pi_\theta(s))$. Then we present a method to construct a Q_{LB} through the Bellman recursion. The target function Q_{target} is a valid control Lyapunov barrier critic which is approximated by:

$$Q_{\text{target}}\left(s_t, a_t\right) = c(s_t, a_t) + \gamma Q'_{\text{LB}}\left(s_{t+1}, \pi(s_{t+1})\right) \quad (1)$$

where Q'_{LB} is the network that has the same structure as Q_{LB}, but parameterized by a different set ϕ', as typically used in the actor-critic methods [35], [36]. The parameter ϕ' is updated through exponential moving average of weights controlled by a hyperparameter $\tau \in \mathbb{R}_{(0,1)}, \phi'_{k+1} \leftarrow \tau\phi_k + (1-\tau)\phi'_k$.

Such that the value function meets the requirements of our main theorem (Theorem 1 in Section IV-B), the tuples

$\{s_t, a_t, c(s_t, a_t), s_{t+1}\}$ are set as follows:

$$\begin{cases} \{s_t, a_t, 0, s_t\} & s_t \in \mathcal{S}_{\text{goal}} \\ \{s_t, a_t, c(s_t, a_t), s_{t+1}\} & s_t \in \mathcal{S}_{\text{safe}} \setminus \mathcal{S}_{\text{goal}} \\ \{s_t, a_t, C, s_t\} & s_t \in \mathcal{S}_{\text{unsafe}} \end{cases} \tag{2}$$

where the terminal cost C is a constant.

B. Data-based CLBF Theorem

In this part, inspired by Definition 3 of CLBF, we propose a novel data-based theorem, on which the constraints should be in the conceptual problem formulation at the beginning of Section IV. Instead of explicitly using a dynamic model, the following theorem provides a sufficient condition for reachability and safety based on samples.

Before presenting the main theorem, we need the following Lemma 1, in addition to (2), on the terminal cost C to hold, so that $V(s_{\overline{\text{unsafe}}}) \geq \hat{c}$ and $V(s_{\overline{\text{safe}}}) < \hat{c}$, as required in (3).

Lemma 1. *Suppose that N is the maximum number of steps in each episode, let $C > \frac{c_{\max}(s,a)\left(1-\gamma^N\right)}{\gamma^N}$, when $\gamma < 1$. Under the controller π, if $s \in \mathcal{S}_{\overline{unsafe}}$, $V(s) \geq \hat{c}$, and $s \in \mathcal{S}_{\overline{safe}}$, $V(s) < \hat{c}$.*

Proof. The proof can be found in Appendix I. □

Theorem 1. *If there exists a function $V(s) : \mathcal{S} \to \mathbb{R}_+$ and positive constants α_1, α_2, α_3, α_4, such that*

$$\begin{aligned} \alpha_1 c_\pi(s) \leq & V(s) < \min\left(\alpha_2 c_\pi(s), \hat{c}\right) < \hat{c}, && \forall s \in \mathcal{S}_{\overline{safe}} \\ \hat{c} \leq & V(s) \leq \hat{c} + \alpha_3 c_\pi(s) < (1+\alpha_3)\hat{c}, && \forall s \in \mathcal{S}_{\overline{unsafe}} \end{aligned} \tag{3}$$

and

$$\begin{aligned} &\mathbb{E}_{s\sim\mu_N}\left(\mathbb{E}_{s'\sim P_\pi} V(s')\mathbb{1}_\Delta(s') - V(s)\mathbb{1}_\Delta(s)\right) \\ &< -\alpha_4 \mathbb{E}_{s\sim\mu_N} c_\pi(s)\mathbb{1}_\Delta(s) \end{aligned} \tag{4}$$

where $c_\pi(s_t) \triangleq \mathbb{E}_{a\sim\pi} c(s_t, a_t)$, and $c_\pi(s) \leq \hat{c}, \forall s \in \mathcal{S}$. The cost function $c(s_t, a_t) = \mathbb{E}_{P(\cdot|s_t,a_t)} \|s_{t+1} - s_{goal}\|$ describes the distance to the goal set. $\mu_N(s)$ denotes the average distribution of s over the finite N time steps,

$$\mu_N(s) \doteq \frac{1}{N}\sum_{t=1}^{N} p(s|\rho, \pi, t)$$

N is the maximum number of steps in each episode. $\mathbb{1}_\Delta(s)$ denotes the function;

$$\mathbb{1}_\Delta(s) = \begin{cases} 1 & s \in \Delta \\ 0 & s \notin \Delta \end{cases}$$

where $\Delta = \mathcal{S} \setminus (\mathcal{S}_{goal} \cup \mathcal{S}_{unsafe})$, $\mathcal{S}_{goal} = \{s \mid c_\pi(s) \leq \delta\} = \{s \mid \|s - s_{goal}\| \leq \delta\}$. Note that $c_\pi(s) > \delta$, $\forall s \in \Delta$.

Then the followings hold: i) if $s_0 \in \mathcal{S}_{\overline{safe}}, V(s_0) \leq \hat{c}$, the system is reachable with tolerance δ and safe within N steps; ii) if $s_0 \in \mathcal{S}_{\overline{unsafe}}, V(s_0) > \hat{c}$, the agent would reach the unsafe areas within N steps.

Proof. The proof can be found in Appendix II. □

C. Lyapunov Barrier Actor-Critic Algorithm

Recent advance in [34] has guaranteed reachability by the Lagrangian relaxation method. Taking inspiration from their work, we extend to safety guarantee by designing an actor-critic RL algorithm. The proposed Algorithm 1 is named Lyapunov barrier actor-critic (LBAC), which gains a value function that satisfies the requirements of Theorem 1, and a corresponding safe controller.

The control Lyapunov barrier critic function Q_{LB} and the actor function (controller) $\pi_\theta(a_t|s_t)$ are parametrized by ϕ and θ, respectively. Note that the stochastic controller π_θ is parameterized by a deep neural network f_θ that depends on s and Gaussian noise ϵ. The goal is to construct the CLBF as the critic function with constraints (4) under the controller $\pi_\theta(a_t|s_t)$. By using the Lagrange relaxation technique [37], Q_{LB} is updated using gradient descent to minimize the following objective function

$$\begin{aligned} J(\phi) = \mathbb{E}_{\mathcal{D}}\Big[&\frac{1}{2}\left(Q_{\text{LB}}(s,a) - Q_{\text{target}}(s,a)\right)^2 \\ &+\lambda(Q_{\text{LB}}(s', f_\theta(\epsilon, s'))\mathbb{1}_\Delta(s') - Q_{\text{LB}}(s,a)\mathbb{1}_\Delta(s) + \alpha_4\hat{c})\Big] \end{aligned} \tag{5}$$

where Q_{target} is the approximation target related to the chosen control Lyapunov barrier candidate, λ is a Lagrange multiplier that controls the relative importance of the inequality condition (4). $\mathcal{D}$ is the set of collected transition pairs that are determined in (2) and Lemma 1. The control Lyapunov barrier candidate acts as a supervision signal to the control Lyapunov barrier critic function.

LBAC is based on the maximum entropy framework [36], which can improve controller exploration during learning. A minimum entropy constraint is added to the above optimization problem to derive the following objective function

$$\begin{aligned} J(\theta) = &\mathbb{E}_{(s,a,s',c)\sim\mathcal{D}}[Q_{LB}(s, f_\theta(\epsilon, s)) \\ &+ \beta(\log(\pi_\theta(f_\theta(\epsilon, s)|s)) + \mathcal{H}_t))] \end{aligned} \tag{6}$$

where β is a Lagrange multiplier that controls the relative importance of the minimum entropy constraint, $\mathcal{H}_t$ is the desired entropy bound.

In the actor-critic framework, the parameters of the controller are updated through stochastic gradient descent, which is approximated by

$$\begin{aligned} \nabla_\theta J(\theta) = &\beta\nabla_\theta \log(\pi_\theta(a|s)) + \beta\nabla_a \log(\pi_\theta(a|s))\nabla_\theta f_\theta(\epsilon, s) \\ &+ \nabla_{a'} Q_{\text{LB}}(s', a')\nabla_\theta f_\theta(\epsilon, s') \end{aligned} \tag{7}$$

Finally, the values of Lagrange multipliers λ and β are adjusted by gradient ascent to maximize the following objectives, respectively,

$$\begin{aligned} J(\lambda) = &\lambda\mathbb{E}_{\mathcal{D}_\Delta}\left[Q_{\text{LB}}\left(s', f_\theta\left(s', \epsilon\right)\right)\mathbb{1}_\Delta\left(s'\right)\right. \\ &\left. - \left(Q_{\text{LB}}(s,a) - \alpha_4\hat{c}\right)\mathbb{1}_\Delta(s)\right], \\ J(\beta) = &\beta\mathbb{E}_{\mathcal{D}}\left[\log\pi_\theta(a \mid s) + \mathcal{H}_t\right] \end{aligned} \tag{8}$$

During training, the Lagrange multipliers are updated by

$$\lambda \leftarrow \max(0, \lambda + \bar{\delta}\nabla_\lambda J(\lambda)), \qquad \beta \leftarrow \max(0, \beta + \bar{\delta}\nabla_\beta J(\beta))$$

where $\bar{\delta}$ is the learning rate. The pseudocode of the proposed algorithm is shown in Algorithm 1.

Algorithm 1 Lyapunov Barrier Actor-Critic (LBAC)

Require: Maximum episode length N; maximum iteration steps M
  repeat
    Sample s_0 according to ρ
    for $t = 0$ to N **do**
      Sample a_t from $\pi_\theta(a_t|s_t)$ and step forward
      Observe s_{t+1}, c_t and store $(s_t, a_t, c_t, s_{t+1}, \mathcal{I})$ in $\mathcal{D}$
    end for
    for $i = 1$ to M **do**
      Sample mini-batches of transitions from $\mathcal{D}$ and update Q_{LB}, π, Lagrange multipliers with (5), (6), (8)
    end for
  until (4) is satisfied

V. Results and Validation

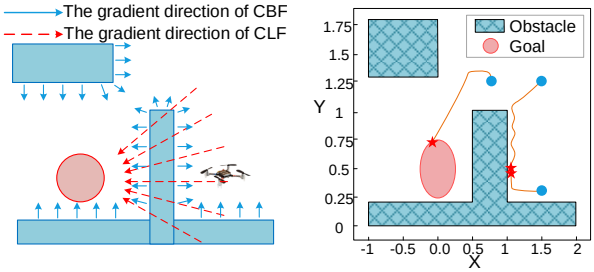
In this section, we consider a 2D quadrotor navigation task, i.e., aiming to reach a target while avoiding obstacles, as illustrated in Figure 1. The experiment setup is detailed in Appendix III. First, we show separate CLFs and CBFs can lead to local optimums by implementing a CLF-CBF based Quadratic Program (CLF-CBF-QP). Then, we show the effectiveness of the proposed LBAC algorithm and evaluate it in the following aspects:

- Training convergence: does the proposed training algorithm converge with random parameter initialization;
- Validation of CLBF: how do the learned CLBFs fit the goal and obstacles in the 2D quadrotor navigation task, and does the reachability and safety condition, i.e., Theorem 1, hold for the learned controllers;
- Sim-to-Real transfer: can we transfer the simulation training result directly to real-world robots, e.g., using a CrazyFlie 2.0 quadrotor.

In this part, the performance of LBAC on the CMDP tasks is evaluated compared with Risk Sensitive Policy Optimization (RSPO) [23], Safety Q-Functions for RL (SQRL) [11], and Reward Constrained Policy Optimization (RCPO) [7]. We use the public codebase of [27] to implement the comparison experiments. The hyperparameters are described in Appendix IV.

A. Conflicts between CLFs and CBFs

To show there exist conflicts between CLFs and CBFs as separate certificates, we implemented a model-based CLF-CBF-QP controller [38] which incorporates a CLF and CBFs as constraints through quadratic programs. As shown in Figure 2(b), the quadrotor easily gets stuck before the wall which is in front of the target. This is because the attraction of the CLF is balanced by the repulsion of CBFs, as illustrated by Figure 2(a). The quadrotor can still successfully reach the target if it luckily avoids conflicting areas. We also tried CLFs and CBFs as separate critics in a multi-objective RL setting, but failed to converge. The failure of the above CLF-CBF controllers motivates our CLBF approach which satisfies both safety and reachability in this 2D quadrotor navigation task, as illustrated in Figure 7(a).

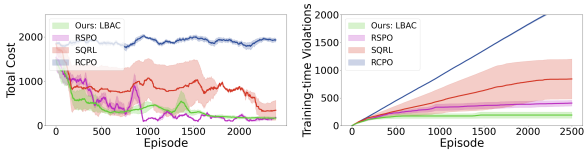


(a) Illustration of CLF-CBF (b) Results of CLF-CBF-QP

Fig. 2. Performance of a CLF-CBF-QP controller. (a) is an intuitive illustration of CLF-CBF. In (b), lines are trajectories. The blue circles stand for the starting points. The red stars represent the final position.

B. Training Convergence

The main criterion we are interested in is the convergence of the controller during the training process. Each approach is trained with five different random seeds. The total cost and number of violations during training are plotted in Figure 3. Among the RL algorithms to be compared, LBAC, RSPO, and SQRL can converge within 2300 episodes, while RCPO fails to converge even in 3000 episodes. As shown in Figure 3, LBAC leads to a fewer number of violations during training than other model-free safe RL methods.



(a) Total Cost (b) Training-time violations

Fig. 3. Total cost and the number of violations during training. The Y-axis indicates the total cost in one episode in (a) and total violation times during training in (b). The X-axis indicates the total episodes. The shaded region shows the 1-SD confidence interval of five random seeds.

C. Validation of CLBF

In this part, we examine the learned control Lyapunov barrier critic function. We pick the controllers and corresponding CLBFs trained in 1000, 1500, and 2000 episodes. The contour plots of the CLBFs are shown in Figure 4 as a function of x and y, where $\{v_x, v_y\}$ is set to $\{0, 0\}$. The white lines are the safety boundaries of the CLBFs, i.e. when $V(s) = \hat{c}$ and $\hat{c}$ is set 2000. As shown in Figure 4, we find that the safety boundary of CLBF where $V(s) = \hat{c}$ gradually approaches the obstacle boundary with increasing training episodes. However, we also noticed some unsafe corner cases are considered as safe (such as the bottom right corner of the left obstacle). This could be due to the exploration and exploitation dilemma LBAC suffers as a model-free RL algorithm.

We also validate the learned CLBF by showing the outcomes of the trajectory rollouts starting from uniformly sampled initial positions. This is because of the well-known fact that it is challenging to initialize uniformly throughout the state space in a model-free setting. For example, we can hardly make a robot have a specific velocity at a particular position. Figure 5 shows that the quadrotors starting from the

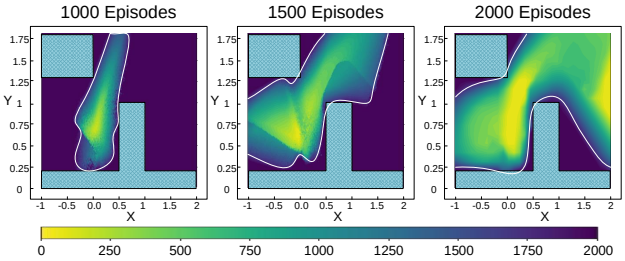


Fig. 4. The contour plots of the CLBF. The white lines show the contour of the learned CLBF. The color bar denotes the function value. From left to right, the contour plots are the CLBFs trained in 1000 episodes, 1500 episodes and 2000 episodes.

unsafe region would be violating, while those that start in the safe region would successfully reach the goal. We present the changes in CLBF values along the trajectories in Figure 6(a), and the averaged changes in CLBF value of these trajectories in Figure 6(b). We can observe that the averaged value has a decreasing trend, which aligns with the theory before. These results indicate that the learned CLBF is valid.

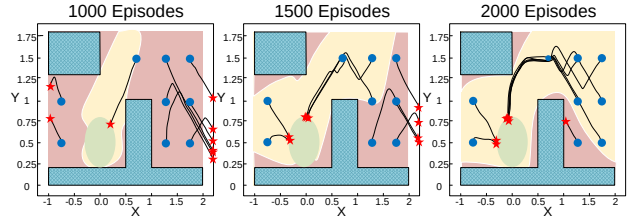
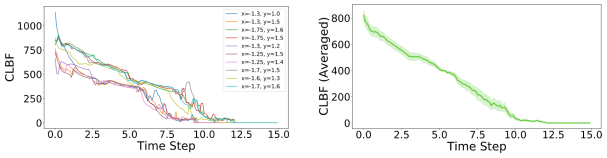


Fig. 5. Trajectories of the learned LBAC controllers in the simulator. The shaded area corresponds to the unsafe region. The green ellipse area stands for the goal. The blue circles are the initial positions, while the red stars are the end positions.



(a) The changes in CLBF value (b) The averaged changes

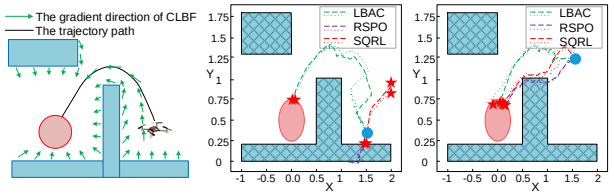
Fig. 6. The changes in CLBF value under different initial conditions. In (a), we show the changes in CLBF value along the trajectories starting from ten different initial positions. In (b), the averaged change in CLBF value of these trails is plotted. The solid line indicates the average value and shadowed region for the 1-SD confidence interval of these trails.

D. Sim-to-Real Transfer

In this part, we evaluate LBAC by deploying controllers learned in the simulators to the physical robot. As shown in Figure 1, a nano Crazyflie 2.0 quadrotor is used to achieve the autonomous navigation task and a motion capture system is used for state estimation in the real world. The trajectories of the Crazyflie starting from different initial positions are shown in Figures 7(b) and 7(c). The controllers trained by LBAC outperform other model-free safe RL algorithms in terms of both reachability and safety.

VI. Conclusion

In this paper, the control Lyapunov barrier function is extended to the constrained Markov decision process, and a data-based theorem is proposed to analyze closed-loop



(a) Illustration of CLBF (b) Height 0.35m (c) Height 1.2m

Fig. 7. Controllers are evaluated in real-world using a Crazyflie 2.0 quadrotor. (a) is an intuitive illustration of CLBF. In (b) and (c), the quadrotor's initial heights are 0.35m and 1.2m. The blue circle represents the starting points, and the red stars stand for the reached positions.

reachability and safety. Based on the theoretical results, a Lyapunov Barrier-based Actor-Critic method is proposed to search for a controller. The proposed algorithm is evaluated on a 2D quadrotor navigation task with safety constraints. Compared to existing model-free RL algorithms, the proposed method can reliably ensure reachability and safety in both simulation and real-world tests. In the future, more experiments will be conducted to validate the effectiveness and scalability of our approach. We also plan to improve the robustness of the learned controller using methods such as domain randomization and adversarial training [39].

Appendix I
Proof of Lemma 1

Proof. When $\hat{s} \in \mathcal{S}_{\overline{\text{safe}}}$, it leads to the goal state within N steps. Thus, $V(\hat{s}) = \mathop{\mathrm{E}}_{a\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t) \mid s_0=\hat{s}] < \sum_{t=0}^{N-1}\gamma^t c_{\max}(s,a) = \frac{c_{\max}(s,a)\left(1-\gamma^N\right)}{1-\gamma}$. In order to have $V(\hat{s}) < \hat{c}$, we set $\frac{c_{\max}(s,a)\left(1-\gamma^N\right)}{1-\gamma} < \hat{c}$. When $\hat{s} \in \mathcal{S}_{\text{unsafe}}$, it leads to unsafe state within N steps. Thus, $V(\hat{s}) \geq \sum_{t=0}^{N-1}\gamma^t c_{\min}(s,a) + \sum_{t=N}^{\infty}\gamma^t C = \frac{c_{\min}(s,a)\left(1-\gamma^N\right)+C\gamma^N}{1-\gamma}$. In order to have $V(\hat{s}) \geq \hat{c}$, we set $\frac{c_{\min}(s,a)\left(1-\gamma^N\right)+C\gamma^N}{1-\gamma} \geq \hat{c}$. Rearranging, we have $C \geq \frac{(1-\gamma)\hat{c}-c_{\min}(s,a)\left(1-\gamma^N\right)}{\gamma^N}$. With $c_{\min}(s,a)=0$, it is simplified to $C \geq \frac{1-\gamma}{\gamma^N}\hat{c} > \frac{c_{\max}(s,a)\left(1-\gamma^N\right)}{\gamma^N}$. To this end, the condition (3) is achieved. □

Appendix II
Proof of Theorem 1

Proof. To prove that N is finite based on the conditions and assumptions where $N = \max\{t : \mathbb{P}(s \in \Delta|\rho,\pi,t) > 0\}$, we will assume that N is infinity and prove by contradiction. $N = \infty$ if for any ϵ there exists an instant $t > \epsilon$ such that $\mathbb{P}(s \in \Delta|\rho,\pi,t) > 0$. In that case, the finite-horizon sampling distribution $\mu_N(s)$ turns into the infinite-horizon sampling distribution $\mu(s) = \lim_{N\to\infty}\mu_N(s) = \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}p(s|\rho,\pi,t)$. The existence of $\mu(s)$ is guaranteed by the existence of $q_\pi(s) = \lim_{t\to\infty}p(s \mid \rho,\pi,t)$, which has been commonly exploited by many RL literature [34], [40]. Since the sequence $\{p(s|\rho,\pi,t), t \in \mathbb{Z}_+\}$ converges to $q_\pi(s)$ as t approaches ∞, then by the Abelian theorem, the sequence $\{\frac{1}{T}\sum_{t=1}^{T}p(s|\rho,\pi,t), T \in \mathbb{Z}_+\}$ also converges and $\mu(s) = q_\pi(s)$. Then one naturally has that

the sequence $\{\mu_N(s)V(s), T \in \mathbb{Z}_+\}$ converges pointwise to $q_\pi(s)V(s)$.

According to Lebesgue's dominated convergence theorem [41], if a sequence $f_n(s)$ converges point-wise to a function f and is dominated by some integrable function g in the sense that,$|f_n(s)| \leq g(s), \forall s \in \mathcal{S}, \forall n$, then one has $\lim_{n\to\infty} \int_{\mathcal{S}} f_n(s)\mathrm{d}s = \int_{\mathcal{S}} \lim_{n\to\infty} f_n(s)\mathrm{d}s$.

Applying this theorem to the left-hand side of (4)

$$
\begin{aligned}
&\mathbb{E}_{s\sim\mu}\left(\mathbb{E}_{s'\sim p_\pi} V(s')\mathbb{1}_\Delta(s') - V(s)\mathbb{1}_\Delta(s)\right) \\
&= \int_{\mathcal{S}} \lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} p(s|\rho,\pi,t)\Big(\int_{\mathcal{S}} p_\pi(s'|s)V(s')\mathbb{1}_\Delta(s')\mathrm{d}s' \\
&\quad - V(s)\mathbb{1}_\Delta(s)\Big)\mathrm{d}s \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} \int_{\mathcal{S}} V(s')\mathbb{1}_\Delta(s') \int_{\mathcal{S}} p_\pi(s'|s)p(s|\rho,\pi,t)\mathrm{d}s\mathrm{d}s' \\
&- \lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} \int_{\mathcal{S}} p(s|\rho,\pi,t)V(s)\mathbb{1}_\Delta(s)\mathrm{d}s \\
&= \lim_{N\to\infty} \frac{1}{N}\Big(\sum_{t=2}^{N+1} \mathbb{E}_{p(s|\rho,\pi,t)} V(s)\mathbb{1}_\Delta(s) \\
&\quad - \sum_{t=1}^{N} \mathbb{E}_{p(s|\rho,\pi,t)} V(s)\mathbb{1}_\Delta(s)\Big) \\
&= \lim_{N\to\infty} \frac{1}{N}\left(\mathbb{E}_{p(s|\rho,\pi,N+1)} V(s)\mathbb{1}_\Delta(s) - \mathbb{E}_{\rho(s)} V(s)\mathbb{1}_\Delta(s)\right)
\end{aligned} \tag{9}
$$

Since $\mathbb{E}_{\rho(s)}V(s)$ is finite, thus the limitation value $\lim_{N\to\infty} \frac{1}{N}(\mathbb{E}_{\rho(s)}V(s)\mathbb{1}_\Delta(s)) = 0$. The above equation equals to $\lim_{N\to\infty} \frac{1}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}V(s)\mathbb{1}_\Delta(s)$. Note that $V(s) \geq \alpha_1 c_\pi(s)$, $\forall s \in \mathcal{S}$, and $c_\pi(s) > \delta$, $\forall s \in \Delta$. Thus, $\lim_{N\to\infty} \frac{1}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}V(s)\mathbb{1}_\Delta(s) \geq \lim_{N\to\infty} \frac{\alpha_1\delta}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}\mathbb{1}_\Delta(s) = 0$

Since $\mu(s) = q_\pi(s)$, the right-hand side of (4) equals to $-\alpha_4\mathbb{E}_{s\sim q_\pi} c_\pi(s)\mathbb{1}_\Delta(s) \leq -\alpha_4\mathbb{E}_{s\sim q_\pi}\delta\mathbb{1}_\Delta(s) = -\alpha_4\delta\lim_{t\to\infty}\mathbb{P}(s \in \Delta|\rho,\pi,t)$. Combining the above inequalities with (4), one has $\lim_{t\to\infty}\mathbb{P}(s \in \Delta|\rho,\pi,t) < 0$, which is contradictory to the fact that $\mathbb{P}(s \in \Delta|\rho,\pi,t)$ is nonnegative. Thus there exist a finite N such that $\mathbb{P}(s \in \Delta|\rho,\pi,t) = 0$ for all $t > N$. In other word, the agent will reach the goal region or the unsafe region within N steps. According to (3), $s_0 \in \mathcal{S}_{\overline{\text{safe}}}, V(s_0) < \hat{c}$ where the agent will reach the goal region and avoid the unsafe region, while $s_0 \in \mathcal{S}_{\overline{\text{unsafe}}}, V(s_0) \geq \hat{c}$ where the agent will reach the unsafe region within N steps. The process of building such function V is described in Section IV-A. □

APPENDIX III
2D QUADROTOR NAVIGATION

The state of the 2D quadrotor model is defined as $s = [p_x, p_y, v_x, v_y]$, with control input $a = [v_{x_{\text{des}}}, v_{y_{\text{des}}}]$. In this experiment, the controller is expected to navigate a 2D quadrotor to the goal set $\boldsymbol{S}_{\text{goal}}$ without colliding with the obstacles. We define the state space as $\mathcal{S} = \{\boldsymbol{s} : \boldsymbol{s}_{\text{lb}} \leq \boldsymbol{s} \leq \boldsymbol{s}_{\text{ub}}\}$ with $\boldsymbol{s}_{\text{lb}} = [-1, 0, -0.25, -0.25]$ and $\boldsymbol{s}_{\text{ub}} = [2, 1.8, 0.25, 0.25]$, representing the lower bound and upper bound of the set of the valid states. The action space is set as $\mathcal{A} = \{\boldsymbol{a} : -\boldsymbol{a}_{\text{b}} \leq \boldsymbol{a} \leq \boldsymbol{a}_{\text{b}}\}$ with $\boldsymbol{a}_{\text{b}} = [0.25, 0.25]$, by considering the real world hardware limitation. The cost function is designed as $c = \sqrt{4p_x^2 + (p_y - 0.5)^2}$. We set the obstacle set $\mathcal{S}_{o1} = \{\boldsymbol{s} : 0.5 \leq p_x \leq 1, 0.2 \leq p_y \leq 1\}$, $\mathcal{S}_{o2} = \{\boldsymbol{s} : -1 \leq p_x \leq 0, 1.3 \leq p_y \leq 1.8\}$ and $\mathcal{S}_{o3} = \{\boldsymbol{s} : p_z \leq 0.2\}$, the unsafe state set $\mathcal{S}_{\text{unsafe}} = \{\boldsymbol{s} : \mathcal{S}_{o1} \cup \mathcal{S}_{o2} \cup \mathcal{S}_{o3}\}$, the goal state set $\mathcal{S}_{\text{goal}} = \left\{\boldsymbol{s} : \sqrt{p_x^2 + (p_y - 0.5)^2} \leq 0.3\right\}$. Once the quadrotor reaches the $\mathcal{S}_{\text{unsafe}}$, the episode ends in advance and the cost function is set as $C = 2000$. The episodes are of maximum length 200 and time step $dt = 0.1$ s. In the experiments, we use Bitcraze's Crazyflie 2.0 quadrotors. We train the controllers in the simulator gym-pybullet-drones [42] based on PyBullet. In the real world, we use a motion capture system for state estimation.

APPENDIX IV
HYPERPARAMETER SETTING

For LBAC, there are two networks: the controller network (actor) and the control Lyapunov barrier network (critic). The controller network is represented by a fully-connected neural network with two hidden layers of size 256 each, with the ReLU activation function, outputting the mean and standard deviations of a Gaussian distribution. A fully-connected neural network represents the control Lyapunov barrier critic network with two hidden layers of size 256, each with a ReLU activation function. We use the vanilla Soft Actor-Critic algorithm [36] for 500 episodes to explore the environment effectively as a warm start. The hyperparameters can be found in Table I

TABLE I
HYPERPARAMETER SETTING IN LBAC

Hyperparameters	2D Quadrotor Navigation
Minibatch size	512
Total episode	2500
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Terminal cost C	2000
Discount factor γ	0.999

In RSPO and SQRL, another safety critic network Q_{risk} is needed to estimate the discounted future probability of constraint violation with discounted γ_{risk}. The safety threshold $\varepsilon_{\text{risk}} \in [0, 1]$ is an upper-bound on the expected risk of the action. In this paper, the safety critic network shares the same architecture as the task critic network, except that a sigmoid activation is added to the output layer to ensure that the outputs are on $[0, 1]$. We use the same hyperparameter settings as LBAC in RSPO, RCPO, and SQRL. The other hyperparameters can be found in Table II.

TABLE II
HYPERPARAMETER SETTING IN SAFE RL

Hyperparameters	2D Quadrotor Navigation
RCPO (γ_{risk} , λ)	$(0.99, 3000)$
RSPO (γ_{risk} , $\varepsilon_{\text{risk}}$, λ)	$(0.99, 0.2, 10000)$
SQRL (γ_{risk} , $\varepsilon_{\text{risk}}$, λ)	$(0.99, 0.2, 5000)$

REFERENCES

[1] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*. PMLR, 2020, pp. 1101–1112.

[2] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[3] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[4] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1471–1478, 2021.

[5] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[6] G. Thomas, Y. Luo, and T. Ma, "Safe reinforcement learning by imagining the near future," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[7] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.

[8] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[9] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[10] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Transactions on robotics*, vol. 35, no. 5, pp. 1186–1205, 2019.

[11] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[12] Y. Shen, M. J. Tobia, T. Sommer, and K. Obermayer, "Risk-sensitive reinforcement learning," *Neural computation*, vol. 26, no. 7, pp. 1298–1328, 2014.

[13] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control lyapunov functions for uncertain robotic systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6878–6884.

[14] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8352–8358.

[15] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 827–834.

[16] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 98–106.

[17] P. Jagtap, G. J. Pappas, and M. Zamani, "Control barrier functions for unknown nonlinear systems using gaussian processes," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3699–3704.

[18] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," in *Robotics: Science and Systems (RSS)*, 2020.

[19] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[20] Y. Meng, Y. Li, M. Fitzsimmons, and J. Liu, "Smooth converse lyapunov-barrier theorems for asymptotic stability with safety constraints and reach-avoid-stay specifications," *Automatica*, vol. 144, p. 110478, 2022.

[21] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Neural certificates for safe control policies," *arXiv preprint arXiv:2006.08465*, 2020.

[22] M. Z. Romdlony and B. Jayawardhana, "Stabilization with guaranteed safety using control lyapunov–barrier function," *Automatica*, vol. 66, pp. 39–47, 2016.

[23] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[24] A. Tamar, Y. Glassner, and S. Mannor, "Policy gradients beyond expectations: Conditional value-at-risk," *arXiv preprint arXiv:1404.3862*, 2014.

[25] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[26] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[27] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[28] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," *arXiv preprint arXiv:2010.14497*, 2020.

[29] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[30] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," *arXiv preprint arXiv:1901.10031*, 2019.

[31] E. Altman, *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

[32] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation." in *Robotics: Science and Systems*, vol. 13. Cambridge, MA, USA, 2017.

[33] Z.-P. Jiang, T. Bian, W. Gao *et al.*, "Learning-based control: A tutorial and some recent results," *Foundations and Trends® in Systems and Control*, vol. 8, no. 3, pp. 176–284, 2020.

[34] M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6217–6224, 2020.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[37] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[38] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.

[39] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, 2021.

[40] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, "Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee," *Automatica*, vol. 129, p. 109689, 2021.

[41] H. L. Royden and P. Fitzpatrick, *Real analysis*. Macmillan New York, 1988, vol. 32.

[42] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

References

[1] https://www.facebook.com/*, *Waymo brings self-driving taxis to San Francisco - with a catch*, Aug. 2021.

[2] P. LeBeau, *Tesla rolls out autopilot technology*, Oct. 2015.

[3] 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 Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 474–488, Jan. 2023, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3216792.

[4] roireshef, *Waymo merging into traffic (repost)*, Jun. 2021.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, Oct. 2017, pp. 1–16.

[6] P. Karle, M. Geisslinger, J. Betz, and M. Lienkamp, "Scenario Understanding and Motion Prediction for Autonomous VehiclesReview and Comparison," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16 962–16 982, Oct. 2022, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3156011.

[7] M. Gulzar, Y. Muhammad, and N. Muhammad, "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving," *IEEE Access*, vol. 9, pp. 137 957–137 969, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3118224.

[8] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 1010–1016. DOI: 10.1109/ITSC.2016.7795679.

[9] P. Bender, Ö.. Ta, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 1386–1392. DOI: 10.1109/IVS.2015.7225909.

[10] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4889–4895. DOI: 10.1109/ICRA.2011.5980223.

[11] Y. Meng, Y. Wu, Q. Gu, and L. Liu, "A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization," *IEEE Access*, vol. 7, pp. 130 530–130 551, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2940271.

[12] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 1879–1884. DOI: 10.1109/IROS.2009.5354448.

[13] H. Fan, F. Zhu, C. Liu, *et al.*, *Baidu Apollo EM Motion Planner*, Jul. 2018. arXiv: 1807.08048 [cs].

[14] 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 *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 632–639.

[15] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2126–2133. DOI: 10.1109/CDC.2017.8263961.

[16] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso–Mora, "Globally guided trajectory planning in dynamic environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 10 118–10 124.

[17] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1208–1214. DOI: 10.1109/ICRA40945.2020.9196996.

[18] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid Trajectory Planning for Autonomous Driving in On-Road Dynamic Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 341–355, Jan. 2021, ISSN: 1558-0016. DOI: 10.1109/TITS.2019.2957797.

[19] R. Tedrake, "Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832," *Working draft edition*, vol. 3, no. 4, p. 2, 2009.

[20] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe Trajectory Generation for Complex Urban Environments Using Spatio-Temporal Semantic Corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, Jul. 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2923954.

[21] Y. Jiang, Z. Liu, D. Qian, H. Zuo, W. He, and J. Wang, "Robust Online Path Planning for Autonomous Vehicle Using Sequential Quadratic Programming," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2022, pp. 175–182. DOI: 10.1109/IV51971.2022.9827017.

[22] Z. Han, Y. Wu, T. Li, *et al.*, *Differential Flatness-Based Trajectory Planning for Autonomous Vehicles*, Aug. 2022. arXiv: 2208.13160 [cs].

[23] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5681–5686. DOI: 10.1109/IROS.2017.8206458.

[24] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous Driving Motion Planning With Constrained Iterative LQR," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 244–254, Jun. 2019, ISSN: 2379-8904. DOI: 10.1109/TIV.2019.2904385.

[25] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 729–15 741, Sep. 2022, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3145389.

[26] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369.

[27] M. Lauri, D. Hsu, and J. Pajarinen, "Partially Observable Markov Decision Processes in Robotics: A Survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, Feb. 2023, ISSN: 1941-0468. DOI: 10.1109/TRO.2022.3200138.

[28] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems*, vol. 23, Curran Associates, Inc., 2010.

[29] Z. Sunberg and M. J. Kochenderfer, "Improving Automated Driving Through POMDP Planning With Human Internal States," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 073–20 083, Nov. 2022, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3182687.

[30] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 558–573, Feb. 2021, ISSN: 0278-3649. DOI: 10.1177/0278364920937074.

[31] M. Naghshvar, A. K. Sadek, and A. J. Wiggers, "Risk-averse behavior planning for autonomous driving under uncertainty," *arXiv preprint arXiv:1812.01254*, 2018.

[32] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 1617–1624. DOI: 10.1109/ITSC.2018.8569729.

[33] J. Fischer, E. Bührle, D. Kamran, and C. Stiller, "Guiding Belief Space Planning with Learned Models for Interactive Merging," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2022, pp. 2542–2549. DOI: 10.1109/ITSC55140.2022.9922488.

[34] Y. Hollander and J. N. Prashker, "The applicability of non-cooperative game theory in transport analysis," *Transportation*, vol. 33, no. 5, pp. 481–496, Sep. 2006, ISSN: 1572-9435. DOI: 10.1007/s11116-006-0009-1.

[35] S. L. Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A Fast Solver for Constrained Dynamic Games," in *Robotics: Science and Systems XVI*, Jul. 2020. DOI: 10.15607/RSS.2020.XVI.091. arXiv: 1910.09713 [cs].

[36] X. Liu, L. Peters, and J. Alonso-Mora, "Learning to Play Trajectory Games Against Opponents With Unknown Objectives," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4139–4146, Jul. 2023, ISSN: 2377-3766. DOI: 10.1109/LRA.2023.3280809.

[37] 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 Transactions on Intelligent Vehicles*, vol. 5, no. 1, pp. 75–89, Mar. 2020, ISSN: 2379-8904. DOI: 10.1109/TIV.2019.2955367.

[38] C. Wei, Y. He, H. Tian, and Y. Lv, "Game Theoretic Merging Behavior Control for Autonomous Vehicle at Highway On-Ramp," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 127–21 136, Nov. 2022, ISSN: 1558-0016. DOI: 10.1109/TITS.2022.3174659.

[39] 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 Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3609–3620, Apr. 2022, ISSN: 1939-9359. DOI: 10.1109/TVT.2022.3148972.

[40] 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 *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 4509–4515. DOI: 10.1109/ICRA48506.2021.9561463.

[41] K. Ji, N. Li, M. Orsag, and K. Han, "Hierarchical and game-theoretic decision-making for connected and automated vehicles in overtaking scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 150, p. 104 109, May 2023, ISSN: 0968-090X. DOI: 10.1016/j.trc.2023.104109.

[42] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency Model Predictive Control for Automated Vehicles," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 717–722. DOI: 10.23919/ACC.2019.8815260.

[43] D. Qiu, Y. Zhao, and C. Baker, "Latent Belief Space Motion Planning under Cost, Dynamics, and Intent Uncertainty," in *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation, Jul. 2020, ISBN: 978-0-9923747-6-1. DOI: 10.15607/RSS.2020.XVI.069.

[44] R. Wang, M. Schuurmans, and P. Patrinos, *Interaction-aware Model Predictive Control for Autonomous Driving*, Nov. 2022. DOI: 10.48550/arXiv.2211.17053. arXiv: 2211.17053 [cs, eess, math].

[45] H. Hu and J. F. Fisac, "Active Uncertainty Reduction for Human-Robot Interaction: An Implicit Dual Control Approach," in *Algorithmic Foundations of Robotics XV*, S. M. LaValle, J. M. O'Kane, M. Otte, D. Sadigh, and P. Tokekar, Eds., ser. Springer Proceedings in Advanced Robotics, Cham: Springer International Publishing, 2023, pp. 385–401, ISBN: 978-3-031-21090-7.

[46] F. Da, "Comprehensive Reactive Safety: No Need For A Trajectory If You Have A Strategy," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 2903–2910. DOI: 10.1109/IROS47612.2022.9981757. arXiv: 2207.00198 [cs].

[47] Y. Chen, U. Rosolia, W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Interactive Multi-Modal Motion Planning With Branch Model Predictive Control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5365–5372, Apr. 2022, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3156648.

[48] K. Leyton-Brown and Y. Shoham, *Essentials of Game Theory: A Concise, Multidisciplinary Introduction* (Synthesis Lectures on Artificial Intelligence and Machine Learning). Cham: Springer International Publishing, 2008. DOI: 10.1007/978-3-031-01545-8.

[49] T. Baar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.

[50] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017, ISBN: 978-0-9759377-3-0.

[51] P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, vol. 4, pp. 1–51, Jan. 1995, ISSN: 1474-0508, 0962-4929. DOI: 10.1017/S0962492900002518.

[52] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006, ISSN: 1436-4646. DOI: 10.1007/s10107-004-0559-y.

[53] P. Polack, F. Altche, B. d'Andrea-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA: IEEE, Jun. 2017, pp. 812–818, ISBN: 978-1-5090-4804-5. DOI: 10.1109/IVS.2017.7995816.

[54] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000. DOI: 10.1103/PhysRevE.62.1805.

[55] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 808–18 821, Oct. 2022, ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2022.3160936.

[56] R. D. Luce, *Individual Choice Behavior* (Individual Choice Behavior). Oxford, England: John Wiley, 1959, pp. xii, 153.

[57] 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," *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, Oct. 2018, ISSN: 1573-7527. DOI: 10.1007/s10514-018-9746-1.

[58] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-end Trajectory Prediction from Dense Goal Sets," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 15 283–15 292, ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.01502.

[59] N. Deo, E. Wolff, and O. Beijbom, "Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals," in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 203–212.

[60] W. Zeng, W. Luo, S. Suo, *et al.*, "End-To-End Interpretable Neural Motion Planner," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 8652–8661, ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00886.

[61] D. Chen and P. Krahenbuhl, "Learning from All Vehicles," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 17 201–17 210, ISBN: 978-1-66546-946-3. DOI: 10.1109/CVPR52688.2022.01671.

[62] Y. Hu, J. Yang, L. Chen, *et al.*, "Planning-Oriented Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[63] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA: IEEE, May 2015, pp. 1670–1677, ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139412.

[64] W. Ding, L. Zhang, J. Chen, and S. Shen, "EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1118–1138, Apr. 2022, ISSN: 1941-0468. DOI: 10.1109/TRO.2021.3104254.

[65] 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 Robotics and Automation Letters*, vol. 6, no. 3, pp. 4978–4985, Jul. 2021, ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3068657.

[66] R. C. Coulter *et al.*, *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.

[67] J. Schulman, Y. Duan, J. Ho, *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364914528132.

[68] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha A local, continuous method," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Jun. 2014, pp. 450–457. DOI: 10.1109/IVS.2014.6856581.

[69] M. Quigley, K. Conley, B. Gerkey, *et al.*, "Ros: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[70] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.

[71] S. Dai, S. Bae, and D. Isele, *Game Theoretic Decision Making by Actively Learning Human Intentions Applied on Autonomous Driving*, Jan. 2023. arXiv: 2301.09178 [cs].

[72] Y. Chen, U. Rosolia, C. Fan, A. Ames, and R. Murray, "Reactive motion planning with probabilisticsafety guarantees," in *Proceedings of the 2020 Conference on Robot Learning*, PMLR, Oct. 2021, pp. 1958–1970.

[73] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020, ISSN: 0020-7179, 1366-5820. DOI: 10.1080/00207179.2016.1222553.

[74] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019, ISSN: 1867-2957. DOI: 10.1007/s12532-018-0139-4.

[75] D. Du, S. Han, N. Qi, H. B. Ammar, J. Wang, and W. Pan, "Reinforcement Learning for Safe Robot Control using Control Lyapunov Barrier Functions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom: IEEE, May 2023, pp. 9442–9448. DOI: 10.1109/ICRA48891.2023.10160991.

[76] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 2018, pp. 1861–1870.