Game-Theoretic and Set-Based Methods for Safe Autonomous Vehicles on Shared Roads

by

Nan Li

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2021

Doctoral Committee:

Associate Professor Anouck R. Girard, Co-Chair
Professor Ilya V. Kolmanovsky, Co-Chair
Dr. Dimitar P. Filev, Ford Motor Company
Professor Jing Sun

Nan Li

nanli@umich.edu

ORCID iD: 0000-0001-7928-8796

Dedication

This dissertation is dedicated to my grandma, Guofen Song, and my parents, Huanyin Li and Mingli Zhang. This journey would not have been possible without your unconditional love and support.

Acknowledgments

Firstly, I would like to thank my advisors, Prof. Anouck Girard and Prof. Ilya Kolmanovsky, for their guidance and support. They have been excellent advisors, mentors, and role models for me. I am privileged to have had the opportunity to learn from them and work with them. Secondly, I would like to thank my committee members, Dr. Dimitar Filev and Prof. Jing Sun, for their continued support and encouragement, and for their insightful comments and suggestions, which have helped me improve my research results and dissertation writing. Also, I would like to thank Prof. Gábor Orosz. His instruction and mentorship during my master's studies inspired my interest in and dedication to the field of dynamic systems and control. Finally, I would like to acknowledge my colleagues for their collaboration that has helped me accomplish my research goals.

TABLE OF CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Figures . . . . . **vi**

List of Tables . . . . . **ix**

List of Appendices . . . . . **x**

Abstract . . . . . **xi**

Introduction **1**

Chapter

1 Game-Theoretic Modeling of Driver Interactions in Multi-Vehicle Traffic Scenarios **4**

1.1 Introduction to game-theoretic modeling of driver interactions 4
1.1.1 Background and overview 4
1.1.2 Definition of a driver model 7
1.1.3 Level-k reasoning theory 8
1.1.4 Leader-follower game 8
1.2 Level-k based modeling of driver interactions on highways 9
1.2.1 Vehicle kinematics 9
1.2.2 Action space 10
1.2.3 Observation space 11
1.2.4 Reward function 12
1.2.5 Constraints 13
1.2.6 Level-k decision making 14
1.2.7 Obtaining level-k policies using reinforcement learning 16
1.2.8 Simulation results of level-k driver models 19
1.3 Leader-follower based modeling of driver interactions at intersections . . 26
1.3.1 Parameterized intersection 26
1.3.2 Vehicle kinematics 28
1.3.3 Reward function 30
1.3.4 Leader-follower decision making 33
1.3.5 Additional modeling considerations 37

1.3.6 Simulation results of leader-follower driver models 39
1.4 Application to verification and validation of autonomous vehicle control systems 48
1.4.1 Highway simulator and virtual V&V results of two AV policies . 49
1.4.2 Integration with the TORCS simulator and calibration 55
1.4.3 Intersection simulator and proof of concept of an adaptive level-k AV policy 58
1.5 Summary and discussion 63

2 Interaction-Aware Autonomous Vehicle Control 64

2.1 Background and introduction 64
2.2 Interaction-aware autonomous vehicle control problem 66
2.3 Interaction-aware AV control as a partially observable decision problem . 68
2.4 An MPC-based solution approach to C-POMDP problems (POMDP-MPC) 70
2.4.1 An approach to general C-POMDP problems 70
2.4.2 Theoretical properties and recursive feasibility 78
2.4.3 Adaptation of POMDP-MPC to AV control problem 82
2.5 Simulation examples of interaction-aware AV control 83
2.5.1 Level-k models for the other vehicle 83
2.5.2 Simulation examples and results 85
2.6 Summary and discussion 89

3 Enhancing Autonomous Vehicle Safety via the Action Governor 93

3.1 Background and introduction 93
3.2 System model and control objective 95
3.3 Action Governor 96
3.3.1 Safe set and unrecoverable sets 97
3.3.2 Offline and online computations 100
3.4 Simulation examples 103
3.4.1 Adaptive cruise control 103
3.4.2 Omni-directional robot obstacle avoidance 105
3.5 Summary and discussion 107

4 Conclusions and future work 108

4.1 Conclusions 108
4.2 Future work 109

Bibliography 110

Appendices 124

LIST OF FIGURES

1.1 The procedure of training level-k driving policies using RL. 19
1.2 Level-0 simulation results: Plots (a)-(c) show snapshots of the simulation at 40[s], 42[s] and 44[s], respectively. 21
1.3 Evolution of the average reward during level-1 and level-2 training. 22
1.4 Simulation of a level-1 ego vehicle in a level-0 traffic environment: Plots (a)-(f) show snapshots of the simulation at 178[s], 179[s], 180[s], 181[s], 184[s], 188[s], respectively. 23
1.5 Simulation of a level-2 ego vehicle in a level-1 traffic environment: Plots (a)-(d) show snapshots of the simulation at 60[s], 66[s], 79[s] and 80[s], respectively. 24
1.6 Validation of level-1 and level-2 driver models with traffic data. The red vehicle uses a level-k policy, and the blue-shaded rectangle represents a real vehicle in traffic data. 25
1.7 Safety violation rates of (a) a level-k ego vehicle in a level-$(k-1)$ traffic environment, and (b) level-0, 1 and 2 ego vehicles in a same traffic environment, which is composed of a mixture of 10% level-0, 60% level-1 and 30% level-2 vehicles. 26
1.8 A four-way intersection modeled by (1.22) and (1.23). The orange dashed lines are the road centerlines[3], the black dashed lines are the lane markings that separate the lanes of traffic moving in the same directions, the black solid lines are the road boundaries, and the shaded polygons are off-road regions. . . 27
1.9 Vehicle kinematics modeled by (1.24)-(1.27). The blue rectangle represents the vehicle's c-zone where the end with double lines is the vehicle's front end. The blue dotted curve represents the pre-planned path $\mathcal{P}$. The states $x(t)$, $y(t)$ and $\theta(t)$ can be computed using the traveled distance along the path $\rho(t)$ and the path geometry (1.24). The green triangles represent the intersection entrance points $(x(\rho^{\text{en}}), y(\rho^{\text{en}}))$ and the red triangles the intersection exit points $(x(\rho^{\text{ex}}), y(\rho^{\text{ex}}))$. 30
1.10 The c-zone (dark blue rectangle) and s-zone (light blue rectangle) of a vehicle. 32
1.11 Leader-follower role assignment. In all of the figures, the red car is the leader and the yellow car is the follower. 35
1.12 Reproducing a real-world traffic scenario with 3 interacting vehicles by the proposed model. 41
1.13 Reproducing a real-world traffic scenario with 4 interacting vehicles by the proposed model. 42
1.14 Completely symmetric case 1. Figures (a-f) show snapshots of the simulation at a series of time steps. 43

1.15 Completely symmetric case 2. Figures (a-d) show snapshots of the simulation at a series of time steps. 44
1.16 Randomized traffic scenarios. Figures (a-b) show snapshots of a simulation in a three-way intersection scenario, figures (c-d) show those in a four-way intersection scenario, and figures (e-f) show those in a five-way intersection scenario. 45
1.17 Statistical evaluation of the vehicle interaction model. Light color: SR, medium color: DR, dark color: CR. 46
1.18 Two failure cases. (a) A deadlock scenario. (b) A collision scenario. 46
1.19 Average completion time (ACT). The black vertical bars represent the standard deviations. 48
1.20 Safety violation rates of the Stackelberg game-based policy and the decision tree-based policy. 50
1.21 Scenarios leading to safety violations. 51
1.22 Trend of safety violation rate. 52
1.23 (a) Average driving speeds and (b) computational costs of the Stackelberg game-based policy and the decision tree-based policy. 53
1.24 Objective function surfaces corresponding to different choices of the weights p_1 and p_2. (a) $p_1 = 1, p_2 = 0$, (b) $p_1 = 0, p_2 = 1$, (c) $p_1 = 0.7, p_2 = 0.3$, and (d) $p_1 = 0.6, p_2 = 0.4$. 54
1.25 Schematics of the lateral motion control. 57
1.26 Controlled lane change response of a TORCS vehicle. 58
1.27 Snapshots of a TORCS simulation with our level-k driver models integrated. . 59
1.28 Average (dark-colored bars) and worst-case (light-colored bars) computation times per vehicle per step. 60
1.29 Simulation of an adaptive level-k vehicle (blue car) versus two leader-follower drivers (red and green cars). Figures (a-d) show snapshots of the simulation at a series of time steps. 61
1.30 Model identification history corresponding to the simulation of Fig. 1.29. . . . 62
1.31 Simulation of two adaptive level-k vehicles (red and green cars) versus one leader-follower driver (blue car). Figures (a-d) show snapshots of the simulation at a series of time steps. 62
1.32 Model identification history corresponding to the simulation of Fig. 1.31. . . . 63

2.1 Simulation results of an intersection crossing scenario. (a-1) and (a-2) show two steps of a simulation where the autonomous ego vehicle (blue) interacts with a level-1 vehicle (red). (b-1) and (b-2) show two steps of a simulation where the autonomous ego vehicle (blue) interacts with a level-2 vehicle (red). 87
2.2 Simulation results of a highway overtaking scenario. (a-1) to (a-4) show four steps of a simulation where the autonomous ego vehicle (blue) overtakes a level-1 vehicle (red). (b-1) to (b-4) show four steps of a simulation where the autonomous ego vehicle (blue) overtakes a level-2 vehicle (red). 91

2.3 Simulation results of a forced merging scenario. (a-1) to (a-4) show four steps of a simulation where the autonomous ego vehicle (blue) merges in front of a level-1 vehicle (red). (b-1) to (b-4) show four steps of a simulation where the autonomous ego vehicle (blue) merges behind a level-2 vehicle (red). 92

3.1 An exclusion zone example, where the union of the red polytopic regions is the exclusion zone X_0. 96
3.2 Adaptive cruise control. 105
3.3 Omni-directional robot obstacle avoidance. 107

C.1 Decision tree diagram. The black arrows indicate the relative velocities of the yellow cars with respect to the red car. 132
C.2 Regions for policy activation criteria. 133

LIST OF TABLES

1.1 Parameter values for highway simulations. 20
1.2 Parameter values for intersection simulations. 40
1.3 Vehicle control-related effectors of TORCS. 55

2.1 Computation times of POMDP-MPC. 90

LIST OF APPENDICES

A Proof of RL Algorithm Convergence **124**

B Functions of Path Model **129**

C Autonomous Vehicle Control Approaches for Highway Driving **131**

D Adaptive Level-k Policy for Intersection Driving **134**

E Lemma 3.1 . . . . . **136**

ABSTRACT

Autonomous vehicle (AV) technology promises safer, cleaner, and more efficient transportation, as well as improved mobility for the young, elderly, and disabled. One of the biggest challenges of AV technology is the development and high-confidence verification and validation (V&V) of decision and control systems for AVs to safely and effectively operate on roads shared with other road users (including human-driven vehicles). This dissertation investigates game-theoretic and set-based methods to address this challenge.

Firstly, this dissertation presents two game-theoretic approaches to modeling the interactions among drivers/vehicles on shared roads. The first approach is based on the "level-k reasoning" human behavioral model and focuses on the representation of heterogeneous driving styles of real-world drivers. The second approach is based on a novel leader-follower game formulation inspired by the "right-of-way" traffic rules and focuses on the modeling of driver intents and their resulting behaviors under such traffic rules and etiquette. Both approaches lead to interpretable and scalable driver/vehicle interaction models. This dissertation then introduces an application of these models to fast and economical virtual V&V of AV control systems.

Secondly, this dissertation presents a high-level control framework for AVs to safely and effectively interact with other road users. The framework is based on a constrained partially observable Markov decision process (POMDP) formulation of the AV control problem, which is then solved using a tailored model predictive control algorithm called POMDP-MPC. The major advantages of this control framework include its abilities to handle interaction uncertainties and provide an explicit probabilistic safety guarantee under such uncertainties.

Finally, this dissertation introduces the Action Governor (AG), which is a novel add-on scheme to a nominal control loop for formally enforcing pointwise-in-time state and control constraints. The AG operates based on set-theoretic techniques and online optimization. Theoretical properties and computational approaches of the AG for discrete-time linear systems subject to non-convex exclusion-zone avoidance constraints are established. The use of the AG for enhancing AV safety is illustrated through relevant simulation case studies.

INTRODUCTION

Autonomous vehicle technology promises safer, cleaner, and more efficient transportation, as well as improved mobility for the young, elderly, and disabled [1]. Thanks to extensive efforts that have been made in both academia and industry over the last few decades to pursue this goal, advances in perception [2], planning and decision-making [3], control [4], and computing systems [5] have made fully autonomous driving a viable option for future transportation [6, 7]. Despite these advances, numerous challenges remain regarding guaranteed safety and performance of autonomous vehicles in a complex, uncertain, and increasingly data-rich and hyper-connected world [8, 9, 10, 11].

It is projected that in the near to medium term, autonomous vehicles will operate on shared roads with other users (including human-driven vehicles) [12, 13]. This imposes a requirement for highly reliable decision and control systems for autonomous vehicles that can appropriately account for/respond to the interactions among road users to achieve safe and effective autonomous vehicle operation. Meanwhile, the development and high-confidence validation of such systems have been recognized by the automotive industry to be among the biggest technological challenges that are currently delaying the advent of fully autonomous driving [14, 15, 16]. The difficulties lie in the accurate modeling and prediction of road user interactions, which are largely due to the complex nature of human behavior, and also in the proper decision-making and control in the presence of these complex interactions.

This dissertation aims to address the challenges in modeling and safe control for autonomous vehicles on shared roads.

The main contributions and outline of this dissertation are as follows:

1. Chapter 1 introduces two novel approaches based on game theory to modeling the interactions among drivers/vehicles in traffic. The first approach is inspired by a human behavioral model called level-k reasoning, and the second approach is inspired by the right-of-way traffic rules. Compared to data-driven approaches to modeling driver behaviors, these game-theoretic approaches lead to models that have better interpretability and (re)usability. Compared to several other game-theoretic approaches previously proposed in the literature, these two new approaches have higher-fidelity in terms of explicitly modeling the dynamic behaviors of vehicles, and have better scalability in terms of being able to model medium-scale traffic scenarios involving tens of interacting vehicles at a reasonable computational cost. The models obtained by these approaches can have multiple applications. The latter part of this chapter introduces their application to simulation-based testing of autonomous vehicle control systems. This application aims to address the above-mentioned challenge of high-confidence validation of such systems.

2. Chapter 2 deals with high-level control[1] of autonomous vehicles in the presence of road user interactions. A novel interaction-aware control approach is proposed that handles interaction uncertainties and vehicle safety under such uncertainties through a constrained partially observable Markov decision process (C-POMDP) framework. A new computational algorithm based on model predictive control using online optimization is developed to solve the formulated C-POMDP problem and leads to theoretically elegant and computationally feasible autonomous vehicle control solutions.

3. Chapter 3 focuses on the safety of more general controlled dynamic systems (including autonomous vehicles). In this chapter, a novel add-on scheme to nominal control loops, called the action governor (AG), is proposed as a safety supervisor that

[1] Also called behavior planning.

monitors and minimally modifies (when necessary) the nominal control signal to enforce pointwise-in-time state and control constraints. The theoretical foundation and computational approach of the AG for discrete-time linear systems with non-convex exclusion-zone avoidance constraints are established, and two autonomous vehicle related examples are considered. Ongoing research reveals the potential of the AG to be used as a general safety supervision framework for enhancing autonomous vehicle safety.

In addition to the above-mentioned methodological developments and advancements for safe autonomous vehicles on shared roads, I have made many other research contributions during my PhD, mainly in the areas of 1) control of systems with constraints, 2) stochastic modeling and control, and 3) learning-enabled control of unknown systems. I have advanced the theory and methods of the reference governor (RG) for nonlinear systems [17, 18, 19, 20, 21, 22], developed new methods for stochastic/chance-constrained systems [23, 24, 25, 26, 27] and *a priori* unknown/black-box systems [28, 29], and investigated their applications in the aerospace and automotive domains [30, 31, 32, 33, 34, 35, 36, 37]. The details of these developments can be found in my journal and conference publications.

CHAPTER 1

Game-Theoretic Modeling of Driver Interactions in Multi-Vehicle Traffic Scenarios

1.1 Introduction to game-theoretic modeling of driver interactions

1.1.1 Background and overview

In traffic scenarios that involve multiple vehicles, the behavior of a vehicle will influence and be influenced by the behaviors of the other vehicles in its vicinity. For instance, at an intersection, two vehicles may end up occupying the same space at the same time, i.e., having a collision, if they both maintain their current speed. In this case, at least one of the two vehicles needs to adjust its speed or change its lane to avoid the collision. And only after at least one of them makes such an adjustment, the other vehicle can safely pass through the intersection. Such mutual influences of vehicle behaviors, including drivers' adjustments of vehicle speeds or lanes to account for such mutual influences, are referred to as *vehicle/driver interactions.* In this chapter, the terms *vehicle interactions* and *driver interactions* are used interchangeably, depending on whether the emphasis is on the vehicles themselves or on the drivers who control the vehicles.

For a human driver or an autonomous driving system to drive safely in these multi-vehicle traffic scenarios, it is necessary to understand and hence be able to predict and account for the driver interactions. This motivates the developments of models that can realistically represent driver interactions.

One way to obtain such driver interaction models is to estimate/learn a model from traffic data. This approach has been pursued in [38, 39, 40, 41, 42, 43]. Limitations of these data-driven models include: 1) A model for a traffic scenario of interest can be learned only when sufficient data for this scenario are available. However, it has been reported

that multi-vehicle interaction scenarios in released traffic datasets are insufficient [44]. 2) A learned model for a specific scenario cannot be easily transferred to modeling other scenarios [45], which largely limits the (re)usability of these learned models.

Another strategy is to develop "first principles" models. Here, "first principles" means that the model is defined by a (or, a set of) mathematical formula(s) that has certain psycho-physical interpretations. For modeling (human) driver interactions, such psycho-physical interpretations often relate to human cognitive and behavioral processes. Compared to data-driven models, a "first principles" model relies only lightly on traffic data (for parameter calibration and validation) and typically can effectively model a range of scenarios, thus overcoming the two limitations of data-driven models mentioned above. The game-theoretic driver models presented in this chapter belong to this category.

Game theory is the study of mathematical models for strategic interaction among rational decision makers [46]. It has applications in many fields of social science, as well as in logic, systems science and computer science. Game theory has also been exploited by several researchers for modeling vehicle/driver interactions in traffic [47, 48, 49, 50, 51, 52, 53, 54, 55]. However, a dynamic game, which means a game involving multiple moves of the players in a time-extended scenario, can be difficult to solve, especially when the game involves more than two players [56]. As a result, many of these previous works choose to model driver interactions in a specific traffic scenario as a one-shot game, i.e., not explicitly accounting for vehicles' dynamic behaviors in the game formulation, to simplify computations. For instance, [47] models driver interactions at an intersection as a one-shot normal-form game, where each driver chooses between "stop" and "go" according to the payoff matrix; for highway scenarios, [48, 49] and [50] model lane changes as instantaneous events, i.e., the duration and vehicle dynamics of lane changes are neglected. The approaches of [51, 52, 53, 54] and [55] explicitly account for the dynamic behaviors of vehicles in their game formulations. The cost is that these approaches do not scale well – they can only model the interaction among two to at most three vehicles due to computational complexity [51, 52, 55] and/or theoretical limitations [53, 54].

In this chapter, we introduce two novel game-theoretic approaches to modeling driver interactions. The first approach is based on a unique combination of level-k reasoning theory [57, 58, 59] and reinforcement learning [60]. The second approach is based on a novel game formulation with multiple concurrent leader-follower pairs, called a leader-follower game, which is partly inspired by Stackelberg game theory [61, 62] and right-of-way traffic rules [63]. These two approaches share the following features:

1. Both approaches produce interactive driver models. A formal definition of a *driver model* is given below. Here, "interactive" means that the driver model takes into con-

sideration the interaction among vehicles explicitly in its decision-making process.

2. Both approaches formulate the multi-vehicle traffic scenario to be modeled as a dynamic game, i.e., explicitly account for vehicles' dynamic behaviors as outcomes of drivers' sequential decision-making using a kinematics model in their game formulations.

3. Both approaches scale reasonably well. They can model medium-scale traffic scenarios involving tens of interacting vehicles at a reasonable computational cost on a desktop.

The bullet points 2 and 3 address the drawbacks of previous game-theoretic approaches to driver interaction modeling mentioned above.

The interactive driver models produced by these approaches can have multiple applications. In Section 1.4, we introduce their application to forming/enhancing traffic simulators for virtual verification and validation (V&V) of autonomous vehicle control systems. This application aims to address the urgent challenge that hundreds of millions of miles need to be driven to demonstrate autonomous vehicle reliability [64] and can support the reduction of autonomous vehicle time-to-market. These driver models can also be used in an autonomous driving algorithm for predicting surrounding vehicles' future trajectories in response to the autonomous ego vehicle's actions. Such an application is discussed in detail in Chapter 2.

These two approaches also have distinct features and to some extent complement each other. In particular, the level-k reasoning based approach focuses more on the modeling of heterogeneous driving styles of real-world drivers (such as aggressive driving versus conservative driving). Meanwhile, the leader-follower game based approach focuses more on the modeling of driver intents and their resulting behaviors under common traffic rules and etiquette (such as to proceed or to yield). We choose to use the level-k reasoning based approach to model driver interactions on highways, where a driver's behavior is deeply influenced by her driving style. For instance, when the vehicle in front drives at a slower speed, an aggressive driver may tend to change lanes to overtake while a conservative driver may tend to slow down to follow. Meanwhile, we choose to use the leader-follower game based approach to model driver interactions at intersections, where traffic rules and etiquette (such as common right-of-way rules) are observed to take a dominant role in determining drivers' behaviors.

The developments of this chapter and related materials have been published in the journal articles [65, 66, 67], book chapter [68], and conference papers [69, 70, 71, 72, 73, 74, 75, 76, 77].

1.1.2 Definition of a driver model

For a traffic scenario $\sigma \in \Sigma$, a driver model is a 3-tuple of state transition model T^σ, observation model O^σ, and driving policy π^σ, i.e.,

$$\langle T^\sigma, O^\sigma, \pi^\sigma \rangle, \tag{1.1}$$

where T^σ represents a model in this driver's mind that characterizes the traffic state evolution with respect to her own action, O^σ describes the way this driver perceives the traffic state, and π^σ delineates how she drives. Specifically, the transition model T^σ is a stochastic map on $\mathcal{S}^\sigma \times \mathcal{A}^\sigma \times \mathcal{S}^\sigma$, where $\mathcal{S}^\sigma$ is a state space and $\mathcal{A}^\sigma$ is an action space, and T^σ defines the probability of transition from a current traffic state $\mathbf{s} \in \mathcal{S}^\sigma$ to a next state $\mathbf{s}^+ \in \mathcal{S}^\sigma$ when the driver applies an action $a \in \mathcal{A}^\sigma$, i.e.,

$$\mathbb{P}(\mathbf{s}^+|\mathbf{s}, a) = T^\sigma(\mathbf{s}, a, \mathbf{s}^+) \quad \text{s.t.} \quad \sum_{\mathbf{s}^+ \in \mathcal{S}^\sigma} \mathbb{P}(\mathbf{s}^+|\mathbf{s}, a) = \sum_{\mathbf{s}^+ \in \mathcal{S}^\sigma} T^\sigma(\mathbf{s}, a, \mathbf{s}^+) = 1. \tag{1.2}$$

The observation model O^σ is a stochastic map from the state space $\mathcal{S}^\sigma$ to an observation space Ω^σ and defines the probability of each observation $o \in \Omega^\sigma$ being received by the driver given the current traffic state $\mathbf{s} \in \mathcal{S}^\sigma$, i.e.,

$$\mathbb{P}(o|\mathbf{s}) = O^\sigma(\mathbf{s}, o) \quad \text{s.t.} \quad \sum_{o \in \Omega^\sigma} \mathbb{P}(o|\mathbf{s}) = \sum_{o \in \Omega^\sigma} O^\sigma(\mathbf{s}, o) = 1. \tag{1.3}$$

The driving policy π^σ is a stochastic map from the observation space Ω^σ to the action space $\mathcal{A}^\sigma$, which defines the probability of each action $a \in \mathcal{A}^\sigma$ being taken by the driver when the driver observes $o \in \Omega^\sigma$, i.e.,

$$\mathbb{P}(a|o) = \pi^\sigma(o, a) \quad \text{s.t.} \quad \sum_{a \in \mathcal{A}^\sigma} \mathbb{P}(a|o) = \sum_{a \in \mathcal{A}^\sigma} \pi^\sigma(o, a) = 1. \tag{1.4}$$

When a stochastic map is actually deterministic, we employ simplified notations. For instance, when $T^\sigma(\mathbf{s}, a, \mathbf{s}^+) \in \{0, 1\}$, we also write $T^\sigma : (\mathbf{s}, a) \mapsto \mathbf{s}^+$, where $\mathbf{s}^+ \in \mathcal{S}^\sigma$ is the state with $\mathbb{P}(\mathbf{s}^+|\mathbf{s}, a) = T^\sigma(\mathbf{s}, a, \mathbf{s}^+) = 1$. The notations $O^\sigma : \mathbf{s} \mapsto o$ and $\pi^\sigma : o \mapsto a$ are used in a similar way.

The maps T^σ, O^σ, π^σ and the spaces $\mathcal{S}^\sigma$, $\mathcal{A}^\sigma$, Ω^σ are all parameterized by the scenario identifier σ, which takes values in a pre-constructed scenario set Σ. The identifier $\sigma \in \Sigma$ facilitates the incorporation of driver models defined for different traffic scenarios within a unified framework. In this chapter, we focus on two traffic scenarios: highway ($\sigma = \mathrm{h}$) and intersection ($\sigma = \mathrm{i}$).

1.1.3 Level-k reasoning theory

Level-k reasoning, or level-k thinking, is a behavioral model that describes human thought processes in interactive scenarios. The level-k model is based on the presumption that humans' behavior can be classified into different levels of reasoning. A level-0 model typically corresponds to non-strategic behavior and follows a simple decision rule. Then, a level-k model, for any $k \geq 1$, behaves as if she best-responds to the belief that all the other participants in the scenario are level-$(k-1)$.

The theory of level-k reasoning was first proposed in the seminal works by Stahl and Wilson (1994,1995) [57, 58] and Nagel (1995) [59], and then followed up, validated, and extended by many researchers, for instance, in [78, 79, 80, 81]. It has been observed in many behavioral experiments that level-k reasoning leads to improved accuracy in predicting human interactive behavior compared to conventional analytic models (including backwards induction and iterated elimination of dominated strategies), which can deviate considerably from actual experimental outcomes.

In this chapter, we exploit level-k reasoning theory to model (human) driver interactive behaviors on highways. We use different reasoning levels, k, to represent the different driving styles of real-world drivers (such as aggressive driving versus conservative driving), that deeply influence a driver's behavior in a certain traffic situation.

1.1.4 Leader-follower game

In many interactive scenarios, participants do not have symmetric roles. Instead, some parties have some sort of advantage over the other. For instance, at an intersection, if a collision occurs between a vehicle that has the right of way and a vehicle that should yield but fails to do so, the driver of the second vehicle is more likely to be determined as at fault.

Motivated by this observation as well as by common traffic rules and etiquette (such as the right-of-way rules), we propose a novel game formulation where the interaction between each pair of interacting vehicles is characterized by a leader-follower relationship. In particular, the leader corresponds to a vehicle that has the right of way and the follower corresponds to a vehicle that is supposed to yield under the right-of-way rules. This leader-follower game formulation is also partly inspired by Stackelberg game theory [61, 62], while relaxes several assumptions of a standard Stackelberg equilibrium which generally do not hold for driver/vehicle interactions in traffic.

In this chapter, we focuses on the application of this leader-follower game formulation to modeling driver interactions at intersections, where the right-of-way rules are observed to take a dominant role in determining drivers' behaviors [63]. Note, however, that this

game formulation may be applied to some other traffic scenarios as well, such as highway forced merging scenarios.

1.2 Level-k based modeling of driver interactions on highways

1.2.1 Vehicle kinematics

We use the following discrete-time EOMs to represent the longitudinal and lateral kinematics of a vehicle for driving on a multi-lane highway,

$$\begin{aligned} x(t+1) &= x(t) + v_x(t)\Delta t, \\ v_x(t+1) &= v_x(t) + a_x(t)\Delta t, \\ y(t+1) &= y(t) + v_y(t)\Delta t, \end{aligned} \tag{1.5}$$

where $x(t)$ ($y(t)$) denotes the vehicle's longitudinal (lateral) position at the discrete time instant $t \in \mathbb{N}_0$, $v_x(t)$ ($v_y(t)$) denotes its longitudinal (lateral) velocity at t, $a_x(t)$ denotes its longitudinal acceleration at t, and $\Delta t > 0$ is the sampling period. In (1.5), $s(t) = (x(t), v_x(t), y(t))$ defines the state of the vehicle at t, and the longitudinal acceleration $a_x(t)$ and lateral velocity $v_y(t)$ are two inputs, the values of which are determined by the driver's maneuver decision made at t. We call the pair $a(t) = (a_x(t), v_y(t))$ the action of the vehicle at t.

Our objective is to model the interactive behaviors of drivers/vehicles in multi-vehicle highway scenarios. To create an environment where the drivers are persistently interacting with each other over a prolonged simulation episode, we assume the vehicles are driving on a cyclic road with a length of $x_{\max}$ (otherwise the distances between vehicles may become very large over a long-period simulation and the interactions between these vehicles will become weak or vanish). Also, we want to restrict the longitudinal velocity v_x of each vehicle to be within a range $[v_{x,\min}, v_{x,\max}]$ to represent typical vehicle speeds during highway driving, and restrict the lateral position y of each vehicle to be within the range $[0, (n_{\text{lane}} - 1)w_{\text{lane}}]$, where $y = 0$ corresponds to the center of the rightmost lane, $n_{\text{lane}} \in \mathbb{N}_0$ denotes the number of lanes, $w_{\text{lane}} > 0$ denotes the lane width, and hence $y = (n_{\text{lane}} - 1)w_{\text{lane}}$ corresponds to the center of the leftmost lane. To account for these

considerations, we modify the kinematics model (1.5) to

$$\begin{aligned} x(t+1) &= \big(x(t) + v_x(t)\Delta t\big) \bmod x_{\max}, \\ v_x(t+1) &= \text{sat}_{[v_{x,\min}, v_{x,\max}]}\big(v_x(t) + a_x(t)\Delta t\big), \\ y(t+1) &= \text{sat}_{[0,(n_{\text{lane}}-1)w_{\text{lane}}]}\big(y(t) + v_y(t)\Delta t\big), \end{aligned} \tag{1.6}$$

where mod denotes the modulo operator, and $\text{sat}_{[\alpha,\beta]}(\cdot)$ represents the saturation function to the range $[\alpha, \beta]$.

For modeling a traffic scenario with n_{v} vehicles, we associate a variable with a subscript i to indicate that this variable corresponds to vehicle i, with $i \in \mathcal{V} = \{1, 2, \ldots, n_{\text{v}}\}$. For instance, $s_i(t)$ ($a_i(t)$) denotes the state (action) of vehicle i at t. Then, the collection of all vehicles' states, $\mathbf{s}(t) = (s_1(t), s_2(t), \ldots, s_{n_{\text{v}}}(t))$, defines the state of the traffic scenario at t, and the EOMs (1.6) of all vehicles determine the evolution of $\mathbf{s}(t)$ as a result of all vehicles' actions $(a_1(t), a_2(t), \ldots, a_{n_{\text{v}}}(t))$, which can be expressed as follows,

$$\mathbf{s}(t+1) = \bar{T}^{\text{h}}\big(\mathbf{s}(t), a_1(t), a_2(t), \ldots, a_{n_{\text{v}}}(t)\big), \tag{1.7}$$

where $\bar{T}^{\text{h}}$ is a deterministic map defined by (1.6).

1.2.2 Action space

Drivers are modeled to have the following 7 basic actions for highway driving:

1. "Maintain" current lane and speed, corresponding to $(a_x, v_y) = (0, 0)$;
2. "Accelerate" at a rate of a_x^1, corresponding to $(a_x, v_y) = (a_x^1, 0)$;
3. "Decelerate" at a rate of $-a_x^1$, corresponding to $(a_x, v_y) = (-a_x^1, 0)$;
4. "Hard accelerate" at a rate of a_x^2, corresponding to $(a_x, v_y) = (a_x^2, 0)$;
5. "Hard decelerate" at a rate of $-a_x^2$, corresponding to $(a_x, v_y) = (-a_x^2, 0)$;
6. "Change lanes to the left," corresponding to $(a_x, v_y) = (0, \frac{w_{\text{lane}}}{t_{\text{lane}}})$;
7. "Change lanes to the right," corresponding to $(a_x, v_y) = (0, -\frac{w_{\text{lane}}}{t_{\text{lane}}})$;

where $a_x^1, a_x^2 > 0$ are acceleration values, and $t_{\text{lane}} > 0$ represents the time needed for a vehicle to complete a lane change. The following action space for $a = (a_x, v_y)$ is defined

based on these 7 actions:

$$\mathcal{A}^{\text{h}} = \Big\{(0,0), (a_x^1, 0), (-a_x^1, 0), (a_x^2, 0), (-a_x^2, 0), (0, \frac{w_{\text{lane}}}{t_{\text{lane}}}), (0, -\frac{w_{\text{lane}}}{t_{\text{lane}}})\Big\}. \tag{1.8}$$

Using such a discrete action space to represent drivers' typical behaviors during highway driving is motivated by the "action point" driver models [82].

1.2.3 Observation space

We assume a driver can observe the following quantities related to her own vehicle (called the "ego vehicle") and the vehicles in the immediate vicinity of her own, and uses them for decision making. In particular, to account for the limited resolution of human vision and/or measurement uncertainty, these quantities are assumed to be measured in discrete, qualitative values as follows:

1. The longitudinal distance from the ego vehicle to the vehicle in the immediate front of and in the same lane with the ego vehicle, d_{fc}, quantified as "close" if $d_{\text{fc}} \leq d_{\text{c}}$, "medium" if $d_{\text{c}} < d_{\text{fc}} \leq d_{\text{f}}$, and "far" if $d_{\text{f}} < d_{\text{fc}} \leq d_{\text{v}}$ (the vehicle is assumed to be out of visual range and unobservable if $d_{\text{fc}} > d_{\text{v}}$);

2. The longitudinal distance to the vehicle in the immediate front and in the lane to the left of that of the ego vehicle, d_{fl}, quantified as "close", "medium" or "far;"

3. The longitudinal distance to the vehicle in the immediate front and in the lane to the right of that of the ego vehicle, d_{fr}, quantified as "close", "medium" or "far;"

4. The longitudinal distance to the vehicle in the immediate rear and in the lane to the left of that of the ego vehicle, d_{rl}, quantified as "close", "medium" or "far;"

5. The longitudinal distance to the vehicle in the immediate rear and in the lane to the right of that of the ego vehicle, d_{rr}, quantified as "close", "medium" or "far;"

6. The longitudinal motion of the vehicle in the immediate front of and in the same lane with the ego vehicle relative to the ego vehicle, v_{fc}, quantified as "approaching" if d_{fc} is decreasing, "stable" if d_{fc} is not changing (within a variation tolerance), and "moving away" if d_{fc} is increasing;

7. The relative longitudinal motion of the vehicle in the immediate front and in the lane to the left of that of the ego vehicle, v_{fl}, quantified as "approaching", "stable" or "moving away;"

8. The relative longitudinal motion of the vehicle in the immediate front and in the lane to the right of that of the ego vehicle, v_{fr}, quantified as "approaching", "stable" or "moving away;"

9. The relative longitudinal motion of the vehicle in the immediate rear and in the lane to the left of that of the ego vehicle, v_{rl}, quantified as "approaching", "stable" or "moving away;"

10. The relative longitudinal motion of the vehicle in the immediate rear and in the lane to the right of that of the ego vehicle, v_{rr}, quantified as "approaching", "stable" or "moving away;"

11. The lane of the ego vehicle, $l_{\text{ego}} \in \{1, \ldots, n_{\text{lane}}\}$.

A case where there is no vehicle in a particular position will be considered by the ego vehicle to be the same as the case where there is a vehicle in that position that is "far" and "moving away."

An observation o is a tuple of measurements of these quantities (in discrete values). Since each quantity can take 3 different values except for the last one, which can take n_{lane} different values, the set of values of o, i.e., the observation space Ω^{h}, has $3^{10} \times n_{\text{lane}}$ elements. It is also clear from the above observation rules that for any particular vehicle $i \in \mathcal{V}$, the observed o_i by its driver is uniquely determined given the current traffic state $\mathbf{s} = (s_1, s_2, \ldots, s_{n_{\text{v}}})$. This defines the observation map $O^{\text{h}} : \mathcal{S}^{\text{h}} \to \Omega^{\text{h}}$ of the driver model.

Such a discrete representation of driver observations is motivated by the "logic-based" driver models [82]. Larger observation spaces with finer meshes can be treated in a similar way. Further discussions on the employment of a discrete observation space can be found in [83, 84].

1.2.4 Reward function

A driver typically has the following considerations during highway driving: 1) maximizing safety, e.g., not having a car crash, 2) minimizing the travel time to her destination, 3) keeping a reasonable headway from preceding vehicles to increase both safety and comfort, and 4) minimizing her driving effort. These considerations are expressed by the following reward function to maximize,

$$R^{\text{h}} = w_1 \hat{c} + w_2 \hat{v} + w_3 \hat{h} + w_4 \hat{e}, \tag{1.9}$$

where $w_{1,2,3,4} > 0$ are weights, and the terms $\hat{c}$, $\hat{v}$, $\hat{h}$ and $\hat{e}$ are explained below:

$\hat{c}$ **(collision avoidance)**: We define a "collision zone" (c-zone) for each vehicle, which is a rectangular area over-bounding the geometric contour of the vehicle with a safety margin. An overlap of two vehicles' c-zones indicates a danger of collision. Therefore, the term $\hat{c}$ takes a value of -1 if the ego vehicle's c-zone overlaps with the c-zone of any of the other vehicles, and a value of 0 otherwise.

$\hat{v}$ **(velocity):** The term $\hat{v}$ takes values according to

$$\hat{v} = \frac{v_x - v_{x,\text{nominal}}}{a_x^1}, \quad v_{x,\text{nominal}} = \frac{v_{x,\min} + v_{x,\max}}{2}, \tag{1.10}$$

where v_x denotes the ego vehicle's longitudinal velocity. Here, the reason for the division by a_x^1 is to make this term to be of the same order of magnitude as the other terms, to facilitate the design of weights.

$\hat{h}$ **(headway):** The term $\hat{h}$ takes values according to

$$\hat{h} = \begin{cases} -1 & \text{if} \quad d_{\text{fc}} = \text{“close,”} \\ 0 & \text{if} \quad d_{\text{fc}} = \text{“medium,”} \\ 1 & \text{if} \quad d_{\text{fc}} = \text{“far.”} \end{cases} \tag{1.11}$$

$\hat{e}$ **(effort):** The term $\hat{e}$ takes a value of 0 if the driver's action is "maintain," a value of e_{hard} if the driver's action is either "hard accelerate" or "hard decelerate," and a value of e_{nominal} otherwise, where $e_{\text{hard}} < e_{\text{nominal}} < 0$. This term discourages the driver from making unnecessary maneuvers. In particular, a higher penalty e_{hard} discourages the driver from unnecessarily applying "hard accelerate" and "hard decelerate." But in the case where a collision cannot be avoided by the other actions, the driver may apply "hard accelerate" or "hard decelerate" to enforce safety.

The weights $w_{1,2,3,4}$ may be tuned depending on the aggressiveness of the driver, but the following relation should be kept,

$$w_1 \gg w_2, w_3, w_4, \tag{1.12}$$

which expresses the fact that safety is typically considered to be most important by a driver.

1.2.5 Constraints

In certain situations, a lane change will be highly likely to cause safety issues such as collisions. Such lane changes are undesirable from a safety perspective, and are also irrational from a human-driver modeling perspective. To eliminate such improper lane changes, we impose the following hard constraints, which make certain actions unavailable in certain

situations:

1. If there is a vehicle that is in the lane to the left of that of the ego vehicle and is either in a side-by-side position with the ego vehicle or is "close" and "approaching," then the ego vehicle cannot "change lanes to the left;"

2. If there is a vehicle that is in the lane to the right of that of the ego vehicle and is either in a side-by-side position with the ego vehicle or is "close" and "approaching," then the ego vehicle cannot "change lanes to the right."

Here, two vehicles are viewed as side-by-side if the projections of their c-zones onto the longitudinal direction overlap with each other. From a practical perspective, eliminating these improper lane changes through the above hard constraints can be more effective than through penalties in the reward function. This treatment can also reduce the time needed for the RL algorithm introduced in Section 1.2.7 to converge when solving for a driving policy.

In addition, to avoid vehicles driving between lanes, once a lane change gets started, it must continue to completion. This means once an action of "change lanes to the left" or "change lanes to the right" has been chosen to apply, the same action must be consecutively applied for $\frac{t_l}{\Delta t}$ steps until the vehicle gets to the center of the target lane.

1.2.6 Level-k decision making

As has been introduced in Section 1.1.3, a level-k decision maker assumes all agents in a decision-making scenario but herself are level-$(k-1)$ decision makers, and makes her own decisions in a way that is optimal under this assumption. Recall that a model representing the dynamics of the traffic scenario has been defined in (1.7) by the deterministic function $\bar{T}^{\mathrm{h}} : \mathcal{S}^{\mathrm{h}} \times (\mathcal{A}^{\mathrm{h}})^{n_{\mathrm{v}}}$. For any vehicle in the traffic scenario, an observation function $O^{\mathrm{h}} : \mathcal{S}^{\mathrm{h}} \to \Omega^{\mathrm{h}}$ has also been defined by the observation rules in Section 1.2.3. We now assume a level-$(k-1)$ driving policy, $\pi^{\mathrm{h},k-1}$, has been available, which is a stochastic map from Ω^{h} to $\mathcal{A}^{\mathrm{h}}$ in the form of (1.4). In this case, to a level-k driver/vehicle, the traffic scenario evolves in a stochastic manner as in (1.2). In particular, the transition map, $T^{\mathrm{h},k-1}$, is uniquely determined by the functions $\bar{T}^{\mathrm{h}}$, O^{h}, and the stochastic decision rules

$$a_i \sim \pi^{\mathrm{h},k-1}(o_i, \cdot), \tag{1.13}$$

where $i \in \mathcal{V}$ designates every vehicle in the traffic scenario except for the level-k ego vehicle. Then, we assume a level-k driver makes decisions according to a policy, $\pi^{\mathrm{h},k}$, that

maximizes *the average reward per unit of time over a long run* associated with an initial state $\mathbf{s}_0$, defined as

$$\bar{R}^{\mathrm{h},k}_{\mathbf{s}_0}(\pi) = \lim_{t_{\max}\to\infty} \frac{\mathbb{E}\left\{\sum_{t=0}^{t_{\max}-1} R^{\mathrm{h}}_t \,\middle|\, \mathbf{s}_0, T^{\mathrm{h},k-1}, \pi\right\}}{t_{\max}}, \tag{1.14}$$

for all initial states $\mathbf{s}_0 \in \mathcal{S}^{\mathrm{h}}$. In (1.14), R^{h}_t represents the reward (1.9) obtained at time t, and $\mathbb{E}\left\{\cdot \,\middle|\, \mathbf{s}_0, T^{\mathrm{h},k-1}, \pi\right\}$ represents the expected value of $\{\cdot\}$ when the initial state is $\mathbf{s}_0$, the state transitions follow the stochastic map $T^{\mathrm{h},k-1}$ corresponding to the scenario where all other vehicles make decisions according to the level-$(k-1)$ decision rule (1.13), and the ego driver makes decisions according to a particular policy π at every time instant. We assume the transition model $T^{\mathrm{h},k-1}$ is *unichain* [85], and in this case the average reward (1.14) becomes state-independent, i.e.,

$$\bar{R}^{\mathrm{h},k}(\pi) := \bar{R}^{\mathrm{h},k}_{\mathbf{s}_0}(\pi) = \bar{R}^{\mathrm{h},k}_{\mathbf{s}'_0}(\pi), \quad \forall\, \mathbf{s}_0, \mathbf{s}'_0 \in \mathcal{S}^{\mathrm{h}}. \tag{1.15}$$

In brief, a level-k driving policy, $\pi^{\mathrm{h},k}$, can be characterized by the following property,

$$\pi^{\mathrm{h},k} \in \arg\max_{\pi}\ \bar{R}^{\mathrm{h},k}(\pi), \tag{1.16}$$

where $k = 1, 2, \ldots$ can be any positive integer.

It can be seen that to construct higher-level policies, one needs to start from defining a level-0 policy, $\pi^{\mathrm{h},0}$, and can then solve for level-k policies, $\pi^{\mathrm{h},k}$, for $k = 1, 2, \ldots$ in a sequential manner according to (1.13)-(1.16). In this study, we consider a deterministic level-0 policy $\pi^{\mathrm{h},0} : \Omega^{\mathrm{h}} \to \mathcal{A}^{\mathrm{h}}$ defined as follows, which represents a conservative driving style:

$$\pi^{\mathrm{h},0}(o) = \begin{cases} \text{“decelerate,”} & \text{if} \quad d_{\mathrm{fc}} = \text{“medium” and } v_{\mathrm{fc}} = \text{“approaching”} \\ & \text{or} \quad d_{\mathrm{fc}} = \text{“close” and } v_{\mathrm{fc}} = \text{“stable,”} \\ \text{“hard decelerate,”} & \text{if} \quad d_{\mathrm{fc}} = \text{“close” and } v_{\mathrm{fc}} = \text{“approaching,”} \\ \text{“maintain,”} & \text{otherwise.} \end{cases} \tag{1.17}$$

In the next subsection, we will introduce a computational approach based on RL to generating level-k driving policies.

1.2.7 Obtaining level-k policies using reinforcement learning

Recall that to obtain a level-k driving policy, we first assign a level-$(k-1)$ policy, $\pi^{\mathrm{h},k-1}$, to all vehicles in the traffic scenario except for the ego vehicle. After this, to the ego driver/vehicle, the traffic scenario evolves according to a stochastic transition model $T^{\mathrm{h},k-1}$ in the form of (1.4). In particular, the transition model $T^{\mathrm{h},k-1}$ has the Markov property, but the ego driver has to make decisions based on only partial information about the traffic state $\mathbf{s}$ contained in her observation o. In other words, the ego driver needs to solve a Partially Observable Markov Decision Process (POMDP) problem.

We employ a Reinforcement Learning (RL) algorithm for POMDP problems developed in [86] to train an optimal policy for such an ego driver, which, by definition, is a level-k driving policy $\pi^{\mathrm{h},k}$. Two major advantages of this RL algorithm are: 1) It is a model-free algorithm. This means we can develop a simulation code based on the EOMs (1.6), the observation rules in Section 1.2.3, and the decision rules (1.13) to generate state transitions, and rely on this simulation code to train the policy, instead of having to know $T^{\mathrm{h},k-1}$ explicitly. Note that although we know the transition model $T^{\mathrm{h},k-1}$ is uniquely determined by these EOMs, observation and decision rules, we have not computed an explicit expression of $T^{\mathrm{h},k-1}$. 2) For a finite-space POMDP problem like the one treated by us, this algorithm guarantees convergence to a policy that is at least locally optimal in terms of maximizing the average reward (1.15). For the sake of completeness, we include the proof of such a convergence guarantee in Appendix A.

This RL algorithm exploits two steps to gradually improve the policy until convergence, including 1) "policy evaluation," where observation-action pairs (o,a) are assigned values based on the cumulative rewards they gain, and 2) "policy improvement," where probabilities of applying actions that have higher reward values are increased. These two steps are further explained below:

Step 1. Estimate the value functions for observations, $V(o|\pi_t)$, and for observation-action pairs, $Q(o,a|\pi_t)$, corresponding to the current policy π_t using the following equations:

$$
\begin{aligned}
\beta_t^{\mathrm{o}}(o) &= \left(1-\frac{\chi_t^{\mathrm{o}}(o)}{K_t^{\mathrm{o}}(o)}\right)\gamma_t\,\beta_{t-1}^{\mathrm{o}}(o)+\frac{\chi_t^{\mathrm{o}}(o)}{K_t^{\mathrm{o}}(o)},\\
V(o|\pi_t) &= \left(1-\frac{\chi_t^{\mathrm{o}}(o)}{K_t^{\mathrm{o}}(o)}\right)V(o|\pi_{t-1})+\beta_t^{\mathrm{o}}(o)\left[R_t^{\mathrm{h}}-\bar{R}^{\mathrm{h},k}(\pi_t)\right],\\
\beta_t^{\mathrm{a}}(o,a) &= \left(1-\frac{\chi_t^{\mathrm{a}}(o,a)}{K_t^{\mathrm{a}}(o,a)}\right)\gamma_t\,\beta_{t-1}^{\mathrm{a}}(o,a)+\frac{\chi_t^{\mathrm{a}}(o,a)}{K_t^{\mathrm{a}}(o,a)},\\
Q(o,a|\pi_t) &= \left(1-\frac{\chi_t^{\mathrm{a}}(o,a)}{K_t^{\mathrm{a}}(o,a)}\right)Q(o,a|\pi_{t-1})+\beta_t^{\mathrm{a}}(o,a)\left[R_t^{\mathrm{h}}-\bar{R}^{\mathrm{h},k}(\pi_t)\right].
\end{aligned}
\tag{1.18}
$$

In (1.18), the subscript t refers to the time step, and the superscript "o" or "a" of a function indicates whether this function is associated with observations or observation-action pairs. The function χ is an indicator function, taking a value of 1 if the observation o or the observation-action pair (o, a) is visited at the current step t, and a value of 0 otherwise. Taking χ_t^{o} as an example, $\chi_t^{\mathrm{o}}(o) = 1$ for $o = o_t$, and $\chi_t^{\mathrm{o}}(o) = 0$ for all $o \neq o_t$. The function K counts the number of times each particular observation o or observation-action pair (o, a) has been visited. Taking K_t^{o} as an example, $K_t^{\mathrm{o}}(o) = K_{t-1}^{\mathrm{o}}(o) + 1$ for $o = o_t$, and $K_t^{\mathrm{o}}(o) = K_{t-1}^{\mathrm{o}}(o)$ for all $o \neq o_t$. The parameter γ_t is a time-dependent discount factor, taking values in $(0, 1)$ and converging to 1 as t goes to infinity. Moreover, R_t^{h} represents the reward (1.9) obtained at the current step t, and $\bar{R}^{\mathrm{h},k}(\pi_t)$ represents the average reward defined in (1.14) and (1.15) evaluated at the current policy π_t. In particular, although the true value of $\bar{R}^{\mathrm{h},k}(\pi_t)$ could be estimated by a Monte Carlo simulation of the current policy π_t, doing so would take a considerable amount of computational cost and make the overall algorithm inefficient. Therefore, in our implementation of (1.18), $\bar{R}^{\mathrm{h},k}(\pi_t)$ is replaced with the following incrementally updated estimate, making use of the fact that the policy is only slowly varying in time,

$$\tilde{R}^{\mathrm{h},k}(\pi_t) = \frac{t}{t+1}\tilde{R}^{\mathrm{h},k}(\pi_{t-1}) + \frac{1}{t+1}R_t^{\mathrm{h}} = \frac{\sum_{\tau=0}^{t} R_\tau^{\mathrm{h}}}{t+1}. \tag{1.19}$$

Step 2. Update the driving policy π_t according to the following equation:

$$\pi_{t+1}(o, a) = (1 - \varepsilon_t)\pi_t(o, a) + \varepsilon_t \hat{\pi}_t(o, a), \quad \forall\, (o, a) \in \Omega^{\mathrm{h}} \times \mathcal{A}^{\mathrm{h}}, \tag{1.20}$$

where $\varepsilon_t \in (0, 1)$ is a learning rate, typically chosen as a time-independent parameter, and $\hat{\pi}_t$ is a greedy policy maximizing

$$J_t(\pi, o) := \sum_{a \in \mathcal{A}^{\mathrm{h}}} \pi(o, a)\big(Q(o, a|\pi_t) - V(o|\pi_t)\big), \tag{1.21}$$

for all $o \in \Omega^{\mathrm{h}}$. In particular, for any o such that $\hat{a}_t(o) := \arg\max_{a \in \mathcal{A}^{\mathrm{h}}} Q(o, a|\pi_t)$ is unique, $\hat{\pi}_t$ satisfies $\hat{\pi}_t\big(o, \hat{a}_t(o)\big) = 1$.

By going through Steps 1 and 2 at every time step t, the driving policy π_t is gradually improved until convergence. The converged policy is, by definition, a level-k driving policy $\pi^{\mathrm{h},k}$. In our implementation, the convergence criterion is based on convergence of the estimated average reward (1.19), i.e., the absolute changes of (1.19) over the most recent certain number of steps being sufficiently small (smaller than a pre-specified threshold).

Specifically, our implementation of the above RL algorithm is presented as the follow-

ing Algorithms 1-2. The overall procedure to train level-k driving policies using RL is illustrated by Fig. 1.1.

Algorithm 1: Training Level-k Policies Using RL

Input : A level-$(k-1)$ driving policy, $\pi^{\mathrm{h},k-1}$
Output: A level-k driving policy, $\pi^{\mathrm{h},k}$

1 Initialize a new policy π as $\pi(o,a) = \frac{1}{|\mathcal{A}^{\mathrm{h}}|}$ for all $(o,a) \in \Omega^{\mathrm{h}} \times \mathcal{A}^{\mathrm{h}}$
2 Initialize the value function estimates as $V(o) = 0$ for all $o \in \Omega^{\mathrm{h}}$ and $Q(o,a) = 0$ for all $(o,a) \in \Omega^{\mathrm{h}} \times \mathcal{A}^{\mathrm{h}}$
3 **for** *episode* $= 1 :$ *maximum number of training episodes* **do**
4 Randomly select a number of vehicles in traffic $n_{\mathrm{v}} \in \{1, \ldots, n_{\mathrm{v,max}}\}$
5 Randomly initialize the states of all vehicles $\mathbf{s} = (s_1, s_2, \ldots, s_{n_{\mathrm{v}}})$
6 Assign the policy π that is being trained to the ego vehicle
7 Assign the level-$(k-1)$ policy $\pi^{\mathrm{h},k-1}$ to all other vehicles
8 Evaluate and improve π according to Algorithm 2
9 **end**
10 Assign each observation o that has been visited less than a threshold number of times a corresponding level-0 action according to (1.17), i.e., $\pi(o) = \pi^{\mathrm{h},0}(o)$
11 Set $\pi^{\mathrm{h},k} = \pi$.

Algorithm 2: Single Episode Simulation

1 **for** $t = 1 : t_{\max}$ **do**
2 **foreach** *vehicle* $i \in \mathcal{V}$ **do**
3 Obtain its observation o_i according to the observation rules in Section 1.2.3
4 Given o_i, determine an action a_i according to its assigned policy and (1.4)
5 Given a_i, update its state s_i according to (1.6)
6 **end**
7 **if** *training a policy* π **then**
8 Evaluate the reward (1.9) obtained by the ego vehicle at the current step
9 Update the value function estimates V and Q according to (1.18) and (1.19)
10 Update the policy π according to (1.20) and (1.21)
11 **end**
12 **if** *the ego vehicle is in a safety-violation state, i.e.,* $\hat{c} = -1$ **then**
13 End the simulation.
14 **end**
15 **end**

When initializing the states of all vehicles $\mathbf{s} = (s_1, s_2, \ldots, s_{n_{\mathrm{v}}})$ in Step 5 of Algorithm 1, we place all vehicles in the centers of lanes and impose a minimum initial separation distance $x_{\min}$ between vehicles that are in the same lane to avoid immediate safety violations. Under given road length $x_{\max}$, number of lanes n_{lane} and this minimum initial separation

distance $x_{\min}$, the maximum number of vehicles $n_{\mathrm{v,max}}$ is chosen such that when $n_{\mathrm{v,max}}$ vehicles are placed on the road, the road is near full capacity. We train the policy π using traffic environments with varying numbers of vehicles n_{v} to increase the opportunity for a large portion of the observation space to be visited sufficiently many times by the ego vehicle (so the ego vehicle can learn from these visits), and also to enable the policy π to handle various traffic densities. Meanwhile, for the observations that are still not visited enough after the maximum number of training episodes, the level-0 actions are assigned to them so that the vehicle can have a reasonable behavior when encountering these rare cases.

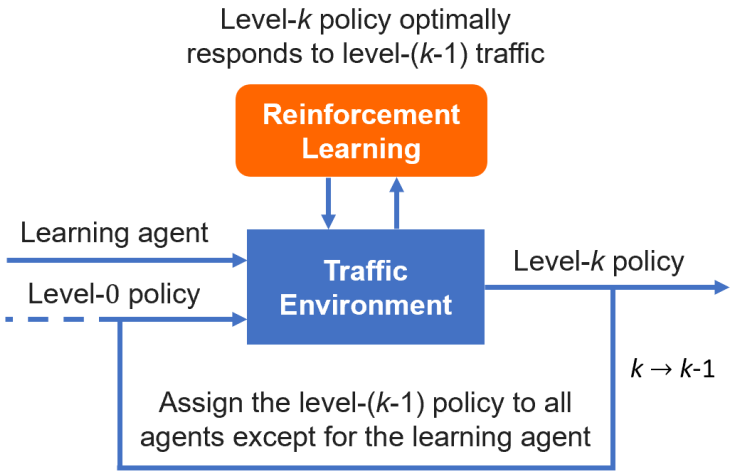


Figure 1.1: The procedure of training level-k driving policies using RL.

1.2.8 Simulation results of level-k driver models

The parameter values used to generate the simulation results in this section are summarized in Table 1.1.

1.2.8.1 Simulation of level-0 driver model

In Fig. 1.2, the red vehicle in the middle is a level-0 ego vehicle, and the yellow vehicles form the traffic environment. The red arrow attached to the red vehicle indicates its travel direction, and the arrow size indicates how fast the vehicle is traveling. The panel on the left represents a speedometer, and the steering wheel on the right indicates the lateral motion of the ego vehicle. The green box and the red box in the middle represent the gas pedal and

Table 1.1: Parameter values for highway simulations.

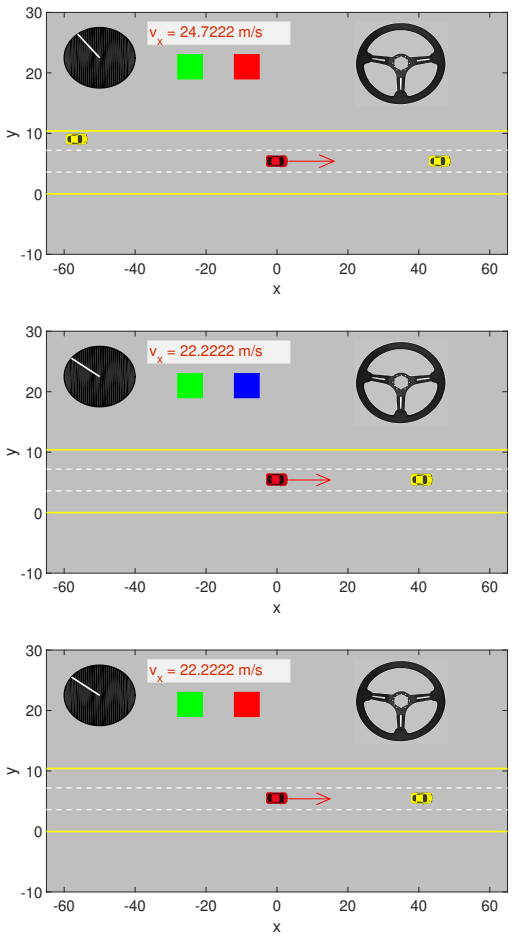
Variable(s)	Value(s)	Unit	Remarks
Δt	1	s	sampling period
$x_{\max}$	1	km	road length
n_{lane}	3		number of lanes
w_{lane}	3.6	m	lane width
$[v_{x,\min}, v_{x,\max}]$	$[62, 98]$	km/h	speed range
a_x^1	2.5	m/s^2	acceleration rate
a_x^2	5	m/s^2	hard acceleration rate
t_{lane}	2	s	lane change duration
d_{c}	21	m	"close" distance upper bound
d_{f}	42	m	"far" distance lower bound
d_{v}	63	m	maximum visibility distance
$w_{1,2,3,4}$	$10000, 5, 1, 1$		reward function weights
$(l_{\text{c}}, w_{\text{c}})$	$(6, 2)$	m	c-zone size
$e_{\text{nominal}}, e_{\text{hard}}$	$-1, -5$		action efforts
$x_{\min}$	30	m	minimum initial separation
$n_{\text{v,max}}$	30		maximum number of vehicles
$t_{\max}$	200		single episode length

the brake pedal, respectively, and turning blue indicates that the pedal is pressed. To better inspect the behavior of the ego vehicle, we attach the x-coordinate axis to the ego vehicle and the motions of the other vehicles can be tracked by their relative positions with respect to the red ego vehicle.

In Fig. 1.2(a), a yellow vehicle in front of the red vehicle is "far" and "approaching" (because the red vehicle is faster). At this moment, neither the gas pedal nor the brake pedal is pressed. In Fig. 1.2(b), this yellow vehicle enters the "medium" distance range, and according to the level-0 policy (1.17), the red vehicle starts to decelerate. In Fig. 1.2(c), the red vehicle gets to a lower speed, and the yellow vehicle is now "stable."

1.2.8.2 Training and simulation of level-k driver models

Fig. 1.3 shows the time history of the estimated average reward (1.19) throughout the training process of level-1 policy and that of level-2 policy. It can be seen that in both cases the estimated average reward $\tilde{R}^{\text{h},k}(\pi_t)$ gradually increases and converges, which implies



(a) 40[s]

(b) 42[s]

(c) 44[s]

Figure 1.2: Level-0 simulation results: Plots (a)-(c) show snapshots of the simulation at 40[s], 42[s] and 44[s], respectively.

the gradual improvement and convergence of the policy π_t. We train policies up to level-2 because, on the one hand, it is shown in experimental studies [37] that level-3 and higher decision makers are rarely encountered in human interactions, and on the other hand, level-0, 1 and 2 policies can already model a sufficiently rich set of driving styles for our targeted use cases.

Fig. 1.4 shows a simulated trajectory of a level-1 ego vehicle in a level-0 traffic environment (i.e., all other vehicles are level-0). In Fig. 1.4(a), a yellow vehicle in front of the red ego vehicle is rapidly "approaching." To avoid a rear-end collision, the red vehicle slows down a bit and begins to steer in Fig. 1.4(b). In Fig. 1.4(c), the red vehicle starts to move into the lane on its left. This lane change takes 2[s] to complete, after which there is another yellow vehicle in the new lane approaching the red vehicle from its front, shown in Fig. 1.4(d). In response, the red vehicle decelerates to an even lower speed to keep a stable

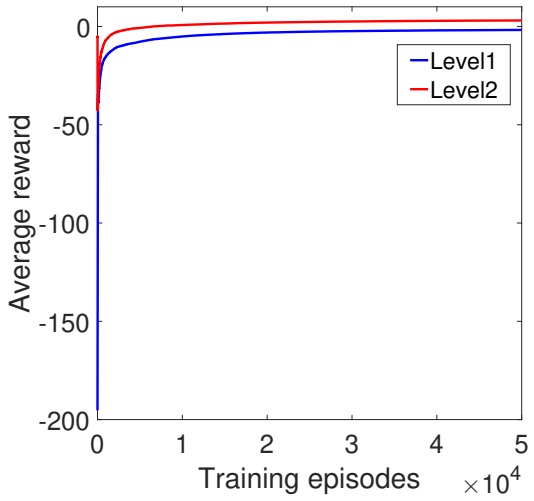


Figure 1.3: Evolution of the average reward during level-1 and level-2 training.

distance from that yellow vehicle, which is shown in Fig. 1.4(e) and (f). It can be seen that the above maneuver sequence of this level-1 ego vehicle represents a realistic response to the traffic situation.

Similarly, Fig. 1.5 shows a simulated trajectory of a level-2 ego vehicle in a level-1 traffic environment. In Fig. 1.5(a), the yellow vehicle in the middle lane is "approaching" the red ego vehicle, while the yellow vehicle in the left lane is "moving away." In this case, the red vehicle decides to move into the left lane. However, a few seconds later, the yellow vehicle in the middle lane also makes a lane change to the left, shown in Fig. 1.5(b), which is attributed to the traffic in its front (not shown in this figure). In response, the red vehicle slows down to let that yellow vehicle cut in. In Fig. 1.5(c), there is no vehicle immediately in front of the red ego vehicle. Under such circumstances, a level-1 ego vehicle would accelerate. However, after taking into account the fact that the longitudinal distance between the ego vehicle and the yellow vehicle in the middle lane is very small at this moment, this level-2 ego vehicle decides to maintain its current speed. Indeed, in this scenario, if the red vehicle decides to accelerate but at the same time the yellow vehicle in the middle lane starts a sudden lane change to the left (which is possible since the yellow vehicle is level-1), a collision between these two vehicles would become inevitable. To avoid such a potential safety violation, the level-2 ego vehicle maintains its current speed. In Fig. 1.5(d), another yellow vehicle in front is moving into the red vehicle's lane, which forces the red vehicle to decelerate.

As a validation of our level-k driver models, we also compare the behavior of our models against the behavior of real human drivers in traffic data. The data we use is from

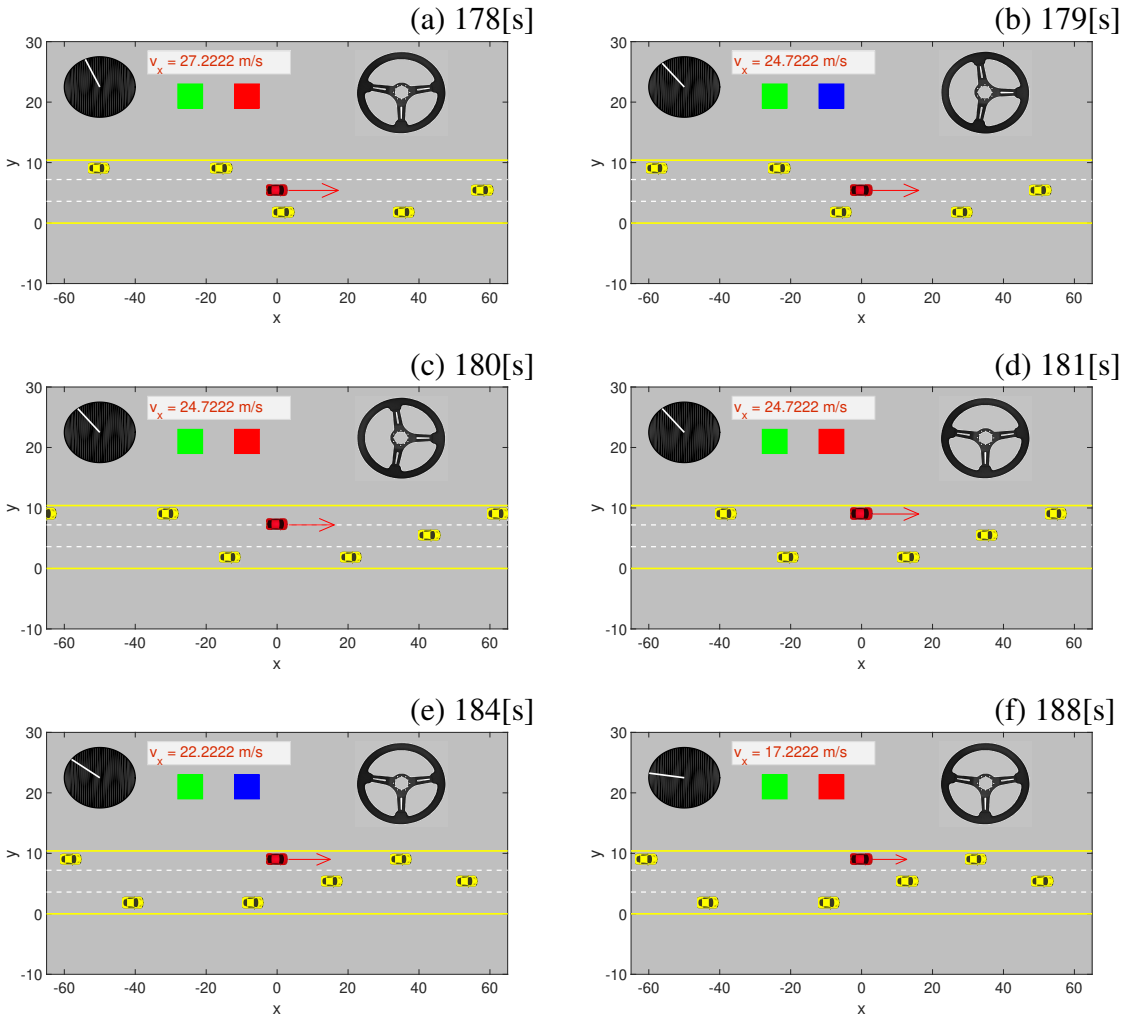


Figure 1.4: Simulation of a level-1 ego vehicle in a level-0 traffic environment: Plots (a)-(f) show snapshots of the simulation at 178[s], 179[s], 180[s], 181[s], 184[s], 188[s], respectively.

the Next Generation Simulation (NGSIM) program [87]. For a given trajectory of traffic data, we replace one vehicle in the data with a level-k vehicle. This level-k vehicle will make decisions and take actions in response to the other vehicles in the data. We then compare the future trajectory of this level-k vehicle against the future trajectory of the original vehicle in the data. In the scenario of Fig. 1.6(a), the level-1 red vehicle and the original vehicle in the data (indicated by the blue-shaded rectangle) both make a lane change from the right lane to the left lane, because the preceding vehicle in the right lane is rapidly "approaching." In the scenario of Fig. 1.6(b), the level-2 red vehicle and the original vehicle in the data both make a lane change from the middle lane to the right lane, because the preceding vehicle in the middle lane is "approaching" while the preceding vehicle in

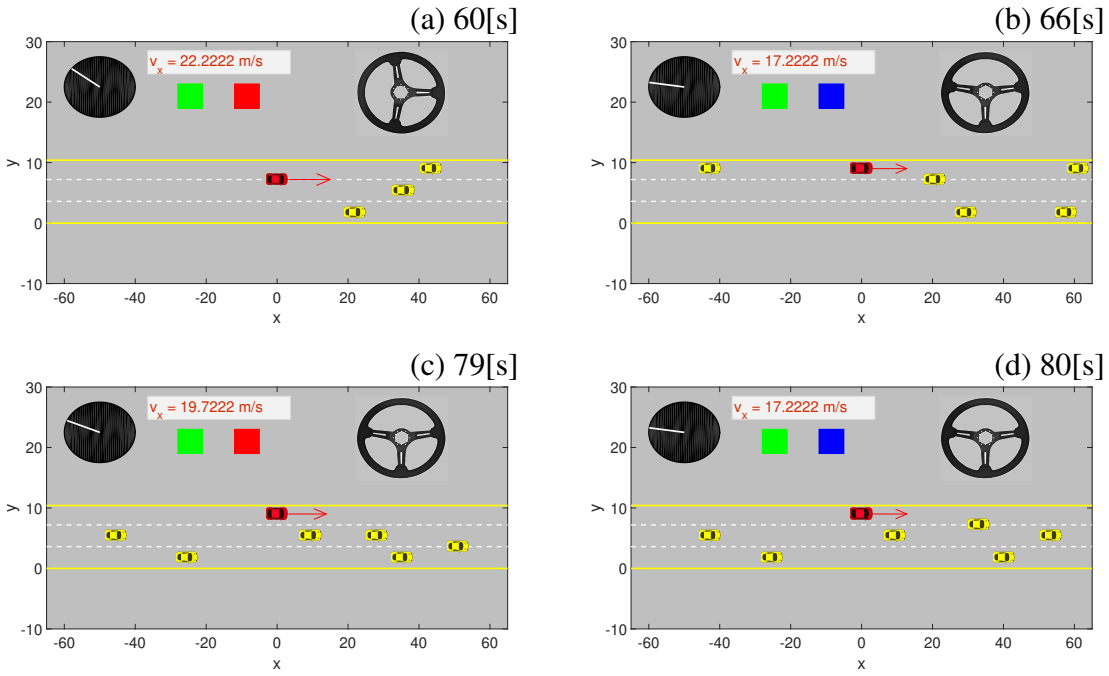


Figure 1.5: Simulation of a level-2 ego vehicle in a level-1 traffic environment: Plots (a)-(d) show snapshots of the simulation at 60[s], 66[s], 79[s] and 80[s], respectively.

the right lane is "moving away."

We also statistically assess our level-k driver models by looking at their safety violation rates. Here, the "safety violation rate" is counted as the percentage of 10000 simulation episodes which end within $t_{\max} = 200$ seconds due to the occurrence of a safety-violation state of the ego vehicle (see Steps 12-14 of Algorithm 2). Fig. 1.7(a) shows the safety violation rates of a level-1 ego vehicle operating in a level-0 traffic environment and of a level-2 ego vehicle operating in a level-1 traffic environment with varying numbers of vehicles $n_{\rm v}$. It can be seen that the level-2 ego vehicle experiences higher safety violation rates than the level-1 ego vehicle in this experiment. One important reason contributing to this is that the level-1 traffic environment (where the level-2 ego vehicle is tested) represents a much more complex and challenging environment than the level-0 traffic environment (where the level-1 ego vehicle is tested), since the former is composed of level-1 vehicles which can accelerate and change lanes while the latter is composed of level-0 vehicles which drive in a fairly conservative manner (see the level-0 policy (1.17)).

Fig. 1.7(b) shows the safety violation rates of level-0, 1, and 2 vehicles when they operate in a same traffic environment. The traffic environment is modeled by mixing 10% level-0, 60% level-1, and 30% level-2 vehicles. It can be seen that level-0 has the lowest safety violation rates in this experiment. This is attributed to the fact that the level-0 policy

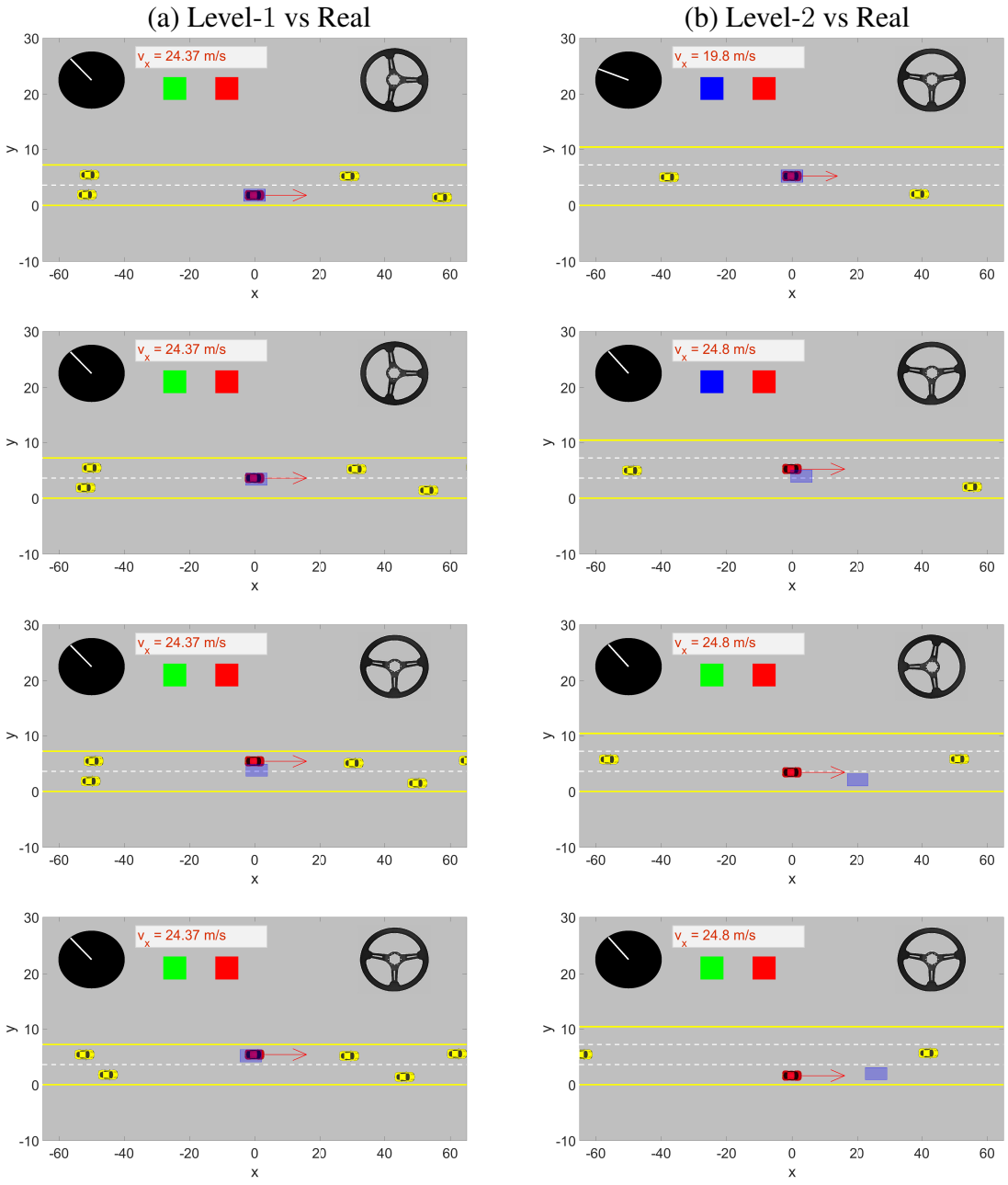


Figure 1.6: Validation of level-1 and level-2 driver models with traffic data. The red vehicle uses a level-k policy, and the blue-shaded rectangle represents a real vehicle in traffic data.

(1.17) represents a fairly conservative driving style, e.g., never accelerating or changing lanes. Comparing the rates of level-1 and level-2, we can see that a level-2 vehicle has a better safety performance than a level-1 vehicle in this mixed traffic. This is because the level-2 policy is trained in a level-1 traffic environment, which is more complex and also more aggressive than the level-0 traffic environment where the level-1 policy is trained, and

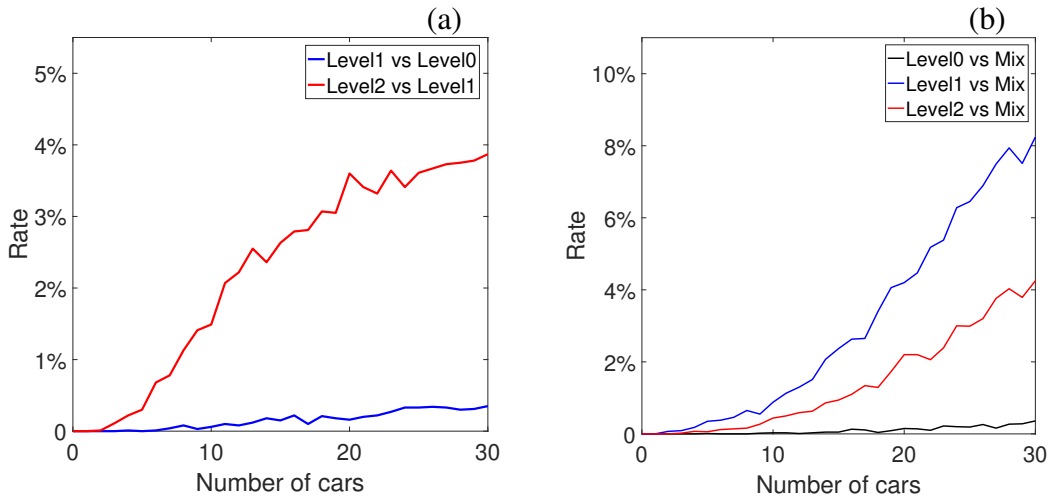


Figure 1.7: Safety violation rates of (a) a level-k ego vehicle in a level-$(k-1)$ traffic environment, and (b) level-0, 1 and 2 ego vehicles in a same traffic environment, which is composed of a mixture of 10% level-0, 60% level-1 and 30% level-2 vehicles.

as a result, a level-2 vehicle can maintain its safety better than a level-1 vehicle when both operating in an unfamiliar traffic environment.

1.3 Leader-follower based modeling of driver interactions at intersections

1.3.1 Parameterized intersection

Intersections in real-world road networks are observed to have varied layouts (e.g., number of intersecting road arms) and geometries (e.g., angles between road arms and lane width). To enable the modeling of driver/vehicle interactions at various intersections, we first introduce a parameterized intersection model. In particular, we model an intersection using the following set of parameters,

$$\big(n_{\rm r}, \{m_{\rm f}^i\}_{i=1}^{n_{\rm r}}, \{m_{\rm b}^i\}_{i=1}^{n_{\rm r}}, \{\phi^i\}_{i=1}^{n_{\rm r}}, w_{\rm lane}\big), \tag{1.22}$$

where $n_{\rm r} \in \{3,4,5\}$ denotes the number of road arms, $m_{\rm f}^i \in \{0,1,2,\dots\}$ and $m_{\rm b}^i \in \{0,1,2,\dots\}$ denote, respectively, the numbers of forward and backward lanes[1] of the ith arm, $i \in \{1,\dots,n_{\rm r}\}$, ϕ^i is the counter-clockwise angle of the ith arm with respect to the x-

[1]A forward lane (a backward lane) is a lane for traffic entering the intersection (moving away from the intersection).

axis, and w_{lane} is the lane width. We consider three-, four- and five-way intersections ($n_{\text{r}} \in \{3, 4, 5\}$) because they are most common in real-world road networks. Note that $m_{\text{f}}^i = 0$ or $m_{\text{b}}^i = 0$ (but not simultaneously equal to 0) represents one-way roads, which are allowed in our model. We assume that the road centerlines[2] of all the road arms intersect at the same point, which is referred to as the intersection center with coordinates $(x, y) = (0, 0)$.

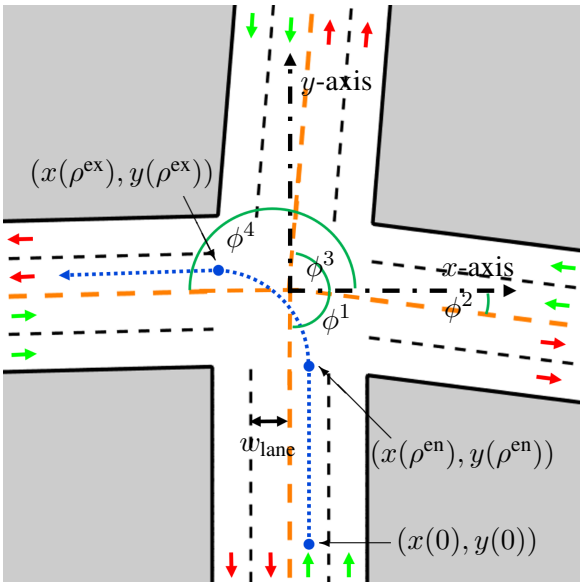


Figure 1.8: A four-way intersection modeled by (1.22) and (1.23). The orange dashed lines are the road centerlines[3], the black dashed lines are the lane markings that separate the lanes of traffic moving in the same directions, the black solid lines are the road boundaries, and the shaded polygons are off-road regions.

Given a set of parameters (1.22), the lane markings and road boundaries of the ith arm are modeled according to

$$x \sin(\phi^i) - y \cos(\phi^i) + \frac{j w_{\text{lane}}}{2} = 0, \tag{1.23}$$

with $j \in \{-2m_{\text{b}}^i, \ldots, 2m_{\text{f}}^i\}$. When $j = 2m_{\text{f}}^i$ (resp. $j = -2m_{\text{b}}^i$), (1.23) corresponds to the right-hand-side road boundary when looking in the forward direction (resp. in the backward direction); when $j \in 2\{m_{\text{f}}^i - 1, \ldots, 1\}$ (resp. $j \in 2\{-m_{\text{b}}^i + 1, \ldots, -1\}$), (1.23) corresponds to a lane marking that separates two lanes of traffic moving in the forward direction (resp. in the backward direction); when $j = 0$, (1.23) corresponds to the road centerline; and when

[2]The road centerlines are the lane markings that separate the lanes of traffic moving in the opposite directions, not necessarily the centerlines in the geometric sense, e.g., for a road arm with different numbers of forward and backward lanes.

$j \in 2\{m_{\mathrm{f}}^i, \ldots, 1\} - 1$ (resp. $j \in 2\{-m_{\mathrm{b}}^i, \ldots, -1\} + 1$), (1.23) corresponds to the geometric center of a forward lane (resp. backward lane). We also assign an entrance point $(x^{\mathrm{en}}, y^{\mathrm{en}})$ to each forward lane and an exit point $(x^{\mathrm{ex}}, y^{\mathrm{ex}})$ to each backward lane, both located in the center of the corresponding lanes and indicating the entrance/exit of the intersection, which will be used when assigning the leader/follower roles to vehicles (see Section 1.3.4.1).

Fig. 1.8 shows an example of four-way intersection modeled with (1.22) and (1.23). We note that the above model corresponds to the right-hand traffic [88]. To model intersections in the context of left-hand traffic requires straightforward modifications.

1.3.2 Vehicle kinematics

As before, we represent a vehicle using a rectangle, referred to as the "collision zone" (c-zone), which bounds the vehicle's geometric contour projected onto the ground. The c-zone of a vehicle is characterized by a 5-tuple, $(x, y, \theta, l_{\mathrm{c}}, w_{\mathrm{c}})$, where (x, y) are the coordinates of its geometric center, θ is the vehicle's heading angle (the counter-clockwise angle of the vehicle's heading direction with respect to the x-axis), and l_{c} (w_{c}) is the length (width) of the rectangle.

We assume that a vehicle plans a path P before entering the intersection and follows this pre-planned path to pass through the intersection [89]. When vehicles have conflicts, they adjust their speeds along the paths to resolve the conflicts. In particular, a path P is a smooth curve starting from an initial point $(x^{\mathrm{ini}}, y^{\mathrm{ini}})$, which is located in the center of the vehicle's origin lane[3], passing through the entrance point $(x^{\mathrm{en}}, y^{\mathrm{en}})$ of the origin lane and the exit point $(x^{\mathrm{ex}}, y^{\mathrm{ex}})$ of the vehicle's target lane[4], and ending at a terminal point $(x^{\mathrm{term}}, y^{\mathrm{term}})$, which is located in the center of the target lane.

For any point (x, y) on the curve P, we use ρ to denote the arc length of the curve segment from $(x^{\mathrm{ini}}, y^{\mathrm{ini}})$ to (x, y). This way, $(x(0), y(0)) = (x^{\mathrm{ini}}, y^{\mathrm{ini}})$, i.e., the initial point is the location of the vehicle when its traveled distance along the path is zero. Also, we denote the ρ values for the entrance point $(x^{\mathrm{en}}, y^{\mathrm{en}})$ by ρ^{en} and for the exit point $(x^{\mathrm{ex}}, y^{\mathrm{ex}})$ by ρ^{ex}, i.e., $(x^{\mathrm{en}}, y^{\mathrm{en}}) = (x(\rho^{\mathrm{en}}), y(\rho^{\mathrm{en}}))$ and $(x^{\mathrm{ex}}, y^{\mathrm{ex}}) = (x(\rho^{\mathrm{ex}}), y(\rho^{\mathrm{ex}}))$. This means the vehicle is right entering (resp. exiting) the intersection when its traveled distance along the path P is ρ^{en} (resp. ρ^{ex}). Then, we define $\Delta\rho^{\mathrm{en}} = \rho^{\mathrm{en}} - \rho$ and $\Delta\rho^{\mathrm{ex}} = \rho^{\mathrm{ex}} - \rho$ so that $\Delta\rho^{\mathrm{en}} < 0$ (resp. $\Delta\rho^{\mathrm{ex}} < 0$) means the vehicle has entered (resp. exited) the intersection.

[3]The lane in which the vehicle is driving before entering the intersection.

[4]The lane to which the vehicle is going after exiting the intersection.

To facilitate following exposition, we write the path P as

$$P: \ \mathbb{R} \to \mathbb{R}^2, \qquad \rho \mapsto \begin{bmatrix} x(\rho) \\ y(\rho) \end{bmatrix}. \tag{1.24}$$

Analytical expressions for the curve (1.24) and for $\Delta\rho^{\text{en}}$ and $\Delta\rho^{\text{ex}}$ as functions of the intersection parameters (1.22) are given in Appendix B. We note that although there are other path models in the literature [3], the above model is simple and suitable for our purpose.

Under the assumption that a vehicle can follow its pre-planned path perfectly, the dynamic behavior of a vehicle is modeled by the following discrete-time EOMs:

$$\begin{aligned} \rho(t+1) &= \rho(t) + v(t)\,\Delta t, \\ v(t+1) &= v(t) + a(t)\,\Delta t, \end{aligned} \tag{1.25}$$

where t denotes the discrete time, $v(t) \in [v_{\min}, v_{\max}]$ and $a(t)$ denote the vehicle's speed and acceleration at t, and Δt is the sampling period.

Using (1.24), the vehicle's location (x, y) and heading angle θ can be written as functions of ρ. In particular,

$$\theta(\rho) = \lim_{h \to 0^+} \operatorname{arctan2}\Big(y(\rho+h) - y(\rho), x(\rho+h) - x(\rho)\Big) = \operatorname{arctan2}\Big(\frac{\mathrm{d}y}{\mathrm{d}\rho}, \frac{\mathrm{d}x}{\mathrm{d}\rho}\Big). \tag{1.26}$$

We now collect all relevant variables and define the state of a vehicle as the following 8-tuple,

$$s(t) = \big(P, \rho(t), v(t), x(\rho(t)), y(\rho(t)), \theta(\rho(t)), \Delta\rho^{\text{en}}(t), \Delta\rho^{\text{ex}}(t)\big). \tag{1.27}$$

To adjust speeds along the path, we assume a vehicle has a finite number of acceleration levels to choose at each time step, i.e.,

$$a(t) \in \mathcal{A}^{\text{i}} = \big\{a^1, a^2, \ldots, a^{|\mathcal{A}^{\text{i}}|}\big\}, \quad \forall\, t. \tag{1.28}$$

Fig. 1.9 illustrates vehicle kinematics at a typical two-lane four-way intersection modeled using (1.24)-(1.27). As before, we associate a variable with a subscript i to indicate that this variable corresponds to vehicle i. For modeling an intersection scenario with n_{v} vehicles, the collection of all vehicles' states, $\mathbf{s}(t) = (s_1(t), s_2(t), \ldots, s_{n_{\text{v}}}(t))$, defines the traffic state, the evolution of which follows

$$\mathbf{s}(t+1) = \bar{T}^{\text{i}}\big(\mathbf{s}(t), a_1(t), a_2(t), \ldots, a_{n_{\text{v}}}(t)\big), \tag{1.29}$$

where $\bar{T}^{\mathrm{i}}$ is a deterministic map defined by a concatenation of all individual vehicles' kinematics models (1.24)-(1.27). Moreover, we assume a driver can observe the state of her own vehicle and the states of all other vehicles at the intersection, i.e., $o_i(t) = (s_i(t), \mathbf{s}_{-i}(t))$. This defines the observation map $O^{\mathrm{i}} : \mathcal{S}^{\mathrm{i}} \to \Omega^{\mathrm{i}}$ of the driver model, which in this case is essentially an identity map. This assumption will be relaxed in Section 1.3.5.2.

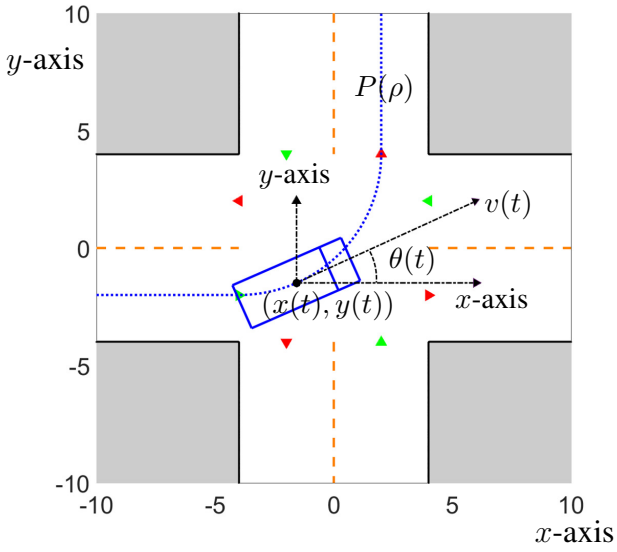


Figure 1.9: Vehicle kinematics modeled by (1.24)-(1.27). The blue rectangle represents the vehicle's c-zone where the end with double lines is the vehicle's front end. The blue dotted curve represents the pre-planned path $\mathcal{P}$. The states $x(t)$, $y(t)$ and $\theta(t)$ can be computed using the traveled distance along the path $\rho(t)$ and the path geometry (1.24). The green triangles represent the intersection entrance points $(x(\rho^{\mathrm{en}}), y(\rho^{\mathrm{en}}))$ and the red triangles the intersection exit points $(x(\rho^{\mathrm{ex}}), y(\rho^{\mathrm{ex}}))$.

1.3.3 Reward function

At an intersection, a driver typically has the following goals: 1) maximizing *safety*, e.g., not having a collision with another vehicle, 2) keeping a reasonable distance from other vehicles to increase safety and comfort, and 3) passing through the intersection and getting to her target lane under traffic rules and in a timely manner (i.e., *liveness*).

We assume common traffic rules (e.g., a left turn can only be made when the vehicle is entering the intersection from the leftmost forward lane) and speed limits have been accounted for in the pre-planned path P and speed bounds $[v_{\min}, v_{\max}]$. Then, the other

goals are expressed by the following reward function to maximize,

$$R^{\mathrm{i}} = w_1\hat{c} + w_2\hat{s} + w_3\hat{v}, \tag{1.30}$$

where $w_{1,2,3} > 0$ are weights, and the terms $\hat{c}$, $\hat{s}$ and $\hat{v}$ are explained below. Here, we consider the case where the ego vehicle i is interacting with one other vehicle j, with $i, j \in \mathcal{V} = \{1, 2, \ldots, n_{\mathrm{v}}\}$. The model will be extended to the case with multiple interacting vehicles, $n_{\mathrm{v}} > 2$, in Section 1.3.4.3.

$\hat{c}$ **(collision avoidance)**: The term $\hat{c}$ takes values according to the following equation:

$$\hat{c} = -(1 + s_{\mathrm{c}} + \hat{w}|v_i v_j|)\,\mathbb{I}(s_{\mathrm{c}} > 0), \tag{1.31}$$

where $s_{\mathrm{c}} \geq 0$ is the area of overlap between vehicle i and j's c-zones, v_i and v_j are the speeds of vehicle i and j, $\hat{w} > 0$ is a weighting parameter, and $\mathbb{I}(\cdot)$ is an indicator function taking 1 if $(\cdot)$ holds true and 0 otherwise.

The term $\hat{c}$ is designed as above so that when there is no collision, $s_{\mathrm{c}} = 0$, we have $\hat{c} = 0$; when there is a collision, $s_{\mathrm{c}} > 0$, the penalty depends on the overlapping area of c-zones and the vehicle speeds. In particular, larger penalties are imposed for larger overlapping areas and for larger absolute values of speeds as they both imply more severe collisions. The parameter $\hat{w} > 0$ adjusts the relative contribution to severity of overlapping area versus speeds. And the addition of 1 ensures a minimum penalty for collisions.

$\hat{s}$ **(separation)**: The term $\hat{s}$ takes values according to the following equation:

$$\hat{s} = -(1 + s_{\mathrm{s}} + \hat{w}|v_i v_j|)\,\mathbb{I}(s_{\mathrm{s}} > 0). \tag{1.32}$$

The expression (1.32) for $\hat{s}$ resembles the expression (1.31) for $\hat{c}$, but replaces s_{c} with $s_{\mathrm{s}} \geq 0$, which is the area of overlap between vehicle i and j's "separation zones" (s-zones). The s-zone of a vehicle is a larger rectangle, which shares the same longitudinal line of symmetry with the vehicle's c-zone and over-bounds the c-zone with a safety margin (see Fig. 1.10). It is characterized by the 6-tuple $(x, y, \theta, l_{\mathrm{s,f}}, l_{\mathrm{s,r}}, w_{\mathrm{s}})$, where $l_{\mathrm{s,f}}, l_{\mathrm{s,r}} \geq \frac{l_{\mathrm{c}}}{2}$ and $w_{\mathrm{s}} \geq w_{\mathrm{c}}$.

$\hat{v}$ **(velocity)**: We use the ego vehicle i's speed along its path to characterize its liveness, since a higher speed corresponds to a shorter time to reach its target lane. In particular, we let

$$\hat{v} = v_i. \tag{1.33}$$

It is clear from the above expressions (1.30)-(1.33) that the reward received by the ego vehicle i at time t depends on the state of vehicle i at t, $s_i(t)$, and the state of the interacting

other vehicle j at t, $s_j(t)$, i.e., $R^{\mathrm{i}}(t) = R^{\mathrm{i}}(s_i(t), s_j(t))$.

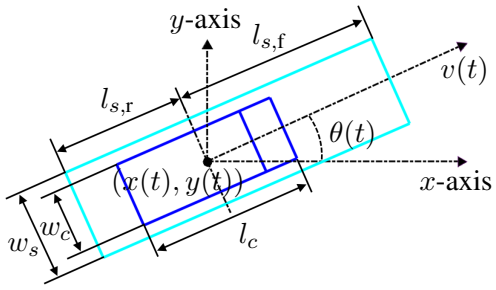


Figure 1.10: The c-zone (dark blue rectangle) and s-zone (light blue rectangle) of a vehicle.

We model the sequential decision-making process of a driver at an intersection as a receding-horizon control strategy. In particular, we assume a driver plans a sequence of actions $\gamma(t) = \{a(\tau|t)\}_{\tau=0}^{N-1} \in \Gamma = (\mathcal{A}^{\mathrm{i}})^N$ to maximize a cumulative reward,

$$\bar{R}^{\mathrm{i}}(t) = \sum_{\tau=1}^{N} \lambda^{\tau-1} R^{\mathrm{i}}(\tau|t), \tag{1.34}$$

where a variable with $\tau|t$ represents a predicted value of the variable at time $t+\tau$ with the prediction made at the current time t, N is the prediction horizon, and $\lambda \in [0, 1]$ is a discount factor. Once an action sequence $\gamma(t)$ has been determined, the driver applies the first element over one sampling period, i.e., $a(t) = a(0|t)$, to update her vehicle state $s(t) \to s(t+1)$. Then, the driver repeats this procedure at the next time instant $t+1$.

Recall that the reward received by a driver depends not only on the state of her own vehicle but also on the state of the interacting other vehicle (in the case of two-vehicle interactions). Hence, in order to maximize the cumulative reward (1.34), a driver needs to predict the interaction behavior of the other vehicle. In what follows, we introduce a model based on a leader-follower game formulation for simultaneous behavior prediction of the other vehicle and decision making for the ego driver.

To facilitate following exposition, we write the cumulative reward (1.34) as

$$\bar{R}^{\mathrm{i}}(t) = \bar{R}^{\mathrm{i}}(\mathbf{s}(t), \gamma_i(t), \gamma_j(t)), \tag{1.35}$$

where $\mathbf{s}(t) = (s_i(t), s_j(t))$ denotes the current traffic state (including the states of the two vehicles i, j in the case of two-vehicle interactions), $\gamma_i(t) = \{a_i(\tau|t)\}_{\tau=0}^{N-1}$ is the predicted action sequence of the ego vehicle i, and $\gamma_j(t) = \{a_j(\tau|t)\}_{\tau=0}^{N-1}$ is the predicted action sequence of the interacting other vehicle j. This can be done because the states of the two

vehicles over the prediction horizon are fully determined by their current states and their action sequences according to the kinematics model (1.24)-(1.27).

1.3.4 Leader-follower decision making

1.3.4.1 Assignment of leader-follower roles

Human drivers resolve their conflicts at intersections by following the "right-of-way" rules [63]. The right-of-way rules help drivers decide who should proceed first to pass through an intersection. Motivated by the right-of-way rules, we assign a leader-follower relationship to each pair of vehicles (i, j) at an intersection based on the following logic (illustrated in Fig. 1.11):

(1) If vehicles i, j have both entered the intersection, the vehicle that is closer to exit of the intersection is the leader.

(2) If at most one of vehicles i, j has entered the intersection, the vehicle that is closer to entrance of the intersection is the leader.

(3) If no leader has been assigned to (i, j) according to (1)-(2), then the vehicle on the right is the leader when the two vehicles are coming from adjacent road arms.

(4) If no leader has been assigned to (i, j) according to (1)-(3), then the vehicle going straight is the leader when the other vehicle is making a turn.

When comparing the distance of a vehicle to entrance (exit) of the intersection, we consider the signed arc length from the vehicle's current location (x, y) to the entrance point $(x^{\text{en}}, y^{\text{en}})$ of its origin lane (the exit point $(x^{\text{ex}}, y^{\text{ex}})$ of its target lane) along its path P. If the vehicle has entered (exited) the intersection, then the signed arc length to the entrance point (the exit point) is the negative of the arc length. For programming purpose, we restate the above leader-follower role assignment logic as Algorithm 3.

In Algorithm 3, we use $i \prec j$ to denote that vehicle i is the leader in the vehicle pair (i, j), and use $i \succeq j$ to denote that i is not the leader (i.e., either j is the leader or no leader role is determined between i and j according to (1)-(4)). The $\delta \geq 0$ is a threshold for differentiating the distances, accounting for the fact that human drivers can only estimate the distances with limited accuracy. In particular, we assume a driver cannot recognize which distance is smaller when $|\Delta\rho_i^{\text{en}}(t) - \Delta\rho_j^{\text{en}}(t)| \leq \delta$ (resp. $|\Delta\rho_i^{\text{ex}}(t) - \Delta\rho_j^{\text{ex}}(t)| \leq \delta$). In line 4, "going straight" and "making a turn" need to be differentiated, which is determined according to the angle between the vehicle's origin and target road arms. In particular,

Algorithm 3: Leader-Follower Role Assignment

Input : An ordered pair of vehicles (i, j) and their states $(s_i(t), s_j(t))$
Output: Whether i is the leader of j

if *(*$\Delta\rho_i^{en}(t) \leq 0$ *and* $\Delta\rho_j^{en}(t) \leq 0$*) and* $\Delta\rho_i^{ex}(t) < \Delta\rho_j^{ex}(t) - \delta$ **then** $i \prec j$;
else if *(*$\Delta\rho_i^{en}(t) > 0$ *or* $\Delta\rho_j^{en}(t) > 0$*) and* $\Delta\rho_i^{en}(t) < \Delta\rho_j^{en}(t) - \delta$ **then** $i \prec j$;
else if i *and* j *are coming from adjacent ways and* i*'s way is on the right of* j*'s way* **then** $i \prec j$;
else if i *is going straight and* j *is making a turn* **then** $i \prec j$;
else $i \succeq j$.

when the clockwise angle from its origin road arm to its target road arm is in the interval $(0, 3\pi/4]$, the vehicle is "making a left turn"; when the angle is in the interval $(3\pi/4, 5\pi/4)$, the vehicle is "going straight"; it is "making a right turn" otherwise.

According to Algorithm 3, at most one of the outcomes $i \prec j$ and $j \prec i$ can take place. It may occur that $i \succeq j$ and $j \succeq i$. In such a case, both vehicles view themselves as followers and make conservative decisions. Also, $\prec$ and $\succeq$ do not have the transitive property, i.e., $i \prec j$ and $j \prec k$ ($i \succeq j$ and $j \succeq k$) do not imply $i \prec k$ ($i \succeq k$). For instance, when four vehicles i, j, k and l coming from different road arms arrive at the entrances of a four-way intersection at the same time, we will have $i \prec j$, $j \prec k$, $k \prec l$ and $l \prec i$. Indeed, this case and similar cases where a cyclic pattern of leader-follower roles occurs are challenging cases for drivers – they may cause deadlocks (i.e., no one proceeds to enter the intersection or everyone gets stuck in the middle of the intersection).

1.3.4.2 Decision model for two-vehicle interactions

We assume a driver makes decisions according to

$$\gamma_i^*(t) \in \arg\max_{\gamma_i \in \Gamma} Q_{\text{leader}}(\mathbf{s}(t), \gamma_i), \tag{1.36}$$

if her vehicle, i, is interacting with one other vehicle, j, and her vehicle i is the leader in the vehicle pair (i, j), where

$$\begin{aligned} Q_{\text{leader}}(\mathbf{s}(t), \gamma_i) &= \min_{\gamma_j \in \Gamma_j^*(\mathbf{s}(t))} \bar{R}_i^{\text{i}}(\mathbf{s}(t), \gamma_i, \gamma_j), \\ \Gamma_j^*(\mathbf{s}(t)) &= \big\{\gamma_j' \in \Gamma : \min_{\gamma_i \in \Gamma} \bar{R}_j^{\text{i}}(\mathbf{s}(t), \gamma_j', \gamma_i) \geq \min_{\gamma_i \in \Gamma} \bar{R}_j^{\text{i}}(\mathbf{s}(t), \gamma_j, \gamma_i), \forall \gamma_j \in \Gamma\big\}. \end{aligned} \tag{1.37}$$

In (1.37), $\bar{R}_i^{\text{i}}(\mathbf{s}(t), \gamma_i, \gamma_j)$ represents the cumulative reward (1.35) received by vehicle i, and $\bar{R}_j^{\text{i}}(\mathbf{s}(t), \gamma_j, \gamma_i)$ represents the cumulative reward (1.35) received by vehicle j (i.e., now

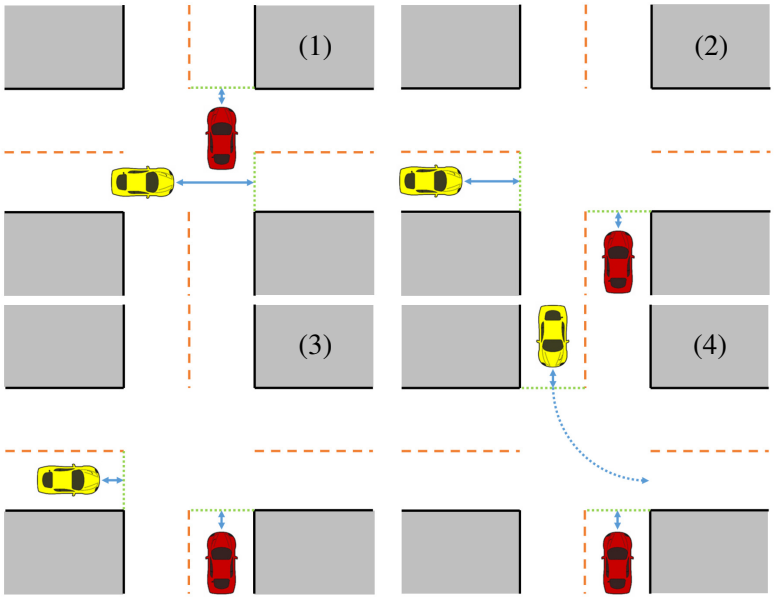


Figure 1.11: Leader-follower role assignment. In all of the figures, the red car is the leader and the yellow car is the follower.

vehicle j takes the role of "ego vehicle" and (1.35) reads $\bar{R}^{\mathrm{i}}(t) = \bar{R}^{\mathrm{i}}(\mathbf{s}(t), \gamma_j(t), \gamma_i(t)))$. We assume she makes decisions according to

$$\gamma_i^*(t) \in \arg\max_{\gamma_i \in \Gamma} Q_{\text{follower}}(\mathbf{s}(t), \gamma_i), \tag{1.38}$$

if her vehicle i is a follower in the vehicle pair (i, j), where

$$Q_{\text{follower}}(\mathbf{s}(t), \gamma_i) = \min_{\gamma_j \in \Gamma} \bar{R}_i^{\mathrm{i}}(\mathbf{s}(t), \gamma_i, \gamma_j). \tag{1.39}$$

The above leader-follower based decision model can be explained as follows: Firstly, as introduced in Section 1.3.4.1, we use a leader-follower relationship to model the right-of-way traffic rules, where a pairwise leader corresponds to a vehicle that has the right of way and a pairwise follower should yield to the leader. Since a leader has the right of way and a follower should yield, we assume a follower must take into account all possible actions of the leader when making her own decision and the way to account for all possible leader actions is modeled as the *maximin* strategy (1.38)-(1.39), i.e., maximizing the worst-case reward. In contrast, we assume a leader is able to predict the *maximin* decision of the follower and will take an optimal response, which leads to the leader decision rule (1.36)-(1.37).

We also note that this leader-follower based decision model is partly inspired by Stackelberg game theory [62]. We do not adopt a standard Stackelberg equilibrium-based decision model because a Stackelberg model relies on several stronger assumptions, including that 1) the follower must be able to observe and respond to the leader's decision immediately after it is made and 2) the leader must know 1) [61], which generally do not hold for vehicle interactions in traffic (e.g., due to vehicle dynamic behavior and human driver reaction delay [90]).

We now make the following technical assumptions:

$$\begin{aligned}
&\text{1) For any } (\mathbf{s}, \gamma_j) \in (\mathcal{S}^{\mathrm{i}})^2 \times \Gamma, \text{ there exists unique } \gamma_i' \in \Gamma \text{ such that} \\
&\quad \bar{R}^{\mathrm{i}}(\mathbf{s}, \gamma_i', \gamma_j) \geq \bar{R}^{\mathrm{i}}(\mathbf{s}, \gamma_i, \gamma_j) \text{ for all } \gamma_i \in \Gamma. \\
&\text{2) For any } \mathbf{s} \in (\mathcal{S}^{\mathrm{i}})^2, \text{ there exists unique } \gamma_i' \in \Gamma \text{ such that} \\
&\quad \min_{\gamma_j \in \Gamma} \bar{R}^{\mathrm{i}}(\mathbf{s}, \gamma_i', \gamma_j) \geq \min_{\gamma_j \in \Gamma} \bar{R}^{\mathrm{i}}(\mathbf{s}, \gamma_i, \gamma_j) \text{ for all } \gamma_i \in \Gamma.
\end{aligned} \tag{1.40}$$

The expressions in (1.40) assume that at any traffic state $\mathbf{s} = (s_i, s_j) \in (\mathcal{S}^{\mathrm{i}})^2$, for either a leader or a follower, there is one decision choice $\gamma_i' \in \Gamma$ that is strictly better than the others with respect to the decision rules (1.36)-(1.37) or (1.38)-(1.39). This assumption typically holds true. Under the assumptions (1.40), our leader-follower based decision model is simplified to

$$\gamma_i^*(t) = \begin{cases} \arg\max_{\gamma_i \in \Gamma} Q_{\text{leader}}(\mathbf{s}(t), \gamma_i), & \text{if } i \prec j, \\ \arg\max_{\gamma_i \in \Gamma} Q_{\text{follower}}(\mathbf{s}(t), \gamma_i), & \text{if } i \succeq j, \end{cases} \tag{1.41}$$

where

$$\begin{aligned}
&Q_{\text{leader}}(\mathbf{s}(t), \gamma_i) = \bar{R}_i^{\mathrm{i}}\big(\mathbf{s}(t), \gamma_i, \gamma_j^*(\mathbf{s}(t))\big), \quad \gamma_j^*(\mathbf{s}(t)) = \arg\max_{\gamma_j \in \Gamma} \min_{\gamma_i \in \Gamma} \bar{R}_j^{\mathrm{i}}(\mathbf{s}(t), \gamma_j, \gamma_i), \\
&Q_{\text{follower}}(\mathbf{s}(t), \gamma_i) = \min_{\gamma_j \in \Gamma} \bar{R}_i^{\mathrm{i}}(\mathbf{s}(t), \gamma_i, \gamma_j).
\end{aligned} \tag{1.42}$$

1.3.4.3 Decision model for multi-vehicle interactions

We now extend the decision model (1.41)-(1.42) for two-vehicle interactions to n-vehicle interactions with $n \geq 2$. Our extension relies on pairwise leader-follower relationships defined for all vehicle pairs at an intersection. In particular, we assume a driver of vehicle

i, for $i \in \mathcal{V} = \{1, 2, \ldots, n_{\text{v}}\}$, makes decisions according to:

$$\begin{aligned}
\gamma_i^*(t) &\in \arg\max_{\gamma_i \in \Gamma} \bar{Q}_i(\mathbf{s}(t), \gamma_i), \\
\bar{Q}_i(\mathbf{s}(t), \gamma_i) &= \min_{j \in \mathcal{V}, j \neq i} Q_{i,j}(\mathbf{s}_{i,j}(t), \gamma_i), \\
Q_{i,j}(\mathbf{s}_{i,j}(t), \gamma_i) &= \begin{cases} Q_{\text{leader}}(\mathbf{s}_{i,j}(t), \gamma_i) & \text{if } i \prec j, \\ Q_{\text{follower}}(\mathbf{s}_{i,j}(t), \gamma_i) & \text{if } i \succeq j, \end{cases}
\end{aligned} \tag{1.43}$$

where the traffic state $\mathbf{s}(t) = (s_1(t), \ldots, s_{n_{\text{v}}}(t))$ contains now the states of all vehicles at the intersection, $\mathbf{s}_{i,j}(t) = (s_i(t), s_j(t))$ represents the state of the vehicle pair (i, j), and $Q_{\text{leader}}(\mathbf{s}_{i,j}(t), \gamma_i)$ and $Q_{\text{follower}}(\mathbf{s}_{i,j}(t), \gamma_i)$ are defined in (1.42).

The decision model (1.43) can be interpreted as follows: A driver of vehicle i predicts the reward of a decision γ_i due to her interactions with vehicle j using $Q_{\text{leader}}(\mathbf{s}_{i,j}(t), \gamma_i)$ if vehicle i is a leader in the vehicle pair (i, j), and using $Q_{\text{follower}}(\mathbf{s}_{i,j}(t), \gamma_i)$ if i is a follower in the pair (i, j). Then, to account for her interactions with all other vehicles at the intersection, the driver maximizes the minimum of predicted rewards for all pairwise interactions. We will show through simulation examples in Section 1.3.6 that this decision model can produce realistic interaction behaviors of vehicles at intersections, and it leads to satisfactory safety and liveness properties (i.e., not being overly aggressive, represented by reasonable collision rates, and not being overly conservative, represented by reasonable deadlock rates). One important advantage of this decision model for multi-agent interactions compared to many other multi-player game-theoretic models (e.g., based on multi-player Nash equilibrium or Stackelberg equilibrium with multiple hierarchies) is its computational efficiency, which will also be shown in Section 1.3.6.

1.3.5 Additional modeling considerations

We incorporate the following additional considerations in our model to increase its fidelity in terms of modeling human driver behavior.

1.3.5.1 Courteous driving

A driver should not intentionally choose an action that would cause a collision when the other vehicles maintain their speeds. We account for this by modifying the decision model (1.43) to

$$\gamma_i^*(t) \in \arg\max_{\gamma_i \in \Gamma_i(\mathbf{s}(t))} \bar{Q}_i(\mathbf{s}(t), \gamma_i), \tag{1.44}$$

where $\Gamma_i(\mathbf{s}(t)) = \mathcal{A}_i^{\mathrm{i}}(\mathbf{s}(t)) \times (\mathcal{A}^{\mathrm{i}})^{N-1}$, in which $\mathcal{A}_i^{\mathrm{i}}(\mathbf{s}(t)) \subset \mathcal{A}^{\mathrm{i}}$ represents the set of courteous actions at the traffic state $\mathbf{s}(t) = (s_1(t), \ldots, s_{n_{\mathrm{v}}}(t))$, defined as

$$\mathcal{A}_i^{\mathrm{i}}(\mathbf{s}(t)) = \big\{a \in \mathcal{A}^{\mathrm{i}} : \text{for every } j \in \mathcal{V},\ j \neq i,\ \text{ if } a_i(t) = a \text{ and } a_j(t) = 0,\ \text{then} \\ \text{the two vehicles } i, j \text{ will not collide at time } t+1\big\} \cup \min\big\{a \in \mathcal{A}^{\mathrm{i}}\big\}. \tag{1.45}$$

In the definition (1.45) of $\mathcal{A}_i^{\mathrm{i}}(\mathbf{s}(t))$, $\min\big\{a \in \mathcal{A}^{\mathrm{i}}\big\}$ represents a fail-safe action: when there are no actions that can avoid a collision, the driver applies maximum braking.

We note that the above modification acts essentially as imposing some hard constraints to the decision problem (1.43) to increase safety. Although collisions have been penalized through the term $\hat{c}$ in the reward function (1.30), improved safety performance can typically be achieved through hard constraints than solely through penalties [91].

1.3.5.2 Local interaction

A driver can consider her interactions with only the other vehicles that are in a certain vicinity of her own vehicle (e.g., due to a limited perception range and information processing ability). To account for this, we further modify the decision model (1.43) according to

$$\bar{Q}_i(\mathbf{s}(t), \gamma_i) = \min_{j \in \mathcal{V}_i(t),\, j \neq i} Q_{i,j}(\mathbf{s}_{i,j}(t), \gamma_i) \tag{1.46}$$

where $\mathcal{V}_i(t) \subset \mathcal{V} = \{1, \ldots, n_{\mathrm{v}}\}$ represents the set of other vehicles whose interactions with the ego vehicle i are considered by the driver, defined as

$$\mathcal{V}_i(t) = \Big\{j \in \mathcal{V} : j \neq i \text{ and } \sqrt{(x_j(t) - x_i(t))^2 + (y_j(t) - y_i(t))^2} \leq \omega_i\Big\}, \tag{1.47}$$

where $\omega_i > 0$ represents an interaction radius of vehicle i.

1.3.5.3 Breaking deadlocks via exploratory actions

In cases where multiple vehicles arrive at an intersection almost simultaneously, a deadlock may occur when they negotiate the right of way – no one proceeds to enter the intersection or everyone gets stuck in the middle of the intersection. The following Algorithm 4 represents a strategy for our driver model to break deadlocks via exploratory actions.

Algorithm 4 is motivated by the observation that when multiple human drivers encounter a deadlock when negotiating the right of way at an intersection, some driver(s) will make a slight movement to seek the possibility of going first, and such exploratory actions can usually lead to an agreement among the drivers on their orders of passing through the

Algorithm 4: Breaking Deadlocks via Exploratory Actions

Input : The states of all vehicles $\mathbf{s}(t) = (s_1(t), \ldots, s_{n_\text{v}}(t))$ and the acceleration decisions produced by (1.43) of all vehicles $(a_1(t), \cdots, a_{n_\text{v}}(t))$

Output: Modified acceleration decisions of all vehicles $(a_1(t), \cdots, a_{n_\text{v}}(t))$

1 $\mathcal{V}_\text{conflict} = \emptyset$;
2 **for** $i \in \mathcal{V}$ **do**
3 **if** *i is the first vehicle coming from its origin lane that has not exited the intersection ($\Delta\rho_i^{ex}(t) > 0$)* **then** add i to $\mathcal{V}_\text{conflict}$;
4 **end**
5 **if** $v_i(t) = 0$ *and* $a_i(t) = 0$, $\forall i \in \mathcal{V}_\text{conflict}$ **then**
6 **for** $i \in \mathcal{V}_\text{conflict}$ **do**
7 **if** $\{a \in \mathcal{A}_i^\text{i}(\mathbf{s}(t)) : a > 0\} \neq \emptyset$ **then**
8 reset $a_i(t)$ based on
$$a_i(t) = \begin{cases} \min\{a \in \mathcal{A}_i^\text{i}(\mathbf{s}(t)) : a > 0\}, & \text{with probability } p_i, \\ 0, & \text{with probability } 1 - p_i. \end{cases}$$
9 **end**
10 **end**
11 **end**

intersection. In Algorithm 4, lines 2-4 aim to identify the vehicles that are in conflict. In particular, a vehicle i that has exited the intersection will not be viewed as a vehicle in conflict. Similarly, if there is another vehicle j that is entering/has entered the intersection from the same lane as vehicle i, drives in front of i[5], and has not exited the intersection, then a vehicle i will not be viewed as a vehicle in conflict. Line 5 aims to recognize the occurrence of a deadlock – a deadlock occurs when all vehicles that are in conflict have stopped and none of them decide to move according to (1.43). Then, lines 6-10 assign the vehicles in conflict positive probabilities of making slight movements to seek the possibility of going first. The effectiveness of Algorithm 4 in breaking deadlocks will be illustrated through simulation examples in Section 1.3.6.

1.3.6 Simulation results of leader-follower driver models

The parameter values used to generate the simulation results in this section are summarized in Table 1.2. These values are manually tuned to produce reasonable vehicle behavior in a few simulation trials and then used for all of the simulation examples in this section. Alternatively, they may be calibrated using data-driven approaches as in [92] or optimized using simulation-based approaches as in [71]. In particular, when a vehicle i is a leader (resp. a

[5]There should be no confusion about the meaning of "in front of" here since vehicles i and j are entering/have entered the intersection from the same lane.

follower) in a vehicle pair (i, j), i assumes the parameters $(l_{\mathrm{s,f}}, l_{\mathrm{s,r}}, w_{\mathrm{s}})$ characterizing the sizes of the s-zones of these two vehicles i, j take the values $(l^{\mathrm{l}}_{\mathrm{s,f}}, l^{\mathrm{l}}_{\mathrm{s,r}}, w^{\mathrm{l}}_{\mathrm{s}})$ (resp. $(l^{\mathrm{f}}_{\mathrm{s,f}}, l^{\mathrm{f}}_{\mathrm{s,r}}, w^{\mathrm{f}}_{\mathrm{s}})$) when i evaluates its reward term $\hat{s}$. We let $l^{\mathrm{l}}_{\mathrm{s,f}} \leq l^{\mathrm{f}}_{\mathrm{s,f}}$, $l^{\mathrm{l}}_{\mathrm{s,r}} \leq l^{\mathrm{f}}_{\mathrm{s,r}}$ and $w^{\mathrm{l}}_{\mathrm{s}} \leq w^{\mathrm{f}}_{\mathrm{s}}$. This way, a pairwise leader tends to take even more aggressive actions than a pairwise follower (since the leader pursues a smaller separation distance than the follower), and this further reduces the occurrence of deadlocks.

Table 1.2: Parameter values for intersection simulations.

Variable(s)	Value(s)	Unit	Remarks
Δt	1	s	sampling period
$[v_{\min}, v_{\max}]$	$[0, 5]$	m/s	speed range
$\mathcal{A}^{\mathrm{i}}$	$\{-4, -2, 0, 2\}$	m/s^2	{hard brake, decelerate, maintain, accelerate}
δ	0.5	m	threshold for differentiating distances
N	2		prediction horizon
λ	0.6		discount factor
$w_{1,2,3}$, $\hat{w}$	$\{100, 5, 1\}$, $\frac{1}{4}$		reward function weights
$(l_{\mathrm{c}}, w_{\mathrm{c}})$	$(6, 2.4)$	m	c-zone size
$(l^{\mathrm{l}}_{\mathrm{s,f}}, l^{\mathrm{l}}_{\mathrm{s,r}}, w^{\mathrm{l}}_{\mathrm{s}})$	$(5, 4, 2.8)$	m	s-zone size for leader
$(l^{\mathrm{f}}_{\mathrm{s,f}}, l^{\mathrm{f}}_{\mathrm{s,r}}, w^{\mathrm{f}}_{\mathrm{s}})$	$(14, 4, 2.8)$	m	s-zone size for follower
ω_i, $\forall i \in \mathcal{V}$	30	m	interaction radius
p_i, $\forall i \in \mathcal{V}$	0.25		probability of taking exploratory action to break deadlock

1.3.6.1 Reproducing real-world traffic scenarios

We first show that our leader-follower based driver model is able to reproduce human-driven vehicle trajectories of real-world traffic scenarios. The scenarios are extracted from the video dataset used in [93], which is collected at a two-lane four-way intersection in Canmore, Alberta.

We initialize states of the vehicles in our simulation according to the positions and velocities of the vehicles in the video, then simulate the evolution of the scenario using our leader-follower based model and compare it with the actual evolution of the video. Fig. 1.12 shows a few snapshots of our simulation versus video data for a scenario involving 3 interacting vehicles, and Fig. 1.13 shows those for a scenario involving 4 interacting vehicles. It can be seen that our model reproduces both scenarios with satisfactory accuracy. For instance, in both cases our model correctly predicts the order in which the interacting

vehicles pass through the intersection, and the time-dependent vehicle positions predicted by our model closely match the video data.

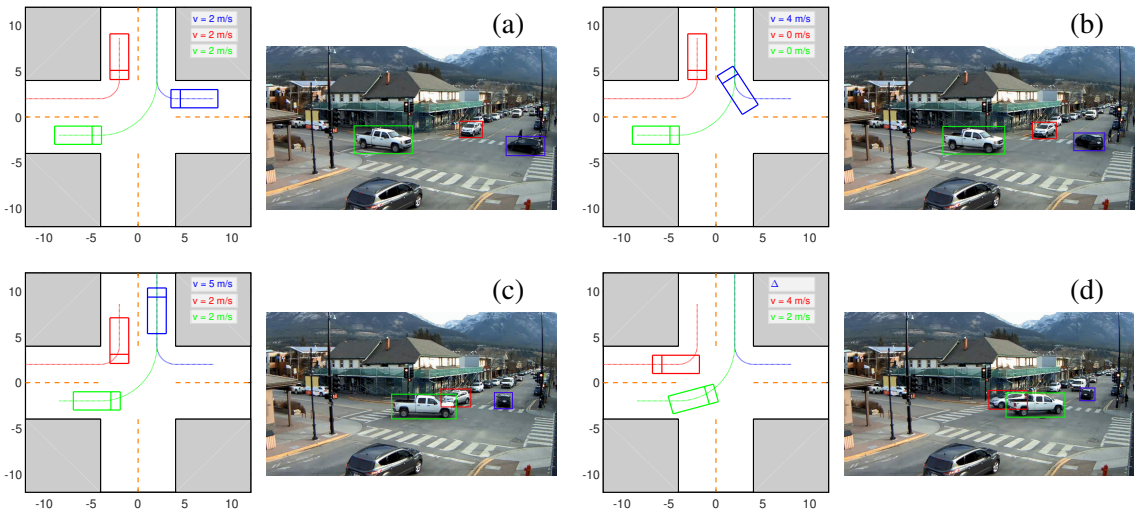


Figure 1.12: Reproducing a real-world traffic scenario with 3 interacting vehicles by the proposed model.

1.3.6.2 Completely symmetric scenarios

Among intersection scenarios, the ones where multiple vehicles arrive at an intersection at the same time are particularly challenging for drivers, because these cases easily cause deadlocks. In this section, we show simulation results of two "completely symmetric" cases.

Both cases involve a geometrically symmetric four-lane (two for each direction) four-way intersection. In the first case, 8 vehicles are approaching the intersection from each of the eight forward lanes with the same initial distance $\Delta\rho^{\text{en}}(0)$ to their corresponding entrance points and the same initial speed $v(0)$. Their goals are all going straight to cross the intersection. In the second case, 4 vehicles are approaching the intersection from each of the four leftmost forward lanes of the road arms with the same $\Delta\rho^{\text{en}}(0)$ and $v(0)$. Their goals all correspond to making a left turn. In both cases, all of the vehicles use the model (1.43) to make initial decisions, and use Algorithm 4 to adjust decisions when encountering a deadlock. The simulation results of these two cases are shown in Figs. 1.14 and 1.15.

In Fig. 1.14(a), the eight vehicles arrive at the intersection at the same time. In Fig. 1.14(b), they all stop at the intersection entrances to yield to the vehicle on their right, and hence a deadlock occurs. In Fig. 1.14(c), the blue vehicle makes a slight movement to seek the pos-

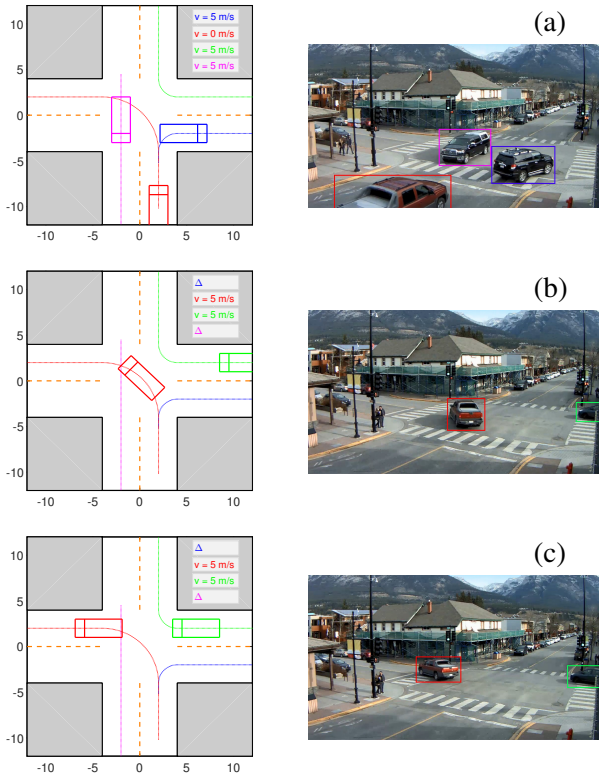


Figure 1.13: Reproducing a real-world traffic scenario with 4 interacting vehicles by the proposed model.

sibility of going first, which is based on Algorithm 4. Due to this movement, the symmetry is broken – the blue vehicle becomes the overall leader among all vehicles, and hence enters and crosses the intersection first, shown in Fig. 1.14(d). The other vehicles then enter and cross the intersection in a clockwise order, shown in Figs. 1.14(e) and (f). In Fig. 1.15(a), after the four vehicles arrive and stop at the intersection entrances, the left purple vehicle makes a slight movement and seeks to go first. As a result, the purple vehicle enters the intersection and gets to its target lane first, shown in Fig. 1.15(b). Similar to the previous case, the other vehicles then pass through the intersection in a clockwise order, shown in Figs. 1.15(c) and (d).

The above results show that our leader-follower game based driver model can produce realistic vehicle interaction behaviors, and has reasonably good capability to resolve vehicle conflicts in challenging intersection scenarios. Note that our model does not rely on a centralized controller/manager to guide the vehicles to resolve their conflicts – the vehicles make their decisions individually and independently. This is consistent with the way human drives in real-world traffic.

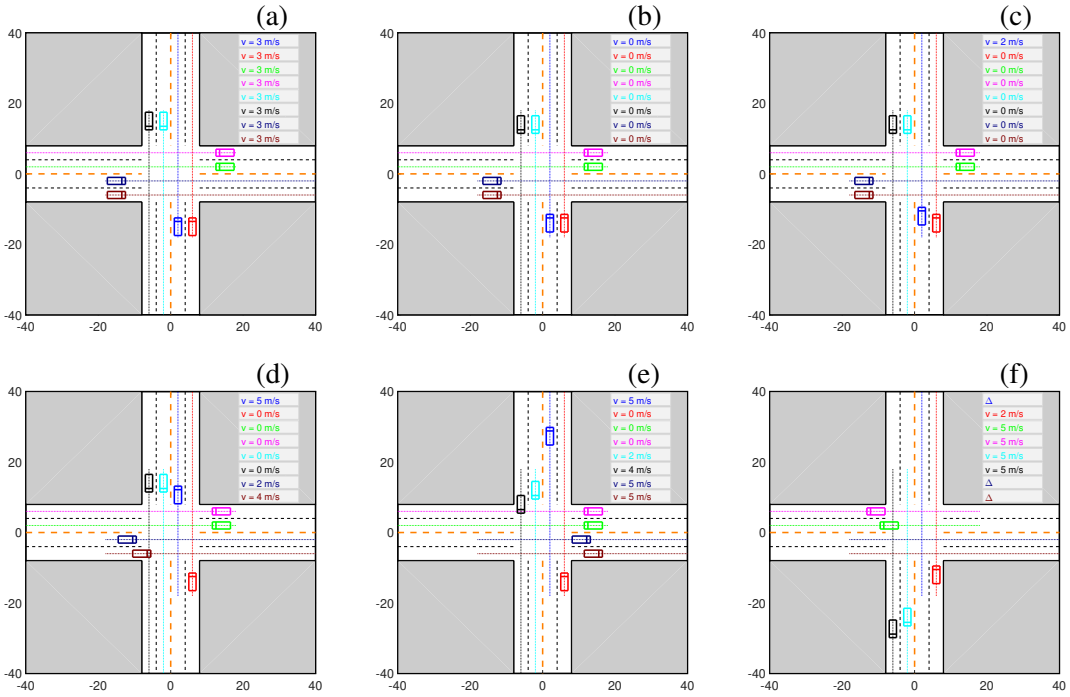


Figure 1.14: Completely symmetric case 1. Figures (a-f) show snapshots of the simulation at a series of time steps.

1.3.6.3 Randomized traffic scenarios and statistical evaluation

We run a batch test with randomized parameters to statistically evaluate our leader-follower game based driver model. We let the number of road arms n_{r} take values in $n_{\mathrm{r}} \in \{3, 4, 5\}$ and let the number of vehicles n_{v} take values in $n_{\mathrm{v}} \in \{2, 4, 6, 8, 10\}$. For each pair of $(n_{\mathrm{r}}, n_{\mathrm{v}})$, we randomly sample the numbers of forward and backward lanes $\{m_{\mathrm{f}}^i\}_{i=1}^{n_{\mathrm{r}}}$, $\{m_{\mathrm{b}}^i\}_{i=1}^{n_{\mathrm{r}}}$ according to the following categorical distributions,

$$m_{\mathrm{f}}^i \sim \mathrm{Cat}\left(\{1, 2, 3\}, \{0.15, 0.7, 0.15\}\right), \quad \xi \in \{\mathrm{f}, \mathrm{b}\}, \tag{1.48}$$

and sample the road arm angles $\{\phi^i\}_{i=1}^{n_{\mathrm{r}}}$ according to the following truncated normal distributions,

$$\phi^i \sim \mathrm{Normal}\left(\frac{2\pi i}{n_{\mathrm{r}}}, \frac{\pi}{24}, \left[\frac{2\pi i}{n_{\mathrm{r}}} - \frac{\pi}{8}, \frac{2\pi i}{n_{\mathrm{r}}} + \frac{\pi}{8}\right]\right), \tag{1.49}$$

to create the intersection. Then, we initialize the states of the vehicles as follows: Firstly, we randomly pick a road arm and a forward lane of this road arm as the origin lane of a vehicle, and randomly pick an "admissible" road arm and an "admissible" backward lane of this road arm as its target lane. Here, "admissible" means that the picked road arm and backward lane satisfy common traffic rules (e.g., a left turn can only be made when the

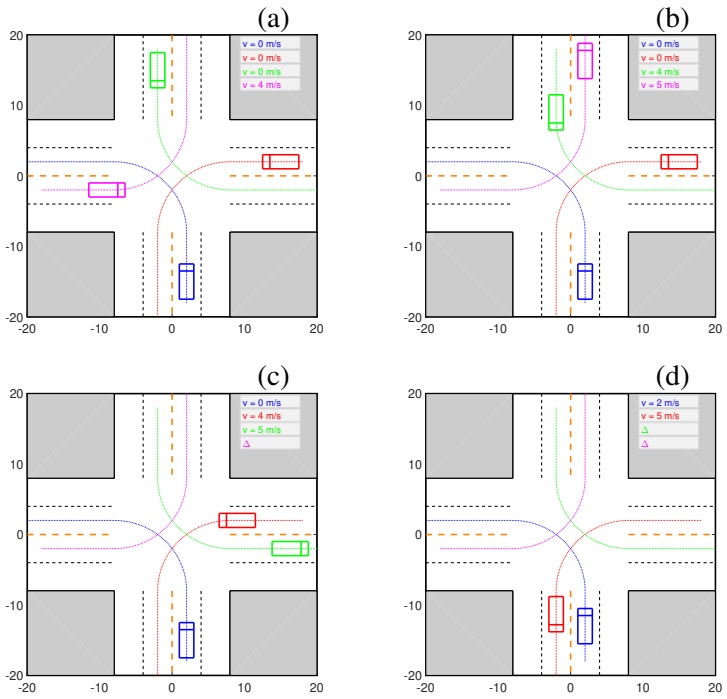


Figure 1.15: Completely symmetric case 2. Figures (a-d) show snapshots of the simulation at a series of time steps.

vehicle is entering the intersection from the leftmost forward lane). Then, we initialize the location and speed of this vehicle according to

$$\Delta\rho^{\text{en}}(0) \sim \text{Uniform}\left([10, 28]\right) \text{ [m]}, \qquad v(0) \sim \text{Uniform}\left([2, 4]\right) \text{ [m/s]}. \tag{1.50}$$

Furthermore, we enforce a minimum initial separation distance ρ^{sep} between any two vehicles that are put in the same origin lane – when we initialize a vehicle i, if it is put in the same origin lane with a vehicle j that has been initialized and $\Delta\rho_i^{\text{en}}(0)$ is sampled in the range of $[\Delta\rho_j^{\text{en}}(0) - \rho^{\text{sep}}, \Delta\rho_j^{\text{en}}(0) + \rho^{\text{sep}}]$, then its origin lane and $\Delta\rho_i^{\text{en}}(0)$ are both resampled according to the above-mentioned procedure until this initial separation requirement is satisfied.

For each pair of $(n_{\text{r}}, n_{\text{v}})$, We run 100 of such randomized simulation trials. A few examples of these simulation trials are illustrated in Fig. 1.16.

We define three metrics to evaluate our driver model in terms of safety and liveness: They are the success rate (SR), the collision rate (CR), and the deadlock rate (DR). The SR is defined as the proportion of simulation trials where all of the vehicles safely (i.e., without colliding with any other vehicles) reach their terminal points $(x^{\text{term}}, y^{\text{term}})$ within 60 [s] of simulation time. The CR is defined as the proportion of simulation trials where at

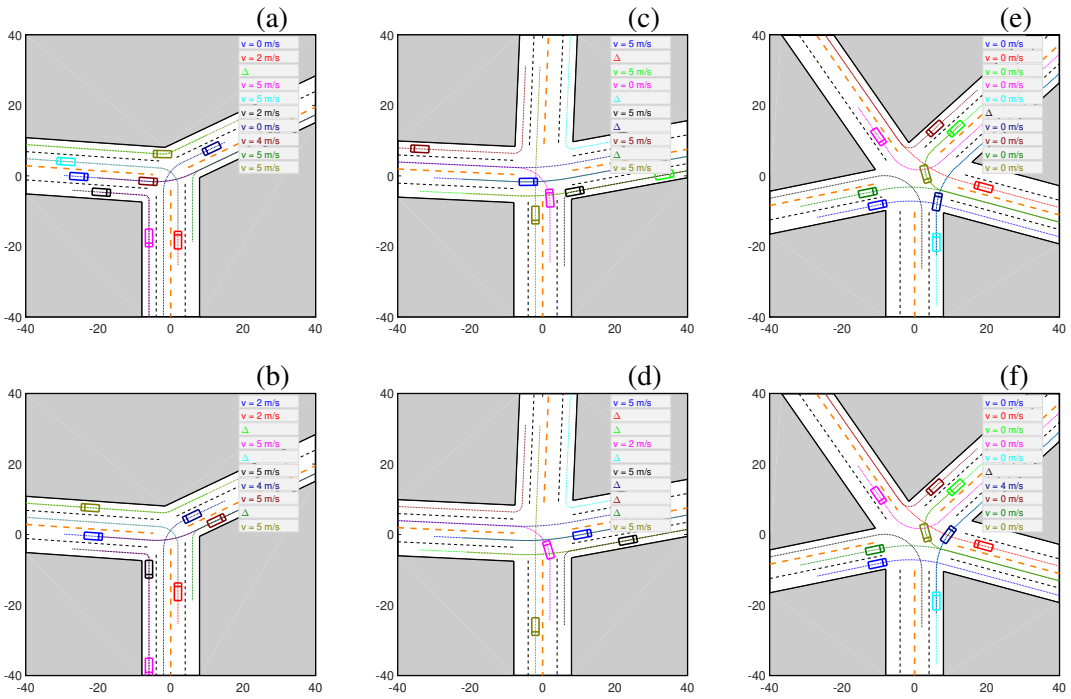


Figure 1.16: Randomized traffic scenarios. Figures (a-b) show snapshots of a simulation in a three-way intersection scenario, figures (c-d) show those in a four-way intersection scenario, and figures (e-f) show those in a five-way intersection scenario.

least one vehicle collision occurs (once a vehicle collision occurs at a simulation step, the simulation stops at that step). The DR is defined as the proportion of simulation trials where no vehicle collision occurs but there is at least one vehicle that does not reach its terminal point $(x^{\text{term}}, y^{\text{term}})$ within 60 [s] of simulation time. We note that based on their definitions, $\text{SR} + \text{CR} + \text{DR} = 1$. A model representing human driver should have reasonably high SR, and reasonably low CR and DR. The evaluation results of our model are shown in Fig. 1.17.

It can be seen that as the numbers of road arms and vehicles increase, which correspond to more complex traffic scenarios, the CRs and DRs also increase. In three-way and four-way intersection scenarios with 2 or 4 vehicles, no collisions or deadlocks are observed. When up to 10 vehicles are interacting at three-way or four-way intersections, the SRs are higher than 0.9. It can also be seen that five-way intersections have higher CRs and DRs than three-way and four-way intersections. For instance, the SR for five-way intersection drops to 0.84 for the case of 10 interacting vehicles. This is because five-way intersections allow more vehicles to arrive at or be inside the intersection at the same time than three-way and four-way intersections, which can cause higher chances of vehicle conflicts. Indeed, as human drivers, we also typically find five-way intersections to be more challenging than three-way and four-way intersections.

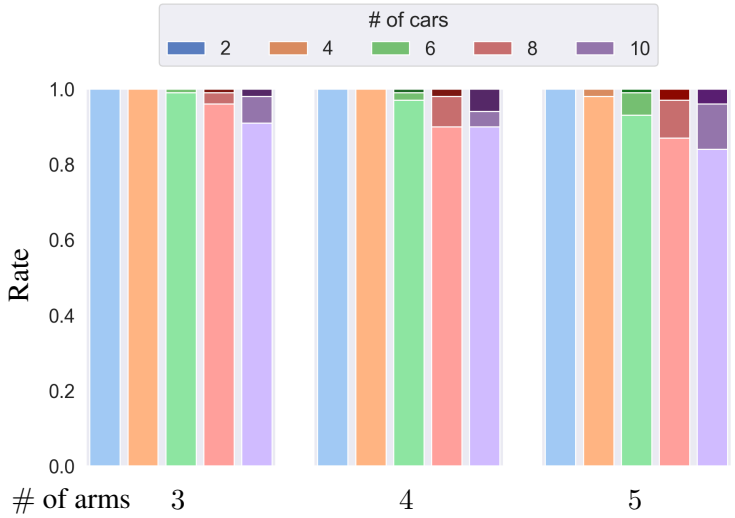


Figure 1.17: Statistical evaluation of the vehicle interaction model. Light color: SR, medium color: DR, dark color: CR.

After investigating the simulation trials with collisions, we find that most collisions are caused by simultaneous exploratory actions of two or more vehicles in deadlock scenarios. Note that in Algorithm 4, a vehicle is not permitted to accelerate if its acceleration would cause a collision when the other vehicles in conflict remain stopped. However, if two or more vehicles accelerate at the same time, it is possible that their simultaneous accelerations cause a collision though each single acceleration would not. Two of the failure cases are shown in Fig. 1.18.

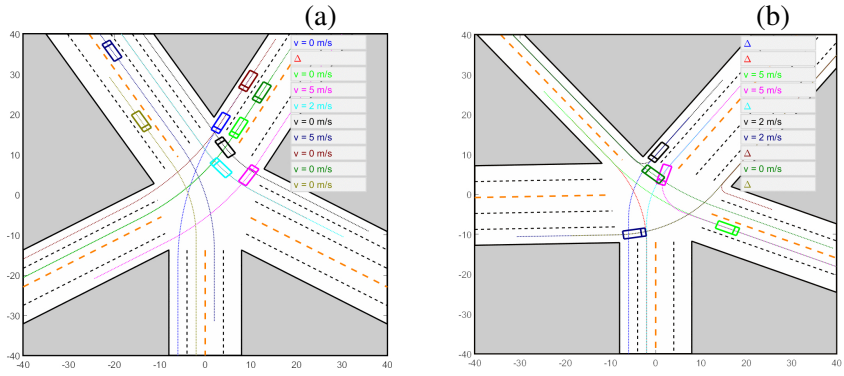


Figure 1.18: Two failure cases. (a) A deadlock scenario. (b) A collision scenario.

We observe higher CRs and DRs in our simulations than what are observed in real-world traffic, especially for the cases with larger numbers of interacting vehicles. For this, we note that modeling human driver behavior, especially for multi-vehicle interaction sce-

narios, is a difficult and open problem, and our results are comparable with (even outperform) several other models for multi-vehicle interactions at intersections in the literature. For instance, for four-way intersection with 4 interacting vehicles, the approach of [94] leads to a series of CRs ranging from 0% to 1.1% and a series of DRs ranging from 0% to 14.3% depending on the simulation settings; while our model has 0% CR and 0% DR for randomized scenarios. For four-way intersection with 2 to 6 interacting vehicles, the approach of [47] leads to collision-free simulations, but at the cost of a rapid increase in DR as the number of interacting vehicles increases – when the number of interacting vehicles increases to 6, almost 50% simulation trials following the approach of [47] end up with deadlocks. In contrast, for four-way intersection with 6 interacting vehicles, our model has only 1% CR and 2% DR.

We also remark that our CR and DR results are for a more complex intersection model than the four-way intersections considered in [47] and [94]. For instance, unlike our intersection model where the number of lanes for each road arm and the angles between road arms can vary, [47] and [94] run their simulation experiments on a simple two-lane (one for each direction) four-way intersection model with orthogonal road arms. On the one hand, the scenarios produced by our intersection model are more complex and hence more challenging for drivers/vehicles to navigate. On the other hand, simulation results for these more complex scenarios can provide more insights into driver/vehicle interactive behaviors. Indeed, the CRs and DRs of our model can be adjusted through tuning the weights for different terms of the reward function (1.30). For instance, it is possible to reduce the number of collisions, which represent more severe failures, by increasing the weights w_1, w_2 for collision avoidance and separation, at the cost of a larger number of deadlocks, which are less severe. However, this is not always desirable. For instance, one of our intended use cases of the developed driver models is for constructing simulated traffic environments for virtual verification and validation of autonomous vehicle control systems. A traffic model with a low CR and a high DR may tend to be overly conservative and less likely to generate challenging test cases for autonomous vehicles.

For the vehicles that safely reach their terminal points within 60 [s] of simulation time, we count their average completion time (CT), which is defined as the duration (in [s] of simulation time) from the simulation initialization to the time instant when the vehicle reaches its terminal point. The average CT can reflect how conservative the driver model is. The average CTs for different numbers of road arms and vehicles are shown in Fig. 1.19.

Firstly, it can be seen that as the number of vehicles increases, the vehicles need more time to pass through an intersection, which is reasonable. We then use the traffic quality rating system called the "level-of-service" (LOS) for unsignalized intersections defined

based on the average control delay [95] to validate that the time needed by our driver model to navigate a vehicle through an intersection in the presence of other vehicle interactions roughly matches that needed by an average human driver. In particular, in the case of 2-4 vehicles, the average CTs exhibited by our model can be mapped to LOS Level-B (10-15 [s] average control delay), which represents traffic with a high degree of freedom and a small amount of interactions [96]. In the case of 6-10 vehicles, the average CTs exhibited by our model can be mapped to LOS Level-C (15-25 [s] average control delay), which represents traffic with restricted freedom due to significant interactions. We can also observe from Fig. 1.19 that among three-, four- and five-way intersections, vehicles spend the least amount of time passing through a four-way intersection. This implies that the right-of-way traffic rules work best for four-way intersections.

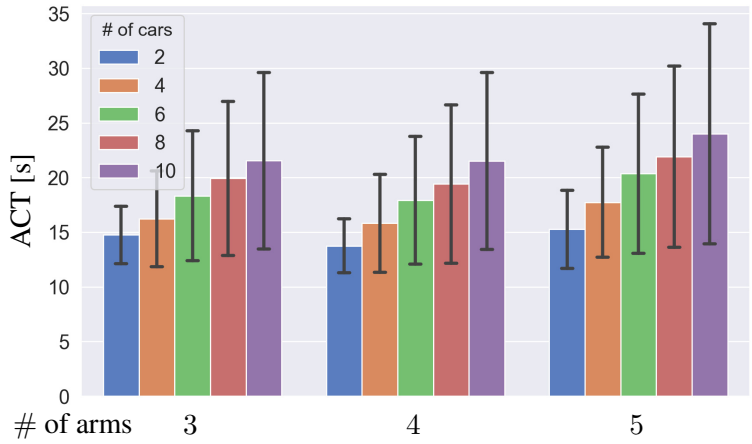


Figure 1.19: Average completion time (ACT). The black vertical bars represent the standard deviations.

1.4 Application to verification and validation of autonomous vehicle control systems

Over the past decade, extensive efforts have been pursued in both academia and industry to develop advanced autonomous driving systems, with the goal to provide safer, cleaner, and more efficient transportation as well as improved mobility for the young, elderly, and disabled. One of the most significant challenges that must be addressed before these systems can be deployed in mass production is their verification and validation (V&V). It is estimated that "autonomous vehicles would have to be driven hundreds of millions of miles

and sometimes hundreds of billions of miles to demonstrate their reliability in terms of fatalities and injuries" [64]. It would be extremely time- and resource-consuming if these testing miles are all established in the real, physical world. On the one hand, a practical solution is to accomplish some portion of them in a virtual world using simulation tools, i.e., via virtual V&V in a simulator. On the other hand, the reliability of virtual V&V depends fundamentally on the fidelity of the simulator, i.e., how representative the simulator is of what a vehicle will encounter in the real world.

In real-world traffic scenarios that involve multiple vehicles, the vehicles share the road and interact with each other. In this case, a simulator should be able to represent the interactive behaviors of drivers/vehicles with reasonable fidelity. In many conventional simulators, only the ego vehicle is a decision maker (i.e., controlled by an autonomous driving algorithm under test) and the behaviors of other vehicles are prescribed either as functions of time or as simple functions of the traffic state (e.g., described as if-then-else rules). In contrast, our game-theoretic driver models developed in previous sections create the opportunity to model all vehicles in a simulated traffic environment as interactive decision makers, where their behaviors are determined by the combination of a reward function, which explicitly models a driver's driving objectives, and an interactive decision-making process (i.e., the level-k or leader-follower based reasoning process). Therefore, we can use our driver models to build up a simulator (or, enhance an existing simulator) with advanced representation of driver/vehicle interactive behaviors, as a platform for autonomous driving system virtual V&V.

1.4.1 Highway simulator and virtual V&V results of two AV policies

In this section, we use our level-k driver models developed in Section 1.2 to form a highway simulator. This simulator provides traffic environments for virtual V&V of autonomous vehicle (AV) control algorithms. As case studies, we use this simulator to test and compare two AV policies, including a Stackelberg game-based policy and a decision tree-based policy.

When setting up a traffic environment, we use the number of simulated vehicles, n_{v}, to control the traffic density. Recall that the length of our cyclic road is $x_{\max}$. In this case, the traffic density can be estimated as $\frac{n_{\mathrm{v}}}{x_{\max}}$. Then, to account for the heterogeneity in driving styles of real-world drivers, when assigning control policies to the vehicles, we let each vehicle have a 10% chance to be assigned a level-0 policy, 60% chance to be assigned a level-1 policy, and 30% chance to be assigned a level-2 policy. This way, the traffic environment is roughly made up of a mixture of 10% level-0, 60% level-1, and 30% level-2

vehicles. These percentages of level-k vehicles are set based on the estimated percentages of level-k reasoners in a human behavior experiment conducted in [79]. The graphical user interface (GUI) of the simulator is the same as those in Figs. 1.2-1.5, where the red vehicle in the middle is now the test vehicle (the vehicle controlled by the autonomous driving algorithm under test).

We now use this simulator with mixed level-k vehicles to test and compare a Stackelberg game-based policy and a decision tree-based policy. These policies were originally proposed in [48, 49] and in [97], respectively. Necessary modifications are made to make them compatible with our simulator. Detailed descriptions of the modified versions of these policies are given in Appendix C.

Firstly, we assess their safety and robustness properties based on their safety violation rates in our simulator, which are defined in the same way as those numbers shown in Fig. 1.7, i.e., as the percentage of 10000 simulation episodes which end within $t_{\max} = 200$ seconds due to the occurrence of a safety-violation state of the test vehicle. Fig. 1.20 shows their safety violation rates versus varied traffic densities. It can be seen that both policies exhibit significant safety violation rates. Comparing Fig. 1.20 with Fig. 1.7(b), we can further see that both policies lead to more safety violations than our level-k driver models when they operate in the same mixed traffic environment.

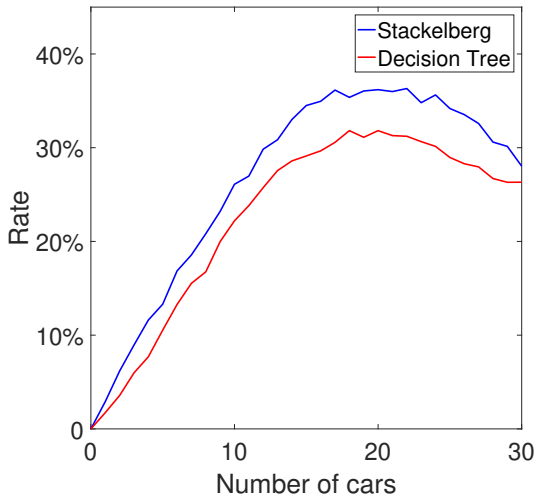


Figure 1.20: Safety violation rates of the Stackelberg game-based policy and the decision tree-based policy.

Reflecting on potential reasons for their high safety violation rates, we realize that our simulated traffic environments with heterogeneous and interactive drivers are more complex and hence more challenging than the traffic models for which the Stackelberg game-

based policy and the decision tree-based policy were originally proposed. Indeed, when we test these policies with a traffic model consisting of only level-0 vehicles, which represents a simpler environment as level-0 vehicles do not make lane changes [see (1.17)], no safety violations are observed. This is also in agreement with the results in [48, 49, 97]. However, when these policies operate in our mixed traffic environments where some other vehicles (i.e., level-1 and 2 vehicles) also make lane changes, safety violations occur.

Fig. 1.21 illustrates two scenarios that can explain many safety violations encountered in our simulations. The red rectangle represents the test vehicle, and the yellow rectangles represent the other vehicles in the traffic environment. The black arrows indicate the longitudinal velocities of the vehicles, and green arrows indicate their lateral motions. The left-hand sides represent the scenarios at the time instant t, and the right-hand sides represent the scenarios at the next time instant $t + 1$. In Fig. 1.21(a), the test vehicle starts a lane change to the left in order to overtake the slower vehicle in its front. Meanwhile, the front vehicle also starts a lane change to the left due to some other vehicle(s) in its front (not shown in this figure). As a result, both vehicles are moving into the left lane while their longitudinal distance keeps decreasing, which eventually leads to a rear-end collision. In Fig. 1.21(b), the test vehicle starts a lane change to the middle lane in order to overtake a slower vehicle in front of it. Meanwhile, another vehicle, which originally drives in the leftmost lane and is in an almost side-by-side position with the test vehicle, also starts a lane change to the middle lane. As a result, both vehicles are moving into the middle lane, which eventually leads to a side collision.

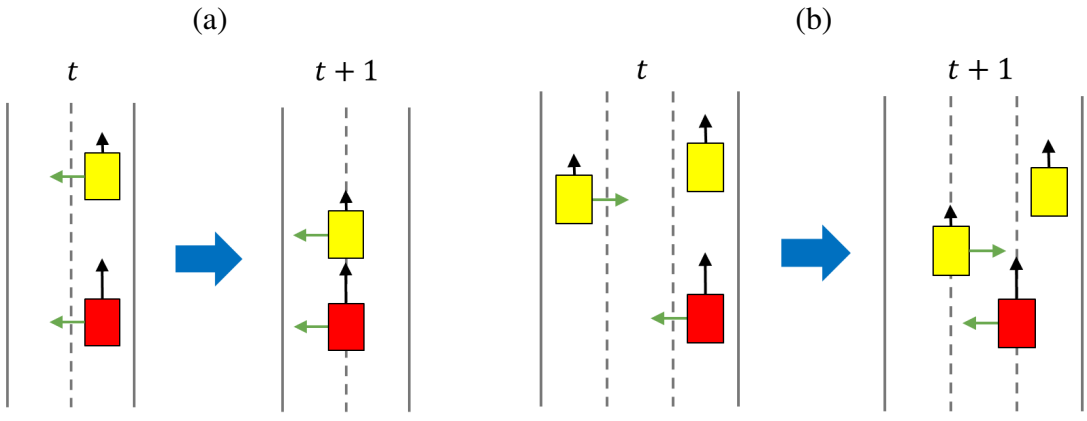


Figure 1.21: Scenarios leading to safety violations.

We note that one reason for the above two scenarios to eventually lead to safety violations is that we have assumed a lane change must continue to completion once it gets

started. The augmentation of a fail-safe mechanism to the AV policies to let the vehicle be able to abort improper lane changes may avoid many of these safety violations. As a matter of fact, the above two scenarios are also challenging scenarios for human drivers. Challenging scenarios automatically generated by our simulator, such as these two, can help us discover faults in an AV control algorithm and hence significantly accelerate the overall V&V procedure.

From Fig. 1.20 we can also observe that the safety violation rates first increase, peak, and then decrease as traffic density continually increases. We now explain this trend with the help of Fig. 1.22: In sparse traffic, vehicles rarely have conflicts, and hence safety violations rarely occur [①]. As traffic density increases, the chances for vehicles to have conflicts increase, and hence the safety violation rate also increases [②] until reaching its peak [③]. When traffic becomes very dense, e.g., in a traffic jam, vehicles drive at low speeds and lane change events become rare. As a result, the rate of safety violations becomes low again [④]. This trend of safety violation rates produced by our simulator matches the statistical analysis results of [98] regarding the relationship between car crash rates and traffic densities in real-world traffic data.

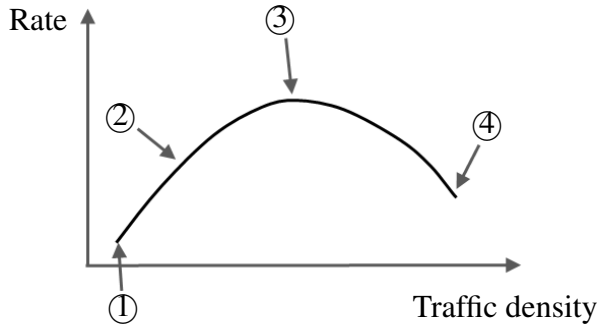


Figure 1.22: Trend of safety violation rate.

In addition to using the safety violation rate as the metric to assess the safety and robustness of the Stackelberg game-based policy and the decision tree-based policy, we also use the test vehicle's average driving speed to assess and compare their liveness. The test vehicle's average driving speeds versus varied traffic densities are shown in Fig. 1.23(a). It can be seen that the decision tree-based policy leads to higher average speeds than the Stackelberg game-based policy in all traffic densities.

Fig. 1.23(b) compares the computational costs of the two AV policies. The numbers in Fig. 1.23(b) indicate the average CPU times for running a $t_{\max} = 200$ sec-long simula-

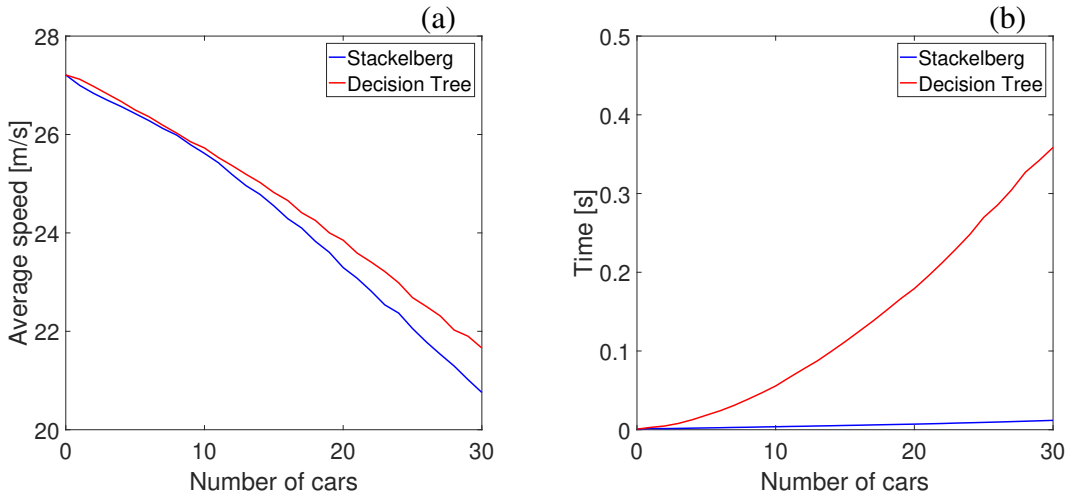


Figure 1.23: (a) Average driving speeds and (b) computational costs of the Stackelberg game-based policy and the decision tree-based policy.

tion episode. We can see that the computational cost of the decision tree-based policy is higher than that of the Stackelberg game-based policy. Combining the results in Figs. 1.20, 1.23(a), and 1.23(b), we can conclude that our implementation of the decision tree-based policy has better safety and liveness properties, while is also more computationally demanding, than our implementation of the Stackelberg game-based policy.

Our simulator is written in Java and run on a desktop with an Intel Core i7-4790 3.60 GHz processor and 16.0 GB RAM using the Eclipse Neon platform. The CPU times shown in Fig. 1.23(b) are calculated using the Java $System.nanoTime()$ function. We can see that our simulator runs very fast – simulating up to 30 vehicles for 200 seconds of simulated time takes only tens to hundreds of milliseconds of real time. Note that all vehicles in our simulator, including the test vehicle and the other vehicles in the environment, are interactive decision makers.

Our simulator can also be used for optimizing the parameters of an autonomous driving algorithm. We now illustrate this by taking the above decision tree-based AV policy as an example and optimizing two of its parameters, including $\frac{w_{l1}}{w_{l2}}$, which represents the weighting between the two decision tree layers, and x_B, which represents a threshold for triggering the policy.

The optimization of an autonomous driving algorithm is necessarily a multi-objective optimization problem, because an AV must operate not only safely not also effectively, i.e., ensuring both safety and liveness. We continue to use the safety violation rate in our simulator, $\bar{c}$, as the metric to measure the safety of an autonomous driving algorithm – a

smaller $\bar{c}$ value represents a higher level of safety, and use the average driving speed, $\bar{v}_x$, to measure the liveness – a larger $\bar{v}_x$ value represents a higher level of liveness. To account for both objectives, we consider the following linear scalarization of them,

$$R_{\text{obj}} = p_1(-\bar{c}) + p_2 \frac{\bar{v}_x - v_{x,\min}}{v_{x,\max} - v_{x,\min}}, \tag{1.51}$$

where $p_1, p_2 \geq 0$ are weights for the two objectives. This linear scalarization is designed in such a way that its terms are all dimensionless.

Fig. 1.24 shows the surfaces of (1.51) corresponding to different choices of the weights p_1 and p_2. These surfaces can be used to pick the best pair of $(\frac{w_{l1}}{w_{l2}}, x_B)$ for a user-specified weighting between safety and liveness. For instance, for maximum safety ($p_1 = 1, p_2 = 0$), one can read from Fig. 1.24(a) that the best pair is $(\frac{w_{l1}}{w_{l2}}, x_B) = (2.5, 23)$.

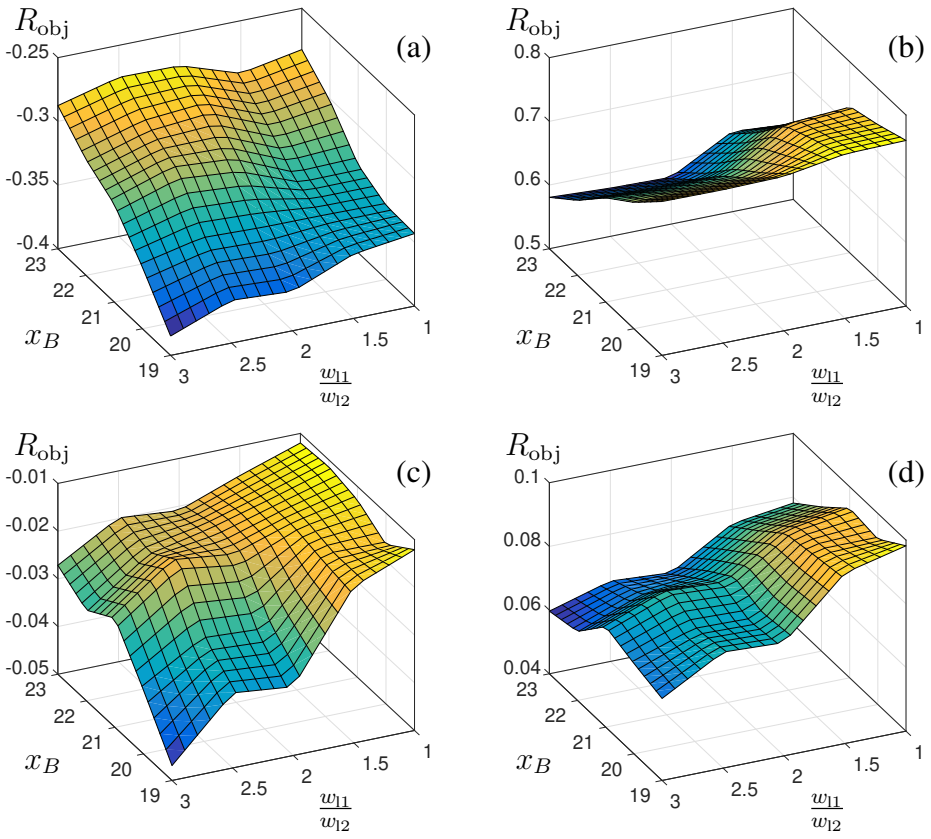


Figure 1.24: Objective function surfaces corresponding to different choices of the weights p_1 and p_2. (a) $p_1 = 1, p_2 = 0$, (b) $p_1 = 0, p_2 = 1$, (c) $p_1 = 0.7, p_2 = 0.3$, and (d) $p_1 = 0.6, p_2 = 0.4$.

1.4.2 Integration with the TORCS simulator and calibration

Our game-theoretic driver models can also be integrated with existing simulators to enhance their representations of driver/vehicle interactive behaviors while taking advantage of their higher-fidelity representations of vehicle dynamics, road and weather conditions, etc., as well as their better graphics. In this section, we describe the integration of our driver models with The Open Racing Car Simulator (TORCS).

TORCS is an open-source car racing simulator designed to enable pre-programmed AI drivers to race against one another and/or a "human driver" (a vehicle controlled by a human user using a keyboard or a steering wheel) [99]. TORCS has a high-fidelity vehicle model, covering 1) rigid-body dynamics related to the mass and rotational inertia of the vehicle, 2) chassis dynamics, including the dynamics of suspensions, links and differentials, 3) tire dynamics for various ground types, and 4) aerodynamics, including slip-streaming and ground effects. We use TORCS as an example to demonstrate the integration of our driver models with existing simulators. Note that our driver models can also be integrated with other simulators, such as CARLA and LGSVL, and the integration procedure will be similar.

For every vehicle in a TORCS simulation, we can read from TORCS application programming interface (API) the following signals: its longitudinal distance from the start line, its lateral position with respect to the center of the track, and the driving speeds of it and all other vehicles in its vicinity. These signals are used to calculate the quantities in Section 1.2.3, which then become the input to our driver model. The effectors of TORCS that are related to the control of a vehicle are listed in Table 1.3.

Table 1.3: Vehicle control-related effectors of TORCS.

Effectors	Description
Accelerator	0 = none, 1 = full throttle
Brake	0 = none, 1 = full brake
Steering	-1 = full right, 1 = full left

The output from our driver model is one of the actions in Section 1.2.2. This action defines a desired state of the vehicle, including a desired speed and a desired lane. We design controls for the effectors in Table 1.3 to generate smooth maneuvers for tracking the desired state. In particular, we use a proportional-integral-derivative (PID) controller to control the vehicle's longitudinal speed and use a proportional-derivative (PD) controller to control the vehicle's lateral motion for lane keeping and lane change. The designed controllers are introduced below.

We note that TORCS uses an update frequency of 50 Hz. This is much higher than our driver model's default decision-making frequency, which is $\frac{1}{\Delta t} = 1$ Hz. Thus, to simplify expressions, we assume TORCS simulations are run in continuous time. In the following two subsections, we use $k \in \mathbb{N}_0$ to denote the discrete/sample time instants where our driver model produces new action commands and use $t \in \mathbb{R}$ to denote continuous time instants.

1.4.2.1 Longitudinal speed control

At each sample time instant k, the action command generated from our driver model defines a desired/reference longitudinal speed, $v_x^{\text{ref}}[k+1]$, through the equation $v_x^{\text{ref}}[k+1] = v_x[k] + a_x[k]\Delta t$. We use $v_x(t)$, $t \in [k\Delta t, (k+1)\Delta t)$, to denote the actual longitudinal speed of the vehicle and define the normalized error between the reference speed and the actual speed as

$$e^v(t) = \frac{v_x^{\text{ref}}[k+1] - v_x(t)}{v_x^{\text{ref}}[k+1]}. \tag{1.52}$$

We normalize the error so that $e^v(t)$ is a dimensionless quantity. Then, the longitudinal control signal is calculated as

$$u^v(t) = \text{sat}_{[-1,1]}\left(k_{\text{p}}^v\, e^v(t) + k_{\text{i}}^v \int_{k\Delta t}^{t} e^v(\tau)\,\text{d}\tau + k_{\text{d}}^v\, \frac{\text{d}e^v}{\text{d}t}(t)\right). \tag{1.53}$$

If $u^v(t) \in [0, 1]$, we set the effector "accelerator" to the value of $u^v(t)$ and set "brake" to 0; if $u^v(t) \in [-1, 0)$, we set "brake" to the value of $-u^v(t)$ and set "accelerator" to 0. The PID gains, k_{p}^v, k_{i}^v and k_{d}^v, are calibrated to achieve satisfactory speed tracking performance.

1.4.2.2 Lateral motion control

At each sample time instant k, the action command generated from our driver model defines a target lane, $l_{\text{target}} \in \{\text{"right"}, \text{"middle"}, \text{"left"}\}$. The lateral motion controller has two tasks, including 1) lane keeping, if the target lane is the same as the current lane, and 2) lane change, if the target lane is different from the current lane.

We consider an angular error, $e^\phi(t)$, which is defined as the angle between the half-line extending from the vehicle's current position $(x(t), y(t))$ to a virtual reference point $(x^{\text{ref}}(t), y^{\text{ref}}(t))$ and the direction of the vehicle's current velocity $(v_x(t), v_y(t))$ (see Fig. 1.25), i.e.,

$$e^\phi(t) = \arctan\frac{y^{\text{ref}}(t) - y(t)}{x^{\text{ref}}(t) - x(t)} - \arctan\frac{v_y(t)}{v_x(t)}, \tag{1.54}$$

where the virtual reference point $(x^{\text{ref}}(t), y^{\text{ref}}(t))$ is determined according to

$$x^{\text{ref}}(t) = \begin{cases} x(t) + l_{\text{car}} + 0.01\,T\,v_x(t), & \text{if lane keeping,} \\ x(t) + 1.2\,l_{\text{car}} + 0.1\,T\,v_x(t), & \text{if lane change,} \end{cases}$$
$$y^{\text{ref}}(t) = y(k\Delta t) + \frac{t - k\Delta t}{T}\left(y^{\text{tar}}[k+1] - y(k\Delta t)\right), \tag{1.55}$$

for $t \in [k\Delta t, k\Delta t + T)$, where l_{car} denotes the length of the vehicle, T is a time constant approximately equal to the duration of a lane change, and $y^{\text{tar}}[k+1]$ represents the target lateral position, which is set as the center of the target lane.

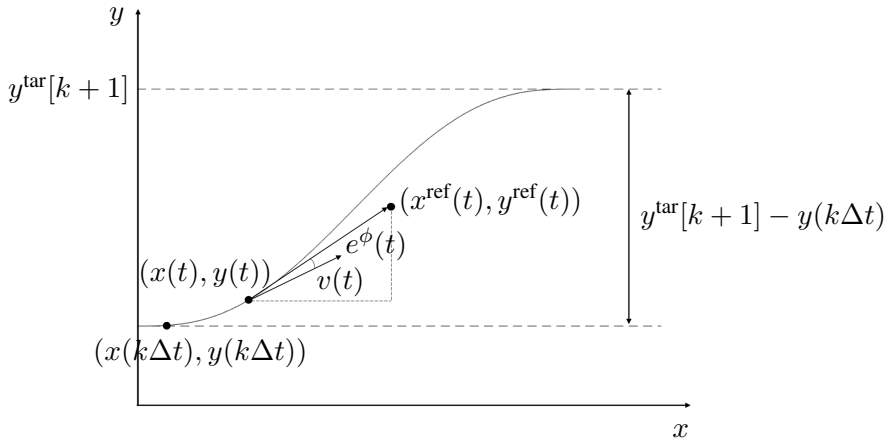


Figure 1.25: Schematics of the lateral motion control.

Note that $t = k\Delta t$ is the continuous time instant that corresponds to the sample time instant k, i.e., it represents the time instant when the action decision is made. For instance, in the case of lane change, $t = k\Delta t$ represents the starting time of lane change. The above $y^{\text{ref}}(t)$ is designed in such a way that $y^{\text{ref}}(k\Delta t) = y(k\Delta t)$ and $y^{\text{ref}}(k\Delta t + T) = y^{\text{ref}}[k+1]$, i.e., the reference point converges from the vehicle's current lateral position at the starting time of lane change to the target lateral position after a duration of T. This can promote smooth lane change maneuvers.

Then, the lateral control signal is calculated as

$$u^y(t) = \text{sat}_{[-1,1]}\left(k_{\text{p}}^y\, e^{\phi}(t) + k_{\text{d}}^y\, \frac{\text{d}e^{\phi}}{\text{d}t}(t)\right). \tag{1.56}$$

We set the effector "steering" to the value of $u^y(t)$. The PD gains, k_{p}^y and k_{d}^y, are calibrated to achieve satisfactory lane keeping and lane change performance.

Fig. 1.26 shows the controlled response of a TORCS vehicle during a lane change. The

blue solid curve represents the vehicle's $(x(t), y(t))$-trajectory during the lane change. The two red dashed lines represent the center of the right lane and the center of the middle lane, respectively. We can see that the vehicle moves from the right lane into the middle lane stably and smoothly. This validates the effectiveness of our designed longitudinal speed and lateral motion controllers.

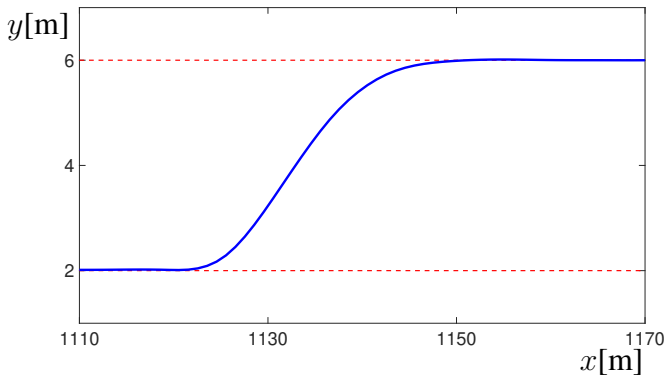


Figure 1.26: Controlled lane change response of a TORCS vehicle.

Fig. 1.27 shows a TORCS simulation after our integration, where the yellow vehicle in the middle is the test vehicle and all blue vehicles are controlled by our level-k driver models + the tracking controllers introduced above.

1.4.3 Intersection simulator and proof of concept of an adaptive level-k AV policy

In this section, we use our intersection model introduced in Section 1.3.1 and our leader-follower game based driver model introduced in Section 1.3.4 to form an intersection simulator and use it for proof of concept of an AV policy based on level-k driver models and an adaptive strategy to online model identification result, called an *adaptive level-k policy*. The GUI of the simulator is the same as those in Figs. 1.12-1.16, where any vehicle(s) in the simulation can now be either a test vehicle (a vehicle controlled by an autonomous driving algorithm under test) or an environmental vehicle (a vehicle controlled by our leader-follower game based driver model).

Firstly, Fig. 1.28 illustrates the computational cost of this intersection simulator. The numbers shown in Fig. 1.28 indicate the average and the worst-case computation times per vehicle per step (in [s] of real time), which are the average and the worst-case CPU times observed in our simulations for one environmental vehicle to confirm its action decision

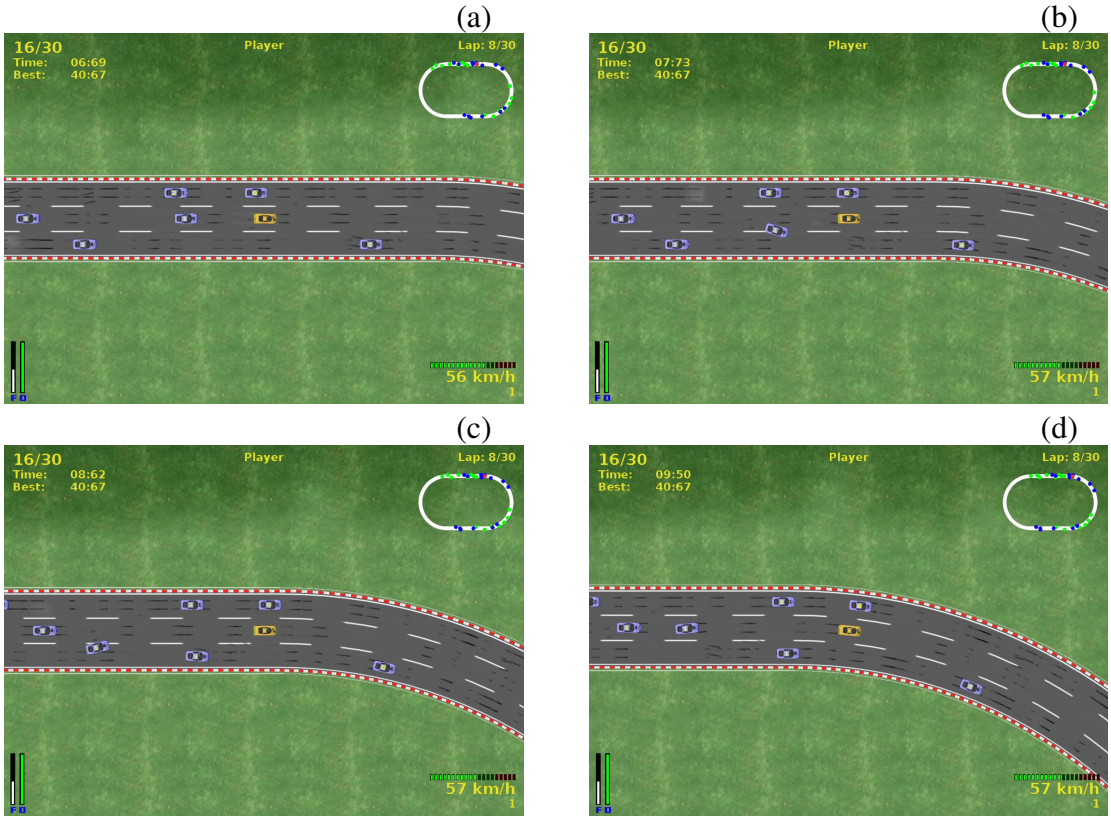


Figure 1.27: Snapshots of a TORCS simulation with our level-k driver models integrated.

for one step (including the time to solve for the initial acceleration decision according to (1.43) with a tree search-based method and the time to compute the modified acceleration decision according to Algorithm 4 when a deadlock occurs). Our simulation codes are written in MATLAB and run on a desktop with an Intel Core i7-4790 3.60 GHz processor and 16.0 GB RAM using the MATLAB R2019a platform. The CPU times in Fig. 1.28 are calculated using the MATLAB *tic-toc* command. A key observation is that the computation time increases only linearly as the number of vehicles increases, which is attributed to the pairwise decoupling of vehicle interactions via the pairwise leader-follower relationships in our model. The computational complexity of many conventional game-theoretic models, including Nash equilibrium-based models and standard Stackelberg equilibrium-based models, increases exponentially as the number of interacting agents increases, which makes them infeasible for modeling traffic scenarios involving multiple interacting vehicles. In contrast, the linear time complexity of our model ensures its feasibility for modeling complex intersection scenarios (for instance, Fig. 1.16 illustrates three simulated scenarios with 10 interacting vehicles).

The adaptive level-k policy was originally proposed in [48, 49], which could only han-

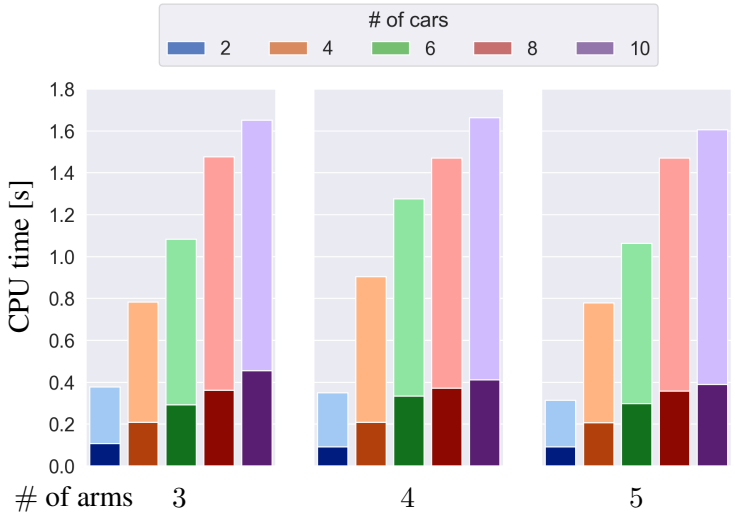


Figure 1.28: Average (dark-colored bars) and worst-case (light-colored bars) computation times per vehicle per step.

dle the interaction between the ego vehicle and one other vehicle. A generalized version that can handle the interactions between the ego vehicle and multiple other vehicles is presented in Appendix D and is tested with our simulator here.

We consider an intersection scenario involving three vehicles as shown in Fig. 1.29(a), where the blue and red vehicles are making left turns, and the green vehicle is going straight. At first, we let the blue vehicle be controlled by the adaptive level-k policy and let the red and green vehicles be controlled by our leader-follower game based driver model. Fig. 1.29 shows snapshots of the simulation at a series of time steps, and Fig. 1.30 shows the model identification history. The blue/red/green curves in Fig. 1.30 represent the estimated degrees of similarity between the observed behavioral pattern of the blue/red/green vehicle and the behavioral patterns of level-k driver models, with $k = 0, 1, 2$, which are interpreted as the confidences that the blue/red/green vehicle's driver/driving policy can be modeled as a level-k driver and hence its future behavior can be correctly predicted using a level-k model.

From Fig. 1.29 we can see that the blue vehicle controlled by the adaptive level-k policy successfully resolves its conflicts with the red and green vehicles and safely passes through the intersection. In particular, it yields to the green vehicle and proceeds before the red vehicle. As a matter of fact, in the scenario of Fig. 1.29(a), the green vehicle should have the right of way. According to our leader-follower role assignment logic in Algorithm 3 and the decision rule (1.43), the green vehicle will proceeds ahead. In this case, the blue

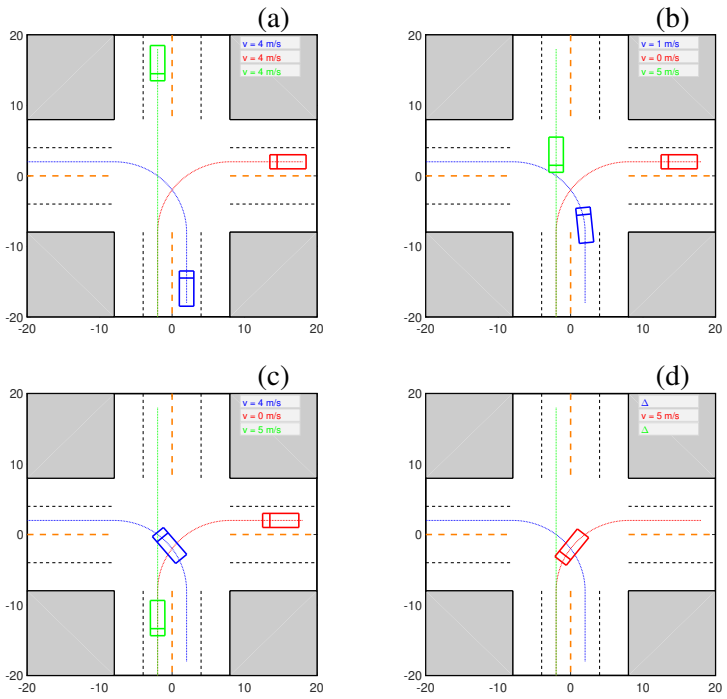


Figure 1.29: Simulation of an adaptive level-k vehicle (blue car) versus two leader-follower drivers (red and green cars). Figures (a-d) show snapshots of the simulation at a series of time steps.

vehicle identifies the green vehicle as 50% chance of being a level-0 vehicle and 50% chance of being a level-2 vehicle (shown in Fig. 1.30). Because both level-0 and level-2 vehicles represent aggressive vehicles, the blue vehicle decides to yield to the green vehicle. Meanwhile, it identifies the red vehicle as most probably being a level-1, conservative vehicle (see Fig. 1.30). Therefore, it decides to proceed before the red vehicle.

We then consider the same scenario but let the red and green vehicles be controlled by the adaptive level-k policy and let the blue vehicle be controlled by our leader-follower game based driver model. Fig. 1.31 shows snapshots of the simulation and Fig. 1.32 shows the model identification history. In this case, the green vehicle passes through the intersection first again, the red vehicle follows it and passes through the intersection second, and the blue vehicle yields to both of them. Comparing the results of Fig. 1.29 and Fig. 1.31 we can see that although the different combinations of policies (adaptive level-k versus leader-follower) lead to different passing orders, the adaptive level-k policy successfully resolves the conflicts between vehicles and navigates the test vehicle(s) to safely pass through the intersection in both cases. This proves the feasibility of concept of this adaptive level-k policy and also motivates us to extend and formalize this concept into an interaction-aware AV control framework in Chapter 2.

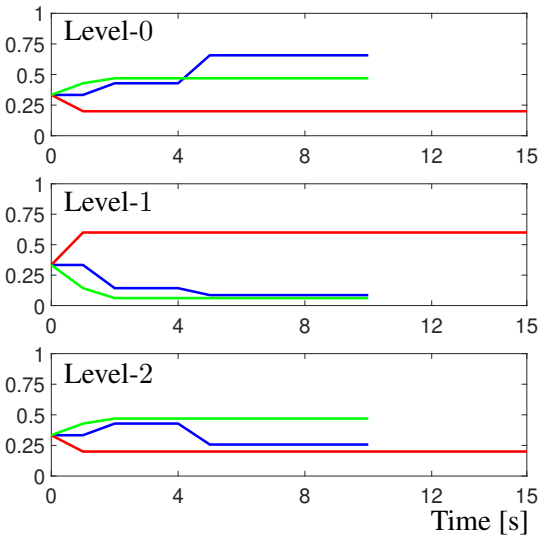


Figure 1.30: Model identification history corresponding to the simulation of Fig. 1.29.

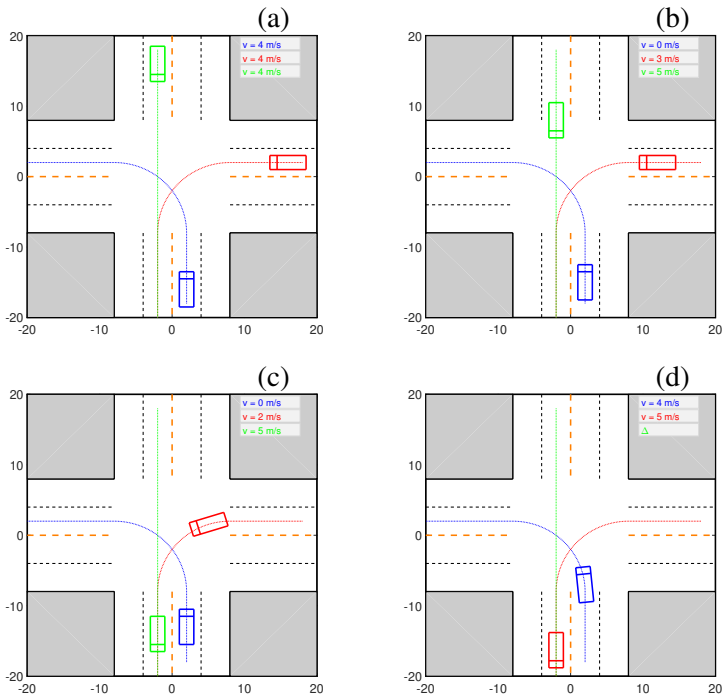


Figure 1.31: Simulation of two adaptive level-k vehicles (red and green cars) versus one leader-follower driver (blue car). Figures (a-d) show snapshots of the simulation at a series of time steps.

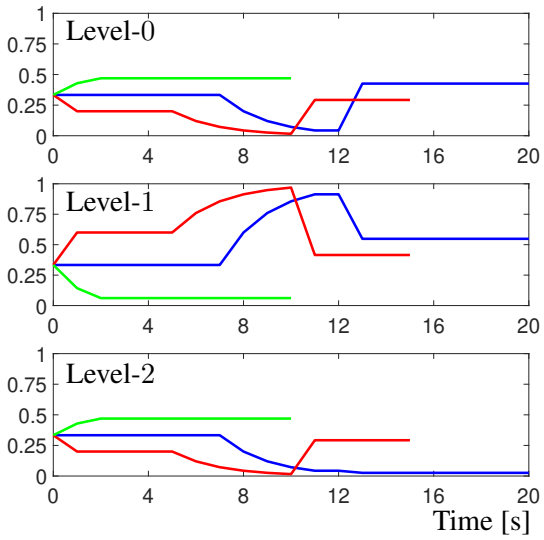


Figure 1.32: Model identification history corresponding to the simulation of Fig. 1.31.

1.5 Summary and discussion

In this chapter, we introduced two novel game-theoretic approaches to modeling driver interactions in traffic. Both approaches lead to interpretable driver interactive behavior models, and they both scale well in terms of being able to model medium-scale traffic scenarios involving tens of interacting vehicles at a reasonable computational cost.

In particular, the first level-k reasoning based approach focuses on the modeling of heterogeneous driving styles of real-world drivers and we have used it to model the driver interactions on highways where a driver's behavior is deeply influenced by her driving style; and the second leader-follower game based approach focuses on the modeling of driver intents and their resulting behaviors under common right-of-way traffic rules and we have used it to model the driver interactions at intersections where traffic rules and etiquette are observed to take a dominant role in determining drivers' behaviors. In this respect, these two approaches complement each other and can be mixed in a traffic model (similarly to what was done in Section 1.4.3).

The models produced by these approaches can have multiple applications. In Section 1.4, we introduced their application to simulation-based testing and V&V of AV control systems. This application addresses the urgent challenge of high-confidence validation of AV control systems and can support the reduction of AV time-to-market.

CHAPTER 2

Interaction-Aware Autonomous Vehicle Control

2.1 Background and introduction

In the near to medium term, autonomous vehicles (AVs) will operate on shared roads with human-driven vehicles [13]. One of the biggest challenges yet to be solved is behavior planning and control[1] for AVs in the presence of complex and uncertain interactions with other road users (such as human-driven vehicles) [3].

Conventional robust control approaches, including reachability-based techniques, can provide safe while conservative solutions through set-based modeling of all possible trajectories of other road users and enforcing the ego vehicle's behavior to respect them all [100, 101, 102, 103]. However, the conservativeness of these approaches may lead to sub-optimal AV behavior or even to standstill, especially in dense traffic and/or scenarios where cooperation among road users is needed (such as at a busy highway on-ramp or at a busy intersection) [104]. This motivates the developments of approaches for AVs to predicting the actions/reactions of surrounding road users and hence enabling less conservative control solutions, which is referred to as *interaction-aware control*.

One way to develop interaction-aware control policies is through reinforcement learning (RL), where the learning agent is put in an environment that has a reasonable representation of road user interactions and is trained by an RL algorithm. The obtained control policy is interaction-aware because it is an optimal policy for this interactive environment. Approaches along this line have been proposed in [105, 106, 107, 108, 109]. Drawbacks of these RL-based approaches include the lack of interpretability and, as a consequence, the lack of safety guarantee of the obtained AV control policies.

Another way is through explicitly incorporating a model (or a set of models) of road user interactions in the control algorithm, which typically leads to more interpretable control solutions [51, 52, 53, 54, 110, 111]. To handle uncertainties in road user interactions

[1] Collectively referred to as *control* in this chapter.

due to, e.g., varied driver styles (such as varied degrees of cooperativeness) and/or driving intents (such as to proceed versus to yield), the partially observable Markov decision process (POMDP) framework is a popular choice, where the uncertainties are modeled as latent variables and estimated based on observed trajectories [112, 113, 114, 115]. In these previous POMDP-based approaches, AV safety is promoted through penalty terms in the reward/cost function for vehicle collisions. This strategy also does not provide a formal safety guarantee (in terms of a percentage chance of safety/collision). Furthermore, POMDPs are in general difficult to solve exactly [116]. It can take several minutes to hours to solve a medium-sized POMDP problem[2] using conventional POMDP solvers based on point-based dynamic programming [117, 118]. In order to simplify computations and hence enable real-time decision making, [113] proposes a multi-policy strategy, where at every time step the AV evaluates and selects the best policy to execute from a finite set of pre-constructed policies that encode common closed-loop driving behaviors (such as lane keeping and lane change). The cost of such a multi-policy strategy for POMDPs is the loss of optimality.

In this chapter, we introduce a novel interaction-aware AV control approach. This approach has the following features:

1. This approach explicitly models the interaction between the ego vehicle and the traffic environment (which represents the unity of all other road users) and leads to interpretable AV control solutions.

2. This approach handles interaction uncertainties due to varied driver styles and intents by formulating the AV control problem into a POMDP problem with the uncertainties treated as latent states and estimated based on Bayesian inference.

3. This approach uses state constraints to represent critical safety requirements (such as collision avoidance) and leads to an explicit probabilistic safety guarantee (a guaranteed percentage chance of safety).

4. The formulated POMDP problem with state constraints (C-POMDP) is solved with an algorithm based on model predictive control using online optimization, called *POMDP-MPC*, which leads to both theoretically elegant and computationally feasible AV control solutions.

The developments of this chapter and related materials have been published in the journal article [119] and conference papers [120, 121, 122].

[2]Referring to problems involving a few hundreds of states.

2.2 Interaction-aware autonomous vehicle control problem

For a traffic scenario $\sigma \in \Sigma$, we represent the sequential decision-making problem for the ego vehicle as the following 6-tuple,

$$\left\langle \Lambda^\sigma, \mathcal{S}^\sigma, \bar{\mathcal{A}}^\sigma, \bar{T}^\sigma, R^\sigma, \mathcal{S}^\sigma_{\text{safe}} \right\rangle, \tag{2.1}$$

where $\Lambda^\sigma = \{1, 2\}$ represents the two decision-makers: 1 corresponds to the ego vehicle and 2 corresponds to the traffic environment; $\mathcal{S}^\sigma$ represents the space for traffic state $\mathbf{s}$; $\bar{\mathcal{A}}^\sigma = \mathcal{A}_1^\sigma \times \bar{\mathcal{A}}_2^\sigma$ with $\mathcal{A}_1^\sigma$ representing the action space of the ego vehicle and $\bar{\mathcal{A}}_2^\sigma$ representing the action space of the environment; $\bar{T}^\sigma$ is a stochastic map on $\mathcal{S}^\sigma \times \bar{\mathcal{A}}^\sigma \times \mathcal{S}^\sigma$ representing the transition of the traffic state $\mathbf{s} \to \mathbf{s}^+$ as a result of an action pair $(a_1, \bar{a}_2) \in \bar{\mathcal{A}}^\sigma$; $R^\sigma = R^\sigma(\mathbf{s}, a_1, \bar{a}_2, \mathbf{s}^+)$ is a reward function representing the decision objective(s) of the ego vehicle; and $\mathcal{S}^\sigma_{\text{safe}} \subset \mathcal{S}^\sigma$ represents a set of *safe* states for the ego vehicle.

In (2.1), we model the traffic environment as a decision-maker 2. When there is a single other vehicle in the environment, then 2 represents this other vehicle, $\bar{\mathcal{A}}_2^\sigma = \mathcal{A}_2^\sigma$ represents its action space, and $\bar{a}_2 = a_2$ represents its action. When there are multiple other vehicles $2, 3, \ldots, n_{\text{v}}$ in the environment, then 2 represents the set of all of these other vehicles, $\bar{\mathcal{A}}_2^\sigma = \mathcal{A}_2^\sigma \times \cdots \times \mathcal{A}_{n_{\text{v}}}^\sigma$ represents the the Cartesian product of their action spaces $\mathcal{A}_i^\sigma$, $i = 2, \ldots, n_{\text{v}}$, and $\bar{a}_2 = (a_2, \ldots, a_{n_{\text{v}}})$ represents the collection of their actions. We treat the set of all other vehicles as a single entity/decision-maker because this can significantly simplify the following exposition, while note that the approach can treat any number of vehicles n_{v}. Also, to further simplify expressions, we drop the superscript σ from all variables for the rest of this chapter, e.g., (2.1) reduces to

$$\left\langle \Lambda, \mathcal{S}, \bar{\mathcal{A}}, \bar{T}, R, \mathcal{S}_{\text{safe}} \right\rangle, \tag{2.2}$$

with $\bar{\mathcal{A}} = \mathcal{A}_1 \times \bar{\mathcal{A}}_2$.

We consider a control strategy for the ego vehicle that is based on the idea of receding-horizon optimal control or model predictive control (MPC) as follows: At each time step t, the ego vehicle solves the following problem,

$$\begin{aligned} \mathbf{a}^*_{1,t} = \left\{a^*_{1,0|t}, \ldots, a^*_{1,N-1|t}\right\} \in \underset{a_{1,\tau|t} \in \mathcal{A}_1}{\arg\max}\ & \mathbb{E}\left\{\sum_{\tau=0}^{N-1} \lambda^\tau R\left(\mathbf{s}_{\tau|t}, a_{1,\tau|t}, \bar{a}_{2,\tau|t}, \mathbf{s}_{\tau+1|t}\right)\right\}, \\ \text{subject to} \quad & \mathbb{P}\left\{\mathbf{s}_{\tau+1|t} \in \mathcal{S}_{\text{safe}},\ \forall \tau = 0, \ldots, N-1\right\} \geq 1 - \varepsilon, \end{aligned} \tag{2.3}$$

where a variable with a subscript $\tau|t$ indicates a predicted value of this variable at time $t+\tau$ with the prediction made at the current time t, $\lambda \in [0,1]$ is a discount factor, and $\varepsilon \in [0,1]$ is a *risk* parameter defining the required confidence level of satisfaction of the constraints $\mathbf{s}_{\tau+1|t} \in \mathcal{S}_{\text{safe}}$ over the planning horizon $\tau = 0, \ldots, N-1$. Note that in (2.3) we maximize the expectation of the cumulative reward $\sum_{\tau=0}^{N-1} \lambda^\tau R\left(\mathbf{s}_{\tau|t}, a_{1,\tau|t}, \bar{a}_{2,\tau|t}, \mathbf{s}_{\tau+1|t}\right)$ and enforce the constraints $\mathbf{s}_{\tau+1|t} \in \mathcal{S}_{\text{safe}}$ over the planning horizon probabilistically because the state transition, $\mathbf{s} \xrightarrow{(a_1,\bar{a}_2)} \mathbf{s}^+$, is assumed to be stochastic (in the most general case). After the optimal action sequence $\mathbf{a}_{1,t}^* = \{a_{1,0|t}^*, \ldots, a_{1,N-1|t}^*\}$ has been determined, the ego vehicle applies the first element for one time step to update its state, i.e., $a_{1,t} = a_{1,0|t}^*$. Then, at the next time step $t+1$, the ego vehicle repeats the above process.

It is clear that the problem (2.3) cannot be solved yet, because the variables $\bar{a}_{2,0|t}, \ldots, \bar{a}_{2,N-1|t}$ are neither controlled nor known. If we assume that a model of the environment, $\bar{\pi}_2$, is available, which is a stochastic map from $\mathcal{S}$ to $\bar{\mathcal{A}}_2$ and defines the probability of each action $\bar{a}_2 \in \bar{\mathcal{A}}_2$ to be taken by the environment at each traffic state $\mathbf{s} \in \mathcal{S}$ according to

$$\mathbb{P}(\bar{a}_2|\mathbf{s}) = \bar{\pi}_2(\mathbf{s}, \bar{a}_2), \tag{2.4}$$

then (2.3) can be solved, where the unknowns $\bar{a}_{2,0|t}, \ldots, \bar{a}_{2,N-1|t}$ are now treated as stochastic disturbances following the state-dependent distribution,

$$\bar{a}_{2,\tau|t} \sim \bar{\pi}_2(\mathbf{s}_{\tau|t}, \cdot). \tag{2.5}$$

We note that the decision process (2.3) is *interaction aware* due to the following two reasons: Firstly, we have modeled the distribution of environment action $\bar{a}_2$ over the planning horizon to be state-dependent according to (2.5). This means when the ego vehicle predicts its reward and safety for different action sequences over the planning horizon, the environment (which represents the other vehicles) will respond differently and correspondingly to the ego vehicle's action sequence. Therefore, the ego vehicle is aware of its interaction with the environment. Secondly, we will exploit game-theoretic driver models, such as the ones developed in Chapter 1, to construct the model $\bar{\pi}_2$ of the environment. In this case, $\bar{\pi}_2$ is a game-theoretic model, which accounts for the interaction between the ego vehicle and the environment.

For a traffic scenario with n_{v} vehicles (including the ego vehicle 1 and multiple other vehicles $2, \ldots, n_{\text{v}}$), such a model of the environment, $\bar{\pi}_2$, can be constructed based on individual driver models. For instance, suppose we have determined a driver model in the

form of (1.1) for each of the other vehicles, then $\bar{\pi}_2$ can be constructed as

$$\begin{aligned}
&\bar{\pi}_2(\mathbf{s}, \bar{a}_2) = \mathbb{P}(\bar{a}_2|\mathbf{s}) = \mathbb{P}(a_2, a_3, \ldots, a_{n_\mathrm{v}}|\mathbf{s}) \\
&= \prod_{i=2}^{n_\mathrm{v}} \mathbb{P}(a_i|\mathbf{s}) = \prod_{i=2}^{n_\mathrm{v}} \sum_{o_i \in \Omega} \mathbb{P}(a_i, o_i|\mathbf{s}) = \prod_{i=2}^{n_\mathrm{v}} \sum_{o_i \in \Omega} \mathbb{P}(o_i|\mathbf{s}) \mathbb{P}(a_i|o_i) \\
&= \prod_{i=2}^{n_\mathrm{v}} \sum_{o_i \in \Omega} O_i(\mathbf{s}, o_i) \pi_i(o_i, a_i),
\end{aligned} \tag{2.6}$$

where we have assumed that given the traffic state $\mathbf{s}$, each driver makes decisions individually and independently.

However, in real traffic, different human drivers may have different driving styles and/or intents. It is reasonable to construct a set of driver models, instead of a single one, to account for these different driving styles and intents. For instance, in Chapter 1 we use level-k driver models with different $k = 0, 1, \ldots$ to represent different driving styles. This will lead to a set of models of the environment, $\Pi_2 = \{\bar{\pi}_2^1, \bar{\pi}_2^2, \ldots, \bar{\pi}_2^{n_\pi}\}$, according to (2.6). Each model $\bar{\pi}_2^i$ leads to a different way for prediction of the environment actions $\bar{a}_{2,0|t}, \ldots, \bar{a}_{2,N-1|t}$ following (2.5) and hence leads to a possibly different optimal action sequence $\mathbf{a}_{1,t}^*$ for the ego vehicle through (2.3). In the following section, we introduce a method to incorporate the predictions of different environment models into the decision-making process (2.3).

2.3 Interaction-aware AV control as a partially observable decision problem

We assume that there is a model $\hat{\pi}_2$ within the model set $\Pi_2 = \{\bar{\pi}_2^1, \bar{\pi}_2^2, \ldots, \bar{\pi}_2^{n_\pi}\}$ that describes the actual way in which the environment takes actions $\bar{a}_2$ given a traffic state $\mathbf{s}$. We treat $\hat{\pi}_2$ as a *latent state* of the traffic system and consider the control problem for the ego vehicle as a decision-making problem under partial observability. Specifically, we define an augmented state of the traffic system as

$$\bar{\mathbf{s}} = (\mathbf{s}, \bar{\pi}_2) \in \mathcal{S} \times \Pi_2, \tag{2.7}$$

where only the *physical state* $\mathbf{s}$ is observable and the latent state $\bar{\pi}_2$ is unobservable.

We construct the transition model of the augmented state, T, based on the transition map of the physical state, $\bar{T} : \mathcal{S} \times (\mathcal{A}_1 \times \bar{\mathcal{A}}_2) \times \mathcal{S} \to [0, 1]$, and each of the environment

models, $\bar{\pi}_2 \in \Pi_2$, as follows,

$$
\begin{aligned}
&T(\bar{\mathbf{s}}, a_1, \bar{\mathbf{s}}^+) = \mathbb{P}(\bar{\mathbf{s}}^+|\bar{\mathbf{s}}, a_1) = \mathbb{P}\left((\mathbf{s}^+, \bar{\pi}_2^+)|(\mathbf{s}, \bar{\pi}_2), a_1\right) \\
&= \mathbb{P}\left(\mathbf{s}^+|\mathbf{s}, \bar{\pi}_2, a_1\right) \mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+) = \sum_{\bar{a}_2 \in \bar{\mathcal{A}}_2} \mathbb{P}\left(\mathbf{s}^+, \bar{a}_2|\mathbf{s}, \bar{\pi}_2, a_1\right) \mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+) \\
&= \sum_{\bar{a}_2 \in \bar{\mathcal{A}}_2} \mathbb{P}\left(\mathbf{s}^+|\mathbf{s}, \bar{\pi}_2, a_1, \bar{a}_2\right) \mathbb{P}\left(\bar{a}_2|\mathbf{s}, \bar{\pi}_2, a_1\right) \mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+) \\
&= \sum_{\bar{a}_2 \in \bar{\mathcal{A}}_2} \mathbb{P}\left(\mathbf{s}^+|\mathbf{s}, (a_1, \bar{a}_2)\right) \mathbb{P}\left(\bar{a}_2|\mathbf{s}, \bar{\pi}_2\right) \mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+) \\
&= \sum_{\bar{a}_2 \in \bar{\mathcal{A}}_2} \bar{T}\left(\mathbf{s}, (a_1, \bar{a}_2), \mathbf{s}^+\right) \bar{\pi}_2\left(\mathbf{s}, \bar{a}_2\right) \mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+),
\end{aligned} \tag{2.8}
$$

where $\mathbb{I}_{\bar{\pi}_2}(\bar{\pi}_2^+)$ is an indicator function, taking 1 if $\bar{\pi}_2^+ = \bar{\pi}_2$ and taking 0 if $\bar{\pi}_2^+ \neq \bar{\pi}_2$. In deriving (2.8), we have made the following three assumptions: 1) The other vehicles do not change their driving styles/intents during their interaction with the ego vehicle. This implies $\bar{\pi}_2^+ = \bar{\pi}_2$ and leads to the second line from the first line. 2) The transition probabilities of $\mathbf{s} \to \mathbf{s}^+$ are determined by which actions $(a_1, \bar{a}_2)$ are taken by the ego vehicle and the environment and does not depend on any other variables. 3) The probability of each action $\bar{a}_2$ to be taken by the environment at a given traffic state $\mathbf{s}$ is determined by the environment model $\bar{\pi}_2$ and does not depend on any other variables. The combination of 2) and 3) leads to the fourth line from the third line. The above model can be extended to handle the case where which model in Π_2 most accurately describes the actual behavioral pattern of the environment may change over time. Such a model change may be due to, e.g., a driver's change of intents. This extension can be achieved by considering probabilistic transitions of the model dependent on the physical state, i.e., $\mathbb{P}(\bar{\pi}_2^+|\bar{\pi}_2, \mathbf{s})$, and adjusting (2.8) accordingly.

Also, we define the following observation model $O : (\mathcal{S} \times \Pi_2) \times \mathcal{S} \to [0, 1]$,

$$
O(\bar{\mathbf{s}}, \mathbf{s}') = \mathbb{P}\left(\mathbf{s}'|\bar{\mathbf{s}}\right) = \mathbb{P}\left(\mathbf{s}'|(\mathbf{s}, \bar{\pi}_2)\right) = \mathbb{I}_{\mathbf{s}}(\mathbf{s}'), \tag{2.9}
$$

where $\mathbb{I}_{\mathbf{s}}(\mathbf{s}')$ is an indicator function, taking 1 if $\mathbf{s}' = \mathbf{s}$ and taking 0 if $\mathbf{s}' \neq \mathbf{s}$. This observation model corresponds to the assumption that the physical state $\mathbf{s}$ can be fully and perfectly measured by the ego vehicle. The case where the ego vehicle can only partially observe the physical state $\mathbf{s}$ due to, e.g., limited perception range, line of sight obstruction, and/or sensor noise, can be handled by making corresponding adjustments to the observation model (2.9).

We now resemble the sequential decision-making problem for the ego vehicle as the

following 6-tuple,

$$\left\langle \bar{\mathcal{S}}, \mathcal{A}_1, T, \Omega, O, R, \bar{\mathcal{S}}_{\text{safe}} \right\rangle, \tag{2.10}$$

where $\bar{\mathcal{S}} = \mathcal{S} \times \Pi_2$ is the space for the augmented state of the traffic system $\bar{\mathbf{s}} = (\mathbf{s}, \bar{\pi}_2)$; $\mathcal{A}_1$ is the action space; $\Omega = \mathcal{S}$ is the observation space, which is equal to the space for the physical traffic state $\mathbf{s}$; T is the state transition map defined in (2.8); O is the observation map defined in (2.9); R is the reward function defined as in (2.2); and $\bar{\mathcal{S}}_{\text{safe}} = \mathcal{S}_{\text{safe}} \times \Pi_2$ is the safe set for the augmented state $\bar{\mathbf{s}}$. This problem is referred to as a partially observable Markov decision process problem with state constraints (C-POMDP). In the next section, we first present a solution approach to general C-POMDP problems and then introduce the adaptation of this general approach to our specific autonomous vehicle control problem (2.10).

2.4 An MPC-based solution approach to C-POMDP problems (POMDP-MPC)

2.4.1 An approach to general C-POMDP problems

In this section, we consider C-POMDP problems represented as the following 6-tuple,

$$\langle \mathcal{S}, \mathcal{A}, T, \Omega, O, R, \mathcal{S}_{\text{safe}} \rangle. \tag{2.11}$$

Throughout this section, we use $s_t \in \mathcal{S}$ to denote the state at the time step $t \in \mathbb{N}_0$, $a_t \in \mathcal{A}$ to denote the action taken at t, and $o_t \in \Omega$ to denote the observation received at t. The maps $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and $O : \mathcal{S} \times \Omega \to [0, 1]$ define the state transition probabilities and observation probabilities according to

$$\begin{aligned} \mathbb{P}(s_{t+1}|s_t, a_t) &= T(s_t, a_t, s_{t+1}), \\ \mathbb{P}(o_t|s_t) &= O(s_t, o_t), \end{aligned} \tag{2.12}$$

for all $t \in \mathbb{N}_0$. In particular, we assume that the probabilities of s_{t+1} conditioned on a given pair of (s_t, a_t) are independent of any other variables and that the probabilities of o_t conditioned on a given s_t are independent of any other variables. They are referred to as the Markov property.

At each time step $t \in \mathbb{N}_0$, we collect the information of all received observations o_τ for $\tau = 0, 1, \ldots, t$ and all previously applied actions a_τ for $\tau = 0, \ldots, t-1$ into the following

data vector,

$$\xi_t = \{o_0, \ldots, o_t, a_0, \ldots, a_{t-1}\}. \tag{2.13}$$

The basic idea of our approach to (2.11) is based on repeatedly solving the following problem at every time step t:

$$\mathbf{a}_t^* = \left\{a_{0|t}^*, \ldots, a_{N-1|t}^*\right\} \in \underset{a_{\tau|t}\in\mathcal{A}}{\arg\max}\ \mathbb{E}\left\{\sum_{\tau=0}^{N-1} \lambda^\tau R\left(s_{\tau|t}, a_{\tau|t}, s_{\tau+1|t}\right) \middle| \xi_t\right\}, \tag{2.14a}$$

$$\text{subject to} \quad \mathbb{P}\left\{\bigwedge_{\tau=0}^{N-1} \left(s_{\tau+1|t} \in \mathcal{S}_{\text{safe}}\right) \middle| \xi_t\right\} \geq 1-\varepsilon, \tag{2.14b}$$

where $\wedge$ designates the "and" operator, and the risk parameter $\varepsilon \in [0, 1]$ determines the required confidence level of constraint satisfaction over the planning horizon. Note that we allow ε to take 0, which represents a robust constraint satisfaction requirement for safety-critical cases. Note also that the expectation in (2.14) and the probability in (2.14a) are conditioned on the current data vector ξ_t to represent the "best" (i.e., posterior) estimates of reward and constraint satisfaction after taking into account all available information.

There are two difficulties associated with the above problem (2.14): Firstly, one needs to answer how to evaluate the objective function (2.14a) and the probabilistic constraint (2.14b) for any solution candidate $\mathbf{a}_t = \left\{a_{0|t}, \ldots, a_{N-1|t}\right\}$. Secondly, one needs to answer how to evolve a solution candidate $\mathbf{a}_t$ according to objective function and constraint evaluations to approach a feasible and optimal solution. Due to the discrete nature of the action space $\mathcal{A}$, (2.14) is a discrete optimization problem. Any exact solution approaches to discrete optimization problems involve one-by-one evaluation and comparison of all solution candidates, which has combinatorial worst-case complexity and hence does not scale well. Therefore, in what follows we introduce an approximate solution approach to (2.14) that relies on a continuous relaxation technique.

To begin with, we define the *belief state* at the time step t, b_t, as the posterior probability distribution of the state s_t conditioned on all available data, i.e.,

$$b_t(s) = \mathbb{P}(s_t = s|\xi_t). \tag{2.15}$$

As time evolves from t to $t+1$, the belief state b_t can be updated according to the following

recursive Bayesian estimation formula,

$$
\begin{aligned}
&b_{t+1}(s') = \mathbb{P}(s_{t+1} = s'|\xi_{t+1}) = \mathbb{P}(s_{t+1} = s'|\xi_t, o_{t+1}, a_t) \\
&= \frac{\mathbb{P}(o_{t+1}|s_{t+1} = s', \xi_t, a_t)\mathbb{P}(s_{t+1} = s'|\xi_t, a_t)}{\mathbb{P}(o_{t+1}|\xi_t, a_t)} \\
&= \frac{\mathbb{P}(o_{t+1}|s_{t+1} = s', \xi_t, a_t)\sum_{s\in\mathcal{S}}\left(\mathbb{P}(s_{t+1} = s'|s_t = s, \xi_t, a_t)\mathbb{P}(s_t = s|\xi_t, a_t)\right)}{\sum_{s''\in\mathcal{S}}\left(\mathbb{P}(o_{t+1}|s_{t+1} = s'', \xi_t, a_t)\mathbb{P}(s_{t+1} = s''|\xi_t, a_t)\right)} \\
&= \frac{\mathbb{P}(o_{t+1}|s_{t+1} = s')\sum_{s\in\mathcal{S}}\left(\mathbb{P}(s_{t+1} = s'|s_t = s, a_t)\mathbb{P}(s_t = s|\xi_t)\right)}{\sum_{s''\in\mathcal{S}}\left(\mathbb{P}(o_{t+1}|s_{t+1} = s'')\sum_{s\in\mathcal{S}}\left(\mathbb{P}(s_{t+1} = s''|s_t = s, a_t)\mathbb{P}(s_t = s|\xi_t)\right)\right)} \\
&= \frac{O(s', o_{t+1})\sum_{s\in\mathcal{S}}\left(T(s, a_t, s')b_t(s)\right)}{\sum_{s''\in\mathcal{S}}\left(O(s'', o_{t+1})\sum_{s\in\mathcal{S}}\left(T(s, a_t, s'')b_t(s)\right)\right)}.
\end{aligned} \tag{2.16}
$$

The recursive update (2.16) starts with $b_0(s) = \mathbb{P}(s_0 = s|\xi_0)$, which is computed based on a prior distribution of the initial state s_0, $b_0^-(s) = \mathbb{P}(s_0 = s)$, according to

$$
b_0(s') = \mathbb{P}(s_0 = s'|o_0) = \frac{\mathbb{P}(o_0|s_0 = s')\mathbb{P}(s_0 = s')}{\sum_{s\in\mathcal{S}}\mathbb{P}(o_0|s_0 = s)\mathbb{P}(s_0 = s)} = \frac{O(s', o_0)b_0^-(s')}{\sum_{s\in\mathcal{S}}O(s, o_0)b_0^-(s)}. \tag{2.17}
$$

Meanwhile, instead of considering deterministic actions a_t, we consider probabilistic action policies, $\gamma_t : \mathcal{A} \to [0, 1]$, such that at each time step t,

$$
\mathbb{P}(a_t = a) = \gamma_t(a). \tag{2.18}
$$

We assume that given an action policy γ_t, the probabilities of a_t are independent of any other variables. This is a reasonable assumption, since a_t is a decision variable, i.e., one can define the way in which a_t takes different values. Note also that after numbering the elements of $\mathcal{A}$ as $a^1, a^2, \ldots, a^{|\mathcal{A}|}$, the map γ_t can be expressed as an $|\mathcal{A}|$-dimensional probability vector,

$$
\gamma_t = \begin{bmatrix}\gamma_t(a^1) & \cdots & \gamma_t(a^{|\mathcal{A}|})\end{bmatrix}^\top. \tag{2.19}
$$

Then, we transfer the original problem (2.14) into the following problem:

$$
\Gamma_t^* = \left\{\gamma_{0|t}^*, \ldots, \gamma_{N-1|t}^*\right\} \in \underset{\gamma_{\tau|t}\in\Delta(\mathcal{A})}{\arg\max}\ \mathbb{E}\left\{\sum_{\tau=0}^{N-1}\lambda^\tau R\left(s_{\tau|t}, a_{\tau|t}, s_{\tau+1|t}\right)\middle|\xi_t\right\}, \tag{2.20a}
$$

$$
\text{subject to}\quad \mathbb{P}\left\{\bigwedge_{\tau=0}^{N-1}\left(s_{\tau+1|t}\in\mathcal{S}_{\text{safe}}\right)\middle|\xi_t\right\} \geq 1-\varepsilon, \tag{2.20b}
$$

where $\Delta(\mathcal{A}) = \left\{\gamma \in [0, 1]^{|\mathcal{A}|} \,|\, \gamma^\top \mathbf{1}_{|\mathcal{A}|} = 1\right\}$ is the $(|\mathcal{A}| - 1)$-dimensional probability simplex [123].

We note that (2.20) can be viewed as a continuously relaxed version of the original discrete optimization problem (2.14). Specifically, (2.14) is almost surely equivalent to (2.20) with the additional requirement that the action policies $\gamma_{\tau|t}$ must be *deterministic policies*, which correspond to zero-one probability vectors (2.19)[3]. Meanwhile, (2.20) is "continuously relaxed" because $\gamma_{\tau|t}$ can take any probability vectors in (2.20). We now discuss the approach to solving (2.20).

Firstly, given a predicted sequence of action policies $\{\gamma_{0|t}, \ldots, \gamma_{N-1|t}\}$, one can predict the distributions of state over the planning horizon, $b_{\tau|t}(s) = \mathbb{P}(s_{\tau|t} = s|\xi_t)$ for $\tau = 1, \ldots, N$, using the following recursive formula,

$$\begin{aligned}
&b_{\tau+1|t}(s') = \mathbb{P}(s_{\tau+1|t} = s'|\xi_t) \\
&= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \big(\mathbb{P}(s_{\tau+1|t} = s'|s_{\tau|t} = s, a_{\tau|t} = a, \xi_t)\mathbb{P}(a_{\tau|t} = a|s_{\tau|t} = s, \xi_t)\mathbb{P}(s_{\tau|t} = s|\xi_t)\big) \\
&= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \big(\mathbb{P}(s_{\tau+1|t} = s'|s_{\tau|t} = s, a_{\tau|t} = a)\mathbb{P}(a_{\tau|t} = a)\mathbb{P}(s_{\tau|t} = s|\xi_t)\big) \\
&= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \big(T(s, a, s')\gamma_{\tau|t}(a)b_{\tau|t}(s)\big)\,,
\end{aligned} \tag{2.21}$$

with the initial condition $b_{0|t} = b_t$.

If we express the state distributions $b_{\tau|t}$ as $|\mathcal{S}|$-dimensional probability vectors,

$$b_{\tau|t} = \Big[b_{\tau|t}\big(s^1\big) \quad \cdots \quad b_{\tau|t}\big(s^{|\mathcal{S}|}\big)\Big]^\top, \tag{2.22}$$

then (2.21) can be written in the following vector-matrix form,

$$b_{\tau+1|t} = \big(I_{|\mathcal{S}|} \otimes (\gamma_{\tau|t})^\top\big)\ \mathrm{diag}\left(\mathbf{T}\big(s^1\big), \ldots, \mathbf{T}\big(s^{|\mathcal{S}|}\big)\right)\ \left(\mathbf{1}_{|\mathcal{S}|} \otimes b_{\tau|t}\right), \tag{2.23}$$

where $I_{|\mathcal{S}|}$ denotes the $|\mathcal{S}|$-by-$|\mathcal{S}|$ identity matrix, $\mathbf{1}_{|\mathcal{S}|}$ denotes the $|\mathcal{S}|$-dimensional vector of all ones, $\otimes$ designates the Kronecker product operator, and

$$\mathbf{T}\big(s^i\big) = \begin{bmatrix} T(s^1, a^1, s^i) & \cdots & T(s^{|\mathcal{S}|}, a^1, s^i) \\ \vdots & \ddots & \vdots \\ T(s^1, a^{|\mathcal{A}|}, s^i) & \cdots & T(s^{|\mathcal{S}|}, a^{|\mathcal{A}|}, s^i) \end{bmatrix}. \tag{2.24}$$

Then, for a given sequence of action policies $\{\gamma_{0|t}, \ldots, \gamma_{N-1|t}\}$, the objective function

[3]With precisely one element being 1 and all other elements being 0.

(2.20a) can be evaluated according to

$$\mathbb{E}\left\{\sum_{\tau=0}^{N-1}\lambda^{\tau}R\left(s_{\tau|t},a_{\tau|t},s_{\tau+1|t}\right)\Big|\xi_t\right\}=\sum_{\tau=0}^{N-1}\lambda^{\tau}\mathbb{E}\left\{R\left(s_{\tau|t},a_{\tau|t},s_{\tau+1|t}\right)\big|\xi_t\right\}$$
$$=\sum_{\tau=0}^{N-1}\lambda^{\tau}\left[\sum_{s,s'\in\mathcal{S},\,a\in\mathcal{A}}\left(R\left(s,a,s'\right)\mathbb{P}\left\{s_{\tau|t}=s,a_{\tau|t}=a,s_{\tau+1|t}=s'\big|\xi_t\right\}\right)\right], \tag{2.25}$$

where

$$\begin{aligned}&\mathbb{P}\left\{s_{\tau|t}=s,a_{\tau|t}=a,s_{\tau+1|t}=s'\big|\xi_t\right\}\\&=\mathbb{P}\left\{s_{\tau+1|t}=s'\big|s_{\tau|t}=s,a_{\tau|t}=a,\xi_t\right\}\mathbb{P}\left\{a_{\tau|t}=a\big|s_{\tau|t}=s,\xi_t\right\}\mathbb{P}\left\{s_{\tau|t}=s\big|\xi_t\right\}\\&=\mathbb{P}\left\{s_{\tau+1|t}=s'\big|s_{\tau|t}=s,a_{\tau|t}=a\right\}\mathbb{P}\left\{a_{\tau|t}=a\right\}\mathbb{P}\left\{s_{\tau|t}=s\big|\xi_t\right\}\\&=T(s,a,s')\gamma_{\tau|t}(a)b_{\tau|t}(s).\end{aligned} \tag{2.26}$$

in which $b_{\tau|t}$, for $\tau=0,\ldots,N-1$, are recursively computed using (2.21).

The above expressions (2.25) and (2.26) immediately imply the following result:

Proposition 2.1: The objective function (2.20a) has the following two properties: (i) It is a smooth (i.e., infinitely differentiable) function of the decision variables $\left\{\gamma_{0|t},\ldots,\gamma_{N-1|t}\right\}$. (ii) It depends on the data vector ξ_t only through the belief state b_t.

Proof: The proof follows from (2.21), (2.25), and (2.26). ■

The above property (ii) is called *separable* in some POMDP literature [124].

We now discuss the way to explicitly evaluate the constraint (2.20b) for any given sequence of action policies. For this, we consider the events $V_{\tau|t}$ for $\tau=1,\ldots,N$, which are defined as

$$V_{\tau|t}=\bigvee_{\sigma=1}^{\tau}\left(s_{\sigma|t}\in\mathcal{S}_{\text{safe}}^{c}\right), \tag{2.27}$$

where $\vee$ designates the "or" operator, and $\mathcal{S}_{\text{safe}}^{c}$ denotes the complement of the safe set $\mathcal{S}_{\text{safe}}$, i.e., $\mathcal{S}_{\text{safe}}^{c}=\mathcal{S}\setminus\mathcal{S}_{\text{safe}}$. Similarly, we use $V_{\tau|t}^{c}$ to denote the complement of $V_{\tau|t}$, i.e.,

$$V_{\tau|t}^{c}=\bigwedge_{\sigma=1}^{\tau}\left(s_{\sigma|t}\in\mathcal{S}_{\text{safe}}\right). \tag{2.28}$$

Note that the constraint (2.20b) can be equivalently expressed as

$$\mathbb{P}\left(V_{N|t}\big|\xi_t\right)\leq\varepsilon. \tag{2.29}$$

Then, for a given sequence of action policies $\left\{\gamma_{0|t},\ldots,\gamma_{N-1|t}\right\}$, one can evaluate (2.29)

using the following theorem:

Theorem 2.1: The probabilities $\mathbb{P}\left(V_{\tau|t}\middle|\xi_t\right)$, for $\tau = 1, \ldots, N$, can be recursively computed as follows:

$$\mathbb{P}\left(V_{\tau|t}\middle|\xi_t\right) = \mathbb{P}\left(V_{\tau-1|t}\middle|\xi_t\right) + \sum_{s' \in \mathcal{S}^c_{\text{safe}}} \mathbb{P}\left((s_{\tau|t} = s') \wedge V^c_{\tau-1|t}\middle|\xi_t\right),$$

$$\mathbb{P}\left((s_{\tau|t} = s') \wedge V^c_{\tau-1|t}\middle|\xi_t\right) = \sum_{s \in \mathcal{S}} \left(\sum_{a \in \mathcal{A}} T(s, a, s') \gamma_{\tau-1|t}(a) \right) \mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t}\middle|\xi_t\right),$$

$$\mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t}\middle|\xi_t\right) = \mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-2|t}\middle|\xi_t\right) \mathbb{I}_{\mathcal{S}_{\text{safe}}}(s), \tag{2.30}$$

with the initial conditions $\mathbb{P}(V_{0|t}|\xi_t) = 0$ and $\mathbb{P}((s_{0|t} = s) \wedge V^c_{0|t}|\xi_t) = \mathbb{P}(s_{0|t} = s|\xi_t) = b_t(s)$.

Proof: Firstly, we have

$$\begin{aligned}\mathbb{P}\left(V_{\tau|t}\middle|\xi_t\right) &= \underbrace{\mathbb{P}\left(V_{\tau|t}\middle|V_{\tau-1|t}, \xi_t\right)}_{=1} \mathbb{P}\left(V_{\tau-1|t}\middle|\xi_t\right) + \mathbb{P}\left(V_{\tau|t} \wedge V^c_{\tau-1|t}\middle|\xi_t\right) \\ &= \mathbb{P}\left(V_{\tau-1|t}\middle|\xi_t\right) + \sum_{s' \in \mathcal{S}} \mathbb{P}\left((s_{\tau|t} = s') \wedge V_{\tau|t} \wedge V^c_{\tau-1|t}\middle|\xi_t\right), \end{aligned} \tag{2.31}$$

where

$$\mathbb{P}\left((s_{\tau|t} = s') \wedge V_{\tau|t} \wedge V^c_{\tau-1|t}\middle|\xi_t\right) = \begin{cases} 0 & \text{if} \quad s' \in \mathcal{S}_{\text{safe}}, \\ \mathbb{P}\big((s_{\tau|t} = s') \wedge V^c_{\tau-1|t}\big|\xi_t\big) & \text{if} \quad s' \in \mathcal{S}^c_{\text{safe}}. \end{cases} \tag{2.32}$$

Therefore, we obtain

$$\mathbb{P}\left(V_{\tau|t}\middle|\xi_t\right) = \mathbb{P}\left(V_{\tau-1|t}\middle|\xi_t\right) + \sum_{s' \in \mathcal{S}^c_{\text{safe}}} \mathbb{P}\left((s_{\tau|t} = s') \wedge V^c_{\tau-1|t}\middle|\xi_t\right). \tag{2.33}$$

We also have

$$\begin{aligned}\mathbb{P}\left((s_{\tau|t} = s') \wedge V^c_{\tau-1|t}\middle|\xi_t\right) &= \sum_{s \in \mathcal{S}} \mathbb{P}\left((s_{\tau|t} = s') \wedge (s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t}\middle|\xi_t\right) \\ = \sum_{s \in \mathcal{S}} \mathbb{P}\left(s_{\tau|t} = s'\middle|s_{\tau-1|t} = s, V^c_{\tau-1|t}, \xi_t\right) &\mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t}\middle|\xi_t\right), \end{aligned} \tag{2.34}$$

where

$$\begin{aligned}
&\mathbb{P}\left(s_{\tau|t} = s' \middle| s_{\tau-1|t} = s, V^c_{\tau-1|t}, \xi_t\right) \\
&= \sum_{a\in\mathcal{A}} \mathbb{P}\left(s_{\tau|t} = s' \middle| s_{\tau-1|t} = s, a_{\tau-1|t} = a, V^c_{\tau-1|t}, \xi_t\right) \mathbb{P}\left(a_{\tau-1|t} = a \middle| s_{\tau-1|t} = s, V^c_{\tau-1|t}, \xi_t\right) \\
&= \sum_{a\in\mathcal{A}} \mathbb{P}\left(s_{\tau|t} = s' \middle| s_{\tau-1|t} = s, a_{\tau-1|t} = a\right) \mathbb{P}\left(a_{\tau-1|t} = a\right) \\
&= \sum_{a\in\mathcal{A}} T(s,a,s')\gamma_{\tau-1|t}(a). \qquad (2.35)
\end{aligned}$$

Finally, we have

$$\begin{aligned}
&\mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t} \middle| \xi_t\right) \\
&= \underbrace{\mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t} \wedge V_{\tau-2|t} \middle| \xi_t\right)}_{=0} + \mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t} \wedge V^c_{\tau-2|t} \middle| \xi_t\right) \\
&= \begin{cases} \mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-2|t} \middle| \xi_t\right) & \text{if} \quad s \in \mathcal{S}_{\text{safe}}, \\ 0 & \text{if} \quad s \in \mathcal{S}^c_{\text{safe}}, \end{cases} \qquad (2.36)
\end{aligned}$$

which can be expressed as

$$\mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-1|t} \middle| \xi_t\right) = \mathbb{P}\left((s_{\tau-1|t} = s) \wedge V^c_{\tau-2|t} \middle| \xi_t\right) \mathbb{I}_{\mathcal{S}_{\text{safe}}}(s). \qquad (2.37)$$

This completes the proof. ■

The following Algorithm 5 represents an implementation of the recursive process (2.30) for evaluating the constraint function $\mathbb{P}(V_{N|t}|\xi_t)$.

Algorithm 5: Probabilistic Safety Constraint Evaluation

Initialize $p^{\text{v}} = 0$ and $b^{\text{v}}_{0|t} = b_t$
for $\tau = 0 : N-1$ **do**
  $b^{\text{v}}_{\tau+1|t} = \left(I_{|\mathcal{S}|} \otimes (\gamma_{\tau|t})^\top\right) \text{ diag } \left(\mathbf{T}\left(s^1\right), \ldots, \mathbf{T}\left(s^{|\mathcal{S}|}\right)\right) \left(\mathbf{1}_{|\mathcal{S}|} \otimes b^{\text{v}}_{\tau|t}\right)$
  Update $p^{\text{v}} \leftarrow p^{\text{v}} + \sum_{s\in\mathcal{S}^c_{\text{safe}}} b^{\text{v}}_{\tau+1|t}(s)$
  Update $b^{\text{v}}_{\tau+1|t}(s) \leftarrow 0$ for all $s \in \mathcal{S}^c_{\text{safe}}$
end
Output $\mathbb{P}(V_{N|t}|\xi_t) = p^{\text{v}}$.

Theorem 2.1 also leads to the following result:

Proposition 2.2: The constraint function (2.29) has the following two properties: (i) It is a multi-affine (hence, smooth) function of the decision variables $\left\{\gamma_{0|t}, \ldots, \gamma_{N-1|t}\right\}$, i.e.,

affine in $\gamma_{\tau|t}$ for each $\tau = 0, \ldots, N-1$. (ii) It depends on the data vector ξ_t only through the belief state b_t, i.e., $\mathbb{P}(V_{N|t}|\xi_t) = \mathbb{P}(V_{N|t}|b_t)$.

Proof: We prove (i) by induction. Firstly, $\mathbb{P}(V_{0|t}|\xi_t) = 0$ and $\mathbb{P}((s_{0|t} = s) \wedge V_{0|t}^c|\xi_t) = b_t(s)$, for all $s \in \mathcal{S}$, are independent of, and hence trivially affine, in $\gamma_{\sigma|t}$, for each $\sigma = 0, \ldots, N-1$. This forms the basis of induction.

We now assume that $\mathbb{P}(V_{\tau-1|t}|\xi_t)$ and $\mathbb{P}((s_{\tau-1|t} = s) \wedge V_{\tau-1|t}^c|\xi_t)$, for all $s \in \mathcal{S}$, are affine in $\gamma_{\sigma|t}$, for each $\sigma = 0, \ldots, N-1$. We are going to show that in this case, $\mathbb{P}(V_{\tau|t}|\xi_t)$ and $\mathbb{P}((s_{\tau|t} = s) \wedge V_{\tau|t}^c|\xi_t)$, for all $s \in \mathcal{S}$, are also affine in $\gamma_{\sigma|t}$, for each $\sigma = 0, \ldots, N-1$. Indeed, we only need to show this for $\gamma_{\sigma|t}$ with $\sigma = 0, \ldots, \tau-1$, because $\mathbb{P}(V_{\tau|t}|\xi_t)$ and $\mathbb{P}((s_{\tau|t} = s) \wedge V_{\tau|t}^c|\xi_t)$ are independent of $\gamma_{\sigma|t}$ for $\sigma = \tau, \ldots, N-1$ due to causality.

Let $s' \in \mathcal{S}$ be arbitrary. Note that for all $s \in \mathcal{S}$, $\sum_{a\in\mathcal{A}} T(s,a,s')\gamma_{\tau-1|t}(a)$ is linear in $\gamma_{\tau-1|t}$ and $\mathbb{P}((s_{\tau-1|t} = s) \wedge V_{\tau-1|t}^c|\xi_t)$ is independent of $\gamma_{\tau-1|t}$ due to causality. Then, for $\sigma = \tau-1$, $\mathbb{P}((s_{\tau|t} = s') \wedge V_{\tau-1|t}^c|\xi_t) = \sum_{s\in\mathcal{S}}(\sum_{a\in\mathcal{A}} T(s,a,s')\gamma_{\tau-1|t}(a))\mathbb{P}((s_{\tau-1|t} = s) \wedge V_{\tau-1|t}^c|\xi_t)$ is linear, hence affine, in $\gamma_{\sigma|t} = \gamma_{\tau-1|t}$. For $\sigma = 0, \ldots, \tau-2$, we have that $\sum_{a\in\mathcal{A}} T(s,a,s')\gamma_{\tau-1|t}(a)$ is independent of $\gamma_{\sigma|t}$ and, by our induction hypothesis above, $\mathbb{P}((s_{\tau-1|t} = s) \wedge V_{\tau-1|t}^c|\xi_t)$ is affine in $\gamma_{\sigma|t}$. Then, according to the same expression as above, $\mathbb{P}((s_{\tau|t} = s') \wedge V_{\tau-1|t}^c|\xi_t)$ is affine in $\gamma_{\sigma|t}$ also for $\sigma = 0, \ldots, \tau-2$.

For each $\sigma = 0, \ldots, \tau-1$, by our induction hypothesis above, $\mathbb{P}(V_{\tau-1|t}|\xi_t)$ is affine in $\gamma_{\sigma|t}$. Then, as a linear combination of affine functions, $\mathbb{P}(V_{\tau|t}|\xi_t) = \mathbb{P}(V_{\tau-1|t}|\xi_t) + \sum_{s'\in\mathcal{S}_{\text{safe}}^c} \mathbb{P}((s_{\tau|t} = s') \wedge V_{\tau-1|t}^c|\xi_t)$ is affine in $\gamma_{\sigma|t}$.

Finally, for each $s \in \mathcal{S}_{\text{safe}}$, we have that $\mathbb{P}((s_{\tau|t} = s) \wedge V_{\tau|t}^c|\xi_t) = \mathbb{P}((s_{\tau|t} = s) \wedge V_{\tau-1|t}^c|\xi_t)$ is affine in $\gamma_{\sigma|t}$, for $\sigma = 0, \ldots, \tau-1$, as shown above. For each $s \in \mathcal{S}_{\text{safe}}^c$, we have $\mathbb{P}((s_{\tau|t} = s) \wedge V_{\tau|t}^c|\xi_t) = 0$, which is trivially affine in $\gamma_{\sigma|t}$. This completes the induction step and hence proves (i).

Then, (ii) follows directly from the recursive process (2.30) for computing $\mathbb{P}(V_{N|t}|\xi_t)$. ■

Some other properties and the recursive feasibility of the problem (2.20) are discussed in the following subsection.

2.4.2 Theoretical properties and recursive feasibility

To facilitate subsequent exposition, we rewrite the problem (2.20) into the following form,

$$\max \quad f_t(\Gamma_t) \triangleq \mathbb{E}\left\{\sum_{\tau=0}^{N-1} \lambda^\tau R\left(s_{\tau|t}, a_{\tau|t}, s_{\tau+1|t}\right) \middle| b_t\right\}, \tag{2.38a}$$

$$\text{subject to} \quad g_t(\Gamma_t) \triangleq \begin{bmatrix} \mathbb{P}(V_{N|t}|b_t) \\ \text{vec}(\Gamma_t) \\ -\text{vec}(\Gamma_t) \end{bmatrix} \leq \begin{bmatrix} \varepsilon \\ \mathbf{1}_{|\mathcal{A}|N} \\ \mathbf{0}_{|\mathcal{A}|N} \end{bmatrix},$$

$$h_t(\Gamma_t) \triangleq \Gamma_t^\top \mathbf{1}_{|\mathcal{A}|} = \mathbf{1}_N, \tag{2.38b}$$

with respect to $\Gamma_t \triangleq [\gamma_{0|t}, \cdots, \gamma_{N-1|t}] \in \mathbb{R}^{|\mathcal{A}|\times N}$, where $\text{vec}(\cdot)$ designates the "vectorization" operator, $\mathbf{1}_{(\cdot)}$ denotes the $(\cdot)$-dimensional vector of all ones, and $\mathbf{0}_{(\cdot)}$ denotes the $(\cdot)$-dimensional vector of all zeros. Note that in (2.38), the conditional expectation in the objective function and the conditional probability in the safety constraint are no longer expressed as conditioned on the data vector ξ_t as in (2.20), but are now expressed as conditioned on the belief state b_t. Recall that by Propositions 2.1(ii) and 2.2(ii) we have shown these two expressions to be equivalent.

Firstly, we have the following result:

Proposition 2.3: The problem (2.38) is a well-defined nonlinear programming problem with smooth objective and constraint functions.

Here, "well-definedness" means that the problem either has no feasible solution or has an optimal solution.

Proof: We have shown in Proposition 2.2 that $\mathbb{P}(V_{N|t}|b_t)$ is a continuous function of Γ_t. Hence, the preimage of $[0, \varepsilon]$ under $\mathbb{P}(V_{N|t}|b_t)$ is a closed set. Then, it is easy to see that the admissible set defined by the constraints in (2.38b), denoted as Λ_t, is both closed and bounded, i.e., compact. Meanwhile, we have shown in Proposition 2.1 that the objective function f_t in (2.38a) is also a continuous function of Γ_t. In this case, according to the Weierstrass extreme value theorem, f_t attains its supremum on Λ_t as long as $\Lambda_t \neq \emptyset$. This proves the well-definedness of the problem (2.38). The smoothness of its objective and constraint functions follow from Propositions 2.1, 2.2 and the fact that the vectorization operator $\text{vec}(\cdot)$ is a linear transformation. ■

Proposition 2.3 also says that one can use standard nonlinear programming solvers (e.g., MATLAB $fmincon$, IPOPT, etc) to solve the problem (2.38).

We now discuss the relationship between the original discrete optimization problem (2.14) and the continuously relaxed problem (2.38):

Proposition 2.4: (i) The relaxed problem (2.38) has feasible solutions precisely when the original problem (2.14) has feasible solutions. (ii) One can transfer a feasible solution to one of (2.14) and (2.38) to a feasible solution to the other in polynomial time.

Proof: Suppose (2.14) has a feasible solution $\bar{\mathbf{a}}_t = \{\bar{a}_{0|t}, \ldots, \bar{a}_{N-1|t}\}$. Then, let us define $\hat{\Gamma}_t = [\hat{\gamma}_{0|t}, \cdots, \hat{\gamma}_{N-1|t}]$ such that for every $\tau = 0, \ldots, N-1$, $\hat{\gamma}_{\tau|t}$ satisfies $\hat{\gamma}_{\tau|t}(\bar{a}_{\tau|t}) = 1$ and all other entries being 0. It is easy to see that $\hat{\Gamma}_t$ is a feasible solution to (2.38).

Now suppose (2.38) has a feasible solution $\bar{\Gamma}_t = [\bar{\gamma}_{0|t}, \cdots, \bar{\gamma}_{N-1|t}]$. We are going to show that in this case, we can always construct another feasible solution to (2.38), $\hat{\Gamma}_t = [\hat{\gamma}_{0|t}, \cdots, \hat{\gamma}_{N-1|t}]$, based on $\bar{\Gamma}_t$ and a polynomial-time algorithm, such that $\hat{\gamma}_{\tau|t}$ is a zero-one probability vector for every $\tau = 0, \ldots, N-1$. We call such a solution a zero-one solution.

Given $\bar{\Gamma}_t$, we denote its associated $\mathbb{P}(V_{N|t}|b_t)$ value as $\bar{\mathbb{P}}(V_{N|t}|b_t)$ and define the function $\tilde{\mathbb{P}}_\sigma(V_{N|t}|b_t)$ for some $\sigma \in \{0, \ldots, N-1\}$, which is a function of $\gamma_{\sigma|t}$ induced from the constraint function $\mathbb{P}(V_{N|t}|b_t)$ after fixing $\{\gamma_{0|t}, \ldots, \gamma_{\sigma-1|t}, \gamma_{\sigma+1|t}, \ldots, \gamma_{N-1|t}\} = \{\bar{\gamma}_{0|t}, \ldots, \bar{\gamma}_{\sigma-1|t}, \bar{\gamma}_{\sigma+1|t}, \ldots, \bar{\gamma}_{N-1|t}\}$. Since $\bar{\Gamma}_t$ is a feasible solution to (2.38), we must have

$$\varepsilon \geq \bar{\mathbb{P}}\left(V_{N|t}\middle|b_t\right) \geq \min_{\gamma_{\sigma|t} \in \Delta(\mathcal{A})} \tilde{\mathbb{P}}_\sigma\left(V_{N|t}\middle|b_t\right). \tag{2.39}$$

According to Proposition 2.2(i), $\tilde{\mathbb{P}}_\sigma(V_{N|t}|b_t)$ is an affine function of $\gamma_{\sigma|t}$. In this case, the problem

$$\min_{\gamma_{\sigma|t} \in \Delta(\mathcal{A})} \tilde{\mathbb{P}}_\sigma\left(V_{N|t}\middle|b_t\right) \tag{2.40}$$

is a linear programming problem and always has a (global) minimizer, $\hat{\gamma}_{\sigma|t}$, that is located at a vertex of the probability simplex $\Delta(\mathcal{A})$, i.e., $\hat{\gamma}_{\sigma|t}$ is a zero-one probability vector [125]. According to (2.39), $\hat{\Gamma}_{t,\sigma} \triangleq [\bar{\gamma}_{0|t}, \cdots, \bar{\gamma}_{\sigma-1|t}, \hat{\gamma}_{\sigma|t}, \bar{\gamma}_{\sigma+1|t}, \cdots, \bar{\gamma}_{N-1|t}]$ is a feasible solution to (2.38).

Then, we can treat $\hat{\Gamma}_{t,\sigma}$ as the $\bar{\Gamma}_t$ above, select a different $\sigma \in \{0, \ldots, N-1\}$, and repeats the above process. For instance, we can first let $\sigma = 0$ and obtain the feasible solution $\hat{\Gamma}_{t,0} = [\hat{\gamma}_{0|t}, \bar{\gamma}_{1|t}, \cdots, \bar{\gamma}_{N-1|t}]$ with $\hat{\gamma}_{0|t}$ being a zero-one probability vector. We then let $\sigma \leftarrow \sigma + 1$, repeats the above process, and obtain the feasible solution $\hat{\Gamma}_{t,0:1} \triangleq [\hat{\gamma}_{0|t}, \hat{\gamma}_{1|t}, \bar{\gamma}_{2|t}, \cdots, \bar{\gamma}_{N-1|t}]$, where $\hat{\gamma}_{0|t}$ and $\hat{\gamma}_{1|t}$ are both zero-one probability vectors now. We can continue this process sequentially for all $\sigma = 0, \ldots, N-1$, after which we will obtain a zero-one feasible solution $\hat{\Gamma}_t = \hat{\Gamma}_{t,0:N-1} = [\hat{\gamma}_{0|t}, \cdots, \hat{\gamma}_{N-1|t}]$.

This zero-one feasible solution to (2.38) corresponds to an action sequence $\hat{\mathbf{a}}_t = \{\hat{a}_{0|t}, \ldots, \hat{a}_{N-1|t}\}$ with probability 1, and it is easy to see that $\hat{\mathbf{a}}_t$ is a feasible solution to the original problem (2.14). This completes the proof of (i).

Note that linear programming problems can be solved in polynomial time using, e.g.,

Karmarkar's algorithm [126]. Since the procedure described above represents a way to construct a feasible solution $\hat{\mathbf{a}}_t$ to (2.14) from an arbitrary feasible solution $\bar{\Gamma}_t$ to (2.38) through solving the linear programming problem (2.40) at most N times (each time for a different σ), (ii) is proved. ■

The optimization problem (2.38) is repeatedly solved at every time step t to determine the immediate action a_t. Recursive feasibility is a desired property and is discussed in what follows.

To the problem (2.38), feasibility at time step t means that for the current belief state b_t, which acts as an "initial condition," there exists Γ_t such that the constraints in (2.38b) can be satisfied. The major difficulty in establishing recursive feasibility lies in that the "initial condition" at the next time step, b_{t+1}, is stochastic. Recursive feasibility requires that for all "possible" realizations of b_{t+1}, (2.38) is feasible at $t+1$.

Establishing recursive feasibility guarantee in an MPC setting usually involves the introduction of extra constraints to the MPC optimization problem. Motivated by the approaches of [127, 128], we augment the problem (2.38) with the following first-step constraint:

$$\left(b_t, \gamma_{0|t}\right) \in \Lambda_{\text{rec}}(\varepsilon), \tag{2.41}$$

where

$$\Lambda_{\text{rec}}(\varepsilon) = \left\{ \begin{bmatrix} b_0 \\ \gamma_{0|0} \end{bmatrix} \;\middle|\; \begin{array}{l} \exists \{\gamma_{1|0}, \ldots, \gamma_{N-1|0}\} \text{ such that } \mathbb{P}(V_{N|0}|b_0) \leq \varepsilon, \text{ and } \forall (a_0, o_1) \in \mathcal{A} \times \Omega \\ \text{with } \mathbb{P}(a_0, o_1|b_0, \gamma_{0|0}) > 0, \text{ it holds that } b_1(b_0, a_0, o_1) \in \text{proj}_b\left(\Lambda_{\text{rec}}(\varepsilon)\right) \end{array} \right\}. \tag{2.42}$$

In (2.42), $\mathbb{P}(V_{N|0}|b_0)$ is computed according to Theorem 2.1,

$$\begin{aligned} \mathbb{P}(a_0, o_1|b_0, \gamma_{0|0}) &= \mathbb{P}(o_1|a_0, b_0, \gamma_{0|0})\mathbb{P}(a_0|b_0, \gamma_{0|0}) = \sum_{s_0, s_1 \in \mathcal{S}} \mathbb{P}(o_1, s_1, s_0|a_0, b_0, \gamma_{0|0})\mathbb{P}(a_0|\gamma_{0|0}) \\ &= \sum_{s_0, s_1 \in \mathcal{S}} \mathbb{P}(o_1|s_1)\mathbb{P}(s_1|s_0, a_0)\mathbb{P}(s_0|b_0)\mathbb{P}(a_0|\gamma_{0|0}) = \sum_{s_0, s_1 \in \mathcal{S}} O(s_1, o_1)T(s_0, a_0, s_1)b_0(s_0)\gamma_{0|0}(a_0), \end{aligned} \tag{2.43}$$

the map $b_1(b_0, a_0, o_1) : \Delta(\mathcal{S}) \times \mathcal{A} \times \Omega \to \Delta(\mathcal{S})$ is determined by the recursive Bayesian estimation formula (2.16), and $\text{proj}_b(\cdot)$ represents the projection operator onto the space of the first $|\mathcal{S}|$ coordinates.

The following Algorithm 6 can be used to compute an outer approximation of the set $\Lambda_{\text{rec}}(\varepsilon)$. The convergence property of Algorithm 6 and the recursive feasibility property of

(2.38) with the first-step constraint (2.41) are discussed in Theorem 2.2.

Algorithm 6: First-Step Constraint Set Computation

1 Initialize $\Lambda_0(\varepsilon)$ as

$$\Lambda_0(\varepsilon) = \left\{(b_0, \gamma_{0|0}) \,\middle|\, \exists\{\gamma_{1|0}, \ldots, \gamma_{N-1|0}\} \text{ such that } \mathbb{P}(V_{N|0}|b_0) \leq \varepsilon\right\} \tag{2.44}$$

2 **for** $t = 1 : t_{\max}$ **do**

3 Compute

$$\bar{\Lambda}_t(\varepsilon) = \left\{(b_0, \gamma_{0|0}) \,\middle|\, \begin{array}{c}\forall(a_0, o_1) \in \mathcal{A} \times \Omega \text{ with } \mathbb{P}(a_0, o_1|b_0, \gamma_{0|0}) > 0, \\ \text{it holds that } b_1(b_0, a_0, o_1) \in \text{proj}_b\left(\Lambda_{t-1}(\varepsilon)\right)\end{array}\right\} \tag{2.45}$$

4 Let $\Lambda_t(\varepsilon) = \Lambda_0(\varepsilon) \cap \bar{\Lambda}_t(\varepsilon)$

5 **end**

6 Output $\tilde{\Lambda}_{\text{rec}}(\varepsilon) = \Lambda_t(\varepsilon)$.

Theorem 2.2: (i) The sequence of sets, $\Lambda_t(\varepsilon)$, produced by Algorithm 6 converges from above to $\Lambda_{\text{rec}}(\varepsilon)$ as t goes to infinity. (ii) If $b_0 \in \text{proj}_b\left(\Lambda_{\text{rec}}(\varepsilon)\right)$, then (2.38) [augmented with the first-step constraint (2.41)] is recursively feasible for all $t \in \mathbb{N}_0$ almost surely.

Proof: To prove (i), we start from showing that $\Lambda_t(\varepsilon)$ is a nonincreasing sequence of sets, i.e., $\Lambda_{t+1}(\varepsilon) \subseteq \Lambda_t(\varepsilon)$ for all $t \in \mathbb{N}_0$. We show this by induction.

Firstly, according to line 4, it holds that $\Lambda_1(\varepsilon) = \Lambda_0(\varepsilon) \cap \bar{\Lambda}_1(\varepsilon) \subseteq \Lambda_0(\varepsilon)$. This forms the basis of induction. We now assume $\Lambda_t(\varepsilon) \subseteq \Lambda_{t-1}(\varepsilon)$ and are going to show that this hypothesis implies $\Lambda_{t+1}(\varepsilon) \subseteq \Lambda_t(\varepsilon)$.

Since $\Lambda_t(\varepsilon) \subseteq \Lambda_{t-1}(\varepsilon)$, it holds that $\text{proj}_b\left(\Lambda_t(\varepsilon)\right) \subseteq \text{proj}_b\left(\Lambda_{t-1}(\varepsilon)\right)$. Then, according to line 3, we have $\bar{\Lambda}_{t+1}(\varepsilon) \subseteq \bar{\Lambda}_t(\varepsilon)$. Now using line 4 again, we obtain that $\Lambda_{t+1}(\varepsilon) = \Lambda_0(\varepsilon) \cap \bar{\Lambda}_{t+1}(\varepsilon) \subseteq \Lambda_0(\varepsilon) \cap \bar{\Lambda}_t(\varepsilon) = \Lambda_t(\varepsilon)$. This completes the induction step and hence proves that the sequence of sets $\Lambda_t(\varepsilon)$ is nonincreasing, which immediately implies the convergence of $\Lambda_t(\varepsilon)$ as t goes to infinity. We denote the limit as $\Lambda_\infty(\varepsilon)$.

According to line 4, the limit $\Lambda_\infty(\varepsilon)$ must satisfy $\Lambda_\infty(\varepsilon) = \Lambda_0(\varepsilon) \cap \bar{\Lambda}_\infty(\varepsilon)$, where

$$\bar{\Lambda}_\infty(\varepsilon) = \left\{(b_0, \gamma_{0|0}) \,\middle|\, \begin{array}{c}\forall(a_0, o_1) \in \mathcal{A} \times \Omega \text{ with } \mathbb{P}(a_0, o_1|b_0, \gamma_{0|0}) > 0, \\ \text{it holds that } b_1(b_0, a_0, o_1) \in \text{proj}_b\left(\Lambda_\infty(\varepsilon)\right)\end{array}\right\}. \tag{2.46}$$

Then, it is easy to see that $\Lambda_\infty(\varepsilon)$ satisfies the definition of $\Lambda_{\text{rec}}(\varepsilon)$ in (2.42). This means the nonincreasing sequence of sets $\Lambda_t(\varepsilon)$ converges indeed to $\Lambda_{\text{rec}}(\varepsilon)$, i.e., (i) is proved.

For (ii), suppose $b_t \in \text{proj}_b\left(\Lambda_{\text{rec}}(\varepsilon)\right)$. Then, there must exist a sequence of action policies, $\{\gamma_{0|t}, \ldots, \gamma_{N-1|t}\}$, that satisfies the following two properties: 1) $(b_t, \gamma_{0|t}) \in \Lambda_{\text{rec}}(\varepsilon)$, i.e., the augmented constraint (2.41) is satisfied, and 2) the probabilistic safety constraint

$\mathbb{P}(V_{N|t}|b_t) \leq \varepsilon$ is satisfied, which is according to the definition of $\Lambda_{\text{rec}}(\varepsilon)$ in (2.42). This means the problem (2.38) [augmented with (2.41)] is feasible at t. Meanwhile, the augmented constraint $(b_t, \gamma_{0|t}) \in \Lambda_{\text{rec}}(\varepsilon)$ also ensures that for any realization of the pair (a_t, o_{t+1}) with positive probability, the resulting b_{t+1} necessarily satisfies $b_{t+1} \in \text{proj}_b\left(\Lambda_{\text{rec}}(\varepsilon)\right)$, which is also due to the definition of $\Lambda_{\text{rec}}(\varepsilon)$ in (2.42). This means the problem must be again feasible at $t+1$. Then, (ii) can be proved by induction. ■

2.4.3 Adaptation of POMDP-MPC to AV control problem

The reward function R in an autonomous vehicle control setting can often be written as a sum of two parts,

$$R(s_{t+1}, a_t) = R_{\text{s}}(s_{t+1}) + R_{\text{a}}(a_t), \tag{2.47}$$

where the first part R_{s} is purely state-dependent and represents safety and liveness considerations, and the second part R_{a} is purely action-dependent and represents driving effort considerations. In this case, the objective function evaluation procedure (2.25)-(2.26) can be simplified and admits the following vector-matrix expression,

$$\begin{aligned}
&\mathbb{E}\left\{\sum_{\tau=0}^{N-1} \lambda^\tau R\left(s_{\tau+1|t}, a_{\tau|t}\right) \middle| b_t\right\} = \sum_{\tau=0}^{N-1} \lambda^\tau \,\mathbb{E}\left\{R_{\text{s}}(s_{\tau+1|t}) \middle| b_t\right\} + \sum_{\tau=0}^{N-1} \lambda^\tau \,\mathbb{E}\left\{R_{\text{a}}(a_{\tau|t}) \middle| b_t\right\} \\
&= \sum_{\tau=0}^{N-1} \lambda^\tau \left(\mathbf{R}_{\text{s}}^\top b_{\tau+1|t}\right) + \sum_{\tau=0}^{N-1} \lambda^\tau \left(\mathbf{R}_{\text{a}}^\top \gamma_{\tau|t}\right) = \mathbf{R}_{\text{s}}^\top \left(\sum_{\tau=0}^{N-1} \lambda^\tau b_{\tau+1|t}\right) + \mathbf{R}_{\text{a}}^\top \left(\sum_{\tau=0}^{N-1} \lambda^\tau \gamma_{\tau|t}\right),
\end{aligned} \tag{2.48}$$

where $b_{\tau+1|t}$ is the vector representation of predicted state distribution, (2.22), which can be computed using (2.23), $\gamma_{\tau|t}$ is the vector representation of predicted action policy, (2.19), and $\mathbf{R}_{\text{s}}$ and $\mathbf{R}_{\text{a}}$ are vector representations of the functions R_{s} and R_{a},

$$\begin{aligned}
\mathbf{R}_{\text{s}} &= \begin{bmatrix} R_{\text{s}}\left(s^1\right) & \cdots & R_{\text{s}}\left(s^{|\mathcal{S}|}\right) \end{bmatrix}^\top, \\
\mathbf{R}_{\text{a}} &= \begin{bmatrix} R_{\text{a}}\left(a^1\right) & \cdots & R_{\text{a}}\left(a^{|\mathcal{A}|}\right) \end{bmatrix}^\top.
\end{aligned} \tag{2.49}$$

The vectors $\mathbf{R}_{\text{s}}$ and $\mathbf{R}_{\text{a}}$ can be pre-computed offline and stored for online use to reduce online computations.

2.5 Simulation examples of interaction-aware AV control

In this section, we use three examples, representing an intersection crossing scenario, a highway overtaking scenario, and a highway forced merging scenario, respectively, to illustrate our interaction-aware autonomous vehicle control approach. In these examples, the autonomous ego vehicle interacts with one other vehicle. Note that in all earlier parts of this chapter our approach has been presented in a way that in theory can handle an arbitrary number of interacting vehicles. However, the computational complexity increases as the vehicle number increases. In Section 2.6, we will discuss the computational complexity of our approach and potential ways to reduce it in more detail.

2.5.1 Level-k models for the other vehicle

Recall that one important ingredient in our interaction-aware control approach is a set of models of the environment, $\Pi_2 = \{\bar{\pi}_2^1, \ldots, \bar{\pi}_2^{n_\pi}\}$, each of which defines a way the environment reacts to the traffic state $\mathbf{s} \in \mathcal{S}$ according to

$$\mathbb{P}(\bar{a}_2|\mathbf{s}) = \bar{\pi}_2(\mathbf{s}, \bar{a}_2). \tag{2.50}$$

In the case of two-vehicle interactions (autonomous ego vehicle versus one other vehicle), the other vehicle is treated as the environment and a model of the environment is essentially a model of the driver/driving behavior of this other vehicle.

In the examples of this section, we model the driver of the other vehicle as a level-k driver, but with an a priori unknown $k \in \{1, \ldots, k_{\max}\}$. In this case, we can include all of level-1 to level-$k_{\max}$ driver models into our model set Π_2, i.e.,

$$\Pi_2 = \{\pi_2^1, \ldots, \pi_2^{k_{\max}}\}, \tag{2.51}$$

where $\pi_2^{(\cdot)}$ represents a level-$(\cdot)$ model.

Recall that a level-k agent assumes all other agents to be level-$(k-1)$ and makes optimal decision under this assumption. Following this principle, we construct a level-k driver model for the other vehicle, π_2^k, according to

$$\pi_2^k(\mathbf{s}, a_2) = \frac{\exp\left(Q_2^k(\mathbf{s}, a_2)\right)}{\sum_{a_2' \in \mathcal{A}_2} \exp\left(Q_2^k(\mathbf{s}, a_2')\right)}, \tag{2.52}$$

in which the Q-function of state-action pairs, Q_2^k, is defined as

$$Q_2^k(\mathbf{s}, a_2) = \max_{a_{2,t} \in \mathcal{A}_2,\, t=1,\ldots,N-1} \mathbb{E}\left\{ \sum_{t=0}^{N-1} \lambda^t R_2\left(\mathbf{s}_t, a_{1,t}, a_{2,t}, \mathbf{s}_{t+1}\right) \,\middle|\, \mathbf{s}_0 = \mathbf{s}, a_{2,0} = a_2, a_{1,t} \sim \pi_1^{k-1}(\mathbf{s}_t, \cdot) \right\}, \tag{2.53}$$

where R_2 is the other vehicle's reward function, which is assumed by the ego vehicle to model the other vehicle's decision objective(s), and π_1^{k-1} is a level-$(k-1)$ driver model for the ego vehicle, which for $k \geq 2$ is defined similarly as π_2^k, i.e.,

$$\pi_1^{k-1}(\mathbf{s}, a_1) = \frac{\exp\left(Q_1^{k-1}(\mathbf{s}, a_1)\right)}{\sum_{a_1' \in \mathcal{A}_1} \exp\left(Q_1^{k-1}(\mathbf{s}, a_1')\right)}, \tag{2.54}$$

in which

$$Q_1^{k-1}(\mathbf{s}, a_1) = \max_{a_{1,t} \in \mathcal{A}_1,\, t=1,\ldots,N-1} \mathbb{E}\left\{ \sum_{t=0}^{N-1} \lambda^t R_1\left(\mathbf{s}_t, a_{1,t}, a_{2,t}, \mathbf{s}_{t+1}\right) \,\middle|\, \mathbf{s}_0 = \mathbf{s}, a_{1,0} = a_1, a_{2,t} \sim \pi_2^{k-2}(\mathbf{s}_t, \cdot) \right\}, \tag{2.55}$$

where R_1 is the ego vehicle's reward function.

We note that instead of using strict level-k decision rules, which take optimal actions with probability 1, we define our level-k driver models in (2.52) and (2.54) according to the *softmax decision rule* [60]. This is done to capture the suboptimality and variability in human driver behavior [129, 130], which can be caused by human's imperfect perception or limited computing ability. Furthermore, this makes $\pi_2^1, \ldots, \pi_2^{k_{\max}}$ to be stochastic, instead of deterministic, models, which increases robustness of the Bayesian estimation process (2.16) against noise. We also note that when constructing level-k driver models, we choose not to use a probabilistic constraint like (2.3) to represent vehicle safety, but account for safety considerations (such as collision avoidance) in the reward functions R_1 and R_2. This is done to simplify computations. Although a safety property achieved through the design of reward function is typically not as strict as that achieved through constraints, this is sufficient for our modeling purpose – the models $\pi_2^1, \ldots, \pi_2^{k_{\max}}$ are used only for prediction and not directly for control.

The set of equations, (2.52)-(2.55), defines a way to construct level-k driver models for the other vehicle, π_2^k, sequentially for $k = 1, \ldots, k_{\max}$ based on a level-0 model for the ego

vehicle, π_1^0, and a level-0 model for the other vehicle, π_2^0. In the examples of this section, we assume a level-0 driver treats other vehicles as stationary obstacles over her planning horizon. This defines a way to predict other vehicles' actions and the evolution of traffic state $\mathbf{s}$ in response to this driver's own actions, and hence enables this driver to optimize her actions according to an optimization problem similar to (2.53). We note that such a level-0 model represents an extremely aggressive driver, who assumes other vehicles will always yield to her. On the basis of this level-0 model, a level-1 driver defined according to (2.52)-(2.55) will behave conservatively, and in turn, a level-2 driver will behave aggressively. We expect that all level-$(2k+1)$ models, $k = 0, 1, \ldots$, represent conservative driving behaviors and all level-$2k$ models represent aggressive driving behaviors. Considering the behavioral similarity between level-$(2k+1)$ models, we only include the level-1 driver model for the other vehicle, π_2^1, in our model set Π_2. And similarly, considering the behavioral similarity between level-$2k$ models, we only include π_2^2 in Π_2. Therefore, $\Pi_2 = \{\pi_2^1, \pi_2^2\}$. We choose to include level-2 instead of level-0 because the level-2 model defined through (2.52)-(2.55) is more sophisticated and is expected to be able to represent real-world human driver behavior better than the baseline level-0 model.

2.5.2 Simulation examples and results

In all the examples of this section, we use the following discrete-time EOMs to represent the longitudinal kinematics of a vehicle,

$$\begin{bmatrix} \rho_{i,j}(t+1) \\ v_{i,j}(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho_{i,j}(t) \\ v_{i,j}(t) \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a_{i,j}(t), \tag{2.56}$$

where ρ represents the vehicle's position, v represents its velocity, a represents its acceleration, $t \in \mathbb{N}_0$ denotes the discrete time instant, $\Delta t = 1$[s] is the sampling period, the first subscript $i \in \{1, 2\}$ is used to indicate whether a variable is associated with the ego vehicle 1 or the other vehicle 2, and the second subscript $j = x$ if the vehicle is driving along the x-direction and $j = y$ if it is driving along the y-direction. Moreover, we model a lane change as an instantaneous event, i.e., a lane change is completed over one time step. An action is a pair of acceleration level and lane change command, with the former taking values in the finite acceleration set $A = \{-2, 0, 2\}$[m/s^2].

2.5.2.1 Intersection crossing

The first example represents an intersection crossing scenario (shown in Fig. 2.1), where the autonomous ego vehicle (blue car) encounters another vehicle (red car) at an unsignalized

four-way intersection. Their goals are both going straight to cross the intersection. In this case, we consider the following reward functions to represent their goals,

$$\begin{aligned} R_1 &= \rho_{1,x}, \\ R_2 &= \rho_{2,y}. \end{aligned} \tag{2.57}$$

Also, to avoid vehicle collisions, we consider a safe set defined as follows:

$$\mathcal{S}_{\text{safe}} = \left\{ (\rho_1, \rho_2) \,\middle|\, \|\rho_1 - \rho_2\|_2 \geq 1.2\, l_{\text{car}} \right\}, \tag{2.58}$$

where $\rho_i = (\rho_{i,x}, \rho_{i,y})$ represents vehicle i's position in the x-y plane, $\|\cdot\|_2$ denotes the Euclidean norm, and $l_{\text{car}} = 5$[m] is the car length.

We consider a planning horizon of $N = 3$, which corresponds to $3\Delta t = 3$[s] look-ahead time. Over the planning horizon, we enforce the following probabilistic constraint,

$$\mathbb{P}\left\{ (\rho_{1,\tau|t}, \rho_{2,\tau|t}) \in \mathcal{S}_{\text{safe}},\, \forall \tau = 1, \ldots, N \,\middle|\, \xi_t \right\} \geq 0.99. \tag{2.59}$$

Note that we impose (2.59) as a constraint only in the autonomous vehicle control problem (2.3). When we construct our level-1 and level-2 driver models through (2.52)-(2.55), we do not impose this constraint, but add a penalty term to the reward functions (2.57) that penalizes the event $(\rho_{1,t}, \rho_{2,t}) \notin \mathcal{S}_{\text{safe}}$, to simplify computations. At the beginning of a simulation, we initialize our beliefs in the level-1 and level-2 models both as 0.5, which represents no prior knowledge of the other vehicle.

Figs. 2.1(a-1) and (a-2) show two steps of a simulation where the autonomous ego vehicle encounters a level-1 vehicle. Recall that our level-1 model represents a conservative driver. In this case, the autonomous ego vehicle passes through the intersection first. Figs. 2.1(b-1) and (b-2) show two steps of a simulation of the autonomous ego vehicle versus a level-2 vehicle. Our level-2 model represents an aggressive driver. In this case, the autonomous ego vehicle yields to this level-2 vehicle and passes through the intersection after it. In both cases, the autonomous ego vehicle successfully resolves its conflict with the other vehicle and safely passes through the intersection. This demonstrates the effective combination in our approach of Bayesian-based identification of the other vehicle's actual model and real-time adaptation of the ego vehicle's plan to the model identification result.

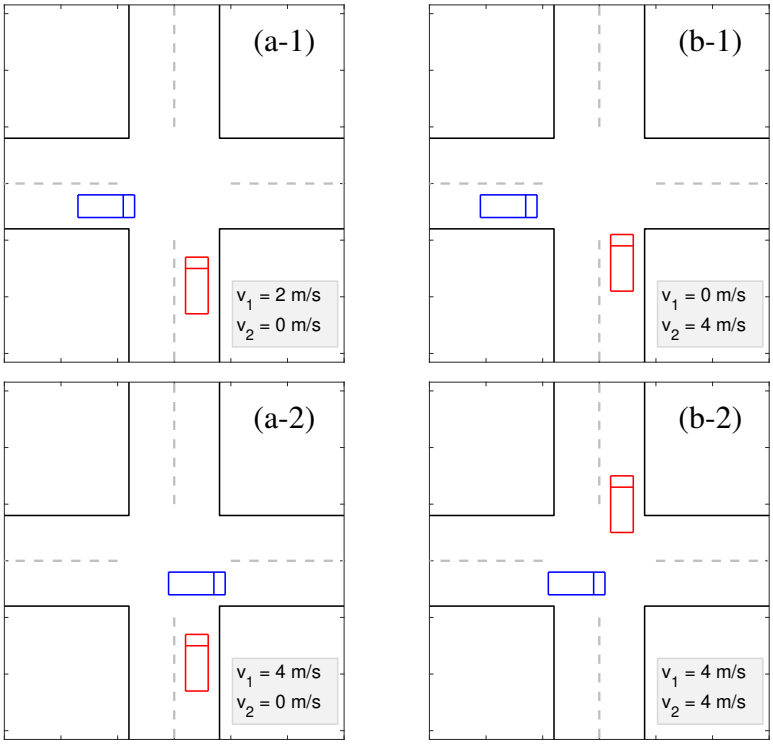


Figure 2.1: Simulation results of an intersection crossing scenario. (a-1) and (a-2) show two steps of a simulation where the autonomous ego vehicle (blue) interacts with a level-1 vehicle (red). (b-1) and (b-2) show two steps of a simulation where the autonomous ego vehicle (blue) interacts with a level-2 vehicle (red).

2.5.2.2 Highway overtaking

The second example represents a highway scenario (shown in Fig. 2.2), where the goal of the autonomous ego vehicle (blue car) is to overtake a slower vehicle in its front (red car). To represent this goal, we consider the following reward function for the ego vehicle,

$$R_1 = 8\rho_{1,x} - \rho_{1,y}, \tag{2.60}$$

where the first term is used to encourage the ego vehicle to overtake the slower vehicle in its front, and the second term is used to penalize driving in the left passing lane so that the ego vehicle is encouraged to come back to the right traveling lane once it has passed the slower vehicle.

We assume the other vehicle (red car) does not change lanes and its goal is to drive at its maximum speed (which is assumed to be smaller than the maximum speed of the ego vehicle). Because a higher speed will lead to a larger longitudinal position, we use the

following reward function to represent its goal,

$$R_2 = \rho_{2,x}. \tag{2.61}$$

As before, we consider a planning horizon of $N = 3$ and enforce the probabilistic constraint (2.59) over the planning horizon, where the safe set $\mathcal{S}_{\text{safe}}$ is now defined as follows:

$$\mathcal{S}_{\text{safe}} = \left\{(\rho_1, \rho_2) \,\middle|\, \left(|\rho_{1,x} - \rho_{2,x}| \geq 1.6\, l_{\text{car}}\right) \vee \left(|\rho_{1,y} - \rho_{2,y}| \geq w_{\text{lane}}\right)\right\}, \tag{2.62}$$

in which $w_{\text{lane}} = 3.6$[m] is the lane width. This safe set definition represents the requirement that one vehicle can pass the other only when they are in different lanes and when they are in the same lane, they must keep a minimum longitudinal distance of $1.6\, l_{\text{car}}$. When constructing level-k driver models, we do not impose constraints but penalize any violation of the set safe $\mathcal{S}_{\text{safe}}$ through an extra penalty term added to the reward functions (2.60) and (2.61).

Figs. 2.2(a-1)-(a-4) show four steps of a simulation where the autonomous ego vehicle overtakes a level-1 (conservative) vehicle, and Figs. 2.2(b-1)-(b-4) show those where the autonomous ego vehicle overtakes a level-2 (aggressive) vehicle. When the front vehicle is level-1, the ego vehicle manages to overtake it quickly, because, as can be seen in Fig. 2.2(a-2), the level-1 vehicle maintains a low speed, which facilitates the ego vehicle's cut-in. When the front vehicle is level-2, the ego vehicle needs to drive in the left passing lane for a longer period of time and accelerate to a higher speed to pass the level-2 vehicle before it can come back to the right traveling lane.

2.5.2.3 Forced merging

The last example represents a forced merging scenario (shown in Fig. 2.3), where the autonomous ego vehicle (blue car) must merge into the left lane before the end of the right lane. We assume there is a vehicle driving in the left lane that is in an almost side-by-side position with the ego vehicle (red car). In this case, the ego vehicle must decide whether to merge in front of it or to merge behind it. We consider the following reward function to represent the goal of the ego vehicle,

$$R_1 = \rho_{1,x} + 10\rho_{1,y}, \tag{2.63}$$

where the first term is used to encourage the autonomous ego vehicle to maintain a reasonable speed (i.e., discouraging it from slowing down too much and stopping), and the second

term is used to encourage it to merge into the left lane. We consider the following safe set,

$$\mathcal{S}_{\text{safe}} = \Bigg\{ (\rho_1, \rho_2) \;\Bigg|\; \Big[\big(|\rho_{1,x} - \rho_{2,x}| \geq 1.6\, l_{\text{car}} \big) \vee \big(|\rho_{1,y} - \rho_{2,y}| \geq w_{\text{lane}} \big) \Big] \bigwedge$$
$$\Big[\big((\rho_{1,x} \leq 20) \wedge (\rho_{1,y} = \frac{w_{\text{lane}}}{2}) \big) \vee \big(20 < \rho_{1,x} \leq 100 \big) \vee \big((\rho_{1,x} > 100) \wedge (\rho_{1,y} = \frac{3w_{\text{lane}}}{2}) \big) \Big] \Bigg\} \tag{2.64}$$

where the conditions in the first line represents vehicle-to-vehicle collision avoidance requirement, and the conditions in the second line represents the requirement that the ego vehicle must complete the merge within the road section marked by the grey-dashed lane marking in Fig. 2.3. The other parameters are the same as in the previous example.

Figs. 2.3(a-1)-(a-4) correspond to a simulation where the vehicle driving in the left lane (red car) is a level-1 (conservative) vehicle. In this simulation, the autonomous ego vehicle accelerates and merges in front of this level-1 vehicle. Figs. 2.3(b-1)-(b-4) correspond to a simulation where the vehicle driving in the left lane is a level-2 (aggressive) vehicle. In this case, the ego vehicle slows down and merges behind this level-2 vehicle, because it predicts that this level-2 vehicle will not yield to let it cut in. Note that in both simulations the ego vehicle has no prior knowledge of the other vehicle but manages to develop a correct belief of the other vehicle's driver model sufficiently quickly based on its observed trajectory over the first few steps.

2.6 Summary and discussion

In this chapter, we introduced a novel interaction-aware control approach for AVs operating on shared roads with other users (such as human-driven vehicles). This approach handles interaction uncertainties due to varied driving styles and intents of other road users and AV safety under such uncertainties through a C-POMDP formulation. A computational algorithm based on MPC using online optimization, referred to as POMDP-MPC, has been developed and used to solve the formulated C-POMDP problem, and leads to AV control solutions with an explicit probabilistic safety guarantee.

Table 2.1 shows the sizes of the formulated C-POMDP problems of the three examples in Section 2.5.2 and the computation times corresponding to using different nonlinear programming solvers. The computation time numbers in Table 2.1 indicate the average computation times (in [s] of real time) for the ego vehicle to solve for an action decision at a time step. The computations are performed on a desktop with an Intel Xeon E3-1246 v3 processor and 16.0 GB RAM.

Table 2.1: Computation times of POMDP-MPC.

Example	State space size	$fmincon$	IPOPT	NLopt(SLSQP)
Intersection	7938	1.0352[s]	0.1849[s]	0.0631[s]
Overtaking	2448	1.6945[s]	0.7015[s]	0.2866[s] (8/230 failed)
Merging	15876	1.8729[s]	0.3439[s]	0.7501[s] (23/250 failed)

First note that these problems involving a few thousands of states are considered as medium to large-sized problems in the POMDP literature, and it can take several minutes to hours to solve such problems using conventional POMDP solvers based on point-based dynamic programming [117, 118]. In contrast, our POMDP-MPC algorithm reduces the computation times to a level that is comparable to the time budget for real-time computation (the decision frequency is $\sim$ 1 Hz). We note that further reduction in these computation times is possible by, for instance, exploiting proper warm-starting approaches, inexact and real-time iteration solution strategies, and symbolic computation techniques [131, 132, 133], the investigation of which is left as a topic for future research.

However, our POMDP-MPC algorithm still suffers from the "curse of dimensionality" [134]. For instance, we have observed an average computation time of ~ 163[s] for an example with 26325 states to be solved with MATLAB $fmincon$ in [119]. This poses a computational challenge for our approach to treat traffic scenarios involving interactions between the ego vehicle and multiple other vehicles. A possible solution is to decouple the interaction between the ego vehicle and the environment (which represents the unity of all other vehicles) into pairwise interactions between the ego vehicle and each of the other vehicles, to prevent the state space size from increasing exponentially as the number of interacting vehicles increases. Another strategy is to use our approach to treat even higher-level planning problems where vehicle physical states are encoded into a small number of Markov states and actions represent higher-level driving behaviors (such as following the front vehicle or overtaking the front vehicle, etc). Moreover, one can choose to solve the formulated C-POMDP problem for a set of state values offline, store the state-action pairs into a dataset, and exploit machine learning techniques to extract an explicit policy from this dataset (e.g., represented by a neural network) used for online control, as has been proposed in [67, 73, 135]. All of these strategies can reduce the online computational burden. Extension of the approach from finite-space setting to continuous-space setting, thus avoiding the computational difficulty associated with discreteness/discretization, is also a topic of ongoing research.

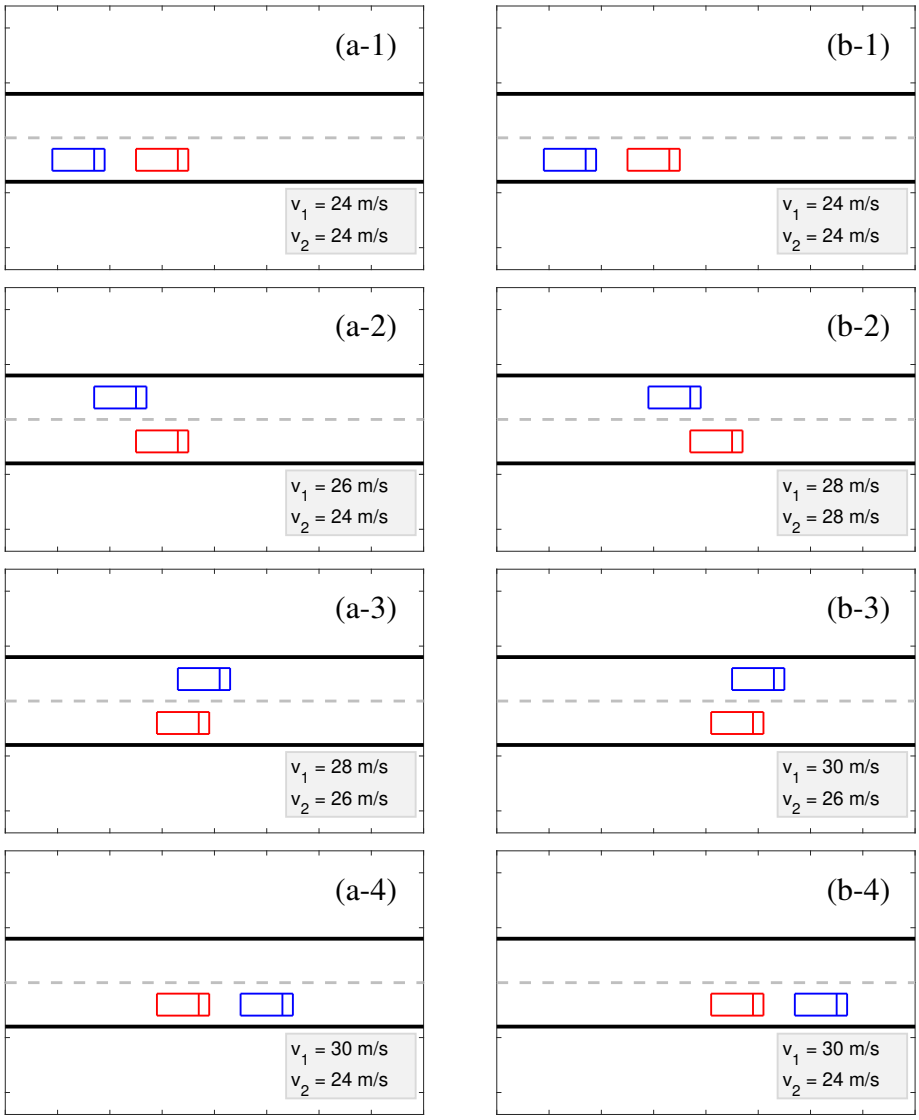


Figure 2.2: Simulation results of a highway overtaking scenario. (a-1) to (a-4) show four steps of a simulation where the autonomous ego vehicle (blue) overtakes a level-1 vehicle (red). (b-1) to (b-4) show four steps of a simulation where the autonomous ego vehicle (blue) overtakes a level-2 vehicle (red).

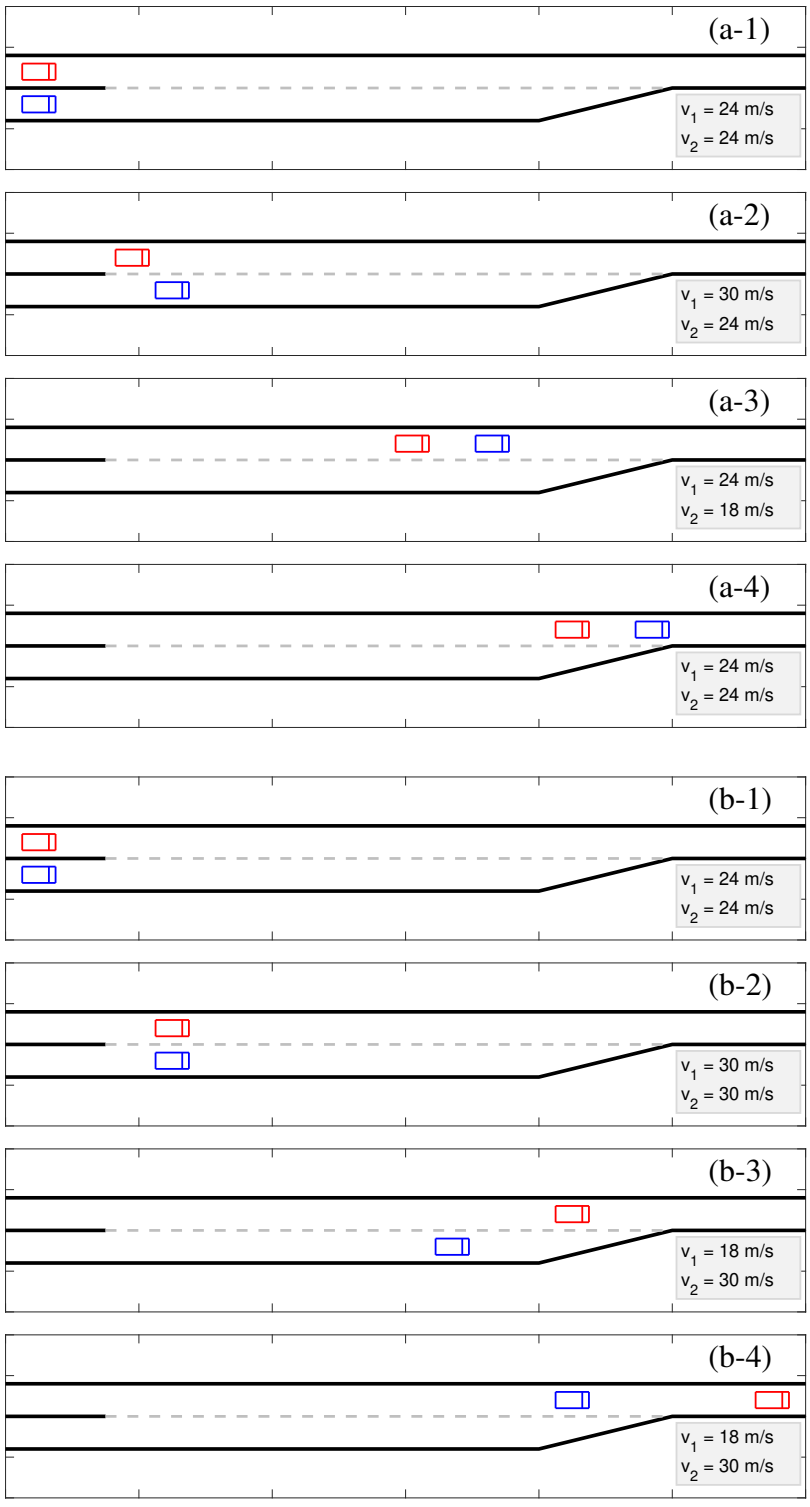


Figure 2.3: Simulation results of a forced merging scenario. (a-1) to (a-4) show four steps of a simulation where the autonomous ego vehicle (blue) merges in front of a level-1 vehicle (red). (b-1) to (b-4) show four steps of a simulation where the autonomous ego vehicle (blue) merges behind a level-2 vehicle (red).

CHAPTER 3

Enhancing Autonomous Vehicle Safety via the Action Governor

3.1 Background and introduction

Many hard specifications for controlled dynamic systems can be imposed in the form of pointwise-in-time state and control constraints. For instance, such constraints can represent variable bounds to ensure safe and efficient system operation, actuator limits, as well as collision/obstacle avoidance requirements. To address such constraints, one route is to take them into account when (re-)designing the controller, e.g., via correct-by-construction synthesis using controlled invariant sets [136] or control Lyapunov/barrier functions [137, 138], or via the model predictive control framework [139, 140]. Another route is to augment a nominal controller with constraint-handling capability via add-on, supervisory schemes [141, 142, 143]. This second route may be preferable in many practical circumstances. For instance, a well-designed, legacy controller may already exist while new specifications are imposed to the system. In this case, the second route can preserve many performance characteristics of the legacy controller, such as stability, frequency-domain responses, etc., and thus significantly reduce the tuning complexity compared to re-designing the controller as in the first route [141].

The Reference Governor (RG) has been shown to be an effective scheme, along the second route, to manage pointwise-in-time state and control constraints [141]. The RG monitors and modifies the reference signal, which is typically an input to the nominal controller and defines the control objective, to enforce these constraints. Alternatively, supervision and modification may also be done to the output of the nominal controller, such as in the approaches of [142, 143].

In this chapter, we propose a novel add-on, supervisory scheme, referred to as *Action Governor* (AG), for discrete-time linear systems to enforce pointwise-in-time constraints.

Following the ideas of [142, 143], the AG enforces constraints by monitoring, and minimally modifying when necessary, the nominal control signal to a constraint-admissible one.

In particular, we focus on exclusion-zone avoidance requirements, where the exclusion zone is assumed to be expressed as a finite union of convex, polytopic sets in the state space. Note that in this case the feasible region is in general non-convex. This differentiates our problem setting from the ones treated in [142, 143]. The particular consideration for such exclusion zones is motivated by those obstacle avoidance scenarios frequently encountered in mobile robot path planning problems [144, 145, 146] as well as vehicle and pedestrian avoidance scenarios in autonomous vehicle control problems [102]. Note, however, that such exclusion zones can indeed represent many state constraint types, such as box constraints (see Fig. 3.1 for an example).

The AG operation is based on set-theoretic techniques and online optimization. Although there has been a rich literature on set-theoretic methods in control covering theoretical properties, computational aspects, and application scenarios, most previous work treated the case where the feasible region is assumed to be compact and convex [147, 148, 149, 150]. In contrast, the feasible region in our problem setting is in general neither bounded nor convex.

Although the RG and our proposed AG are both prediction-based supervisory schemes and non-convex constraints have also been considered within the RG framework [151, 152, 153], our AG scheme has the following distinguishing features: 1) The AG modifies the output of the nominal controller, whereas the RG modifies the reference input to the controller. 2) Unlike the RG, the AG does not restrict the signal modification over the prediction horizon to a constant. A direct consequence is that the AG yields a larger feasible set, leaving greater flexibility to control, and thereby, can potentially achieve better control performance (as illustrated by an example in Section 3.4). 3) The RG typically assumes the closed-loop system (plant + controller) to be linear + time-invariant (LTI), whereas the AG assumes only the plant to be LTI, i.e., permitting the controller to be nonlinear and evolving with time. This allows controller variability, as well as online learning of the control policy, without needing to redesign/retune the AG.

On the other hand, graph search-based, sampling-based, and potential field-based approaches have been extensively used in path planning for mobile robots with obstacle avoidance requirements [144, 154]. The first two typically use simplified kinematic models for motion prediction and leave the control of system dynamics to a lower level. Although potential field-based approaches can deal with dynamic models, they do not easily lend themselves to theoretical guarantees. In contrast, our AG approach handles dynamic mod-

els in the form of discrete-time linear systems and provides theoretical safety guarantees under suitable assumptions.

In summary, the contributions of this chapter include: 1) establishing the theoretical foundation of the AG scheme, 2) discussing its computational realization, including its offline computational tasks and two online optimization algorithms of different complexity, and 3) illustrating its operation and effectiveness using an automotive and a mobile robot related examples.

The developments of this chapter have been published in the journal article [155].

3.2 System model and control objective

We consider systems that can be represented as the following discrete-time linear model:

$$x(k+1) = f\left(x(k), u(k)\right) = Ax(k) + Bu(k), \tag{3.1}$$

where $x(k) \in \mathbb{R}^n$ represents the system state at the discrete time instant $k \in \mathbb{N}_0$, and $u(k) \in \mathbb{R}^m$ represents the control input. We assume that a nominal control policy ϕ has been defined for the system (3.1),

$$u_\phi(k) = \phi\left(x(k), r(k), k\right), \tag{3.2}$$

where $r(k) \in \mathbb{R}^p$ represents a reference signal determining the control objective. The control policy ϕ may be nonlinear and time-varying, which may be due to specific control objectives such as state/control constraints, finite-time convergence requirements, etc., or due to online evolution/learning of the control policy [156, 157, 158].

Furthermore, we assume that the system is subject to an *exclusion-zone avoidance* requirement of the form

$$x(k) \notin X_0, \quad \forall k \in \mathbb{N}_0. \tag{3.3}$$

In particular, we assume X_0 can be written as a finite union of convex, open, and polytopic sets, i.e.,

$$X_0 = \bigcup_{i=1}^{N} \left\{x \in \mathbb{R}^n : G_i x < g_i\right\}, \tag{3.4}$$

where $G_i \in \mathbb{R}^{q_i \times n}$ and $g_i \in \mathbb{R}^{q_i}$. Note that such an X_0 can indeed represent many exclusion zone types (see Fig. 3.1 for an example).

The exclusion-zone avoidance requirement (3.3) may not have been incorporated when defining the nominal control policy (3.2). Our objective is to develop a control algorithm

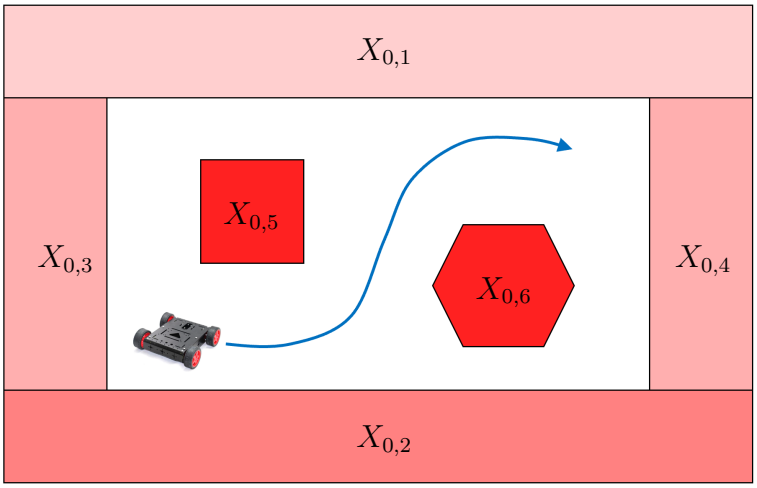


Figure 3.1: An exclusion zone example, where the union of the red polytopic regions is the exclusion zone X_0.

to enforce (3.3).

3.3 Action Governor

The solution that we propose is a supervisory scheme, referred to as *Action Governor* (AG), which monitors and minimally modifies, if necessary, the nominal control signal $u_\phi(k)$ to enforce the exclusion-zone avoidance requirement (3.3).

In particular, the AG operates based on the following constrained optimization problem:

$$u(k) = \underset{u \in U}{\arg\min} \, \|u - u_\phi(k)\|_S^2, \tag{3.5a}$$

$$\text{subject to } Ax(k) + Bu \in X_{\text{safe}}, \tag{3.5b}$$

where the set $U \subset \mathbb{R}^m$ represents the range of control authority, the function $\|\cdot\|_S = \sqrt{(\cdot)^\top S(\cdot)}$ with $S \in \mathbb{R}^{m \times m}$ being positive-definite penalizes the difference between the modified control signal $u(k)$ and the nominal control signal $u_\phi(k)$, and $X_{\text{safe}} \subset \mathbb{R}^n$ is a *safe set* which will be introduced later. In particular, we assume U to be a convex, closed, and polytopic set as follows, which is a common assumption in the control literature [139, 140, 141],

$$U = \{u \in \mathbb{R}^m : Hu \leq h\}, \tag{3.6}$$

where $H \in \mathbb{R}^{s\times m}$ and $h \in \mathbb{R}^s$.

3.3.1 Safe set and unrecoverable sets

To enforce both present and future safety, the safe set X_{safe} is characterized by the following requirements: For any $x(k) \in X_{\text{safe}}$, there exists $u(k) \in U$ such that

1. The next state satisfies the exclusion-zone avoidance requirement (3.3), i.e., $x(k+1) = Ax(k) + Bu(k) \notin X_0$.

2. Future exclusion-zone avoidance is possible, i.e., given $x(k+1) = Ax(k) + Bu(k)$, there exists a control sequence $\{u(k+1), u(k+2), \dots\} \subset U$ such that $\{x(k+2), x(k+3), \dots\} \cap X_0 = \emptyset$.

The explicit determination of X_{safe} relies on the following sequence of sets, referred to as *unrecoverable sets*, defined sequentially as

$$\begin{aligned} X_k &= X_0 \cup \{x \in \mathbb{R}^n : \text{ for each } u \in U, Ax + Bu \in X_j \text{ for some } j = 0, \dots, k-1\} \\ &= X_0 \cup \left\{ x \in \mathbb{R}^n : Ax + Bu \in \bigcup_{j=0}^{k-1} X_j, \, \forall u \in U \right\} \\ &= X_0 \cup \left\{ x \in \mathbb{R}^n : Ax \in \left(\bigcup_{j=0}^{k-1} X_j \right) \sim BU \right\}, \quad k = 1, 2, \dots \end{aligned} \tag{3.7}$$

where $\sim$ denotes the P(ontryagin)-difference operation between sets [147].

The sets X_k satisfy the following properties, which also explain why they are called "unrecoverable sets":

Proposition 3.1: If $x(0) \in X_k$, then for any sequence $\{u(0), \dots, u(k-1)\} \in U \times \cdots \times U$ there exists $0 \le j \le k$ such that $x(j) \in X_0$.

Proof: The proof is by induction. For $k = 1$, $x(0) \in X_1$ implies either $x(0) \in X_0$ or $x(1) = Ax(0) + Bu(0) \in X_0$ for any $u(0) \in U$. Suppose the statement has been proven for X_j, $1 \le j \le k$. For $k+1$, if $x(0) \in X_{k+1}$, then either $x(0) \in X_0$ or for each $u(0) \in U$, $x(1) = Ax(0) + Bu(0) \in X_j$ for some $0 \le j \le k$, which is by the definition of X_{k+1}. For the latter, since $x(1) \in X_j$, by our induction hypothesis, for any $\{u(1), \dots, u(j)\} \in U \times \cdots \times U$, there exists $0 \le i \le j$ such that $x(i+1) \in X_0$. Thus, we have shown that for any $\{u(0), u(1), \dots, u(k)\} \in U \times U \times \cdots \times U$, there exists $0 \le j' = i + 1 \le j + 1 \le k + 1$ such that $x(j') \in X_0$. This proves the statement for $k+1$. ■

Proposition 3.2: Let $x(0)$ be given. If for any sequence $\{u(0), \ldots, u(k-1)\} \in U \times \cdots \times U$ there exists $0 \leq j \leq k$ such that $x(j) \in X_0$, then $x(0) \in X_k$.

Proof: The proof is by induction. For $k = 1$, $x(0) \in X_0$ or $x(1) = Ax(0)+Bu(0) \in X_0$ for any $u(0) \in U$ implies $x(0) \in X_1$, which is by the definition of X_1. Suppose the statement has been proven for k. For $k+1$, if $x(0) \in X_0$, then $x(0) \in X_{k+1}$, which follows from the fact that $X_0 \subset X_{k+1}$. Otherwise, fix an arbitrary $u(0) \in U$ and let $x(1) = Ax(0)+Bu(0)$. The proposition statement assumes that for any $\{u(1), \ldots, u(k)\} \in U \times \cdots \times U$, there exists $0 \leq j \leq k$ such that $x(j+1) \in X_0$, which implies $x(1) \in X_k$ by our induction hypothesis. Thus, we have shown that for each $u(0) \in U$, $Ax(0) + Bu(0) \in X_k$, which implies $x(0) \in X_{k+1}$ by the definition of X_{k+1}. This proves the statement for $k+1$. ■

The combination of Propositions 3.1 and 3.2 says that there exists no solution for the control sequence $\{u(0), \ldots, u(k-1)\} \in U \times \cdots \times U$ to avoid the state trajectory entering the exclusion zone X_0 over the steps $0, \ldots, k$ if and only if $x(0) \in X_k$. Or equivalently, there exists a control sequence $\{u(0), \ldots, u(k-1)\} \in U \times \cdots \times U$ to avoid X_0 over the steps $0, \ldots, k$ if and only if $x(0) \in \mathbb{R}^n \setminus X_k$. Furthermore, the following convergence property of X_k holds:

Proposition 3.3: For each $k = 0, 1, \ldots$, $X_k \subset X_{k+1}$, i.e., X_k is an increasing sequence of sets. In turn, $X_\infty = \lim_{k\to\infty} X_k$ exists (in the set-theoretic sense [159]) and satisfies $X_k \subset X_\infty$ for all k.

Proof: The proof follows from

$$\begin{aligned} X_k &= X_0 \cup \left\{ x \in \mathbb{R}^n : Ax \in \left(\bigcup_{j=0}^{k-1} X_j \right) \sim BU \right\} \\ &\subset X_0 \cup \left\{ x \in \mathbb{R}^n : Ax \in \left(\bigcup_{j=0}^{k} X_j \right) \sim BU \right\} = X_{k+1}. \quad \blacksquare \end{aligned} \tag{3.8}$$

Using the unrecoverable sets X_k, we define the safe set as $X_{\text{safe}} = \mathbb{R}^n \setminus X_\infty = \lim_{k\to\infty}(\mathbb{R}^n \setminus X_k)$. On the basis of Propositions 3.1 and 3.2, X_{safe} has the following properties:

Proposition 3.4: For any $x \in X_{\text{safe}}$, it holds that (i) $x \notin X_0$, and (ii) there exists $u \in U$ such that $Ax + Bu \in X_{\text{safe}}$.

Proof: The former $x \in X_{\text{safe}} \implies x \notin X_0$ follows from

$$X_{\text{safe}} = \mathbb{R}^n \setminus X_\infty = \mathbb{R}^n \setminus \left(\bigcup_{k=0}^{\infty} X_k \right) \subset \mathbb{R}^n \setminus X_0. \tag{3.9}$$

For the latter, let x be given and assume that for any $u \in U$, $Ax + Bu \in \mathbb{R}^n \setminus X_{\text{safe}} = \bigcup_{k=0}^{\infty} X_k$, i.e., $Ax \in \left(\bigcup_{k=0}^{\infty} X_k\right) \sim BU = \bigcup_{k=0}^{\infty} \left(X_k \sim BU\right)$, where the last equality holds according to Lemma 3.1 in Appendix E. This means there must exist k' such that $Ax \in X_{k'} \sim BU = \left(\bigcup_{k=0}^{k'} X_k\right) \sim BU$. Then, according to (3.7), $x \in X_{k'+1} \subset \bigcup_{k=0}^{\infty} X_k = \mathbb{R}^n \setminus X_{\text{safe}}$. Thus, if $x \in X_{\text{safe}}$, then there must exist $u \in U$ such that $Ax + Bu \notin \mathbb{R}^n \setminus X_{\text{safe}}$. ■

Proposition 3.4 ensures that if the AG operates based on (3.5) to modify the control input $u(k)$, then 1) a feasible solution exists to (3.5) for all k, and 2) the exclusion-zone avoidance requirement (3.3) is satisfied for all k.

Note that the exact determination of X_{safe} relies on the set X_k iteratively computed according to (3.7) with $k \to \infty$. In practice, X_{safe} can be approximated by $\tilde{X}_{\text{safe},k'} = \mathbb{R}^n \setminus X_{k'}$ with k' being sufficiently large. Moreover, the following result says that, under a few additional assumptions, such a finitely determinable approximation of X_{safe} suffices for implementation.

Proposition 3.5: Assume $0 \in U$ and define $R = \bigoplus_{k=0}^{\infty} A^k BU$, where $\oplus$ denotes the Minkowski sum operation of sets [147]. Suppose 1) there exists k' such that

$$R \cap (X_\infty \setminus X_{k'}) = \emptyset, \tag{3.10}$$

2) $x(0) \in R \cap \tilde{X}_{\text{safe},k'}$, and 3) the AG operates based on

$$u(k) = \arg\min_{u \in U} \|u - u_\phi(k)\|_S^2, \tag{3.11a}$$

$$\text{subject to } Ax(k) + Bu \in \tilde{X}_{\text{safe},k'}. \tag{3.11b}$$

Then, (i) $x(k) \notin X_0$, and (ii) there exists $u \in U$ such that $Ax(k) + Bu \in \tilde{X}_{\text{safe},k'}$, for all $k = 0, 1, \ldots$

Proof: Firstly, (i) follows from $x(k) \in \mathbb{R}^n \setminus X_{k'} \subset \mathbb{R}^n \setminus X_0$. Now assume $x(k-1) \in R \cap \tilde{X}_{\text{safe},k'} = R \cap (\mathbb{R}^n \setminus X_{k'})$. Note that (3.10) implies

$$\begin{aligned} & R \cap (X_\infty \setminus X_{k'}) = R \cap (\mathbb{R}^n \setminus X_{k'}) \cap X_\infty = \emptyset \\ \implies & R \cap (\mathbb{R}^n \setminus X_{k'}) \subset \mathbb{R}^n \setminus X_\infty = X_{\text{safe}}. \end{aligned} \tag{3.12}$$

Thus, we have $x(k-1) \in X_{\text{safe}}$, which by Proposition 3.4 ensures the existence of $u \in U$ such that $Ax(k-1) + Bu \in X_{\text{safe}} = \mathbb{R}^n \setminus X_\infty \subset \mathbb{R}^n \setminus X_{k'} = \tilde{X}_{\text{safe},k'}$. This proves (ii) for $k-1$. Also, by the invariance of R [147], $x(k-1) \in R$ implies $Ax(k-1) + Bu \in R$ for any $u \in U$. Therefore, if the AG operates based on (3.11) at $k-1$, we must have

$x(k) = Ax(k-1) + Bu(k-1) \in R \cap \tilde{X}_{\text{safe},k'}$. Then, the proof of (ii) for all $k = 0, 1, \ldots$ can be completed by induction. ■

We remark that our definition of the safe set X_{safe} is similar to the *viability kernel* considered in [145]. In [145], the computation and utilization of the viability kernel are based on finite state and control spaces (or based on finite discretization of the original spaces). In contrast, we deal with continuous state and control spaces, and the computational algorithms introduced in what follows do not rely on discretization of the spaces. Furthermore, our theoretical results, Propositions 3.1-3.6, are not in [145].

3.3.2 Offline and online computations

Given X_0 as a convex, polytopic set (3.4), the sets X_k for $k = 1, 2, \ldots$ are iteratively computed offline based on the following proposition:

Proposition 3.6: Assume A is invertible (see Remark 3.1) and U is a polytopic set. Then, for each $k = 1, 2, \ldots$, we have (i) X_k can be represented as the union of a finite number of polytopic sets, i.e., $X_k = \bigcup_{j=1}^{r_k} X_{k,j}$ where $X_{k,j}$ is a polytopic set for each $j = 1, \ldots, r_k$; and (ii) X_k can be numerically computed using Algorithm 7.

Proof: Assume that X_{k-1} can be represented as the union of a finite number of polytopic sets. Using the fact that $X_j \subset X_{j+1}$ for all $j = 0, 1, \ldots, k-2$, we can rewrite (3.7) as

$$X_k = X_0 \cup \{x \in \mathbb{R}^n : Ax \in X_{k-1} \sim BU\}. \tag{3.13}$$

Note that BU, as the image of a polytopic set U under the linear transformation B, is also a polytopic set. Then, $X_{k-1} \sim BU$ is the P-difference between a finite union of polytopic sets and a polytopic set. Thus, according to Theorem 4.4 of [140], the lines 1-5 of Algorithm 7 compute $\mathcal{G} = X_{k-1} \sim BU$, which is also a finite union of polytopic sets. Then, since A is invertible, the preimage of $\mathcal{G}$ under A, $A^{-1}\mathcal{G}$, is still a finite union of polytopic sets. Furthermore, we have

$$A^{-1}\mathcal{G} = \{x \in \mathbb{R}^n : Ax \in X_{k-1} \sim BU\}. \tag{3.14}$$

Finally, since X_0 is a finite union of polytopic sets, we obtain that $X_k = X_0 \cup A^{-1}\mathcal{G}$ is a finite union of polytopic sets. Then, (i) and (ii) are simultaneously proved for k. The proof can be extended to all $k = 1, 2, \ldots$ by induction. ■

Remark 3.1: We remark that the assumption of A being invertible typically holds true for a practical system, e.g., when the model (3.1) is discretized from continuous-time dynamics. We also remark that the set operations involved in Algorithm 7, including P-

difference between polytopic sets in line 2, and convex hull, set difference, Minkowski sum and union of finite unions of polytopic sets in lines 1, 3-6 can be efficiently computed using corresponding functions of the Multi-Parametric Toolbox 3 (MPT3) [160].

Algorithm 7: Offline Computation for X_k

Input : A, B, X_0, X_{k-1}, U
Output: X_k
1 $\mathcal{H} \leftarrow \text{convhull}(X_{k-1})$
2 $\mathcal{D} \leftarrow \mathcal{H} \sim (BU)$
3 $\mathcal{E} \leftarrow \mathcal{H} \setminus X_{k-1}$
4 $\mathcal{F} \leftarrow \mathcal{E} \oplus (-BU)$
5 $\mathcal{G} \leftarrow \mathcal{D} \setminus \mathcal{F}$
6 $X_k \leftarrow X_0 \cup A^{-1}\mathcal{G}$

From now on, we assume the AG operates based on $\tilde{X}_{\text{safe},k'} = \mathbb{R}^n \setminus X_{k'}$ for some k'. According to Proposition 3.6, $X_{k'}$ is a finite union of polytopic sets, i.e., can be written as

$$X_{k'} = \bigcup_{j=1}^{r_{k'}} \bigcap_{i=1}^{s_j} \{x \in \mathbb{R}^n : G_{i,j}x < g_{i,j}\}, \tag{3.15}$$

where $G_{i,j} \in \mathbb{R}^{1\times n}$, $g_{i,j} \in \mathbb{R}$, and in turn,

$$\tilde{X}_{\text{safe},k'} = \mathbb{R}^n \setminus X_{k'} = \bigcap_{j=1}^{r_{k'}} \bigcup_{i=1}^{s_j} \{x \in \mathbb{R}^n : G_{i,j}x \geq g_{i,j}\}. \tag{3.16}$$

Then, the constraint $Ax(k) + Bu \in \tilde{X}_{\text{safe},k'}$ is equivalent to the following set of constraints:

$$G_{i,j}\left(Ax(k) + Bu\right) \geq g_{i,j} - M(1 - \delta_{i,j}), \tag{3.17a}$$

$$\delta_{i,j} \in \{0, 1\}, \quad \forall i = 1, \ldots, s_j, \, \forall j = 1, \ldots, r_{k'}, \tag{3.17b}$$

$$\sum_{i=1}^{s_j} \delta_{i,j} = 1, \quad \forall j = 1, \ldots, r_{k'}, \tag{3.17c}$$

where $M > 0$ is a sufficiently large positive number.

Thus, the AG online problem (3.11) can be solved as a Mixed-Integer Quadratic Programming (MIQP) problem with $(u, \delta_{i,j})$ as the decision variables.

Furthermore, under the following practical assumption, a computationally lighter approach, presented as Algorithm 8, can be used to approximately solve (3.11).

Assumption 3.1: A safe-mode control policy ψ has been defined for the system (3.1),

$$u_\psi(k) = \psi\left(x(k), k\right), \tag{3.18}$$

possibly being conservative, such that for any $x(k) \in \tilde{X}_{\text{safe},k'}$ we have

$$Ax(k) + Bu_\psi(k) \in \tilde{X}_{\text{safe},k'}. \tag{3.19}$$

Assumption 3.1 is reasonable for many practical systems. For instance, for an adaptive cruise control (ACC) system where $x(k) \in X_0$ represents a rear-end collision to the preceding vehicle, the safe-mode policy ψ may correspond to an automatic emergency braking (AEB) system.

Algorithm 8 aims to find a feasible point $u(k)$ along the line segment connecting $u_\phi(k)$ and $u_\psi(k)$ that is as close to $u_\phi(k)$ as possible through a bisection method. It is similar to the Algorithm 1 in [161]. Two important properties of Algorithm 8 are: 1) algorithm convergence is guaranteed, i.e., $\underline{\lambda}$ and $\overline{\lambda}$ are converging to the same value $\in [0, 1]$; and 2) algorithm output $u(k)$ is always feasible, in terms of satisfying that $Ax(k) + Bu(k) \in \tilde{X}_{\text{safe},k'}$. For more details regarding these two properties, the reader is referred to [161, 162]. We also remark that the set containment condition in line 4 can be efficiently checked using the *PolyUnion.contains()* function of MPT3 [160].

Algorithm 8: Online Computation for $u(k)$

Input : $A, B, \tilde{X}_{\text{safe},k'}, x(k), u_\phi(k), u_\psi(k)$
Output: $u(k)$

1 $\underline{\lambda} \leftarrow 0$, $\overline{\lambda} \leftarrow 1$, $\lambda \leftarrow 1$
2 **while** $\overline{\lambda} - \underline{\lambda} > \delta$ **do**
3 $u \leftarrow \lambda u_\phi(k) + (1-\lambda)u_\psi(k)$
4 **if** $Ax(k) + Bu \in \tilde{X}_{safe,k'}$ **then**
5 $\underline{\lambda} \leftarrow \lambda$
6 **else**
7 $\overline{\lambda} \leftarrow \lambda$
8 **end**
9 $\lambda \leftarrow (\underline{\lambda} + \overline{\lambda})/2$
10 **end**
11 $u(k) \leftarrow \underline{\lambda} u_\phi(k) + (1-\underline{\lambda})u_\psi(k)$

3.4 Simulation examples

3.4.1 Adaptive cruise control

The first example we consider represents an ACC system for automated highway driving. The relative motion between the leading vehicle and the following ego vehicle is written in discrete-time as

$$\begin{bmatrix} \Delta s(k+1) \\ \Delta v(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta s(k) \\ \Delta v(k) \end{bmatrix} - \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} u(k), \tag{3.20}$$

where $\Delta t = 0.25$[s] is the sampling period, Δs and Δv denote, respectively, the longitudinal distance and relative speed between the leading and the ego vehicles, and u is the control input representing the ego vehicle's acceleration. The following feedback policy is defined as the nominal control,

$$u_\phi(k) = K \begin{bmatrix} \Delta s(k) - \Delta s_r(k) \\ \Delta v(k) \end{bmatrix}, \tag{3.21}$$

where Δs_r is the reference signal and represents the desired car-following distance, and the feedback gain K is computed as the linear quadratic regulator (LQR) gain with $Q = \text{diag}(10, 1)$ and $R = 20$.

To avoid rear-end collision and promote passenger comfort, the following constraints are imposed,

$$\Delta s(k) \geq 2\text{[m]}, \quad -2\text{[m/s}^2\text{]} \leq u(k) \leq 2\text{[m/s}^2\text{]}, \quad \forall k. \tag{3.22}$$

Clearly, the nominal policy (3.21) does not account for the constraints (3.22). We now consider both the applications of AG and RG to enforcing (3.22) and compare them.

On the one hand, the constraints (3.22) can be handled by AG with considering the following definition for the exclusion set X_0,

$$X_0 = \left\{ \begin{bmatrix} \Delta s \\ \Delta v \end{bmatrix} \in \mathbb{R}^2 : \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta v \end{bmatrix} < \begin{bmatrix} 2 \\ M \\ M \\ M \end{bmatrix} \right\}, \tag{3.23}$$

where $M > 0$ is a sufficiently large positive number, and the control authority set $U = [-2, 2]$. The reason for including the 2nd-4th inequalities as "virtual constraints" is to make X_0 a polytopic set so that enable the numerical computations of Algorithm 7. In

this example, the online determination of the control input u is through solving the MIQP (3.11) and (3.17).

On the other hand, the RG considers the closed-loop system after the nominal control (3.21) is applied, i.e.,

$$\begin{bmatrix}\Delta s(k+1)\\ \Delta v(k+1)\end{bmatrix} = \left(\begin{bmatrix}1 & \Delta t\\ 0 & 1\end{bmatrix} - \begin{bmatrix}\frac{1}{2}\Delta t^2\\ \Delta t\end{bmatrix} K\right)\begin{bmatrix}\Delta s(k)\\ \Delta v(k)\end{bmatrix} + \begin{bmatrix}\frac{1}{2}\Delta t^2\\ \Delta t\end{bmatrix} K(1)\,\Delta s_r(k), \quad (3.24)$$

where $K(1)$ denotes the first entry of K. The RG monitors and modifies, if necessary, the reference signal $\Delta s_r(k)$ to a constraint-admissible one $\Delta s_v(k)$ to enforce constraints. For this, it utilizes a set typically called O_∞ [141]. In particular, to handle the constraints (3.22) and also enable the numerical computation of O_∞, we consider the following constraints on the state-reference pair,

$$\begin{bmatrix}-1 & 0\\ K(1) & K(2)\\ -K(1) & -K(2)\\ 1 & 0\\ 0 & 1\\ 0 & -1\end{bmatrix}\begin{bmatrix}\Delta s(k)\\ \Delta v(k)\end{bmatrix} + \begin{bmatrix}0\\ -K(1)\\ K(1)\\ 0\\ 0\\ 0\end{bmatrix}\Delta s_v(k) \le \begin{bmatrix}-2\\ 2\\ 2\\ M\\ M\\ M\end{bmatrix}. \quad (3.25)$$

We consider the following initial condition and constant reference signal,

$$(\Delta s(0), \Delta v(0)) = (18[\text{m}], -4[\text{m/s}]), \quad \Delta s_r(k) \equiv 2.5[\text{m}]. \quad (3.26)$$

The simulation results are presented in Fig. 3.2. The state trajectory using AG (green dash-dotted) versus that using RG (blue solid) is shown in Fig. 3.2(a). It can be seen that the green curve converges to the desired steady state $(2.5, 0)$ while being kept outside the set X_∞ (red shaded) for all time. In particular, it sometimes rides on the boundary of X_∞ but never enters into X_∞. This guarantees the satisfaction of the state constraint $\Delta s(k) \ge 2$, whose boundary is marked by the black dashed vertical line. The blue curve also satisfies the constraint $\Delta s(k) \ge 2$ by being kept inside the set O_∞ (magenta shaded) for all time and converges to the desired steady state. However, it is clear that the region where the state is allowed to reach using AG ($\mathbb{R}^2 \setminus X_\infty$) is strictly larger than that using RG (O_∞), and this results in faster response and convergence with AG than RG, which can be seen from the control input trajectories plotted in Fig. 3.2(b). Also, both the control input trajectory using AG and that using RG satisfy the control constraints $-2 \le u(k) \le 2$. To achieve this, as well as to enforce the state constraint, AG sometimes modifies the nominal control u_ϕ, the

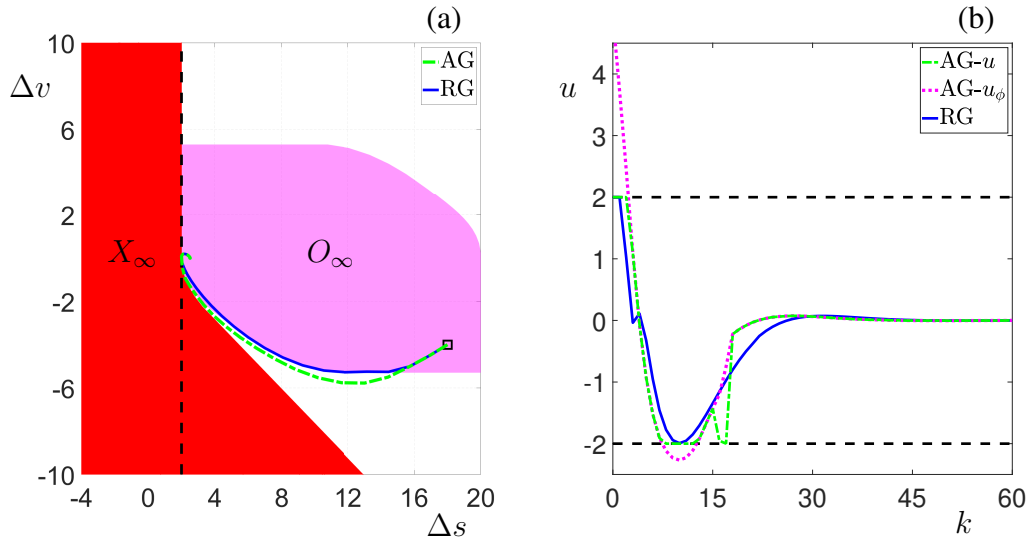


Figure 3.2: Adaptive cruise control.

profile of which is shown by the magenta dotted curve.

3.4.2 Omni-directional robot obstacle avoidance

The dynamics of an omni-directional robot are modeled in continuous-time as

$$\ddot{s}_1 = u_1, \tag{3.27a}$$

$$\ddot{s}_2 = u_2, \tag{3.27b}$$

where (s_1, s_2) represent the global positions of the robot in the x- and y-directions, and (u_1, u_2) represent the accelerations and are the control inputs.

After being first written in first-order differential equations and then discretized with a sampling period $\Delta t = 1$ and assuming zero-order hold on the inputs, (3.27) is converted to a discrete-time model in the form of (3.1) with $(s_1, s_2, \dot{s}_1, \dot{s}_2)$ as the vector state and (u_1, u_2) as the vector control input. To track a desired position $(s_{1,r}, s_{2,r})$, a nominal control policy is defined as: for $i = 1, 2$,

$$u_{i,\phi}(k) = \text{sat}_{R_i(k)} \left(K(i,:) \begin{bmatrix} s_1(k) - s_{1,r}(k) \\ s_2(k) - s_{2,r}(k) \\ \dot{s}_1(k) \\ \dot{s}_2(k) \end{bmatrix} \right), \tag{3.28}$$

where K is the LQR gain with $Q = \text{diag}(1, 1, 1, 1)$ and $R = \text{diag}(1, 1)$, $K(i,:)$ denotes its

ith row, and $\text{sat}_{R_i(k)}(\cdot)$ is the saturation function to the range

$$R_i(k) = \left[\max\big(-2, -\frac{1}{\Delta t}(4 + \dot{s}_i(k))\big), \min\big(2, \frac{1}{\Delta t}(4 - \dot{s}_i(k))\right]. \tag{3.29}$$

We remark that the control policy defined by (3.28) and (3.29) is a modified LQR control law equipped with the capability of enforcing the velocity and acceleration constraints

$$-4 \leq \dot{s}_i(k) \leq 4, \quad -2 \leq u_i(k) \leq 2, \quad \forall k. \tag{3.30}$$

We consider a scenario similar to the one studied in [146]. It is assumed that a diamond-shaped obstacle blocks the straight-line path from the robot's initial position $\big(s_1(0), s_2(0)\big) = (-10, 0)$ to the target position $(s_{1,r}, s_{2,r}) = (10, 0)$, as shown in Fig. 3.3(a). To avoid collision with such an obstacle, we use an AG to supervise the control signal. In particular, the polytopic exclusion set X_0 is determined by the diamond-shaped obstacle and the velocity ranges $\dot{s}_i \in [-4, 4]$, and the box-shaped control authority set U is determined by the acceleration ranges $u_i \in [-2, 2]$.

In this example, the online determination of the control inputs (u_1, u_2) is through Algorithm 2, which relies on a safe-mode control policy ψ. We construct ψ based on a simple repulsive-force field surrounding the obstacle [144], which is illustrated by the black arrows in Fig. 3.3(a). Specifically, the control $(u_{1,\psi}, u_{2,\psi})$ is determined first as a vector along the direction of the arrow at the robot's current position with magnitude proportional to the arrow length (which is constant everywhere in the considered repulsive-force field), then saturated to the ranges $u_i \in [-2, 2]$. We remark that one important difference of our approach from other obstacle avoidance approaches based on repulsive-force/potential fields is that the trade-off between position tracking and collision avoidance is optimized online through Algorithm 8 in our approach rather than pre-designed offline, which is typical in those approaches [144].

The simulation result is shown in Fig. 3.3. It can be seen from Fig. 3.3(a) that the robot safely travels from the start position (marked by the red square) to the target position (marked by the green triangle) without colliding with the obstacle. Figs. 3.3(b) and (c) show the control input histories. In particular, the green dotted curves correspond to the nominal policy ϕ, the red dashed curves to the safe-mode policy ψ, and the blue solid curves are the optimized convex combinations of ϕ and ψ obtained by Algorithm 8.

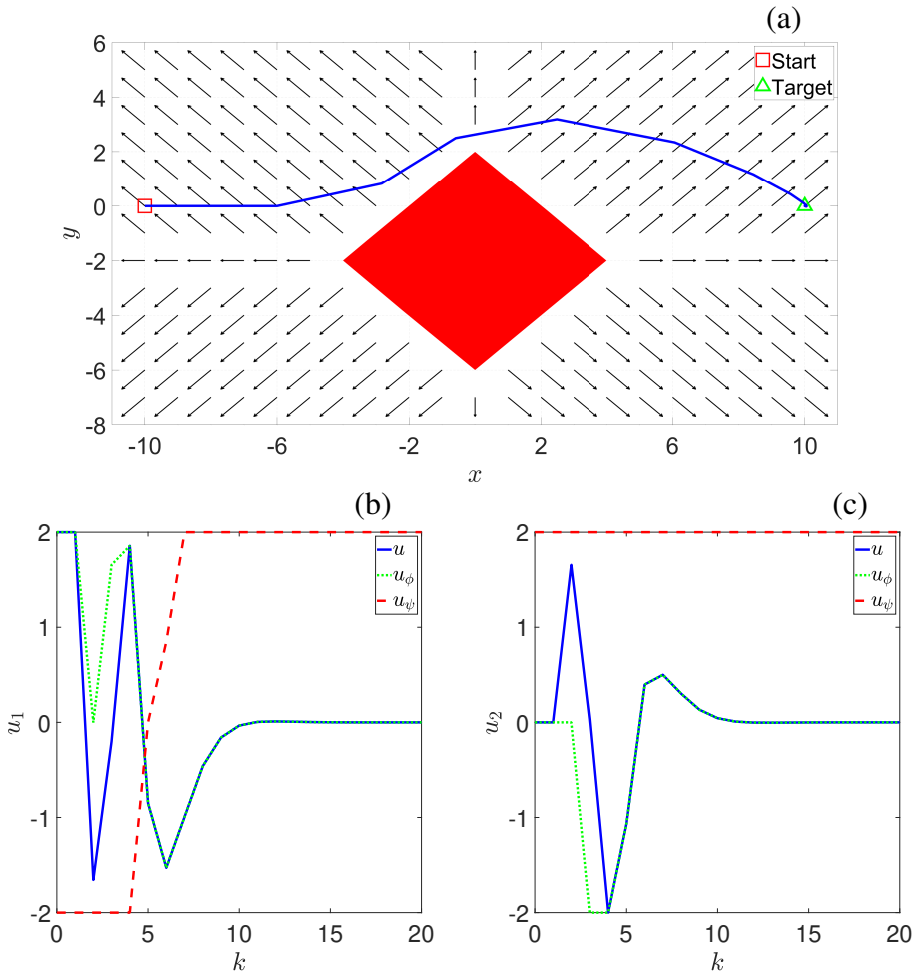


Figure 3.3: Omni-directional robot obstacle avoidance.

3.5 Summary and discussion

In this chapter, we introduced the Action Governor (AG), which is a novel add-on scheme to nominal control loops that monitors and minimally modifies (when necessary) the nominal control signal to enforce pointwise-in-time state and control constraints. We established the theoretical foundation and computational approach of the AG for discrete-time linear systems subject to non-convex exclusion-zone avoidance constraints, and used automotive and mobile robot related examples to illustrate its operation and effectiveness.

Ongoing research focuses on extending the AG scheme to linear systems with additive disturbances and/or parametric uncertainties, and to nonlinear systems. Also, using the AG to achieve safe reinforcement learning is of interest and is being investigated [163]. The goal is to develop the AG into a general safety supervision framework for enhancing autonomous vehicle safety.

CHAPTER 4

Conclusions and future work

4.1 Conclusions

In order to ensure safe and effective operation on shared roads, autonomous vehicles need to be equipped with highly reliable decision and control systems that can appropriately account for/respond to road user interactions. This dissertation addresses urgent challenges in the development and high-confidence validation of such systems from the following three angles:

Firstly, we developed new approaches based on game theory for modeling the interactions among drivers/vehicles in traffic. The models produced by these approaches have been used to build up highway and intersection traffic simulators with improved representation of driver/vehicle interactions, which can be used for fast and economical simulation-based testing and validation of autonomous vehicle control systems.

Secondly, we proposed a new autonomous vehicle (high-level) control approach that enables an autonomous ego vehicle to appropriately respond to its interaction with the traffic environment. The major advantages of this approach include its abilities to handle interaction uncertainties and to provide an explicit probabilistic safety guarantee under such uncertainties.

Thirdly, we introduced the Action Governor (AG) as an add-on scheme to nominal control loops for enforcing pointwise-in-time state and control constraints. In this dissertation, we established the theoretical foundation and computational approach of the AG for discrete-time linear systems with non-convex exclusion-zone avoidance constraints. The goal of ongoing research is to extend the AG into a general safety supervision framework for enhancing autonomous vehicle safety.

4.2 Future work

Virtual testing using a simulator can effectively discover "faults" of an autonomous driving algorithm[1] through fast generation and simulation of a rich set of randomized scenarios. One of the next steps is the development of methodology and toolchain that can quantitatively and/or formally certify certain levels of safety and robustness of an autonomous driving algorithm based on measures of coverage and performance of its virtual testing results and provide guideline and/or data for automatic test-case generation for future on-road testing. This will be a major step towards an effective integration of virtual testing and on-road testing into a unified verification and validation (V&V) framework for autonomous vehicles that can provide highly accelerated and reliable V&V procedures and results.

To overcome the scalability issue of the proposed interaction-aware autonomous vehicle control approach is one of the focuses of ongoing research. On the one hand, proper abstraction and decoupling of vehicle interactive behaviors can lead to decision problems of smaller sizes and hence enable handling of more interacting vehicles. On the other hand, developing more efficient C-POMDP solvers based on extension and integration of recent advances in real-time optimization, parallel and distributed computation, and/or machine learning techniques, and developing the continuous-space counterpart of the approach are promising directions to achieve improved scalability and are of general interest.

Extensions of the Action Governor (AG) to handle set-bounded/stochastic disturbances and parametric uncertainties and to handle a broader class of systems, such as piecewise-affine nonlinear systems, are being pursued. Also, using the AG to achieve safe reinforcement learning is an interesting and relevant topic. For instance, it will enable autonomous vehicles to safely evolve their control policies online to account for effects of vehicle component aging as well as passengers' individual preferences. This will be more thoroughly investigated in future research.

[1] For instance, conditions under which the algorithm can lead to unsafe driving behaviors

BIBLIOGRAPHY

[1] Anderson, J. M., Nidhi, K., Stanley, K. D., Sorensen, P., Samaras, C., and Oluwatola, O. A., *Autonomous vehicle technology: A guide for policymakers*, Rand Corporation, 2014.

[2] Van Brummelen, J., O'Brien, M., Gruyer, D., and Najjaran, H., "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, Vol. 89, 2018, pp. 384–406.

[3] Schwarting, W., Alonso-Mora, J., and Rus, D., "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 1, 2018, pp. 187–210.

[4] Guanetti, J., Kim, Y., and Borrelli, F., "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, Vol. 45, 2018, pp. 18–40.

[5] Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., and Shi, W., "Computing systems for autonomous driving: State-of-the-art and challenges," *IEEE Internet of Things Journal*, 2020, pp. 1–1.

[6] Buehler, M., Iagnemma, K., and Singh, S., *The 2005 DARPA grand challenge: The great robot race*, Vol. 36, Springer, 2007.

[7] Buehler, M., Iagnemma, K., and Singh, S., *The DARPA urban challenge: Autonomous vehicles in city traffic*, Vol. 56, Springer, 2009.

[8] Koopman, P. and Wagner, M., "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, Vol. 9, No. 1, 2017, pp. 90–96.

[9] Hussain, R. and Zeadally, S., "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 2, 2018, pp. 1275–1313.

[10] Parkinson, S., Ward, P., Wilson, K., and Miller, J., "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, No. 11, 2017, pp. 2898–2915.

[11] Ersal, T., Kolmanovsky, I., Masoud, N., Ozay, N., Scruggs, J., Vasudevan, R., and Orosz, G., "Connected and automated road vehicles: State of the art and future challenges," *Vehicle System Dynamics*, Vol. 58, No. 5, 2020, pp. 672–704.

[12] SAE International, *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*, June 2018, https://doi.org/10.4271/J3016_201806.

[13] Litman, T., "Autonomous vehicle implementation predictions: Implications for transport planning," Tech. rep., 2020.

[14] Plumer, B., *5 big challenges that self-driving cars still have to overcome*, Vox Media, April 2016, https://www.vox.com/2016/4/21/11447838/self-driving-cars-challenges-obstacles.

[15] Sovani, S., *Top 3 challenges to produce level 5 autonomous vehicles*, Ansys Blog, December 2018, https://www.ansys.com/blog/challenges-level-5-autonomous-vehicles.

[16] Hao, K., *The three challenges keeping cars from being fully autonomous*, MIT Technology Review, April 2019, https://www.technologyreview.com/2019/04/23/103181/the-three-challenges-keeping-cars-from-being-fully-autonomous/.

[17] Li, N. I., Kolmanovsky, I., and Girard, A., "A reference governor for nonlinear systems based on quadratic programming," *Dynamic Systems and Control Conference*, Vol. 50695, American Society of Mechanical Engineers, 2016, p. V001T02A005.

[18] Li, N., Kolmanovsky, I. V., and Girard, A., "A reference governor for nonlinear systems with disturbance inputs based on logarithmic norms and quadratic programming," *IEEE Transactions on Automatic Control*, Vol. 65, No. 7, 2019, pp. 3207–3214.

[19] Kalabić, U. V., Li, N. I., Vermillion, C., and Kolmanovsky, I. V., "Reference governors for chance-constrained systems," *Automatica*, Vol. 109, 2019, pp. 108500.

[20] Liu, K., Li, N., Rizzo, D., Garone, E., Kolmanovsky, I., and Girard, A., "Model-free learning to avoid constraint violations: An explicit reference governor approach," *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 934–940.

[21] Liu, K., Li, N., Kolmanovsky, I., Rizzo, D., and Girard, A., "Model-free learning for safety-critical control systems: A reference governor approach," *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 943–949.

[22] Li, N., Girard, A., and Kolmanovsky, I., "Chance-constrained controller state and reference governor," *arXiv preprint arXiv:2010.01710*, 2020.

[23] Li, N., Kolmanovsky, I., and Girard, A., "Detection-averse optimal and receding-horizon control for Markov decision processes," *Automatica*, Vol. 122, 2020, pp. 109278.

[24] Li, N., Kolmanovsky, I., Girard, A., and Filev, D., "Fuzzy encoded Markov chains: Overview, observer theory, and applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[25] Li, N., Li, Z., Dong, L., Girard, A., Kolmanovsky, I., and Lu, R., "Triggered measurements in Markov processes for entropy-constrained state estimation with application to precision agriculture," *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 3611–3616.

[26] Li, N., Kolmanovsky, I. V., and Girard, A., "An analytical safe approximation to joint chance-constrained programming with additive Gaussian noises," *IEEE Transactions on Automatic Control*, 2021.

[27] Li, X., Li, N., Kolmanovsky, I., and Epureanu, B. I., "Stochastic model predictive control for remanufacturing system management," *Journal of Manufacturing Systems*, Vol. 59, 2021, pp. 355–366.

[28] Hudson, J., Gupta, R., Li, N., and Kolmanovsky, I., "Iterative model and trajectory refinement for orbital trajectory optimization," *Optimal Control Applications and Methods*, Vol. 38, No. 6, 2017, pp. 1132–1147.

[29] Li, N., Kolmanovsky, I., and Girard, A., "LQ control of unknown discrete-time linear systems – A novel approach and a comparison study," *Optimal Control Applications and Methods*, Vol. 40, No. 2, 2019, pp. 265–291.

[30] Berning, A. W., Li, N. I., Girard, A., Leve, F. A., Petersen, C. D., and Kolmanovsky, I., "Spacecraft relative motion planning using chained chance-constrained admissible sets," *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 4938–4944.

[31] Li, N., Kolmanovsky, I., and Girard, A., "Model-free optimal control based automotive control system falsification," *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 636–641.

[32] Li, N., Girard, A., and Kolmanovsky, I., "Optimal control based falsification of unknown systems with time delays: A gasoline engine A/F ratio control case study," *IFAC-PapersOnLine*, Vol. 51, No. 31, 2018, pp. 252–257.

[33] Maldonado, B. P., Li, N., Kolmanovsky, I., and Stefanopoulou, A. G., "Learning reference governor for cycle-to-cycle combustion control with misfire avoidance in spark-ignition engines at high exhaust gas recirculation-diluted conditions," *International Journal of Engine Research*, Vol. 21, No. 10, 2020, pp. 1819–1834.

[34] Tian, R., Li, N., Girard, A., and Kolmanovsky, I., "Controller mode and reference governor for constraint and failure management in vehicle platoon systems," *2020*

IEEE Conference on Control Technology and Applications (CCTA), IEEE, 2020, pp. 660–665.

[35] Tang, S., Li, N., Kolmanovsky, I., and Girard, A., "Trajectory optimization for falsification: A case study of vehicle rollover test generation based on black-box models," *21st IFAC World Congress*, IFAC, 2020.

[36] Liu, K., Li, N., Kolmanovsky, I., Rizzo, D., and Girard, A., "Tanker truck rollover avoidance using learning reference governor," Tech. rep., SAE Technical Paper, 2021.

[37] Liu, K., Li, N., Kolmanovsky, I., Rizzo, D., and Girard, A., "Safe learning reference governor for constrained systems with application to fuel truck rollover avoidance," *arXiv preprint arXiv:2101.09298*, 2021.

[38] Deo, N. and Trivedi, M. M., "Convolutional social pooling for vehicle trajectory prediction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[39] Sun, L., Zhan, W., and Tomizuka, M., "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 2111–2117.

[40] Li, J., Ma, H., Zhan, W., and Tomizuka, M., "Generic probabilistic interactive situation recognition and prediction: From virtual to real," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3218–3224.

[41] Hu, Y., Zhan, W., Sun, L., and Tomizuka, M., "Multi-modal probabilistic prediction of interactive behavior via an interpretable model," *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 557–563.

[42] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A., "Social GAN: Socially acceptable trajectories with generative adversarial networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[43] Dai, S., Li, L., and Li, Z., "Modeling vehicle interactions via modified LSTM models for trajectory prediction," *IEEE Access*, Vol. 7, 2019, pp. 38287–38296.

[44] Wang, W., Liu, C., and Zhao, D., "How much data are enough? A statistical approach with case study on longitudinal driving behavior," *IEEE Transactions on Intelligent Vehicles*, Vol. 2, No. 2, 2017, pp. 85–98.

[45] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q., "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, Vol. 109, No. 1, 2020, pp. 43–76.

[46] Myerson, R. B., *Game theory*, Harvard university press, 2013.

[47] Mandiau, R., Champion, A., Auberlet, J.-M., Espié, S., and Kolski, C., "Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation," *Applied Intelligence*, Vol. 28, No. 2, 2008, pp. 121–138.

[48] Yoo, J. H. and Langari, R., "Stackelberg game based model of highway driving," *Dynamic Systems and Control Conference*, Vol. 45295, American Society of Mechanical Engineers, 2012, pp. 499–508.

[49] Yoo, J. H. and Langari, R., "A Stackelberg game theoretic driver model for merging," *Dynamic Systems and Control Conference*, Vol. 56130, American Society of Mechanical Engineers, 2013, p. V002T30A003.

[50] Talebpour, A., Mahmassani, H. S., and Hamdar, S. H., "Modeling lane-changing behavior in a connected environment: A game theory approach," *Transportation Research Procedia*, Vol. 7, 2015, pp. 420–440.

[51] Bahram, M., Lawitzky, A., Friedrichs, J., Aeberhard, M., and Wollherr, D., "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Transactions on Vehicular Technology*, Vol. 65, No. 6, 2015, pp. 3981–3992.

[52] Yu, H., Tseng, H. E., and Langari, R., "A human-like game theory-based controller for automatic lane changing," *Transportation Research Part C: Emerging Technologies*, Vol. 88, 2018, pp. 140–158.

[53] Sadigh, D., Sastry, S., Seshia, S. A., and Dragan, A. D., "Planning for autonomous cars that leverage effects on human actions," *Robotics: Science and Systems*, Vol. 2, 2016.

[54] Fisac, J. F., Bronstein, E., Stefansson, E., Sadigh, D., Sastry, S. S., and Dragan, A. D., "Hierarchical game-theoretic planning for autonomous vehicles," *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9590–9596.

[55] Dreves, A. and Gerdts, M., "A generalized Nash equilibrium approach for optimal control problems of autonomous cars," *Optimal Control Applications and Methods*, Vol. 39, No. 1, 2018, pp. 326–342.

[56] Başar, T. and Zaccour, G., *Handbook of dynamic game theory*, Springer, 2018.

[57] Stahl II, D. O. and Wilson, P. W., "Experimental evidence on players' models of other players," *Journal of Economic Behavior & Organization*, Vol. 25, No. 3, 1994, pp. 309–327.

[58] Stahl, D. O. and Wilson, P. W., "On players' models of other players: Theory and experimental evidence," *Games and Economic Behavior*, Vol. 10, No. 1, 1995, pp. 218–254.

[59] Nagel, R., "Unraveling in guessing games: An experimental study," *The American Economic Review*, Vol. 85, No. 5, 1995, pp. 1313–1326.

[60] Sutton, R. S. and Barto, A. G., *Reinforcement learning: An introduction*, MIT press, 2018.

[61] Von Stackelberg, H., *Market structure and equilibrium*, Springer Science & Business Media, 2010.

[62] Basar, T. and Olsder, G. J., *Dynamic noncooperative game theory*, Vol. 23, SIAM, 1999.

[63] National Highway Traffic Safety Administration, "Right-of-way Rules," 2018, https://one.nhtsa.gov/DOT/NHTSA/NTI/Article/RightOfWay/RightOfWayRules.pdf.

[64] Kalra, N. and Paddock, S. M., "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, Vol. 94, 2016, pp. 182–193.

[65] Li, N., Oyler, D. W., Zhang, M., Yildiz, Y., Kolmanovsky, I., and Girard, A. R., "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on Control Systems Technology*, Vol. 26, No. 5, 2017, pp. 1782–1797.

[66] Li, N., Yao, Y., Kolmanovsky, I., Atkins, E., and Girard, A. R., "Game-theoretic modeling of multi-vehicle interactions at uncontrolled intersections," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[67] Tian, R., Li, N., Kolmanovsky, I., Yildiz, Y., and Girard, A. R., "Game-theoretic modeling of traffic in unsignalized intersection network for autonomous vehicle control verification and validation," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[68] Li, N., Zhang, M., Yildiz, Y., Kolmanovsky, I., and Girard, A., "Game theory-based traffic modeling for calibration of automated driving algorithms," *Control Strategies for Advanced Driver Assistance Systems and Autonomous Driving Functions*, Springer, 2019, pp. 89–106.

[69] Oyler, D. W., Yildiz, Y., Girard, A. R., Li, N. I., and Kolmanovsky, I. V., "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 1705–1710.

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

[71] Zhang, M., Li, N., Girard, A., and Kolmanovsky, I., "A finite state machine based automated driving controller and its stochastic optimization," *Dynamic Systems and Control Conference*, Vol. 58288, American Society of Mechanical Engineers, 2017, p. V002T07A002.

[72] Li, N., Kolmanovsky, I., Girard, A., and Yildiz, Y., "Game theoretic modeling of vehicle interactions at unsignalized intersections and application to autonomous vehicle control," *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 3215–3220.

[73] Tian, R., Li, S., Li, N., Kolmanovsky, I., Girard, A., and Yildiz, Y., "Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts," *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 321–326.

[74] Su, G., Li, N., Yildiz, Y., Girard, A., and Kolmanovsky, I., "A traffic simulation model with interactive drivers and high-fidelity car dynamics," *IFAC-PapersOnLine*, Vol. 51, No. 34, 2019, pp. 384–389.

[75] Albaba, M., Yildiz, Y., Li, N., Kolmanovsky, I., and Girard, A., "Stochastic driver modeling and validation with traffic data," *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 4198–4203.

[76] Li, H., Li, N., Kolmanovsky, I., and Girard, A., "Energy-efficient autonomous vehicle control using reinforcement learning and interactive traffic simulations," *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 3029–3034.

[77] Tian, R., Li, N., Kolmanovsky, I., and Girard, A., "Beating humans in a penny-matching game by leveraging cognitive hierarchy theory and Bayesian learning," *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 4652–4657.

[78] Costa-Gomes, M. A. and Crawford, V. P., "Cognition and behavior in two-person guessing games: An experimental study," *American Economic Review*, Vol. 96, No. 5, 2006, pp. 1737–1768.

[79] Costa-Gomes, M. A., Crawford, V. P., and Iriberri, N., "Comparing models of strategic thinking in Van Huyck, Battalio, and Beil's coordination games," *Journal of the European Economic Association*, Vol. 7, No. 2-3, 2009, pp. 365–376.

[80] Arad, A. and Rubinstein, A., "The 11-20 money request game: A level-k reasoning study," *American Economic Review*, Vol. 102, No. 7, 2012, pp. 3561–73.

[81] Shapiro, D., Shi, X., and Zillante, A., "Level-k reasoning in a generalized beauty contest," *Games and Economic Behavior*, Vol. 86, 2014, pp. 308–329.

[82] Brackstone, M. and McDonald, M., "Car-following: A historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, Vol. 2, No. 4, 1999, pp. 181–196.

[83] Kikuchi, S. and Chakroborty, P., "Car-following model based on fuzzy inference system," *Transportation Research Record*, 1992, pp. 82–82.

[84] McDonald, M., Wu, J., and Brackstone, M., "Development of a fuzzy logic based microscopic motorway simulation model," *Proceedings of Conference on Intelligent Transportation Systems*, 1997, pp. 82–87.

[85] Kallenberg, L., "Classification problems in MDPs," *Markov processes and controlled Markov chains*, Springer, 2002, pp. 151–165.

[86] Jaakkola, T., Singh, S. P., and Jordan, M. I., "Reinforcement learning algorithm for partially observable Markov decision problems," *Advances in Neural Information Processing Systems*, 1995, pp. 345–352.

[87] Federal Highway Administration, "NGSIM – Next generation simulation," Tech. rep., United States Federal Government, Washington, DC, USA. https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[88] Kincaid, P., "The rule of the road: an international guide to history and practice," 1986.

[89] Chen, L. and Englund, C., "Cooperative intersection management: A survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 2, 2016, pp. 570–586.

[90] Ma, X. and Andréasson, I., "Estimation of driver reaction time from car-following data: Application in evaluation of General Motor-type model," *Transportation Research Record*, Vol. 1965, No. 1, 2006, pp. 130–141.

[91] Rawlings, J., Mayne, D., and Diehl, M., *Model predictive control: Theory, computation, and design*, Nob Hill Publishing, 2017.

[92] Kuderer, M., Gulati, S., and Burgard, W., "Learning driving styles for autonomous vehicles from demonstration," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2641–2646.

[93] Ren, X., Wang, D., Laskey, M., and Goldberg, K., "Learning traffic behaviors by extracting vehicle trajectories from online video streams," *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2018, pp. 1276–1283.

[94] Pruekprasert, S., Zhang, X., Dubut, J., Huang, C., and Kishida, M., "Decision making for autonomous vehicles at unsignalized intersection in presence of malicious vehicles," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 2299–2304.

[95] Quiroga, C. A. and Bullock, D., "Measuring control delay at signalized intersections," *Journal of Transportation Engineering*, Vol. 125, No. 4, 1999, pp. 271–280.

[96] Transportation Research Board, *HCM 2010: Highway capacity manual*, Transportation Research Board, Washington, D.C., 5th ed., 2000.

[97] Claussmann, L., Carvalho, A., and Schildbach, G., "A path planner for autonomous driving on highways using a human mimicry approach with binary decision diagrams," *2015 European Control Conference (ECC)*, IEEE, 2015, pp. 2976–2982.

[98] Lord, D., Manar, A., and Vizioli, A., "Modeling crash-flow-density and crash-flow-V/C ratio relationships for rural and urban freeway segments," *Accident Analysis & Prevention*, Vol. 37, No. 1, 2005, pp. 185–199.

[99] Wymann, B., Dimitrakakisy, C., Sumnery, A., and Guionneauz, C., "TORCS: The open racing car simulator," 2015.

[100] Althoff, M. and Dolan, J. M., "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, Vol. 30, No. 4, 2014, pp. 903–918.

[101] Althoff, M. and Magdici, S., "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, Vol. 1, No. 2, 2016, pp. 187–202.

[102] Chen, Y., Peng, H., and Grizzle, J. W., "Fast trajectory planning and robust trajectory tracking for pedestrian avoidance," *IEEE Access*, Vol. 5, 2017, pp. 9304–9317.

[103] Ahn, H., Berntorp, K., Inani, P., Ram, A. J., and Di Cairano, S., "Reachability-based decision-making for autonomous driving: Theory and experiments," *IEEE Transactions on Control Systems Technology*, 2020, pp. 1–15.

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

[105] Nishi, T., Doshi, P., and Prokhorov, D., "Merging in congested freeway traffic using multipolicy decision making and passive actor-critic learning," *IEEE Transactions on Intelligent Vehicles*, Vol. 4, No. 2, 2019, pp. 287–297.

[106] Hoel, C.-J., Driggs-Campbell, K., Wolff, K., Laine, L., and Kochenderfer, M. J., "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, Vol. 5, No. 2, 2019, pp. 294–305.

[107] You, C., Lu, J., Filev, D., and Tsiotras, P., "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, Vol. 114, 2019, pp. 1–18.

[108] Nageshrao, S., Tseng, H. E., and Filev, D., "Autonomous highway driving using deep reinforcement learning," *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019, pp. 2326–2331.

[109] Hu, Y., Nakhaei, A., Tomizuka, M., and Fujimura, K., "Interaction-aware decision making with adaptive strategies under merging scenarios," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 151–158.

[110] Evestedt, N., Ward, E., Folkesson, J., and Axehill, D., "Interaction aware trajectory planning for merge scenarios in congested traffic situations," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2016, pp. 465–472.

[111] Bae, S., Saxena, D., Nakhaei, A., Choi, C., Fujimura, K., and Moura, S., "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network," *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 1209–1216.

[112] Bandyopadhyay, T., Won, K. S., Frazzoli, E., Hsu, D., Lee, W. S., and Rus, D., "Intention-aware motion planning," *Algorithmic foundations of robotics X*, Springer, 2013, pp. 475–491.

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

[114] Hubmann, C., Schulz, J., Becker, M., Althoff, D., and Stiller, C., "Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction," *IEEE Transactions on Intelligent Vehicles*, Vol. 3, No. 1, 2018, pp. 5–17.

[115] Lefkopoulos, V., Menner, M., Domahidi, A., and Zeilinger, M. N., "Interaction-aware motion prediction for autonomous driving: A multiple model Kalman filtering scheme," *IEEE Robotics and Automation Letters*, Vol. 6, No. 1, 2021, pp. 80–87.

[116] Hauskrecht, M., "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, Vol. 13, 2000, pp. 33–94.

[117] Pineau, J., Gordon, G., Thrun, S., et al., "Point-based value iteration: An anytime algorithm for POMDPs," *IJCAI*, Vol. 3, Citeseer, 2003, pp. 1025–1032.

[118] Kurniawati, H., Hsu, D., and Lee, W. S., "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." *Robotics: Science and systems*, Vol. 2008, Citeseer, 2008.

[119] Li, N., Girard, A., and Kolmanovsky, I., "Stochastic predictive control for partially observable Markov decision processes with time-joint chance constraints and application to autonomous vehicle control," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 141, No. 7, 2019.

[120] Li, N., Kolmanovsky, I., and Girard, A., "Tractable stochastic predictive control for partially observable Markov decision processes with time-joint chance constraints," *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 3276–3282.

[121] Li, S., Li, N., Girard, A., and Kolmanovsky, I., "Decision making in dynamic and interactive environments based on cognitive hierarchy theory, Bayesian inference, and predictive control," *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 2181–2187.

[122] Liu, K., Li, N., Kolmanovsky, I., and Girard, A., "A vehicle routing problem with dynamic demands and restricted failures solved using stochastic predictive control," *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 1885–1890.

[123] Boyd, S. and Vandenberghe, L., *Convex optimization*, Cambridge university press, 2004.

[124] Kumar, P. R. and Varaiya, P., *Stochastic systems: Estimation, identification, and adaptive control*, SIAM, 2015.

[125] Lee, J., *A first course in combinatorial optimization*, No. 36, Cambridge University Press, 2004.

[126] Adler, I., Resende, M. G., Veiga, G., and Karmarkar, N., "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming*, Vol. 44, No. 1, 1989, pp. 297–335.

[127] Lorenzen, M., Allgöwer, F., Dabbene, F., and Tempo, R., "An improved constraint-tightening approach for stochastic MPC," *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 944–949.

[128] Lorenzen, M., Allgöwer, F., Dabbene, F., and Tempo, R., "Scenario-based stochastic MPC with guaranteed recursive feasibility," *2015 54th Conference on Decision and Control (CDC)*, IEEE, 2015, pp. 4958–4963.

[129] Beck, J. M., Ma, W. J., Pitkow, X., Latham, P. E., and Pouget, A., "Not noisy, just wrong: the role of suboptimal inference in behavioral variability," *Neuron*, Vol. 74, No. 1, 2012, pp. 30–39.

[130] Acerbi, L., Vijayakumar, S., and Wolpert, D. M., "On the origins of suboptimality in human probabilistic inference," *PLOS Computational Biology*, Vol. 10, No. 6, 2014, pp. e1003661.

[131] Wang, Y. and Boyd, S., "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2009, pp. 267–278.

[132] Dontchev, A. L., Huang, M., Kolmanovsky, I. V., and Nicotra, M. M., "Inexact Newton-Kantorovich methods for constrained nonlinear model predictive control," *IEEE Transactions on Automatic Control*, Vol. 64, No. 9, 2018, pp. 3602–3615.

[133] Walker, K., Samadi, B., Huang, M., Gerhard, J., Butts, K., and Kolmanovsky, I., "Design environment for nonlinear model predictive control," *SAE Technical Paper*, 2016, pp. 2016–01–0627.

[134] Ernest, B. R., *Dynamic programming*, Courier Dover Publications, 2003.

[135] Li, N., Chen, H., Kolmanovsky, I., and Girard, A., "An explicit decision tree approach for automated driving," *Dynamic Systems and Control Conference*, Vol. 58271, American Society of Mechanical Engineers, 2017, p. V001T45A003.

[136] Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A. D., Grizzle, J. W., Ozay, N., Peng, H., and Tabuada, P., "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, Vol. 24, No. 4, 2015, pp. 1294–1307.

[137] Tee, K. P., Ge, S. S., and Tay, E. H., "Barrier Lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, Vol. 45, No. 4, 2009, pp. 918–927.

[138] Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P., "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, Vol. 62, No. 8, 2016, pp. 3861–3876.

[139] Camacho, E. F. and Alba, C. B., *Model predictive control*, Springer Science & Business Media, 2013.

[140] Borrelli, F., Bemporad, A., and Morari, M., *Predictive control for linear and hybrid systems*, Cambridge University Press, 2017.

[141] Garone, E., Di Cairano, S., and Kolmanovsky, I., "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, Vol. 75, 2017, pp. 306–328.

[142] Chen, Y., Hereid, A., Peng, H., and Grizzle, J., "Enhancing the performance of a safe controller via supervised learning for truck lateral control," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 141, No. 10, 2019.

[143] Li, Z., Kalabić, U., and Chu, T., "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," *American Control Conference*, IEEE, 2018, pp. 6390–6395.

[144] Latombe, J.-C., *Robot motion planning*, Vol. 124, Springer Science & Business Media, 2012.

[145] Bouguerra, M. A., Fraichard, T., and Fezari, M., "Viability-based guaranteed safe robot navigation," *Journal of Intelligent & Robotic Systems*, Vol. 95, No. 2, 2019, pp. 459–471.

[146] Hermand, E., Nguyen, T. W., Hosseinzadeh, M., and Garone, E., "Constrained control of UAVs in geofencing applications," *26th Mediterranean Conference on Control and Automation*, IEEE, 2018, pp. 217–222.

[147] Kolmanovsky, I. and Gilbert, E. G., "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical Problems in Engineering*, Vol. 4, No. 4, 1998, pp. 317–367.

[148] Blanchini, F., "Set invariance in control," *Automatica*, Vol. 35, No. 11, 1999, pp. 1747–1767.

[149] Raković, S., Kerrigan, E. C., Mayne, D. Q., and Kouramas, K. I., "Optimized robust control invariance for linear discrete-time systems: Theoretical foundations," *Automatica*, Vol. 43, No. 5, 2007, pp. 831–841.

[150] Rakovic, S. V. and Baric, M., "Parameterized robust control invariant sets for linear systems: Theoretical advances and computational remarks," *IEEE Transactions on Automatic Control*, Vol. 55, No. 7, 2010, pp. 1599–1614.

[151] Tedesco, F., Raimondo, D. M., and Casavola, A., "Collision avoidance command governor for multi-vehicle unmanned systems," *International Journal of Robust and Nonlinear Control*, Vol. 24, No. 16, 2014, pp. 2309–2330.

[152] Lucia, W., Franzè, G., and Sznaier, M., "A hybrid command governor scheme for rotary wings unmanned aerial vehicles," *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 2, 2020, pp. 361–375.

[153] Romagnoli, R., Couto, L. D., Goldar, A., Kinnaert, M., and Garone, E., "A feedback charge strategy for Li-ion battery cells based on reference governor," *Journal of Process Control*, Vol. 83, 2019, pp. 164–176.

[154] Elbanhawi, M. and Simic, M., "Sampling-based robot motion planning: A review," *IEEE Access*, Vol. 2, 2014, pp. 56–77.

[155] Li, N., Han, K., Girard, A., Tseng, H. E., Filev, D., and Kolmanovsky, I., "Action governor for discrete-time linear systems with non-convex constraints," *IEEE Control Systems Letters*, Vol. 5, No. 1, 2020, pp. 121–126.

[156] Bristow, D. A., Tharayil, M., and Alleyne, A. G., "A survey of iterative learning control," *IEEE Control Systems Magazine*, Vol. 26, No. 3, 2006, pp. 96–114.

[157] Di Cairano, S., Bernardini, D., Bemporad, A., and Kolmanovsky, I. V., "Stochastic MPC with learning for driver-predictive vehicle control and its application to HEV energy management," *IEEE Transactions on Control Systems Technology*, Vol. 22, No. 3, 2013, pp. 1018–1031.

[158] Rosolia, U. and Borrelli, F., "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, Vol. 63, No. 7, 2017, pp. 1883–1896.

[159] Folland, G. B., *Real analysis: Modern techniques and their applications*, Vol. 40, John Wiley & Sons, 1999.

[160] Herceg, M., Kvasnica, M., Jones, C. N., and Morari, M., "Multi-Parametric Toolbox 3.0," *European Control Conference*, IEEE, 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[161] Cotorruelo, A., Limon, D., and Garone, E., "Output admissible sets and reference governors: Saturations are not constraints!" *IEEE Transactions on Automatic Control*, Vol. 65, No. 3, 2020, pp. 1192–1196.

[162] Bemporad, A., "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 3, 1998, pp. 415–419.

[163] Li, Y., Li, N., Tseng, H. E., Girard, A., Filev, D., and Kolmanovsky, I., "Safe reinforcement learning using robust action governor," *arXiv preprint arXiv:2102.10643*, 2021.

APPENDIX A

Proof of RL Algorithm Convergence

Consider a finite-space *unichain* [85] Markov decision process (MDP) and the following *average reward per unit of time over a long run* associated with a policy π,

$$\bar{R}^{\pi} = \lim_{t_{\max}\to\infty} \frac{\mathbb{E}\left\{\sum_{t=0}^{t_{\max}-1} R^{\pi}(s_t)\right\}}{t_{\max}}. \tag{A.1}$$

Define the value functions for states s, observations o, state-action pairs (s,a), and observation-action pairs (o,a) corresponding to policy π as follows,

$$\begin{aligned}
V^{\pi}(s) &= \lim_{t_{\max}\to\infty} \sum_{t=0}^{t_{\max}-1} \mathbb{E}\left\{R^{\pi}(s_t) - \bar{R}^{\pi}\middle| s_0 = s\right\}, \\
V^{\pi}(o) &= \mathbb{E}\left\{V^{\pi}(s)\middle| o\right\} = \sum_{s} \mathbb{P}^{\pi}(s|o) V^{\pi}(s), \\
Q^{\pi}(s,a) &= R(s,a) - \bar{R}^{\pi} + \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s'), \\
Q^{\pi}(o,a) &= \mathbb{E}\left\{Q^{\pi}(s,a)\middle| o\right\} = \sum_{s} \mathbb{P}^{\pi}(s|o) Q^{\pi}(s,a),
\end{aligned} \tag{A.2}$$

where $\mathbb{P}(s'|s,a)$ is the MDP state transition kernel, and $\mathbb{P}^{\pi}(s|o)$ is the probability distribution of state s given observation o in steady state. Note that because the MDP is assumed to be unichain, $\mathbb{P}^{\pi}(s|o)$ exists, is unique, and depends on the policy π.

According to the above definitions, the following equality can be shown [86],

$$\sum_{a} \pi(o,a) Q^{\pi}(s,a) = V^{\pi}(s), \tag{A.3}$$

which also implies

$$\begin{aligned}
&\sum_{a} \pi(o,a) Q^{\pi}(o,a) = \sum_{a} \pi(o,a) \left(\sum_{s} \mathbb{P}^{\pi}(s|o) Q^{\pi}(s,a) \right) \\
&= \sum_{s} \mathbb{P}^{\pi}(s|o) \left(\sum_{a} \pi(o,a) Q^{\pi}(s,a) \right) = \sum_{s} \mathbb{P}^{\pi}(s|o) V^{\pi}(s) = V^{\pi}(o),
\end{aligned}$$

$$\sum_a \pi(o,a)\left(\sum_s \mathbb{P}^\pi(s|o)\left(Q^\pi(s,a)-V^\pi(s)\right)\right)=0. \tag{A.4}$$

Define

$$J(\hat{\pi},\pi,o)=\sum_a \hat{\pi}(o,a)\left(Q^\pi(o,a)-V^\pi(o)\right). \tag{A.5}$$

Consider a policy $\hat{\pi}$ such that $J(\hat{\pi},\pi,o)\geq 0$ for all o and $J(\hat{\pi},\pi,o)>0$ for some o. For instance, $\hat{\pi}$ can be a greedy policy with probability 1 to choose $a^*(o)$ and probability 0 to choose other actions, where $a^*(o)\in\arg\max_a\left(Q^\pi(o,a)-V^\pi(o)\right)$.

We update the policy from π to π^ε according to

$$\pi^\varepsilon=(1-\varepsilon)\pi+\varepsilon\hat{\pi}, \tag{A.6}$$

where $\varepsilon>0$ is a (sufficiently small) update step size.

We will show that such an update leads to $\bar{R}^{\pi^\varepsilon}>\bar{R}^\pi$.

Because the MDP is assumed to be finite-space, $\max_{s,a}R(s,a)$ exists. It is easy to see from (A.1) that $\bar{R}^\pi$, over all policies π, is upper bounded by $\max_{s,a}R(s,a)$. Therefore, if we update the policy as above iteratively and ensure $\bar{R}^{\pi_{t+1}}>\bar{R}^{\pi_t}$ for all $t=0,1,\ldots$, the average reward must converge as $t\to\infty$.

Note also that a policy $\hat{\pi}$ such that $J(\hat{\pi},\pi,o)>0$ may not exist. On the one hand, a greedy policy guarantees $J(\hat{\pi},\pi,o)\geq 0$. On the other hand, if $J(\hat{\pi},\pi,o)\leq 0$ for all $\hat{\pi}$ and o, then π is already a locally optimal policy.

We now show

$$\bar{R}^{\pi^\varepsilon}-\bar{R}^\pi=\varepsilon\sum_o \mathbb{P}^\pi(o)J(\hat{\pi},\pi,o)+\mathcal{O}(\varepsilon^2), \tag{A.7}$$

where $\mathbb{P}^\pi(o)$ is the probability distribution of observation o in steady state. If (A.7) holds true, then our policy update procedure (A.5)-(A.6) with sufficiently small $\varepsilon>0$ guarantees $\bar{R}^{\pi^\varepsilon}>\bar{R}^\pi$, and therefore, average reward convergence as discussed above.

Firstly, consider

$$J'(\pi^\varepsilon,\pi,o)=\sum_a \pi^\varepsilon(o,a)\left(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o)\left(Q^\pi(s,a)-V^\pi(s)\right)\right). \tag{A.8}$$

According to (A.4), we have

$$J'(\pi^\varepsilon,\pi^\varepsilon,o)=\sum_a \pi^\varepsilon(o,a)\left(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o)\left(Q^{\pi^\varepsilon}(s,a)-V^{\pi^\varepsilon}(s)\right)\right)=0. \tag{A.9}$$

Then, expand $J'(\pi^\varepsilon, \pi, o)$ as

$$
\begin{aligned}
&J'(\pi^\varepsilon, \pi, o) = J'(\pi^\varepsilon, \pi, o) - J'(\pi^\varepsilon, \pi^\varepsilon, o) \\
&= \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \left(Q^\pi(s,a) - V^\pi(s) - Q^{\pi^\varepsilon}(s,a) + V^{\pi^\varepsilon}(s) \right) \Bigg) \\
&= \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \left(Q^\pi(s,a) - Q^{\pi^\varepsilon}(s,a) \right) \Bigg) \\
&\quad - \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) \Bigg) \qquad \text{(A.10)} \\
&= \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \big(Q^\pi(s,a) - Q^{\pi^\varepsilon}(s,a) \big) \Bigg) - \sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \big(V^\pi(s) - V^{\pi^\varepsilon}(s) \big) .
\end{aligned}
$$

Using the definition of $Q^\pi(s,a)$ in (A.2), we further get

$$
\begin{aligned}
&J'(\pi^\varepsilon, \pi, o) \\
&= \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \Bigg(\bar{R}^{\pi^\varepsilon} - \bar{R}^\pi + \sum_{s'} \mathbb{P}(s'|s,a) \left(V^\pi(s') - V^{\pi^\varepsilon}(s') \right) \Bigg) \Bigg) \\
&\quad - \sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) \\
&= \left(\bar{R}^{\pi^\varepsilon} - \bar{R}^\pi \right) + \sum_a \pi^\varepsilon(o,a) \Bigg(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \Bigg(\sum_{s'} \mathbb{P}(s'|s,a) \left(V^\pi(s') - V^{\pi^\varepsilon}(s') \right) \Bigg) \Bigg) \\
&\quad - \sum_s \mathbb{P}^{\pi^\varepsilon}(s|o) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) . \qquad \text{(A.11)}
\end{aligned}
$$

Weight $J'(\pi^\varepsilon, \pi, o)$ by $\mathbb{P}^{\pi^\varepsilon}(o)$ and sum up over all o,

$$
\begin{aligned}
&\sum_o \mathbb{P}^{\pi^\varepsilon}(o) J'(\pi^\varepsilon, \pi, o) \\
&= \left(\bar{R}^{\pi^\varepsilon} - \bar{R}^\pi \right) + \sum_{o,a,s,s'} \mathbb{P}^{\pi^\varepsilon}(o) \pi^\varepsilon(o,a) \mathbb{P}^{\pi^\varepsilon}(s|o) \mathbb{P}(s'|s,a) \left(V^\pi(s') - V^{\pi^\varepsilon}(s') \right) \\
&\quad - \sum_{o,s} \mathbb{P}^{\pi^\varepsilon}(o) \mathbb{P}^{\pi^\varepsilon}(s|o) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) \\
&= \left(\bar{R}^{\pi^\varepsilon} - \bar{R}^\pi \right) + \sum_{o,a,s,s'} \mathbb{P}(s',s,a,o) \left(V^\pi(s') - V^{\pi^\varepsilon}(s') \right) - \sum_{o,s} \mathbb{P}^{\pi^\varepsilon}(s,o) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) \\
&= \left(\bar{R}^{\pi^\varepsilon} - \bar{R}^\pi \right) + \sum_{s'} \mathbb{P}(s') \left(V^\pi(s') - V^{\pi^\varepsilon}(s') \right) - \sum_s \mathbb{P}^{\pi^\varepsilon}(s) \left(V^\pi(s) - V^{\pi^\varepsilon}(s) \right) \\
&= \bar{R}^{\pi^\varepsilon} - \bar{R}^\pi , \qquad \text{(A.12)}
\end{aligned}
$$

where we have used

$$\begin{aligned}&\mathbb{P}^{\pi^\varepsilon}(o)\pi^\varepsilon(o,a)\mathbb{P}^{\pi^\varepsilon}(s|o)\mathbb{P}(s'|s,a) = \mathbb{P}^{\pi^\varepsilon}(o)\left(\mathbb{P}^{\pi^\varepsilon}(a|o)\mathbb{P}^{\pi^\varepsilon}(s|o)\right)\mathbb{P}(s'|s,a)\\&= \mathbb{P}^{\pi^\varepsilon}(o)\mathbb{P}^{\pi^\varepsilon}(s,a|o)\mathbb{P}(s'|s,a,o) = \mathbb{P}(s',s,a,o)\end{aligned} \tag{A.13}$$

to derive the third line from the second line.

As a summary, we have shown that

$$\bar{R}^{\pi^\varepsilon} - \bar{R}^{\pi} = \sum_o \mathbb{P}^{\pi^\varepsilon}(o) J'(\pi^\varepsilon, \pi, o). \tag{A.14}$$

According to (A.6), we have

$$\max_{o,a} |\pi^\varepsilon(o,a) - \pi(o,a)| = \varepsilon \max_{o,a} |\hat{\pi}(o,a) - \pi(o,a)| \leq \varepsilon. \tag{A.15}$$

It can be shown that under (A.15), there exists a constant $C > 0$ such that [86]

$$\begin{aligned}&\max_{s,o} \left|\mathbb{P}^{\pi^\varepsilon}(s|o) - \mathbb{P}^{\pi}(s|o)\right| \leq C\varepsilon,\\&\max_{o} \left|\mathbb{P}^{\pi^\varepsilon}(o) - \mathbb{P}^{\pi}(o)\right| \leq C\varepsilon.\end{aligned} \tag{A.16}$$

We now express $J'(\pi^\varepsilon, \pi, o)$ as

$$\begin{aligned}&J'(\pi^\varepsilon, \pi, o) = \sum_a \pi^\varepsilon(o,a)\left(\sum_s \mathbb{P}^{\pi^\varepsilon}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\right)\\&= \sum_{a,s} \pi^\varepsilon(o,a)\mathbb{P}^{\pi^\varepsilon}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\\&= \sum_{a,s} \left(\pi(o,a) + \Delta\pi(o,a)\right)\left(\mathbb{P}^{\pi}(s|o) + \Delta\mathbb{P}^{\pi}(s|o)\right)\left(Q^\pi(s,a) - V^\pi(s)\right)\\&= \sum_{a,s} \Big(\pi(o,a)\mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right) + \Delta\pi(o,a)\mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\\&\quad + \pi(o,a)\Delta\mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right) + \Delta\pi(o,a)\Delta\mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\Big)\\&= \sum_a \pi(o,a)\left(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\right)\\&\quad + \sum_a \Delta\pi(o,a)\left(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right)\right)\\&\quad + \sum_s \Delta\mathbb{P}^{\pi}(s|o)\left(\sum_a \pi(o,a)\left(Q^\pi(s,a) - V^\pi(s)\right)\right)\\&\quad + \sum_{a,s} \Delta\pi(o,a)\Delta\mathbb{P}^{\pi}(s|o)\left(Q^\pi(s,a) - V^\pi(s)\right),\end{aligned} \tag{A.17}$$

where $\Delta\pi(o,a) = \pi^{\varepsilon}(o,a) - \pi(o,a)$ and $\Delta\mathbb{P}^{\pi}(s|o) = \mathbb{P}^{\pi^{\varepsilon}}(s|o) - \mathbb{P}^{\pi}(s|o)$.

According to (A.4), the first term of (A.17) is 0. According to (A.3), the third term of (A.17) is also 0. According to (A.15) and (A.16), the last term of (A.17) is $\mathcal{O}(\varepsilon^2)$. As a summary, we have shown that

$$J'(\pi^{\varepsilon},\pi,o) = \sum_a \Delta\pi(o,a)\bigg(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^{\pi}(s,a) - V^{\pi}(s)\right)\bigg) + \mathcal{O}(\varepsilon^2). \tag{A.18}$$

Then, using (A.2), (A.4), (A.5), and (A.6), we obtain

$$\begin{aligned}
J'(\pi^{\varepsilon},\pi,o) &= \sum_a \left(\pi^{\varepsilon}(o,a) - \pi(o,a)\right)\bigg(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^{\pi}(s,a) - V^{\pi}(s)\right)\bigg) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon\sum_a (\hat{\pi}(o,a) - \pi(o,a))\bigg(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^{\pi}(s,a) - V^{\pi}(s)\right)\bigg) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon\sum_a \hat{\pi}(o,a)\bigg(\sum_s \mathbb{P}^{\pi}(s|o)\left(Q^{\pi}(s,a) - V^{\pi}(s)\right)\bigg) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon\sum_a \hat{\pi}(o,a)\left(Q^{\pi}(o,a) - V^{\pi}(o)\right) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2).
\end{aligned} \tag{A.19}$$

Combining (A.14) and (A.19) leads to

$$\begin{aligned}
\bar{R}^{\pi^{\varepsilon}} - \bar{R}^{\pi} &= \sum_o \mathbb{P}^{\pi^{\varepsilon}}(o) J'(\pi^{\varepsilon},\pi,o) = \sum_o \mathbb{P}^{\pi^{\varepsilon}}(o)\left(\varepsilon J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2)\right) \\
&= \varepsilon\sum_o \mathbb{P}^{\pi^{\varepsilon}}(o) J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2).
\end{aligned} \tag{A.20}$$

Then, using (A.16), we obtain

$$\begin{aligned}
\bar{R}^{\pi^{\varepsilon}} - \bar{R}^{\pi} &= \varepsilon\sum_o (\mathbb{P}^{\pi}(o) + \mathbb{P}^{\pi^{\varepsilon}}(o) - \mathbb{P}^{\pi}(o)) J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon\sum_o \mathbb{P}^{\pi}(o) J(\hat{\pi},\pi,o) + \varepsilon\sum_o (\mathbb{P}^{\pi^{\varepsilon}}(o) - \mathbb{P}^{\pi}(o)) J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2) \\
&= \varepsilon\sum_o \mathbb{P}^{\pi}(o) J(\hat{\pi},\pi,o) + \mathcal{O}(\varepsilon^2),
\end{aligned} \tag{A.21}$$

i.e., we have shown (A.7). This completes the convergence proof of the applied reinforcement learning algorithm. ■

APPENDIX B

Functions of Path Model

Firstly, we determine the entrance points $(x^{\text{en}}, y^{\text{en}})$ of forward lanes according to the following procedure: We first locate the n_{r} intersection corners[1]. The line segment connecting a pair of adjacent corners is the "entrance line" of the corresponding road arm. The entrance point $(x^{\text{en}}, y^{\text{en}})$ of each forward lane is determined as the intersection point of its center and the entrance line of the road arm it belongs to.

We now describe our path model. A path P is a smooth curve composed of three segments: The first segment is a line segment connecting the vehicle's initial point $(x^{\text{ini}}, y^{\text{ini}})$ and the entrance point $(x^{\text{en}}, y^{\text{en}})$ of the vehicle's origin lane. The third segment is also a line segment with the vehicle's terminal point $(x^{\text{term}}, y^{\text{term}})$ as one of its end points and extending in the direction of the vehicle's target lane. The second segment is an arc that connects the first segment and the third segment, tangential to the first segment at $(x^{\text{en}}, y^{\text{en}})$ and also tangential to the third segment, where the point of tangency is determined as the exit point $(x^{\text{ex}}, y^{\text{ex}})$[2]. The three segments of P can be described by the following equations:

$$\begin{aligned} &a_1 x + b_1 y + c_1 = 0 && \text{for} \quad 0 \le \rho < \rho^{\text{en}}, \\ &(x - x_{\text{c}})^2 + (y - y_{\text{c}})^2 = r^2 && \text{for} \quad \rho^{\text{en}} \le \rho < \rho^{\text{ex}}, \\ &a_2 x + b_2 y + c_2 = 0 && \text{for} \quad \rho \ge \rho^{\text{ex}}, \end{aligned} \tag{B.1}$$

where (a_1, b_1, c_1) and (a_2, b_2, c_2) are parameters for the line segments, which correspond to the coefficients $\big(\sin(\phi^i), -\cos(\phi^i), \frac{jw_{\text{lane}}}{2}\big)$ of (1.23) with i and j determined by the vehicle's origin lane and target lane, and $(x_{\text{c}}, y_{\text{c}}, r)$ are the coordinates of the center and the radius of

[1] An "intersection corner" is the intersection point of two adjacent road boundaries.

[2] So $(x^{\text{ex}}, y^{\text{ex}})$ is the other end point of the third segment.

the arc segment, which are determined by the following set of functions:

$$
\begin{aligned}
x_{\mathrm{c}} &= \frac{a_1 b_2 x^{\mathrm{ex}} - a_2 b_1 x^{\mathrm{en}} + a_1 a_2 y^{\mathrm{en}} - a_1 a_2 y^{\mathrm{ex}}}{a_1 b_2 - a_2 b_1} \\
y_{\mathrm{c}} &= \frac{b_1 b_2 x^{\mathrm{ex}} - b_1 b_2 x^{\mathrm{en}} + a_1 b_2 y^{\mathrm{en}} - a_2 b_1 y^{\mathrm{ex}}}{a_1 b_2 - a_2 b_1} \\
r &= \frac{\sqrt{a_1^2 + b_1^2}\big(a_2 y^{\mathrm{en}} - b_2 x^{\mathrm{en}} + b_2 x^{\mathrm{ex}} - a_2 y^{\mathrm{ex}}\big)}{a_1 b_2 - a_2 b_1} \\
x^{\mathrm{ex}} &= -\frac{-b_2^2 x_{\mathrm{c}} + a_2 b_2 y_{\mathrm{c}} + a_2 c_2}{a_2^2 + b_2^2} \\
y^{\mathrm{ex}} &= -\frac{-a_2^2 y_{\mathrm{c}} + a_2 b_2 x_{\mathrm{c}} + b_2 c_2}{a_2^2 + b_2^2},
\end{aligned}
\tag{B.2}
$$

where $(x^{\mathrm{en}}, y^{\mathrm{en}})$ are determined by the intersection layout as introduced above. Then, ρ^{ex} is determined as follows:

$$
\begin{aligned}
&\rho^{\mathrm{ex}} = \rho^{\mathrm{en}} + r\Delta\phi, \quad \Delta\phi = \arccos\left(\frac{u^\top v}{\|u\|\|v\|}\right), \\
&u = \big[x^{\mathrm{en}} - x_{\mathrm{c}}, y^{\mathrm{en}} - y_{\mathrm{c}}\big]^\top, \quad v = \big[x^{\mathrm{ex}} - x_{\mathrm{c}}, y^{\mathrm{ex}} - y_{\mathrm{c}}\big]^\top.
\end{aligned}
\tag{B.3}
$$

APPENDIX C

Autonomous Vehicle Control Approaches for Highway Driving

Stackelberg game-based policy

The Stackelberg game-based policy considers a game involving three players – the ego vehicle and two other vehicles immediately following it. The three players are assigned roles as the "leader," "first follower," and "second follower," and they choose actions to apply from the action space in Section 1.2.2 sequentially: the leader chooses its action first, followed by the first follower, and finally the second follower. Each player evaluates actions according to a utility function that consists of two parts. The first part, referred to as the positive utility, is defined as follows:

$$U_{\text{pos}} = \begin{cases} \min(d_{\triangle}, d_{\text{v}}), & \text{if there is a vehicle in front,} \\ d_{\text{v}}, & \text{otherwise,} \end{cases} \tag{C.1}$$

where $d_{\triangle}$ is the longitudinal distance to the vehicle in front, and d_{v} is the maximum visibility distance. The second part, referred to as the negative utility, is defined as follows:

$$U_{\text{neg}} = d_{\triangledown} - v_{\triangledown} T - d_{\min}, \tag{C.2}$$

where $d_{\triangledown}$ and $v_{\triangledown}$ are the distance to and the relative velocity of the vehicle in the immediate rear, T is a prediction time window, and $d_{\min}$ is the minimum distance required to allow a lane change and is set to be equal to the length of the vehicle's c-zone. This way, overtaking vehicles are taken into account and lane changes that cut off overtaking vehicles are discouraged.

The actions chosen by the leader, first follower, and second follower are denoted as a_{ℓ}, a_{f1}, and a_{f2}, respectively. The leader chooses its action to maximize the worst-case utility it may receive due to the uncertain actions of the two followers, i.e.,

$$a_{\ell}^{*} \in \arg\max_{a_{\ell}} \min_{a_{\text{f1}}, a_{\text{f2}}} \left[U_{\text{pos}} + U_{\text{neg}} \right]. \tag{C.3}$$

The two followers maximize their own utilities with the known choice of a_{ℓ}^{*}.

In particular, the autonomous ego vehicle is treated as the leader of the game, and the two vehicles immediately behind it (they can be in any lanes) are the followers.

Decision tree-based policy

The decision tree-based policy builds a tree of admissible action sequences (each sequence is referred to as an "action profile") and evaluates each action profile according to a metric function.

In our implementation, the decision tree has two layers. Each layer contains the seven actions in Section 1.2.2 as nodes. Therefore, $7^2 = 49$ action profiles are evaluated and compared. The evaluation metric is as follows:

$$R_{\text{total}} = w_{\text{l1}} R_{\text{l1}} + w_{\text{l2}} R_{\text{l2}}, \tag{C.4}$$

where $R_{\text{l1}}, R_{\text{l2}}$ are the reward (1.9) received after the layer-1 action is applied and after the layer-2 action is applied, respectively, and $w_{\text{l1}}, w_{\text{l2}} \geq 0$ are weights for the two layers.

After evaluating all action profiles, the ego vehicle applies the layer-1 action of the profile that has the highest total reward for one time step, updates its state, and then repeats this procedure at the next time step. In particular, when evaluating the action profiles, the ego vehicle assumes all other vehicles apply the action "maintain" over its planning horizon.

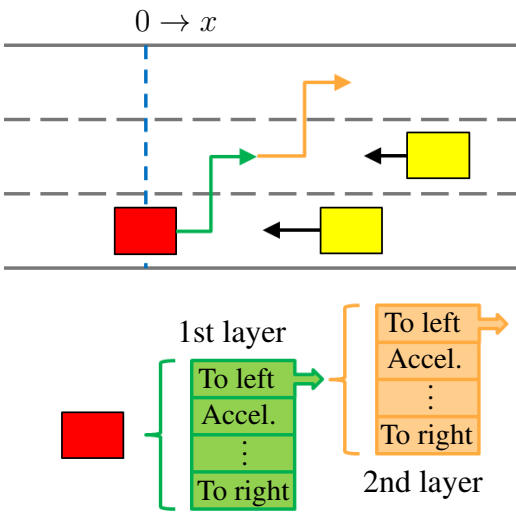


Figure C.1: Decision tree diagram. The black arrows indicate the relative velocities of the yellow cars with respect to the red car.

Policy activation criteria

Both the Stackelberg game-based policy and the decision tree-based policy are activated only when necessary and/or beneficial. When they are not activated, the autonomous vehicle drives according to simpler rules to reduce computational footprints. Their activation criteria are as follows:

1. Policy is activated if there are vehicles in region A and no vehicles in region B,
2. "Accelerate" until maximum speed/speed limit if there are no vehicles in region A,
3. "Safe mode," if there are vehicles in region B.

The above criteria are designed based on the following ideas: When there are no other vehicles within a certain neighborhood (region A) of the ego vehicle, the ego vehicle can accelerate and drive at its maximum speed/the speed limit safely. When there are some other vehicle(s) being very close to the ego vehicle (within region B), the ego vehicle should drive conservatively to increase safety, i.e., switch to a "safe mode." In our implementation, region A is designed to cover the center lines of adjacent lanes and region B is designed to cover the boundary lines of the current lane. This way, when some other vehicle in the vicinity is changing lanes into the ego vehicle's lane, the ego vehicle becomes aware of this vehicle before this vehicle enters its lane. Furthermore, we use the level-0 policy, (1.17), as the "safe mode" policy.

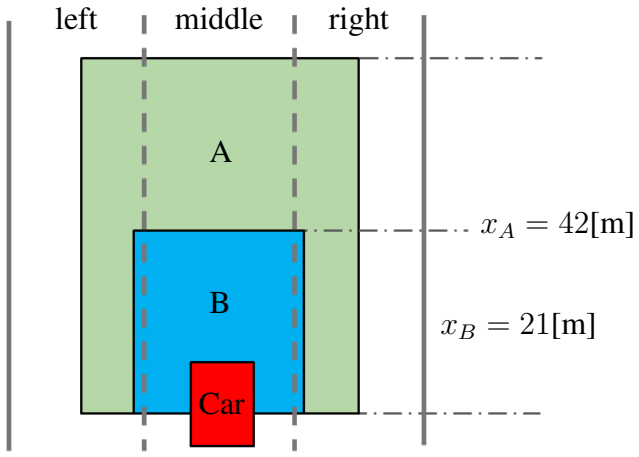


Figure C.2: Regions for policy activation criteria.

APPENDIX D

Adaptive Level-k Policy for Intersection Driving

We first define a level-k decision of vehicle i, for $k \geq 1$, as follows:

$$\gamma_i^k(t) = \{a_i^k(\tau|t)\}_{\tau=0}^{N-1} \in \arg\max_{\gamma_i \in \Gamma} \bar{R}_i^{\mathrm{i}}\left(\mathbf{s}(t), \gamma_i, \gamma_{-i}^{k-1}\right), \tag{D.1}$$

where $\gamma_{-i}^{k-1} = (\gamma_j^{k-1})_{j \neq i}$ represents the collection of level-$(k-1)$ decisions of all other vehicles interacting with vehicle i. The above formula defines a way to compute level-k decisions of all vehicles sequentially for $k = 1, 2, \ldots$, given a level-0 decision rule as the starting point.

Then, we consider the following decision process for the autonomous ego vehicle (vehicle i):

$$\gamma_i^{\mathfrak{K}}(t) \in \arg\max_{\gamma_i \in \Gamma} \sum_{\sigma_j \in \{0,\ldots,k_{\max}\}, j \neq i} \left[\left(\prod_{j \neq i} \mathbb{P}(k_j = \sigma_j|t) \right) \bar{R}_i^{\mathrm{i}}\left(\mathbf{s}(t), \gamma_i, (\gamma_j^{\sigma_j})_{j \neq i}\right) \right], \tag{D.2}$$

where $\mathbb{P}(k_j = \sigma_j|t)$ represents the ego vehicle's current belief in that vehicle j can be modeled as level-σ_j. The beliefs $\mathbb{P}(k_j = \sigma_j|t)$ are updated after each time step t according to the following algorithm:

For each $j \neq i$, if there exist $k, k' \in \{0, \ldots, k_{\max}\}$ such that $a_j^k(0|t) \neq a_j^{k'}(0|t)$, then

$$\tilde{\mathbb{P}}(k_j = \sigma_j|t+1) = \begin{cases} \mathbb{P}(k_j = \sigma_j|t) + \Delta\mathbb{P}, & \text{for each } \sigma_j \in \arg\min_k \left| a_j^k(0|t) - a_j^{\text{actual}}(t) \right|, \\ \mathbb{P}(k_j = \sigma_j|t), & \text{for all other } \sigma_j \in \{0, \ldots, k_{\max}\}, \end{cases}$$

$$\mathbb{P}(k_j = \sigma_j|t+1) = \frac{\tilde{\mathbb{P}}(k_j = \sigma_j|t+1)}{\sum_{k=0}^{k_{\max}} \tilde{\mathbb{P}}(k_j = k|t+1)}, \quad \text{for all } \sigma_j = 0, \ldots, k_{\max}; \tag{D.3}$$

otherwise, $\mathbb{P}(k_j = \sigma_j|t+1) = \mathbb{P}(k_j = \sigma_j|t)$ for all $\sigma_j = 0, \ldots, k_{\max}$. In (D.3), $a_j^{\text{actual}}(t)$ represents the observed acceleration of vehicle j (observed by the ego vehicle), and $\Delta\mathbb{P} > 0$ is an update step size.

The above model estimation algorithm is to increase the ego vehicle's belief in the level-σ_j model whose prediction most accurately matches the actual behavior of vehicle j. The update is triggered only when some level-k model predicts a different $a_j^k(0|t)$ value than others. This is because if all models predict the same $a_j^k(0|t)$ value, the ego vehicle will have no useful information to improve its beliefs.

The decision process (D.2) maximizes the weighted sum of rewards corresponding to all possible level combinations of the interacting vehicles, where the weights are the ego vehicle's current belief in each level combination, i.e., (D.2) maximizes the reward expectation.

APPENDIX E

Lemma 3.1

Lemma 3.1: Given an increasing sequence of sets X_k and an arbitrary U, we have $(\bigcup_{k=0}^{\infty} X_k) \sim U = \bigcup_{k=0}^{\infty} (X_k \sim U)$.

Proof: Note first that $Y_r = (\bigcup_{k=0}^{r} X_k) \sim U$ and $Z_r = \bigcup_{k=0}^{r} (X_k \sim U)$ are both increasing sequences of sets, and thus, their limits exist as $r \to \infty$ (in the set-theoretic sense). For each r, we have

$$Y_r = \bigcup_{j=0}^{r} Y_j = \bigcup_{j=0}^{r} \left(\left(\bigcup_{k=0}^{j} X_k \right) \sim U \right) = \bigcup_{j=0}^{r} (X_j \sim U) = Z_r, \tag{E.1}$$

where we have used the monotone increase of X_k to derive the third equality of (E.1). Since $Y_r = Z_r$ for every r, it must hold that $\lim_{r\to\infty} Y_r = \lim_{r\to\infty} Z_r$, i.e., $(\bigcup_{k=0}^{\infty} X_k) \sim U = \bigcup_{k=0}^{\infty} (X_k \sim U)$. ■