# INTELLIGENT DECISION MAKING FOR AUTONOMOUS DRIVING IN DYNAMIC INTERACTIVE ENVIRONMENTS

MINGFENG YUAN

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN
EARTH AND SPACE SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2024

# Abstract

The decision-making module is crucial for safe and efficient driving in autonomous vehicles (AVs). However, AVs face significant challenges in coexisting with human road users, making fast and optimal driving decisions, and operating in unknown traffic environments with only incomplete information. Unsignalized intersection and lane-changing scenarios are particularly representative of such challenges, which involves complex dynamic interactions. AVs need to assess and predict the driving preferences of nearby vehicles to optimize and adaptively adjust their own driving policies while considering uncertainties arising from incomplete observations. This dissertation investigates game-theoretic and learning-based methods to address these challenges.

In this dissertation, a novel approach for integrating game-theoretic decision-making with deep reinforcement learning (DRL) is proposed to enable AVs to navigate unsignalized intersections using an onboard sensor. The game model predicts the surrounding vehicles' reactions to the ego-vehicle's movements without relying on coordination or vehicle-to-vehicle communication. The proposed algorithm employs cognitive hierarchy reasoning and a DRL algorithm to achieve a self-play training mode for getting a near-optimal driving policy in a realistic simulator before transferring to the real world.

Second, this dissertation introduces a practical algorithm based on DRL for enhancing lane-changing decision-making, addressing low sample efficiency in DRL, and improving

the generalization capability in Imitation Learning (IL). To narrow the gap between simulation and reality (sim-to-real gap), a digital twin platform is developed for simulating LiDAR sensing, model training, and algorithm evaluations. To tackle multi-objective optimization and imbalanced data concerns, a hierarchical decision-making framework is proposed, breaking down the complex decision-making problem into subtasks for improved convergence of driving policies.

Third, a robust adaptive game-theoretic decision-making algorithm by utilizing receding horizon optimization and level-k game theory is presented. To reduce the potential safety risk arising from an inaccurate motion prediction of surrounding vehicles, the proposed approach can estimate driving aggressiveness of surrounding vehicles online. Then, the generated trustworthiness is used to formulate a safe, efficient, and robust adaptive driving policy. Additionally, a switching interaction graph is introduced into the adaptive level-k framework to reduce the computational complexity.

Validation on both a high-fidelity simulator and hardware confirms the feasibility, effectiveness, and real-time performance of the proposed methods. Overall, this dissertation contributes novel approaches to address decision-making challenges in AVs. The integration of game theory and model-based/model-free optimization showcases the potential for improving safety and efficiency of AVs' operation in dynamic traffic environments.

# Dedication

*To my parents, sister's family, and wife*

# Acknowledgments

Words cannot express my gratitude to my supervisor Prof. Jinjun Shan for the close guidance and unwavering support both academically and personally. He has been excellent advisor, mentor, and role model for me. I have been so privileged to have the opportunity to learn from and work with him. I would like to extend my sincere thanks to my supervisory committee, Prof. Costas Armenakis and Prof. Ryan Orszulik, for their support and insightful comments and suggestions. Also, I would like to thank Prof. Enzeng Dong (RIP), whose mentorship during my master's studies inspired my interest in and dedication to the field of complex system modeling and control. My gratitude also goes to everyone in the Spacecraft Dynamics Control and Navigation Laboratory (SDCNLab) since I had the pleasure of working with Drs. Ti, Shiyuan, Marc, Hassan, Adeel, Shuo, Junjie and Xiaoyu as well as Samira, William, Yibo, Hao, Hunter, Huarong, Ingredy, Amal, Yida, and Reza. You are an important part of my journey. I would also like to express my gratitude to my family, who was the best support along the way. I owe my deepest gratitude to my parents, my sister, and brother-in-law, who always trust, encourage and help me to pursue what I believe in. Without you, I would not be the person I am today. I thank my darling wife Shengwen for her love, understanding and constant support. Thank you for being my proofreader and sounding board. But most of all, thank you for being my best friend. My accomplishments will always be fuelled by you ... and for you.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| AVs | Autonomous Vehicles |
| ACC | Adaptive Cruise Control |
| ADAS | Advanced Driver Assistance Systems |
| AD | Autonomous Driving |
| BEV | Bird's-Eye View |
| CA | Collision Avoidance |
| CNN | Convolutional Neural Network |
| CSI | Camera Serial Interface |
| DQN | Deep Q-Network |
| DRL | Deep Reinforcement Learning |
| DA | Domain Adaptation |
| DR | Domain Randomization |
| DQfD | Deep Q-Learning from Demonstration |
| DRQfD | Deep Recurrent Q-Network from Demonstration |
| GAN | Generative Adversarial Network |
| HV | Host Vehicles |
| HDVs | Human-Driven Vehicles |
| IV | Interactive Vehicle |

| | |
|---|---|
| IL | Imitation Learning |
| IMU | Inertial Measurement Unit |
| ISW | Importance Sampling Weights |
| LSTM | Long Short-Term Memory |
| LiDAR | Light Detection and Ranging |
| LV | Leading Vehicle |
| MPC | Model Predictive Control |
| MDP | Markov Decision Processes |
| MARL | Multi-Agent Reinforcement Learning |
| OV | Other Vehicles |
| POMDP | Partially Observable Markov Decision Process |
| PID | Proportional-Integral-Derivative |
| PER | Prioritized Experience Replay |
| RD3QN PER | Recurrent Dueling Double Deep Q Network with PER |
| RNN | Recurrent Neural Network |
| ROS | Robot Operating System |
| Sim-to-Real | Simulation to Real World |
| TD-error | Temporal Difference Error |
| SDCNLab | Spacecraft Dynamics Control and Navigation Laboratory |
| UAVs | Unmanned Aerial Vehicles |
| URDF | Unified Robotics Description Format |

# 1 Introduction

## 1.1 Motivation

Recent advancements and progress in the field of Advanced Driver Assistance Systems (ADAS) have shown significant strides towards achieving widespread autonomous driving capabilities. Most recent ADAS of various car manufacturers have raised the level of autonomy over the past years. Ongoing projects are now targeting a SAE level of 4 or higher [1], signifying the pursuit of fully autonomous vehicles, which requires ADAS that are capable of handling complex situations. The goals of autonomous driving are to significantly increase safety in individual transport, to increase ride comfort and fuel efficiency as well as to improve mobility for the young, elderly, and disabled.

An AV is a sophisticated intelligent system that combines technologies for environmental sensing, decision-making and planning, and motion control [2]. Throughout this dissertation, AVs are sometimes referred to as ego vehicles, with both terms representing the same concept. The decision-making system serves as the "brain" of AVs, playing a crucial role in ensuring their safe and efficient operation. Consequently, designing a highly intelligent and robust decision system has become the central focus of research in the field of autonomous driving (AD). The decision-making module is designed to generate driving actions that align with experienced human drivers, taking into account the environment,

predicted trajectories of other road users, and the state estimation of the ego vehicle. These driving actions are subsequently integrated by the motion control module to ensure the satisfactory performance of the AVs [3, 4]. In addition, it has been estimated that AVs need to be running for 275 million miles without a fatality to assure the same rate of reliability as existing human drivers [5, 6]. There is no doubt that the road-testing phase is already costly and energy-consuming, not to mention poses a threat to pedestrians. Therefore, modelling various driving behaviors and testing decision-making algorithms in a high-fidelity simulator are of great significance to improve the safety of AVs, as well as reduce their dependence on road tests. To enable the AVs to navigate safely and reliably in complex real-world scenarios, various challenges need to be taken into account.

In the near future, AVs, together with other road users (e.g. human-driven vehicles (HDVs) and pedestrians), will be employed in traffic scenarios where the interactions of AVs and road users will constantly occur. Current mainstream autonomous driving solutions with level-2 autonomy [7] usually limits these interactions rather than accelerate them. For example, in complicated interactive cases, the AV decides to slow down and pause rather than spontaneously find another way through [8]. The main difficulties of decision-making for AVs in dynamic interactive traffic scenarios are twofold: 1) the behavioral patterns and intents of other vehicles are complex and cannot be directly observed, therefore, the capability of interaction with surrounding vehicles is needed and 2) the perception module of the AVs is uncertain due to noise and occlusions.

Solving decision-making problems for AVs in dynamic and interactive environments is challenging since it is almost impossible to achieve high automation in AVs by utilizing hard-coding or rule-based methods. The interaction process of AVs in real traffic scenarios has the following characteristics. First, the actions to be taken by AVs will be affected by the actions of surrounding vehicles, and vice versa. Secondly, vehicles not only cooperate

to avoid collision but also compete due to their different driving strategies, thus producing rich dynamic interaction behaviors. The driving policies of HDVs are unknown to AVs, which can only be estimated by observing actions taken by surrounding vehicles. Therefore, how to design robust, optimal, and computationally efficient decision-making algorithms has become a research hotspot.

In general, decision-making methods can be divided into classical methods and learning-based methods. Usually, AVs are operating in complex, dynamic environments with cooperation with other traffic participants. Classical methods such as control-based methods or optimization-based methods are not always effective in such driving environments due to poor robustness to uncertainty, high computational complexity, and difficulty in scaling to multi-vehicle scenarios. In addition, with the emergence of new powerful computational technologies, learning-based methods have indeed gained significant traction in the realm of AVs due to their ability to adapt and improve based on data. However, they come with their own set of challenges, including data dependency and interpretability. Therefore, it is essential to combine learning-based methods with classical methods judiciously to ensure safety, reliability, and robustness. The optimal approach should be leveraging the strengths of both paradigms while mitigating their respective weaknesses.

## 1.2 Related Work and Challenges

### 1.2.1 Collision Avoidance

The most crucial task of decision-making systems is to avoid collisions. There are four main categories of collision avoidance (CA) methods: motion planning-based, risk assessment-based, game-theoretic-based, and learning-based (supervised and DRL). Studies have been conducted in each category [9]. Motion planning methods are commonly used to solve CA

problems [10], but their application can be limited because the CA constraints they rely on are typically non-linear and non-convex. This can make the problem NP-hard [11].

Two methods for risk assessment have been developed, namely the deterministic approach and the probabilistic approach. The deterministic approach predicts whether a collision will occur or not by using a pre-determined threshold of specific indicators such as time headway or time to collision [12]. Although this approach has a low computational burden, it is not effective in more complex scenarios and does not model input data uncertainties. On the other hand, the probabilistic approach [13] uses empirical criteria as safety thresholds, limiting its adaptiveness in different traffic scenarios and leading to over-conservative actions. ***Existing safety methods often neglect the vehicle's ability to learn about surrounding road users during runtime interactions (make static modeling assumptions), leading to unsafe driving behaviors (overly conservative or aggressive)***.

### 1.2.2 Game-Theoretic Decision-Making

Game theory, mathematical models of strategic interaction among rational decision-makers, is a promising way to study the strategic reasoning of multiple vehicles and to model the interaction behavior between vehicles in mixed traffic scenarios. In traditional approaches, the interaction behavior between vehicles is modeled by a one-shot normal-form game, in which each vehicle will choose driving actions ("move" and "wait") through the payoff matrix without considering the dynamic characteristics of the vehicle. Nash-equilibrium [14, 15], Stackelberg game, and differential game approaches [16] have been used to model driving conflicts as well as to model interaction behaviors of vehicles at different traffic scenes. Level-k game theory is a cognitive hierarchy game model, which can be used to formulate vehicle interactions as a dynamic game. It considers the states of surrounding vehicles,

predicts their actions, and finds the optimal response, which differs from previous game models as it breaks down the Nash-equilibrium rational-expectations logic and assumes that drivers regard others as less sophisticated than themselves. There have been numerous previous works in the field of AD, which have utilized a combination of level-k game theory and receding horizon control. For instance, in [17], a multi-vehicle interaction model was proposed to address driving conflicts at unsignalized intersections, but only two-car interactions were considered due to computational limitations. There have been some studies using [17,18] game-theoretic approach to model vehicle interactions in highway or intersection cases. [19] modeled the interactions among vehicles at unsignalized intersections using the leader-follower game. [20] proposed an adaptive control strategy that accounts for the uncertainties in the vehicle dynamic model and the driver model estimation. ***However, taking MPC as an example, as the number of vehicles and prediction horizon increase, the computational complexity of such algorithms grows exponentially, making real-time decision-making for AVs challenging.***.

One promising way is to move works of driver-type estimation and behavior prediction to offline training and use a trained model online to reduce the run time of algorithms. In [21], an explicit online implementation scheme was proposed that uses function approximation techniques to avoid optimization problems in real-time. In [22], the algorithm was extended to different intersection shapes, and an IL-based algorithm for level-k control policies was proposed. [23] proposes a method to accurately predict lane-changing behaviors in complex transportation environments by integrating driving environments and drivers' cognitive processes using a fuzzy inference system and long short-term memory neural network (LSTM). But, performance is limited by the quality of training data, and the learning-based method lacks interpretability. ***To the best of our knowledge, there has been insufficient research on reducing the computational complexity of level-k***

***theory based decision-making algorithms while maintaining interpretability as well as increasing scalability.***

### 1.2.3 Reinforcement Learning based Decision-Making

In recent years, DRL-based autonomous driving technology has become a research hotspot in both academia and industry. Compared to IL, optimizing long-term goals and policy exploration give DRL a great potential advantage in achieving human-like driving behavior, which can cover extreme driving conditions. However, DRL training requires policy exploration. Training samples of colliding or nearly colliding is essential to the learning process, enabling future policy rollout to incorporate these critical experiences. How are we to provide safe multi-vehicle learning experiences without forgoing the realism of high-fidelity training data? There is a shortage of work that addresses this challenge.

***Applying DRL to the decision-making of AVs still faces several challenges that need to be addressed, including i) gap in simulation to the real world (sim-to-real) transfer, leading to difficulties in transferring models from simulation to reality; ii) low sample efficiency, requiring a massive amount of interactions; iii) incomplete observation information, leading to unstable training.***

#### 1.2.3.1 Sim-to-Real Transfer and End-to-End Training

The unapologetic nature of the trial-and-error process in DRL compounds the difficulty of ensuring functional safety. These adversities call for learning that first takes place in simulation before transferring to the real world [24]. Sim-to-real transfer is a class of methods to bridge the reality gap, connecting and integrating digital entities in simulations with their physical counterparts in the real world. ***The most challenging problem of this lies in the sim-to-real gap, which comes from three aspects: (i) the***

***difference in perception level; real traffic scenarios are more complex than the simulated environment. (ii) The raw data generated by the sensor hardware is noisy. (iii) The difference between the interaction features in the simulated environment and the real scene*** *[25].*

Currently, existing studies on sim-to-real transfer can be divided into two categories, namely Domain Adaptation (DA) and Domain Randomization (DR). Generative Adversarial Network (GAN) is a popular technique in the field of DA for transforming synthetic images to resemble those captured from the Target Domain [26, 27]. DR is a simple yet effective concept that operates by randomizing the dynamic properties of the Source Domain while undergoing training [27–29]. Recently, some researchers also start focusing on multi-agent reinforcement learning (MARL) sim-to-real transfer. They use DR to develop their multiple AVs and multiple unmanned aerial vehicles (UAVs) for different application backgrounds [30, 31].

ALVINN [32,33] is the first work of IL for an end-to-end AD. After ALVINN, more complex and successful end-to-end driving systems were developed in [34–36], which utilized multiple cameras, enabling the system to extract distance information and learn to control the lateral motion of vehicle in an end-to-end fashion. Recently, various DRL methods have been used to train Light Detection and Ranging(LiDAR) information-based end-to-end AD and navigation policies. In [37], the authors utilize unsupervised contrastive learning to differentiate between similar and dissimilar pairs of high-dimensional LiDAR data to learn representations of environments. While, in [38], the author used a convolutional neural network (CNN) to learn environmental features by converting LiDAR point clouds into gray images, achieving motion control in a static environment. To train the local navigation policy, in [39], a single frame of laser scan is combined with polar coordinates of waypoints to achieve collision-free exploration tasks. End-to-end algorithms map relationships

between observations and vehicle operations [40, 41], showing improvements in real-time performance and flexibility for multi-vehicle scenarios. ***However, uncertainties from unknown driving preferences were not considered in these works, potentially reducing AV safety.***

#### 1.2.3.2 Improving Learning Efficiency via Human Demonstration

Instead of learning from scratch, there are three categories of methods that can help DRL methods to speed up training process: (1) rule-based guidance [42], aiming to reduce unreasonable exploration behavior; (2) human-based guidance [43], incorporating human intervention during the training process; (3) IL based guidance [44, 45], utilizing human demonstration to pre-train and/or train DRL network.

Some researchers have proposed using prior knowledge from humans to guide the interaction of DRL in a training environment, as opposed to training policies from scratch. To prevent unsafe exploration, [42, 46] suggested the addition of a rule-based safety check module to the DRL-based control system to achieve fast training. In [47], a human-guidance-based learning method allowing human experts to intervene in the interaction process in real-time was proposed to speed up the training of RL agents. However, this approach comes at the cost of increasing human workload. To solve this problem, Hug-DRL [43] was proposed with the aim of reducing the human workload and enhancing the performance of RL for training and testing on AD by utilizing intermittent guidance. Another promising method is to use limited demonstration data to pre-train an expert policy that can achieve a reasonable level of performance. IL is an effective method to mimic expert behavior. However, it performs poorly in some scenarios where the training data is not covered, especially in some extreme conditions (e.g., collision, near-collision). Therefore, a reasonable idea is to combine the strengths of IL and DRL to efficiently train a

robust model. In [44, 45, 48], the authors employed an imitative expert policy to aid in the learning process of the actor-critic-based DRL agent for different traffic scenarios. Another approach is to design a loss function that enables the IL algorithm to learn the value function of actions in certain states, thereby achieving pre-training of the DRL network [49], rather than learning the expert behavior policy based on a classification task.

#### 1.2.3.3 Multi-Objective Learning Problem

Driving policy modeling, such as lane-changing policy, is a typical multi-objective problem, including adaptive cruise control (ACC), switching lanes, and merging. An effective way to solve such problems is to design each modularized skill through a hierarchical decision-making framework. The high-level strategic module is mainly responsible for macro-level decision-making with optimization goals including travel efficiency and safety. It will trigger a specific task that needs to be executed by technical level submodules in real-time based on the environment states. When the ACC module is activated, it will control the longitudinal motion of the AV with optimization goals including comfort, driving speed, and safety. When the lane-changing task is triggered, it will plan a collision-free and smooth trajectory to switch lanes. Currently, the industry commonly adopts a hierarchical decision-making framework for modular ADAS consisting of perception, decision-making, and control. Although rule-based methods can create policies for each submodule effectively, they face challenges when scaling to complex scenarios, exhibiting overly conservative driving behaviors.

There have been several studies addressing the decision-making problem for AD through a hierarchical decision framework. In [50], the authors used the Deep Q-Network (DQN) algorithm to learn the longitudinal control policy in a highway case with some assumptions about the high-level policy. In [51], the authors proposed an effective state-action abstrac-

tion and a hierarchical training framework for DRL to achieve multi-lane cruising, and demonstrated that the model trained in a non-dynamic simulation environment has good transferability in a more realistic simulator. However, a challenge faced by the previous methods is that the decision module overly relies on the performance of the perception module. The failure of the perception module may result in fatal traffic accidents [52]. The hierarchical decision-making framework used in this work is different from previous research. Decision-making and motion planning modules of AVs are trained in an end-to-end manner with LiDAR-based observation. These modules directly learn useful features from the raw sensor data, thereby preventing an excessive dependence on the performance of the perception module, especially the accuracy in predicting the future trajectories of surrounding vehicles.

## 1.3 Contributions

Figure 1.1 summarizes several mainstream methods currently used to address the challenges of human-vehicle interaction in driving scenarios, along with their respective advantages and disadvantages. This dissertation will focus on developing new decision-making algorithms by leveraging the strengths of these methods and addressing their respective shortcomings. The goal is to enhance the safety and robustness of autonomous driving, to reduce computational complexity without sacrificing performance and to improve the scalability of the algorithms.

This dissertation aims to address the research gaps reported in Section 1.2 to help AVs make fast and optimal driving decisions in complex and unknown traffic environments, and eventually be used in daily transportation. Specifically, the main contributions of this dissertation are as follows:

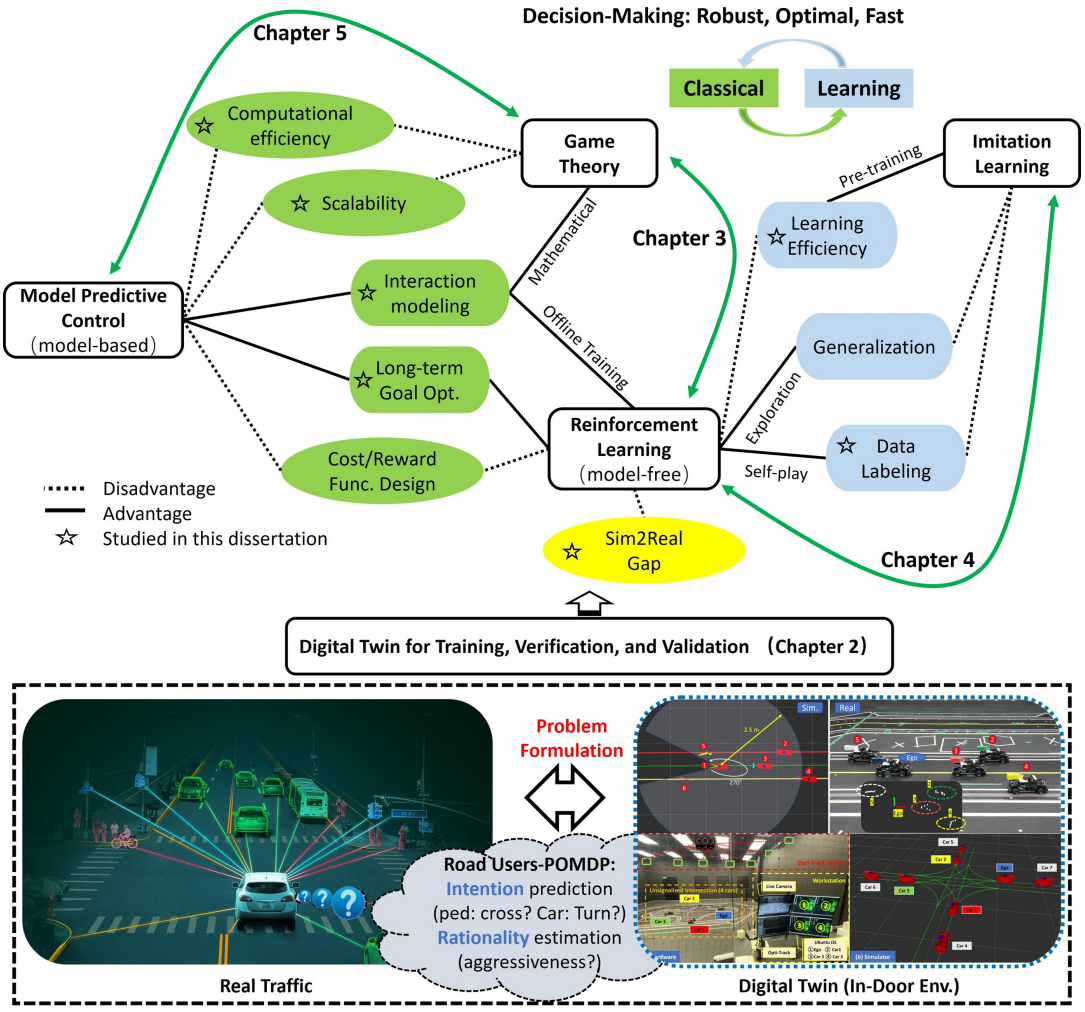



Figure 1.1: Methodology and challenges

### 1.3.1 Digital Twin for Training, Verification, and Validation

- To reduce the gap in sim-to-real transfer, a digital twin platform is developed for self-play training, and algorithm verification & validation, which largely differs from previous DRL-based works that usually rely on relatively simple simulations without considering the noise existing in observation or dynamics in physical AVs, thus imposing a large gap on their application to real vehicles;

- Sim-to-real transfer is successfully achieved by applying DR. Experimental verification is carried out, which indicates that the trained DRL models can be directly applied on the hardware platform after training, exhibiting driving behaviour consistent with it in the simulator;

- The developed digital twin platform supports various traffic scenarios (including unsignalized intersections and highways), which is capable of simulating different sensors, vehicle dynamics, replaying real traffic data, and evaluating decision-making algorithms in terms of scalability, multi-vehicle interaction, interpretability, and real-time implementation;

### 1.3.2 Reinforcement Learning Based Decision-Making

- An adaptive game-theoretic decision-making strategy with DRL is proposed for the AVs sharing the road with other drivers in a multi-agent traffic scenario, where all the drivers make strategic decisions simultaneously, instead of modeling the AV as a decision-maker and assuming predetermined actions for the rest of the drivers;

- A learning-efficient Deep Recurrent Q-learning from Demonstration (DRQfD) framework is proposed to model lane-changing decisions as a partially observable Markov decision process (POMDP), in which a small amount of expert data is employed to pre-train the DRL network parameters and train the IL policy to guide early policy exploration for the vehicle;

- An end-to-end modularized skill-based decision-making framework with two layers of hierarchy (strategic and tactical planner) is proposed to address the multi-objective problem in complex lane-changing scenarios;

- The decision-making scheme proposed in this dissertation relies solely on the onboard

sensors, without assuming any coordination, communication, or shared control with the surrounding cars. Also, the actual actions and movements of the vehicle are performed on a realistic simulator;

### 1.3.3 Scalable Game Theoretic Decision-Making

- An adaptive decision-making algorithm is designed using receding horizon optimization, level-k game theory, and a directed switching graph to address interactions between AV and other vehicles with varying driving preferences in complex unsignalized intersections;
- By utilizing the switching interaction graph, AVs can establish instantaneous interactive connections with other vehicles, allowing them to adapt their decision-making strategies to complex traffic situations, which effectively addresses the challenges of computational complexity and scalability faced by previous level-k based algorithms;
- The proposed adaptive strategy adjusts the size of anticipated disturbances based on the aggressiveness of other interacting vehicles, which provides a 'balanced' control action for the AV that is safer than aggressive strategies while also being more efficient than conservative strategies.

## 1.4 Dissertation Organization

This dissertation is organized in five chapters in addition to the Introduction. Chapter 2 mainly introduces the preliminary knowledge of learning algorithms and the AD platform. Section 2.1 introduces a DRL algorithm to be employed in this dissertation. Section 2.2 provides an overview of our developed digital twin platform used for training, verification, and validation.

Chapter 3 is about the DRL-based game theoretic decision-making algorithm for AVs in a dynamic interactive environment. Section 3.1 defines a decision-making problem at an unsignalized intersection scenario. Section 3.2 applies level-k game theory, LSTM network along with DRL algorithm to model decision-making for multiple cars. The algorithm is tested in Section 3.3 by both a simulator and hardware. The work is summarized and concluded in Section 3.4. The developments listed in this chapter, and related materials have been published in the journal article [40].

Chapter 4 is about improving the learning efficiency of the DRL algorithm for addressing decision-making problems of AVs in urban lane-changing scenario by leveraging human demonstrations. Section 4.1 introduces the research problem at a lane-changing scenario and a learning efficient DQRfD algorithm. Section 4.3 describes the human driving data set and the implementation details. Section 4.4 provides the experimental results. Section 4.5 concludes the work. The developments listed in this chapter, and related materials have been published in the journal article [4], book chapter [53], and conference paper [54].

Chapter 5 is about solving the scalability, robustness, and computational complexity of decision-making algorithm for AVs in complex multi-vehicle interaction scenarios. Section 5.1 presents the problem formulation at an unprotected left-turn scenarios. Section 5.2 introduces a kinematic model with pure pursuit controller. Section 5.3 describes the details of a scalable adaptive game-theoretic decision-making framework. Hardware and simulation results are provided in Section 5.4 to show the effectiveness of proposed method. Section 5.5 concludes this work. The developments listed in this chapter, and related materials have been published in the journal article [3].

Finally, the conclusion and the future work are presented in Chapter 6, followed by the references.

# 2 Learning Algorithm and Digital Twin Development

This chapter will introduce a DRL algorithm, which is used to optimize long-term goals and to solve POMDP problem. In addition, a digital twin platform will be also introduced, including hardware configurations, simulated traffic scenarios, sensor configurations, validation scheme for decision-making algorithms, as well as low-level control modules.

## 2.1 Recurrent Dueling Double Deep Q Network with Prioritized Experience Replay Algorithm

Recurrent dueling double deep Q-network with prioritized experience replay algorithm (RD3QN PER) combines the strengths of various DQN variants developed over the past decade, including neural networks for handling large state spaces, Double DQN for mitigating overestimation issues, Dueling DQN for effective learning, PER for efficient sampling and Deep Q-Learning from Demonstration (DQfD) for improving learning efficiency by utilizing expert demonstration. The history of DQN algorithms is illustrated in Figure 2.1. The key components of the RD3QN PER algorithm are briefly described below.

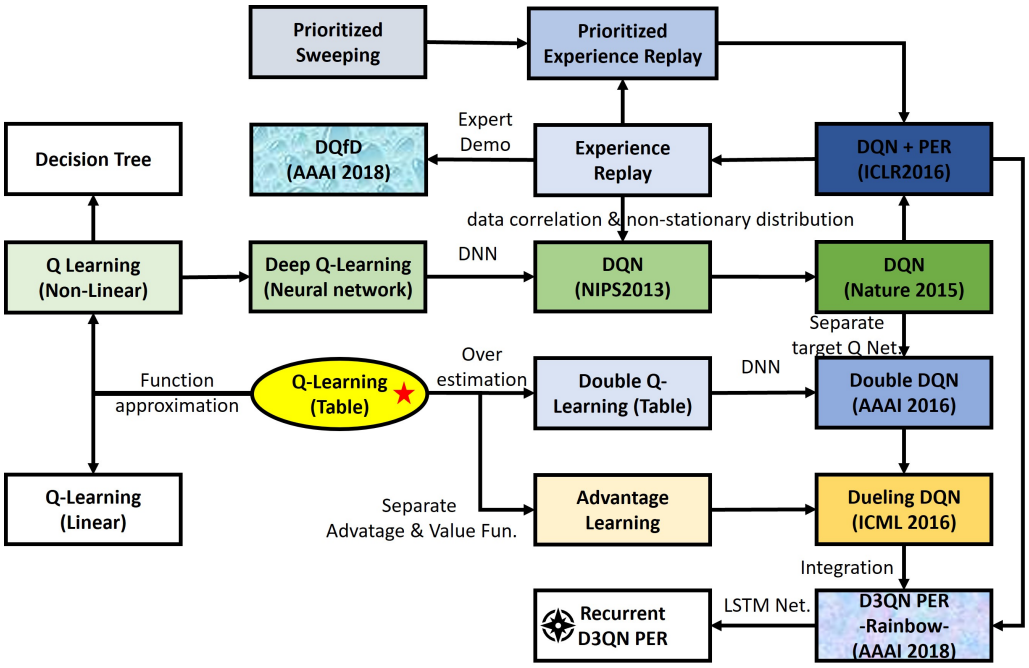



Figure 2.1: The history of DQN

### 2.1.1 Vanilla DQN Algorithm

The Vanilla DQN algorithm [55] is designed to address challenges in learning optimal policies for Markov Decision Processes (MDPs) with large state and action spaces. It leverages a neural network, known as the Q-network, to approximate the Q-function, which represents the expected cumulative reward for taking a particular action in a given state.

The Vanilla DQN algorithm follows a Q-learning framework with the addition of deep neural networks. It employs experience replay and target networks to stabilize training and improve sample efficiency. The key components of the algorithm include:

- **Q-Network:** A neural network that takes the state as input and outputs Q-values for each possible action. The network is trained to minimize the temporal difference error.

- **Experience Replay:** A technique where past experiences, represented as tuples of state, action, reward and next state, are stored in a replay buffer. During training, random batches are sampled from this buffer to break temporal correlations in the data.

- **Target Network:** To enhance stability during training, two separate networks are used – the Q-network and the target network. The target network's parameters are periodically updated to the Q-network's parameters.

The DQN algorithm uses Q-learning to provide labeled samples for the deep Q-network: $Q(s, a; \theta)$ with parameters $\theta$, which can be estimated by optimizing the following loss function Eq. (2.1) at iteration $i$:

$$L_i(\theta_i) = \mathbb{E}\left[\left(y_i^{DQN} - Q(s, a; \theta_i)\right)^2\right], \tag{2.1}$$

with

$$y_i^{DQN} = r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right), \tag{2.2}$$

where $r$ is the reward for taking action $a$ in given state $s$; $a'$ is next action taking in next state $s'$; $\gamma$ is discount factor; $\theta^-$ represents the parameters of target network $Q\left(s', a'; \theta^-\right)$ which are frozen for a fixed number of iterations while updating the online network $Q(s, a; \theta_i)$ by gradient descent. The specific gradient update is given by Eq. (2.3)

$$\nabla L_i(\theta_i) = \mathbb{E}\left[\left(y_i^{DQN} - Q(s, a; \theta_i)\right) \nabla_{\theta_i} Q(s, a; \theta_i)\right]. \tag{2.3}$$

### 2.1.2 Double DQN with PER

In Vanilla DQN, the Q-value estimates are used both for action selection and value estimation, which can lead to overestimation of action values. To address this issue, the

Double DQN algorithm [56] was introduced. The fundamental idea behind Double DQN is to decouple the selection and evaluation of actions.

The Double DQN algorithm modifies the Q-value update equation by using two separate networks: one for action selection and another for value estimation. The network responsible for action selection is used to choose the best action, while the network for value estimation is utilized to calculate the Q-value of that chosen action. This decoupling helps mitigate the overestimation bias observed in traditional DQN, resulting in more stable and accurate Q-value estimates.

The update rule for the Double DQN is given by Eq. (2.4):

$$Q(s,a) \leftarrow r + \gamma Q\left(s', \arg\max_{a'} Q\left(s', a'; \theta\right); \theta^{-}\right) \tag{2.4}$$

Double DQN has been shown to provide more accurate Q-value estimates and improved learning performance in various reinforcement learning tasks compared to traditional DQN.

The idea behind the PER [57] is that some experiences may be more important than others for training. We can take in priority experience that has a big error between deep Q-network and target network instead of selecting the experiences randomly. To generate the probability of being chosen for a replay, we have Eq. (2.5)

$$P(i) = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}}, \tag{2.5}$$

where $p_i = |\delta_i| + e$ and $|\delta_i|$ is the absolute temporal difference error (TD-error). Small constant $e$ assures that no experience has 0 probability to be taken. If $\alpha = 1$, it selects the experiences with the highest priorities while $\alpha = 0$ for pure uniform randomness. Since we use priority sampling, which leads to the bias toward high-priority samples. To correct this bias, importance sampling weights (ISW) is used to adjust the updating by reducing the weights of the often seen samples [57], according to Eq. (2.6)

$$\omega_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta} \tag{2.6}$$

where $\beta$ controls how much the ISW affect learning, and $N$ is the size of replay buffer.

### 2.1.3 Dueling DQN

Dueling DQN [58] introduces a novel architecture that disentangles the estimation of the value and advantage functions, enhancing the learning process in reinforcement learning tasks. Traditional DQN architectures face difficulties in distinguishing the importance of different actions in a given state. The Dueling DQN architecture aims to overcome this limitation by explicitly separating the estimation of the state value and the advantage of each action. This separation allows the neural network to better understand the intrinsic value of being in a particular state and the additional value associated with taking specific actions in that state.

The key innovation of Dueling DQN lies in the decomposition of the Q-function into two distinct components: the state value function $V(s)$ and the advantage function $A(a)$. The Q-function is then reconstructed using Eq. (2.7):

$$Q(s,a) = V(s) + \left( A(a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A\left(a'\right) \right) \tag{2.7}$$

where, $Q(s,a)$ represents the state-action value, $V(s)$ is the state value, $A(a)$ is the advantage of taking action $a$ in state $s$, and $|\mathcal{A}|$ denotes the total number of possible actions.

Dueling DQN offers several advantages over traditional DQN architectures. By modeling the state value and advantage, the algorithm gains improved stability and efficiency during training. This proves particularly beneficial in scenarios where certain actions have little impact on the overall outcome, as Dueling DQN enables the network to discern and assign appropriate values to each action independent of the state value.

### 2.1.4 Long Short-Term Memory Networks

LSTM networks [59] are a type of recurrent neural network (RNN) [60] architecture designed to overcome the vanishing gradient problem associated with traditional RNNs. LSTMs are well-suited for processing and predicting sequences of data due to their ability to capture long-range dependencies.

The key feature of LSTM networks is their memory cell, which can store information over extended periods. This is achieved through a set of gates, including an input gate, a forget gate, and an output gate. The input gate regulates the flow of new information into the cell, the forget gate controls the retention or removal of existing information, and the output gate determines the information to be output from the cell.

LSTMs have found widespread application in various domains, including natural language processing, time series analysis, and, relevant to autonomous driving, in modeling and predicting vehicle trajectories. Their ability to effectively handle sequential data makes them valuable for capturing temporal dependencies in dynamic driving scenarios.

## 2.2 ROS-Enabled Digital Twin

Digital twin for AV refers to a virtual replica or simulation of an AV and its driving environments, used for training, testing, and ensuring the safety and reliability of AV systems. It can be used to conduct comprehensive assessments, refine algorithms, and simulate various scenarios without real-world risks. Its advantages include accelerated development cycles, cost savings, enhanced safety, and the ability to explore edge cases and rare scenarios efficiently. By leveraging learning based methods like RL algorithm within the digital twin framework, AVs can learn and adapt their behaviour based on feedback from simulated environments. This approach enables AVs to improve decision-

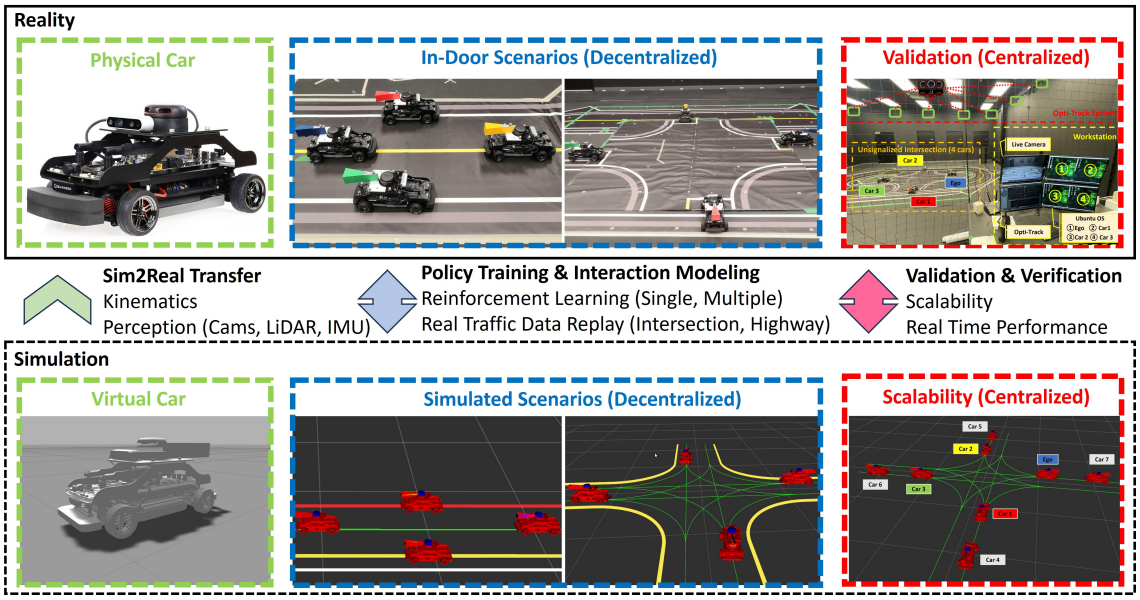



Figure 2.2: Leveraging computer simulations to improve real-world AD performance using three types of techniques: sim-to-real, self-play training, and algorithm validation.

making, navigate complex scenarios, and enhance overall performance more efficiently. Additionally, RL within the digital twin facilitates safer experimentation and iteration, contributing to the advancement of AV technology.

Traditional AD systems are typically composed of several key functional modules, including perception, mapping, prediction, decision-making & planning, and control [61]. For a considerable period, each module underwent internal technological iterations and solution evolution. However, with the expansion of AD applications and increased demands for functionality and performance, various issues with modular system architecture have become evident. These challenges include vague interface definitions between modules [62], difficulty aligning performance metrics and inconsistencies between optimization goals within modules. End-to-end unified modeling in AD integrates perception, decision-making, and control into a single model, simplifying integration, optimizing performance,

and improving adaptability compared to a modular scheme [33,35,38]. It reduces latency, handles uncertainty better, and offers holistic optimization. Despite challenges like data requirements, it's seen as the ultimate solution for AD architecture due to its efficiency and effectiveness [63–65].

Currently, numerous emerging technological solutions, such as DRL algorithms, offer significant advantages in enhancing overall AD performance [43, 66–68]. However, until these algorithms mature, there is substantial economic and research value in their preliminary application in indoor and simulation scenarios.

Inspired by this demand, this dissertation establishes a comprehensive digital twin AD research platform for training, validation, and verification of various AD algorithms as shown in Figure 2.2. The platform possesses the following advantages:

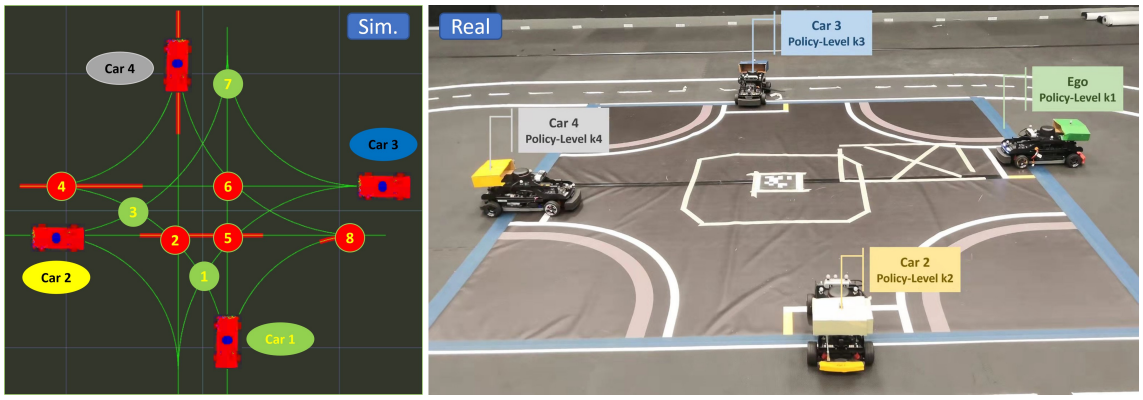
(i) A high-fidelity simulator developed based on ROS-Gazebo that can simulate the vehicle's kinematics and onboard sensors, including inertial measurement unit (IMU), LiDAR, and cameras.

(ii) A decentralized control scheme is employed to simulate complex dynamic interactions among different vehicles in diverse traffic scenarios.

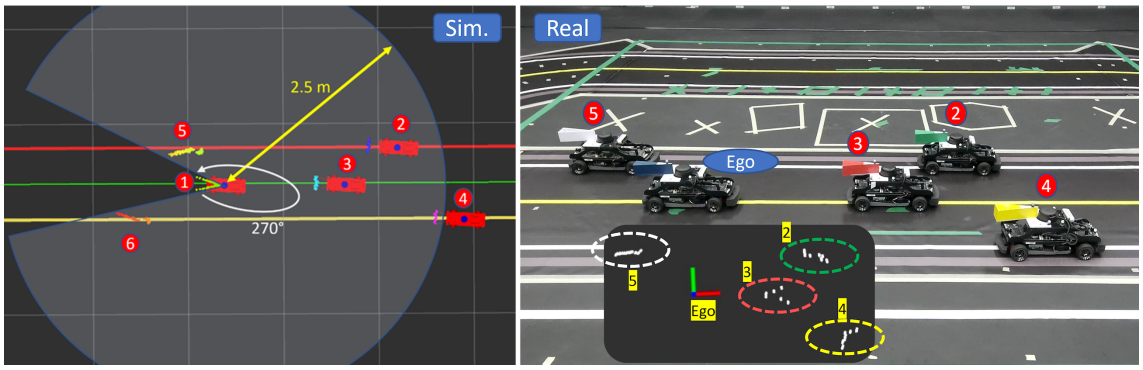
(iii) Algorithm validation encompasses real-time decision-making, control performance, and computation complexity of algorithms.

### 2.2.1 Sim-to-Real Transfer

To train diverse driving policies across various traffic scenarios through an end-to-end approach, a high-fidelity simulator is needed capable of simulating vehicle dynamics and sensor information. This is essential for the eventual transfer of trained driving policies from simulation to real-world scenarios. Our simulator supports various sensors, including




(a) Four Way Intersection (Studied in Chapter 3)




(b) Highway (Studied in Chapter 4)

Figure 2.3: Simulation and hardware environments

an IMU, an RGB-D camera (D435), four CSI cameras providing a 360 view of the vehicle, and a 2D LiDAR. The LiDAR can operate in two scan modes: standard mode with 360 points per scan and boost mode with 1160 points per scan.

In this dissertation, 2D LiDAR mounted on the top of the car will be used to provide observation information about the environment. To enhance the point cloud data from the LiDAR detection of surrounding cars, we mounted an isosceles triangle frame (or box) behind LiDAR. The scanning region of LiDAR is [135°, -135°] and the scanning range is set to [0.1, 3.0] m. The scanning frequency is 12 Hz. Figure 2.3a depicts an unsignalized

intersection scene, where each vehicle can be individually controlled and follows its own driving preferences (e.g., conservative or aggressive) according to its chosen path (turning left, going straight, or turning right). The left image in Figure 2.3b shows a bird's-eye view (BEV) of the simulated environment, displaying the LiDAR scan of surrounding vehicles from the perspective of the ego vehicle (denoted as "1"), with other numbers representing surrounding vehicles. The right image shows the corresponding hardware testing environment, where the ego vehicle is in the middle lane with a blue triangular frame. The point clouds scanned by the ego LiDAR hardware is provided in the right figure. Surrounding vehicles are distinguish with dashed circles in different colors.

### 2.2.2 Naturalistic Human Driving Data Set Replay

In a simulation environment, there are two schemes for validating algorithms. The first is modeling different driving behaviours and assigning these strategies to surrounding vehicles to validate the decision-making algorithm. Chapter 3 will detail how to achieve this goal through DRL and game theory. The second is real traffic datasets contain rich human driving information and can be used for open-loop [69] test to verify the algorithm's robustness by replaying traffic data in the simulation environment. The digital twin platform supports both of these requirements. The following describes how to preprocess traffic data and use it in the simulator. Chapter 4 will provide a detailed explanation of how to replay traffic data to test whether the decision-making algorithm successfully models human-like driving behaviours.

Taking highway as an example, vehicles with lane-changing behaviours can be randomly selected from real traffic data. The data used in this dissertation was recorded by eight cameras for 10 minutes of all vehicle information in the US101 road segment with a length of 640 m containing 3000 vehicles. The data includes vehicle number, global time, position,

speed, and lane number. A total of 25 types of information were recorded [70]. The road consists of six lanes, each with a lane width of 3.66 m. The sixth lane is the on-ramp, the off-ramp, and the auxiliary lane between them. The actual road structure is shown in Figure 2.4 [70–72]. Since some vehicles need to leave the expressway via the off-ramp, their lane-changing policies are different from that of other vehicles running on the first three lanes. Therefore, only these vehicles on the first three lanes are considered, and all selected trajectories are shown in Figure 2.5. To train/test lane-changing policies, a road section is reconstructed in the simulator as shown in Figure 2.6, each with a lane width of 0.366 m. Since our scaled car is 10 times smaller than real vehicles, we need to scale the traffic data proportionally in order to replay real traffic data for testing in the simulation environment.

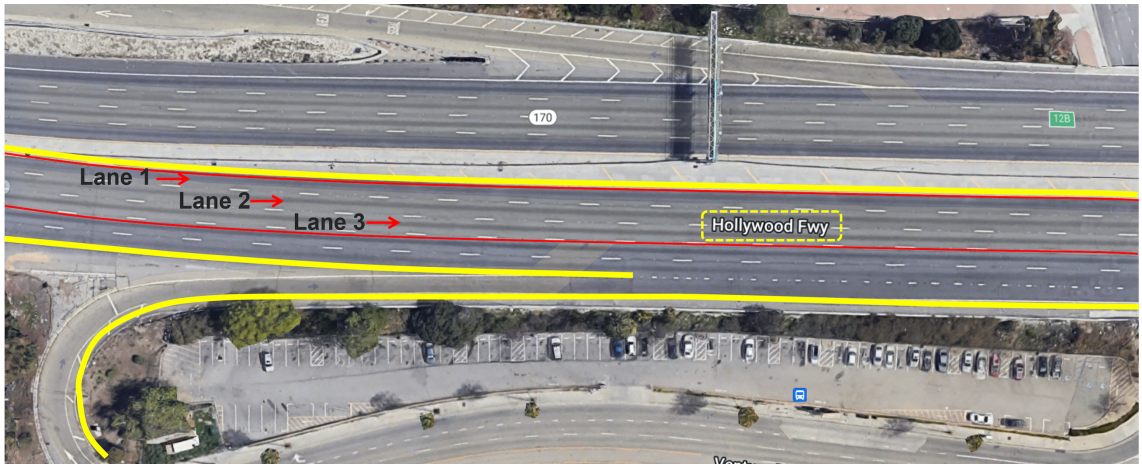



Figure 2.4: US-101 road segment

Considering the size of our in-door environment, the speed limit of our scaled car is set to 1.5 m/s in the simulator (equivalent to 15 m/s (54 km/h) for full-size vehicle in real scenario), therefore, we remove all cases with vehicle speeds above 15 m/s from the real traffic data. During the testing, if policy under test selects the same lane-changing action as the human driver in a given scenario, we consider that the driving behaviour

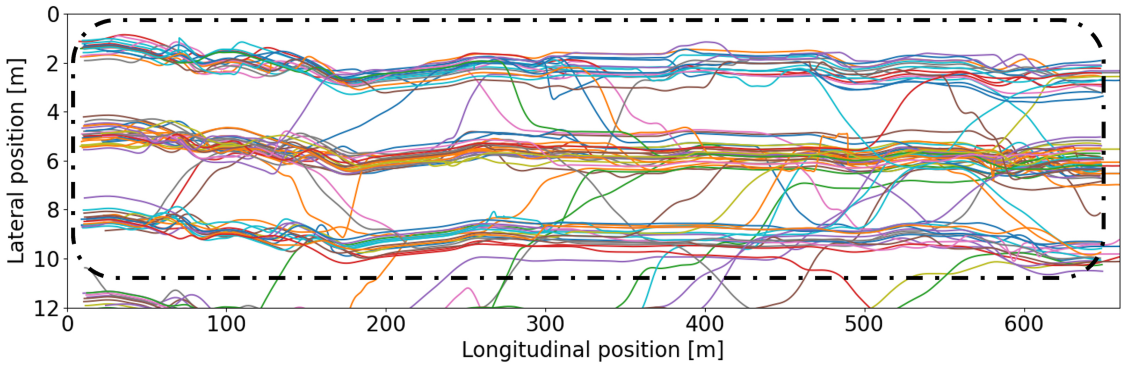



Figure 2.5: The trajectories of 100 randomly selected vehicles

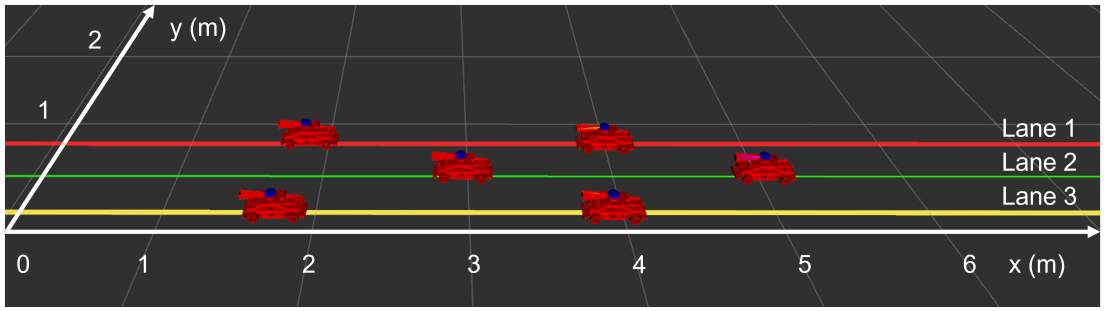



Figure 2.6: Simulated environment

of the current case is successfully modeled. Due to the presence of noise in the original data, the trajectories of surrounding vehicles are fitted by a fifth-order polynomial, which is introduced in Section 2.2.4.1. To obtain the boundary conditions for generating the trajectories of surrounding vehicles, we can directly scale the speed and distance in real traffic data by the ratio of the vehicle size $\lambda$ ($\lambda = 10$ in our case). The duration required for a vehicle to complete a lane-changing in the scenario remains consistent with real traffic data. Assuming that the initial and final time of the lane-changing in the real traffic data are $t_0$ and $t_n$, the testing duration for each scenario is denoted by Eq. (2.8)

$$\Delta t = t_n - t_0 \tag{2.8}$$

The position and speed information of vehicles in real traffic dataset can be expressed

as $\tilde{\boldsymbol{p}}^i(t) = \left(\tilde{p}_x^i(t), \tilde{p}_y^i(t)\right)$ and $\tilde{\boldsymbol{v}}^i(t) = \left(\tilde{v}_x^i(t), \tilde{v}_y^i(t)\right)$. $i$ is the vehicle number. $i = 0$ indicates a lane-changing vehicle. The boundary conditions including position and speed of vehicles in simulated environment can be calculated by Eq. (2.9) and Eq. (2.10):

$$\left(p_x^i(t), p_y^i(t)\right) = \frac{1}{\lambda}\left(\tilde{p}_x^i(t), \tilde{p}_y^i(t)\right) \tag{2.9}$$

and

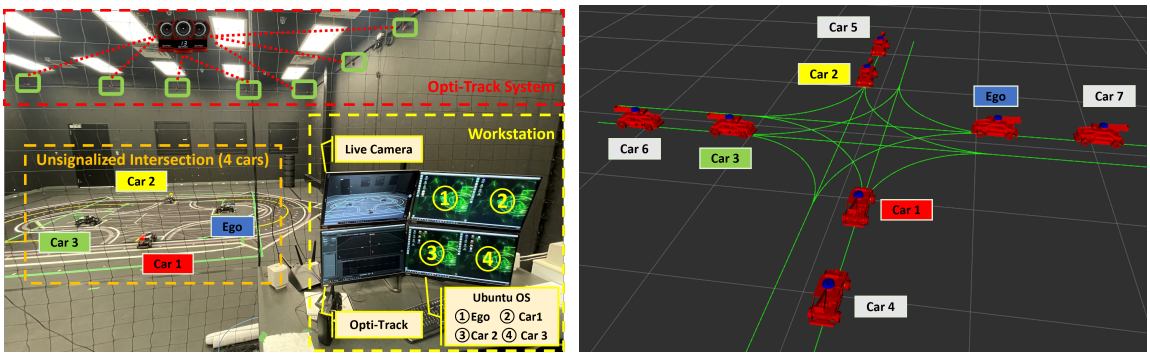
$$\left(v_x^i(t), v_y^i(t)\right) = \frac{1}{\lambda}\left(\tilde{v}_x^i(t), \tilde{v}_y^i(t)\right) \tag{2.10}$$

### 2.2.3 Validation and Verification

This digital twin platform supports the validation of both decentralized and centralized algorithms. The physical car is an open-architecture scaled model vehicle, powered with NVIDIA Jetson TX2 supercomputer, and equipped with a wide range of sensors, including cameras, LiDAR, encoders, and user-expandable IO. Algorithms can be run on each car with onboard computer, supporting a set of software tools including Python, Matlab, and Robot Operating System (ROS).

To experimentally validate the effectiveness of centralized algorithms, scaled model vehicles can also receive control commands via Wi-Fi from a workstation with Intel(R) Core(TM) i7-7700 CPU, see Figure 2.7a. An OptiTrack system consisting of 16 Flex 13 cameras is used to capture motion states of all cars and feed them back to the workstation via USB cables.

The high-fidelity simulator not only has the capability to simulate various sensors, vehicle kinematics, and different traffic scenarios but also possesses the advantage of easy scalability, as shown in Figure 2.7b. In the simulator, an arbitrary number of vehicles can be generated to validate the algorithms in terms of scalability and computational complexity. The ROS-Gazebo environment is developed on Ubuntu 18.04 workstation with Intel Xeon W-1290P CPU.




(a) Hardware platform (b) Intersection

Figure 2.7: Autonomous driving research studio (studied in chapter 5)

### 2.2.4 Low Level Control

The control performance needs to be guaranteed using controllers once the decision-making module generates acceleration and steering commands. This subchapter will introduce the proportional-integral-derivative (PID) controller at the low-level module for longitudinal control and the Stanley controller, illustrated in Figure 2.8, for steering control.

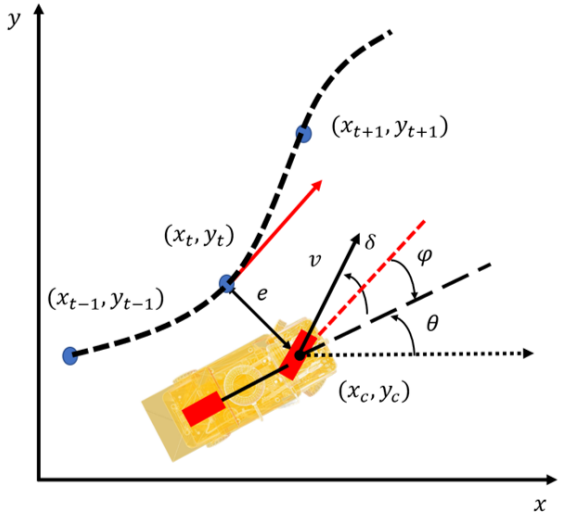



Figure 2.8: Stanley controller

#### 2.2.4.1 Trajectory Generation Module

To achieve smoother driving, a fifth-order polynomial [73] is utilized to create a trajectory for lane-changing. This approach enables us to set six boundary conditions (position, velocity, acceleration) at both $t_i = 0$ and $t_f = T$, which represents the time it takes to complete the lane-changing. The reference trajectory can be represented as Eq.(2.11), which includes six coefficients for each equation.

$$\begin{aligned} \mathrm{x(t)} &= a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \\ \mathrm{y(t)} &= b_5 t^5 + b_4 t^4 + b_3 t^3 + b_2 t^2 + b_1 t + b_0 \end{aligned} \tag{2.11}$$

Then the time-dependent parameter matrix can be defined as Eq.(2.12):

$$M_{6\times 6} = \begin{bmatrix} t_i^5 & t_i^4 & t_i^3 & t_i^2 & t_i & 1 \\ 5t_i^4 & 4t_i^3 & 3t_i^2 & 2t_i & 1 & 0 \\ 20t_i^3 & 12t_i^2 & 6t_i^1 & 2 & 0 & 0 \\ t_f^5 & t_f^4 & t_f^3 & t_f^2 & t_f^1 & 1 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 & 2t_f^1 & 1 & 0 \\ 20t_f^3 & 12t_f^2 & 6t_f^1 & 2 & 0 & 0 \end{bmatrix} \tag{2.12}$$

The module mentioned above will be used in Chapter 4 to generate smooth trajectories for an AV to perform lane changes from the current lane to a target lane.

#### 2.2.4.2 Lateral Control

The Stanley control method achieves lateral control by minimizing heading error $\psi$(t) and cross-track error $e(t)$, as illustrated in Figure 2.8. $\psi$(t) is the angle between the car heading and the trajectory heading, while $e(t)$ is the distance between the front wheels' center reference point $(x_c, y_c)$ and the path $(x_t, y_t)$ at the current time $t$. The linear speed of the front wheels is denoted as $v(t)$, and the steering angle as $\delta(t)$. The steering angle is limited by the vehicle's angle constraint, $\delta(t) \in [\delta_{\min}, \delta_{\max}]$. The controller equation is expressed as Eq.(2.13) [74].

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{k_s + v(t)}\right) \tag{2.13}$$

where softening constant $k_s$ is added to ensure the denominator be non-zero, and $k$ is a constant parameter serving as the weight of the cross track error.

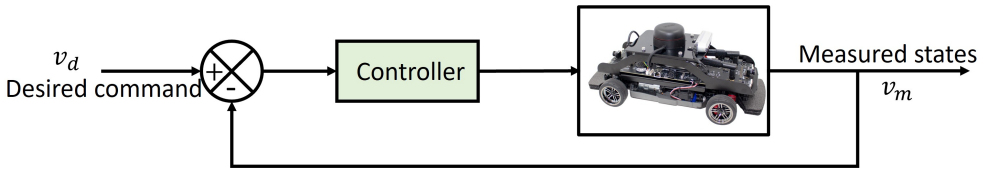



Figure 2.9: Longitudinal control

#### 2.2.4.3 Longitudinal Control

To regulate speed, the controller examines the car's velocity and modifies the throttle to align with the predetermined speed determined by the decision-making module (see Figure 2.9). The PID consists of three components [75]: the proportional, integral, and derivative terms. The proportional term $K_p$ is in direct proportion to the difference between the desired and actual speeds, while the integral term $K_I$ is proportional to the error's integral, and the derivative term $K_d$ is proportional to the error's derivative. The controller can be represented as Eq.(2.14)

$$u = K_P\left(v_d - v\right) + K_I \int_0^t \left(v_d - v\right) dt + K_D \frac{d\left(v_d - v\right)}{dt} \tag{2.14}$$

In motion control, there are two types of tracking errors used to assess the performance of robots: trajectory tracking error and path tracking error. The difference lies in that trajectory tracking typically involves four states: positional information ($x$ and $y$), velocity ($v$), and temporal information ($t$). However, path tracking error typically only considers positional deviation without accounting for time. The control performance of the scaled car for both longitudinal and lateral controllers is shown in Figure 2.10 and 2.11. It is

noticeable that the speed tracking error measures below 0.02 m/s, the trajectory tracking error measures below 0.05 m, and the path tracking error measures below 0.01 m. These values satisfy the criteria for self-driving vehicles.

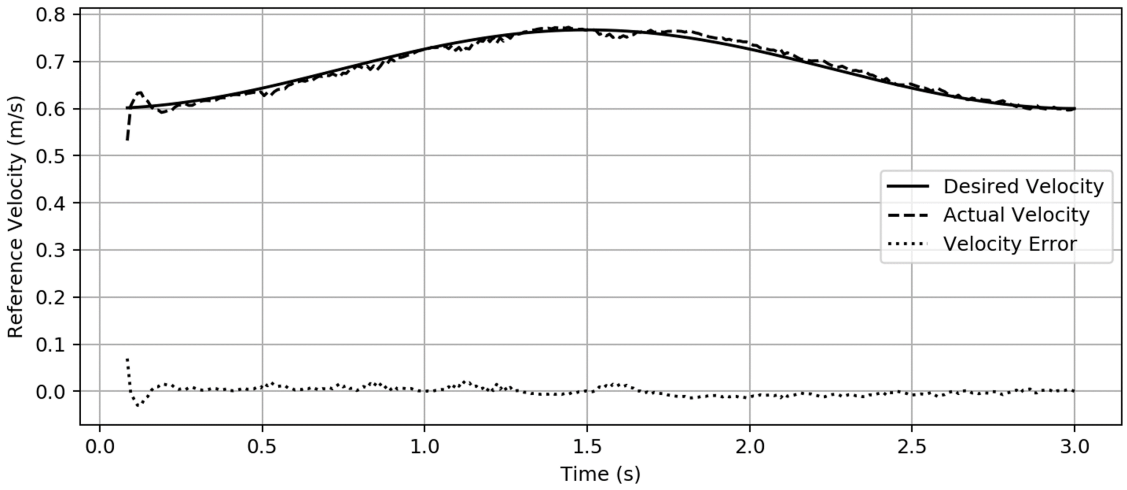



Figure 2.10: Speed control

## 2.3 End-to-End Training Scheme

Previous two sections introduced several variants of the DQN algorithm, the LSTM neural network for processing temporal information, and the digital twin platform. To train driving policy in an end-to-end manner, a multi-sensor fusion solution along with recurrent D3QN PER algorithm will be introduced in this section. Firstly, the features of 2D LiDAR point cloud data and car states are extracted through LSTM to generate latent space information as observation of DRL. If visual information needs to be incorporated into the decision-making framework, image information can be added through the stacking of images. After processing through a CNN, it can be concatenated with the output of the LSTM. Secondly, Q value is output through the full connection layer with dueling structure. Finally, the best action in each state can be generated by choosing the maximum

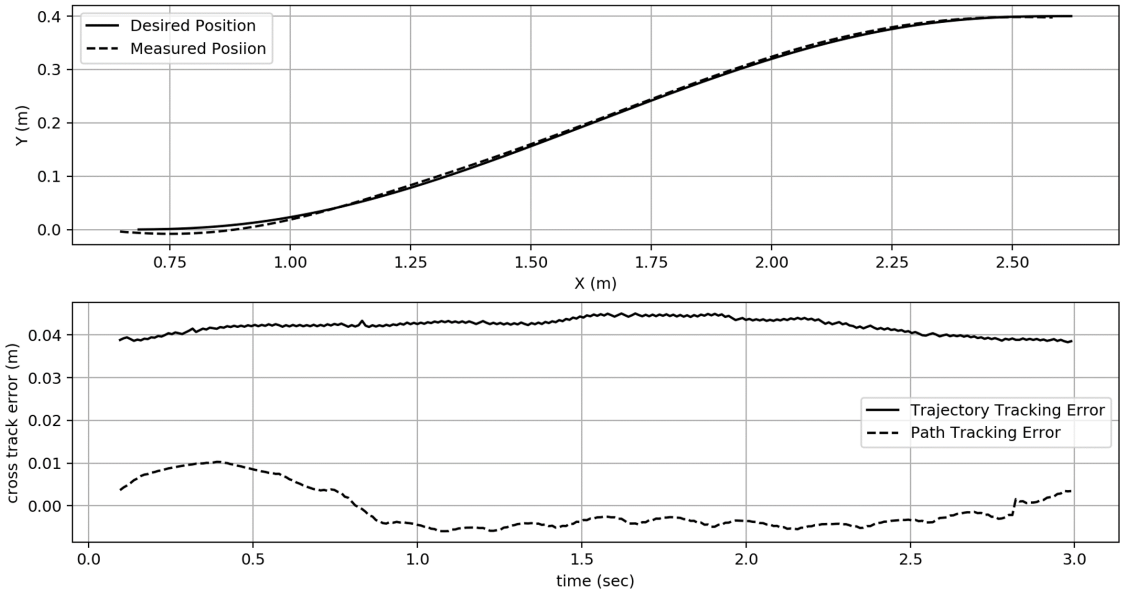



Figure 2.11: Trajectory tracking

Q value. Since the road centerline can be represented by a series of waypoints, we can use lateral controllers (e.g. pure pursuit controller and Stanley controller) to generate steering commands to drive the car on the road. The overall structure of RD3QN PER algorithm is shown in Figure 2.12. In Chapter 3 and Chapter 4, detailed studies will be provided on how to train driving policies with different reasoning levels to address decision-making problems related to unsignalized intersection and lane-changing scenarios.

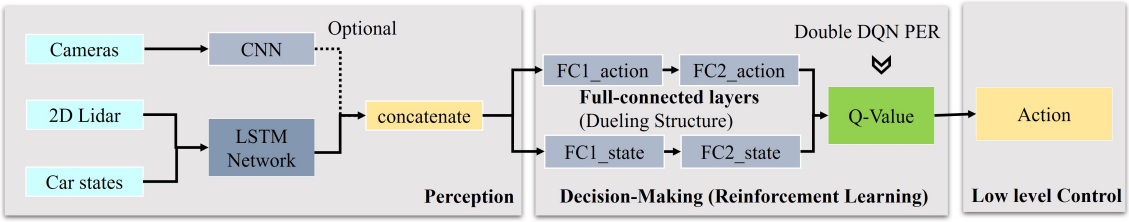



Figure 2.12: End-to-end training scheme with RD3QN PER algorithm

# 3 Reinforcement Learning Based Game Theoretic Decision-Making for AVs

This chapter applies the RD3QN PER algorithm, integrated with game theory, to model decision-makers at various reasoning levels within an unsignalized intersection scenario. Unlike approaches that isolate the ego vehicle as the sole decision-maker with predefined policy for surrounding vehicles, this method treats all drivers as strategic vehicles acting simultaneously in a multi-move environment. Through self-play training, different driving modes are modeled, ultimately leading to an adaptive driving policy. This policy can successfully navigate complex environments with a mix of conservative and aggressive driving behaviors, relying solely on onboard sensors to achieve safe and efficient driving.

## 3.1 Problem Definition

### 3.1.1 Modeling Scenarios

In this work, an unsignalized intersection is chosen as the scenario where each car chooses to enter the intersection area, and drivers constantly interact with the surrounding road users to safely and efficiently cross the intersection, since it is much more complex than other traffic scenarios.

The unsignalized intersection scenario has been introduced in Section 2.2.1, it consists

of 12 paths and four cars, among which car 1 is ego vehicle and the rest of the vehicles are opponents as shown in Figure 2.3a. To better show the results, the problem can be simplified as follows. The number of opponents ego encounters at the intersection is generated randomly, which can be 0, 1, 2 or 3. Ego has three tasks: turning left, going straight, and turning right. All opponents go straight but have two optional trained policies, namely conservative and aggressive driving behavior which is unknown to the Ego vehicle. The details about training of various driving policies are presented in Section 3.2. It's worth noting that the method proposed in this chapter can naturally be applied to more complicated scenarios, where all cars have no limitation of path. In this research, there are 81 scenarios in total as mentioned above since there are 24 combinations in the case of ego versus three opponents, 36 combinations in the case of ego versus any two opponents, and 18 combinations in the case of ego versus one opponent. There are also 3 combinations when the ego passes through an empty intersection. To focus more on complex scenarios, the probability of having four cars, three cars, two cars, and one car interaction scenario will occur with probability 40%, 30%, 20%, and 10% respectively during the training.

### 3.1.2 Observation Space

LiDAR is one of the most important sensors in the development of self-driving cars because of its ability to adapt to different lighting conditions and its robustness to the environment. The point clouds generated by LiDAR belong to long sequences information. To process the point cloud data, the LSTM network with 512 cell units is used in this chapter to deal with partially observed environments. Due to various strategies of the opponent cars, the ego car may be confused if it is heavily rewarded for selecting an action in one state and then penalised for choosing the same action in the same situation next time, making the training process unstable. Therefore, the action chosen by ego depends not only on the

current observation but also on a fixed number of the most recent observations.

The point cloud information is controlled to be 360 dimensions; the frequency is 12 Hz; the detection distance is [0.1, 3.0] m. In the intersection scenario, there are 8 collision areas where ego vehicle and opponent vehicle's path overlap, as shown in Figure 2.3a.

To combat limitations of using only 2D LiDAR, each vehicles' distance to the collision points are added. Also, vehicle's own state information, and velocities of the opponent vehicles, represented as an array [$v_1$, $v_2$, $v_3$, $v_4$, $p_1$, $p_2$, $p_3$, $p_4$], where $v_1$ to $v_4$ are the speeds of car 1 to car 4 at each time step, and $p_1$ to $p_4$ correspond to the path each car selected (obtaining from turn signal) to further supplement the observation space. It can be extended to more complex cases without loss of generality.

### 3.1.3 Action Space

The action space includes 5 possible actions that each vehicle can undertake:

(i) Maintain: Maintain current speed.

(ii) Accelerate: Increase speed of vehicle at 1.5 m/s$^2$, ignored if maximum velocity is reached.

(iii) Fast Accelerate: Increase speed of vehicle at 3 m/s$^2$, ignored if maximum velocity is reached.

(iv) Brake: Reduce speed of vehicle at 1.5 m/s$^2$, ignored if vehicle is stationary.

(v) Hard brake: Reduce speed of vehicle at 3 m/s$^2$, ignored if vehicle is stationary.

## 3.2 Driver Interaction Model

The driver interaction model developed in this section enables the modeling of driver-to-AV interactions through the use of level-k reasoning and RD3QN PER algorithm. The model is a "policy" which is a stochastic map from the observation space of the driver to their action space (see Section 3.1). In other words, this map assigns a probability distribution over possible actions for every observation. In the following sections, we explain how this model is trained.

### 3.2.1 Level-k Reasoning

In order to model the strategic decision-making process of human drivers, a game-theoretical concept named level-k reasoning is used. The level-k approach is a hierarchical decision-making concept and presumes that different levels of reasoning exist for different humans. The lowest level of reasoning in this concept is called level-0 reasoning. A level-0 agent is a non-strategic/naive agent since their decisions are not based on other agents' possible actions but consist of predetermined moves. In one level higher, a strategic level-1 agent exists, who determines their actions by assuming that the other agents' reasoning levels are level-0. Hence, the actions of a level-1 agent are the best responses to level-0 actions. Similarly, a level-2 agent considers other agents as level-1 and makes their decisions according to this prediction. The process continues following the same logic for higher levels. In some experiments, humans are observed to have at most level-3 reasoning, which may, of course, depend on the type of game being played. To generalize, all level-k agents, except level-0, presume that the rest of the agents are level-(k-1) and make their decisions based on this belief. Since this belief may not always hold true, the agents have bounded rationality [76, 77].

### 3.2.2 Combining Level-k Reasoning with RD3QN PER

To generate vehicles with different levels of reasoning, the RD3QN PER algorithm is run in the developed simulator, where the ego vehicle is the level-k learner. According to level-k theory, trained level-(k-1) policies (or predefined level-0 behavior) are assigned to the rest of the vehicles consisting the training environment.

In the proposed approach, the predetermined, non-strategic level-0 policy is the anchoring policy from which all the higher levels are derived using RD3QN PER. To obtain the level-1 policy, a traffic scenario is created where all drivers are level-0 agents except the ego car that is to be trained to best respond to the level-0 policy. There are 5 predefined velocities (0.3 m/s, 0.6 m/s, 0.9 m/s,1.2 m/s, and 1.5 m/s) that opponents can choose randomly. At level $k = 0$ of reasoning, opponent vehicles travel at the selected constant speed without considering the motions of others. To avoid collisions between opponents affecting the training process of the ego vehicle, we set priority for speed selection of each opponent. For example, car 2 can select a speed first, and then car 4 will randomly pick a speed from our pre-defined "safe speed set" to avoid collisions with car 2 at collision area 2 (see Figure 2.3a). Finally, car 3 will randomly select a safe speed to avoid collision with car 4. Once the training is completed, the ego becomes a level-1 driver. The procedure for obtaining the level-k policy through the proposed combination of level-k reasoning and RD3QN PER is explained in Table 3.1, where $n_d$ is the number of drivers.

Now, we have trained level-1 and level-2 policies in turn, but the problem is that these trained models are based on the assumption that all opponents are playing level-(k-1). If the true strategies chosen by opponents do not meet this assumption, the conflict between them will not be well resolved. The test results in Section 3.3 can well reflect this issue. To solve the problem, all opponents will choose policy among trained models following uniform distribution, and the ego car explores the adaptive strategy in this mixed

Table 3.1: Training of level-k agent

| **Algorithm 1** Obtaining the Level-k Policy |
|---|
| 1: Load the trained level-(k-1) (or rule-based level-0) policy, $\pi_{k-1}$; |
| 2: Set the reasoning levels of all opponents in the environment, $p_i$, to level-(k-1): $\pi_{p_i} = \pi^{k-1}, i = 2, 3, \ldots, n_d$; |
| 3: Initialize the ego driver's policy to a uniform action probability distribution over all states: $\pi_{p_1} = \pi^{uniform}$; |
| 4: Train the ego driver using RD3QN PER algorithm; |
| 5: At the end of the training, ego driver learns to best respond to $\pi^{k-1}$,therefore the resulting policy is the level-k policy, $\pi^k$. |

environment through RD3QN PER. Since each vehicle cannot access the driving policies of others, all vehicles can observe only a partial state of the traffic via the LiDAR sensor.

An LSTM recurrent neural network inside of the RD3QN PER algorithm is used to resolve the hidden state by making the chosen action that depends not only on the current observation but also on a fixed number of the most recent observations which is a black box way to learn the pattern of various driving policy instead of using Bayesian-based method to estimate the belief of driver model of other vehicles. Training process of adaptive policy is described in Table 3.2.

Table 3.2: Training of adaptive agent

| **Algorithm 2** Obtaining the Adaptive Policy |
|---|
| 1: Load the previously obtained level-1 and level-2 policy randomly, $\pi^1$ and $\pi^2$ (see Algorithm 1); |
| 2: Set the agents in the environment, $p_i$, as level-1 agents or level-2 agents: $\pi_{p_i} = \pi^1$ or $\pi_{p_i} = \pi^2, i = 2, 3, \ldots, n_d$; |
| 3: Initialize the ego driver's policy to a uniform action probability distribution over all states: $\pi_{p_1} = \pi^{uniform}$; |
| 4: Train the ego driver using RD3QN PER algorithm; |
| 5: At the end of the training, ego driver learns to best respond to $\pi^1$ and $\pi^2$. Thus, the resulting policy is the adaptive policy, $\pi^{adap}$. |

It is noted that the hierarchical learning process explained above decreases the computational cost since at each stage of learning, the agents other than the ego agent use previously trained policies and hence become parts of the environment. This helps to obtain traffic scenarios, containing a mixture of different levels, where all the vehicles are simultaneously making strategic decisions. This sharply contrasts conventional decision-making approaches, in crowded traffic, where one driver is strategic decision-maker and the rest are assigned predefined policies that satisfy certain kinematic constraints. A visual representation of the process of combining level-k reasoning and RD3QN PER is given in Figure 3.1.

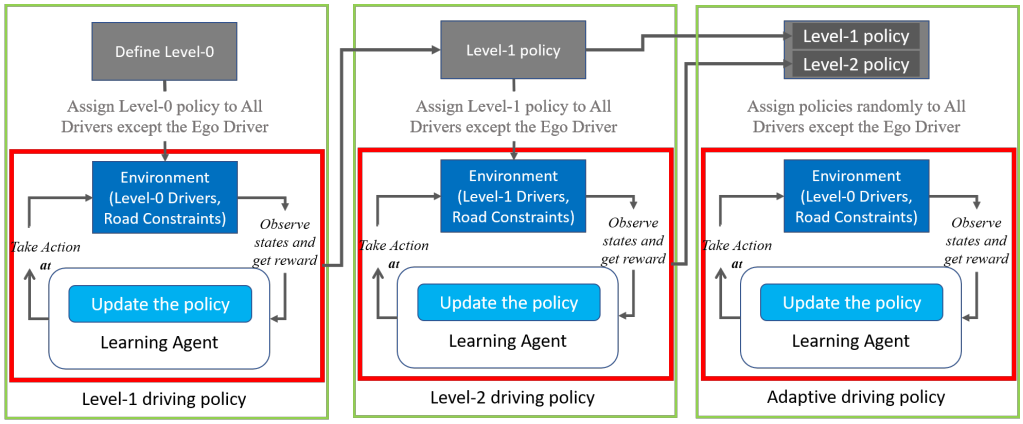



Figure 3.1: Combination of level-k theory and RL

### 3.2.3 Setting of Algorithm Parameters and Reward Function

Table 3.3 shows the parameter settings of the RD3QN PER algorithm. To speed up the training process, each sequential training after obtaining the level-1 policy was done by loading the previous model to the ego vehicle and assigning the starting value of $\epsilon$ to 0.5 instead of 1.

Table 3.3: Parameter settings

| **Parameters** | **Values** |
|---|---|
| Discount factor ($\gamma$) | 0.95 |
| Learning rate | 0.001 |
| Starting (ending) value of $\epsilon$ greedy policy ($\epsilon$) | 1 (0.01) |
| Number of actions | 5 |
| Size of replay memory | 6000 |
| Maximum value of training steps | 150 |
| Number of steps to update the target network | 500 |
| Mini-batch size | 32 |
| Training steps ($N_{training}$) | 15000 |
| $e$ | 0.00001 |
| Priority ($\alpha$) | 0.6 |
| Adjusting the deviation ($\beta$) | 0.4 |

In order to avoid insufficient exploration and to accelerate convergence, the parameter of the $\epsilon$-greedy method decrease from 1 linearly according to the training steps, as shown in Eq. (3.1), and remain unchanged until it equals ending value.

$$\epsilon = \epsilon - \frac{1.0}{N_{training}}. \tag{3.1}$$

Parameter $e$ in the PER algorithm is used to prevent the saved experience from not being replay after TD-error equals 0. The exponent $\alpha$ determines how much prioritization is used, with $\alpha = 0$ corresponding to the uniform case. Parameter $\beta$ fully compensates for the non-uniform probabilities $P(i)$ if $\beta = 1$. It's increased linearly according to Eq. (3.2).

$$\beta = \beta + \frac{1.0}{N_{training}}. \tag{3.2}$$

The reward function shown in Table 3.4 is used to evaluate the performance of the ego vehicle, which encourages ego vehicles to learn efficient human-driving behaviours. A reward function was designed to penalize collisions or being in dangerous states, and reward efficient behaviours, such as reaching the destination or progressing.

Table 3.4: Setting of reward function

| **Conditions** | **Values** |
|---|---|
| Collision status | -5 |
| Dangerous zone status and car approaching | -0.1*$\Delta(t)$ |
| Changes in acceleration | -0.05*($\|a_t - a_{t-1}\|$) |
| Reach destination | 2 |
| Reach each checkpoint | 0.1 |

According to the setting of reward function, the ego car will receive a reward of 0.1 for reaching each checkpoint. If ego successfully reaches the destination, when going straight, it receives a reward of 2 points for passing the 20 checkpoints. Similarly, for turning left or right, the ego vehicle receives a reward of 1.5 or 1.2 points. When a collision occurs, 5 points are deducted. When ego vehicles and opponents cars are in a deadlock, 0.1 point will be deducted for each time-step, which will be a total of -1.5 points for the maximum number of steps each episode. When an ego vehicle reaches the destination, the score will be rewarded 2 points.

## 3.3 Simulation and Experimental Results

### 3.3.1 Comparison of Different Algorithms

To compare learning efficiency, all algorithms were trained 8,000 episodes in an environment where all opponents were level-0 reasoning. All curves in Figure 3.2 are smoothed with a moving average over 300 episodes. We can find that the prioritized reply and dueling structure are the two most crucial components of the RD3QN PER algorithm, in that removing either component caused a large drop in learning performance. Nature DQN and Dueling DQN perform worst during training, which could be caused by the overestimation mentioned in Section 3.2.

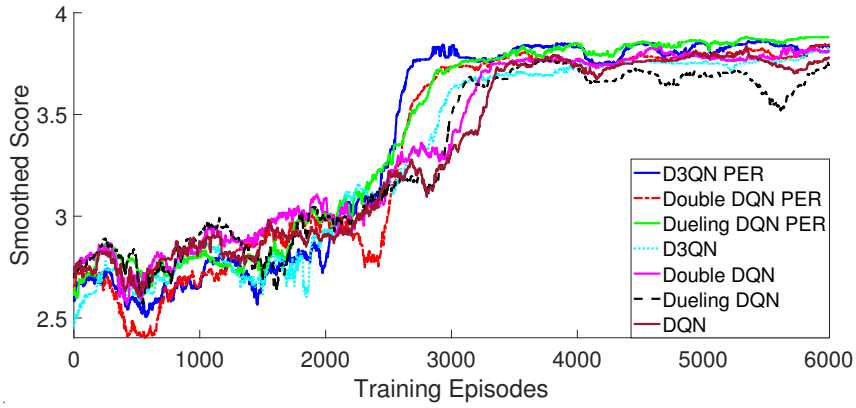



Figure 3.2: Comparison of various DQN algorithms

### 3.3.2 Simulation Results

According to the scheme proposed in Section 3.2, we have successfully trained level-1, level-2, and adaptive driving strategies, respectively [1]. After obtaining the trained polices, we first tested the two scenarios of ego vehicles being of level-k and opponents' vehicles being level-(k-1) each for 1000 episodes, when k =1, 2. They all have a success rate of over 96%, which meets expectations of level-k theory. Then, the trained level-1 policy and level-2 policy were tested on level-k versus level-k scenarios for 1000 episodes each, k =1, 2. These scenarios have a much lower success rate, which is reasonable because all the cars have wrong assumptions about the driving policy of others. And the level-1 policies often result in deadlock due to conservative driving behaviours, and level-2 policies often result in collisions due to aggressive driving behaviours (see Table 3.6).

We also compared the performance of our algorithm with the autonomous vehicle controller which is based on the driver interaction models and online model estimation proposed in [17], and their results are shown in the third column of Table 3.5 named AV controller. We named our algorithm GTDRL since it is based on GT and DRL. The com-

[1] Video for both training and testing is available at `https://youtu.be/mPtoojXh2-s`

parison shows that our end-to-end scheme integrating perception, decision, and control has better performance in terms of the success rate than the results shown in [17], which focuses on a simpler two-car interaction without considering the noise existing observation information.

Table 3.5: Comparison with existing work

| **No.** | **Interaction type** | **GTDRL** | **AV controller** [17] |
|---|---|---|---|
| 1 | L1 vs. L0 | 96.2% | **99%** |
| 2 | L2 vs. L1 | **99.6%** | 95% |
| 3 | L1 vs. L1 | **94.9%** | 84% |
| 4 | L2 vs. L2 | **85.9%** | 57% |
| 5 | L2 vs. L0 | **83.1%** | 41% |
| 6 | Adapt. vs. Mixed Env. | **96.2%** | 94-95% |

Finally, we tested the level-1, level-2, and adaptive policy 1000 times in the 81 scenarios following the distribution mentioned in Section 3.1. As shown in the Table 3.6, the adaptive policy has the highest success rate in all three policies, which measures the ego vehicle's ability to pass through the intersection without collision and deadlock. The deadlock rate is highest for level-1 policy because it's conservative driving behaviour that tends to react by decelerating to a halt. Level-2 has the highest collision rate because it models an aggressive driving behaviour that tends to collide with other vehicles. The adaptive policy successfully combines the advantage of both level-1 and level-2 policies to reduce deadlock and collision when interacting in the mixed scenario.

Table 3.6: Comparison of policies in mixed environment

| **Interaction type** | **Deadlock (%)** | **Collision (%)** | **Success (%)** |
|---|---|---|---|
| L1 vs. Mixed | 4.3% | 1.7% | 94.0% |
| L2 vs. Mixed | 2.3% | 9.3% | 88.4% |
| Adapt. vs. Mixed | **3.5%** | **0.3%** | **96.2%** |

### 3.3.3 Hardware Implementation

To show the performance of the trained model, owing to space constraints, four scenarios with four cars are selected to show the interactions among level-k vehicles at unsignalized four-way intersection (see Figure 2.3a left). Four vehicles are controlled by different policies. It can be observed from Figure 3.4 that when level-1 (L1) and level-2 (L2) vehicles interact with each other, the conflicts between them can be resolved. This is expected since level-1 vehicles, representing cautious drivers, will yield the right of way and level-2 vehicles, representing aggressive drivers, will proceed ahead.

Figure 3.4 (a)-(d) show five subsequent steps in a hardware testing where each vehicle can be controlled by level-1 or leve-2 policies that are pretrained in the simulator. Figure 3.3 shows the corresponding time histories of the four vehicles' motion state. All paths are divided into 200 waypoints. For each point the car reaches, the number of passed waypoints increases by one, e.g., the number of passed waypoints is 1 when the car is in its initial position and it's 200 when the car reaches its destination.

All vehicles are located outside the intersection at $t = 0$ s. (a) column shows the interactions of car 1 controlled by the level-1 policy (conservative) with three cars controlled by the level-2 policy (aggressive). Because car 2, 3 and 4 all use the level-2 policy, they usually choose to pass the intersection as quickly as possible. By observing the motion state of each vehicle in Figure 3.3, we can find that car 4 pass the intersection first and does not take any deceleration action. Although car 2 and car 3 also adopt level-2 policy, since car 4 enter the collision area first, car 2 and car 3 chose to adopt deceleration actions of different degrees at around $t = 4$ s based on their observation. Car 3 is in front of car 1 at around $t = 6$ s, therefore, car 1 takes deceleration action at around $t = 5$ s, and car 2 chooses to wait for car 1 to pass the collision area again. Finally, all four cars safely pass through the intersection in turn. Similarly, (b) column shows car 1 and car 4 controlled

by level-1 policy interact with car 2 and car 3 controlled by level-2 policy. Car 2 and car 3 enter the collision area first, and car 1 and car 4 take deceleration action to wait them to pass at around $t = 3$ s. (c) column shows car 2 controlled by level-2 interacts with others controlled by level-1 with car 1 turning left. (d) column shows car 1 turning right controlled by level-2 interacts with others controlled by level-1. The last two situations are similar: the car adopting level-2 strategy will pass through the intersection first, and the other vehicles with level-1 policy will pass through the intersection subsequently.

The examples above show that trained models obtained from simulator can deal with complex interaction scenarios without knowing the policies of others, which verifies the feasibility of the proposed method for modeling different driving behaviours proposed in this chapter.

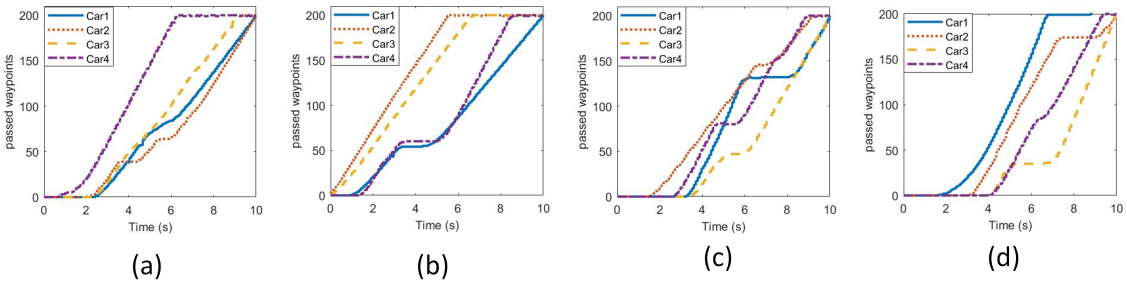



Figure 3.3: Travel efficiency under: (a) policy [l1, l2, l2, l2]; (b) policy [l1, l2, l2, l1]; policy [l1, l2, l1, l1]; (d) policy [l2, l1, l1, l1]

## 3.4 Summary and Discussion

An adaptive game-theoretic decision-making strategy with DRL has been proposed for the ADVs sharing the road with other drivers in a multi-agent traffic scenario. The interactions between vehicles are modeled using a level-k game-theoretic framework. The ego estimates the driver model of opponents at each time step based on real sensor data and is shown to

use it to adapt its behaviour in both simulation and hardware implementation.

Both simulation results and hardware tests were reported and showed that the vehicle interaction model exhibited reasonable behaviour expected in traffic. The performance of the model was then evaluated based on several ways, including the success rate, collision, deadlock, and snapshot of hardware testing. It was shown that the adaptive model had reasonably high rates of success in resolving traffic conflicts matching the expected behaviour of each reasoning levels.

The framework proposed in this chapter for modeling multi-vehicle interactions can be used as simulation tool for calibration, validation and verification of autonomous driving systems. In addition, it may also be used in high-level decision-making algorithms of ADVs, and to support intersection automation/autonomous intersection management. Moreover, vehicle interactions in some other traffic scenarios, such as highway merging and driving in parking lots, can be modeled based on the proposed framework with modified road layouts and geometries.

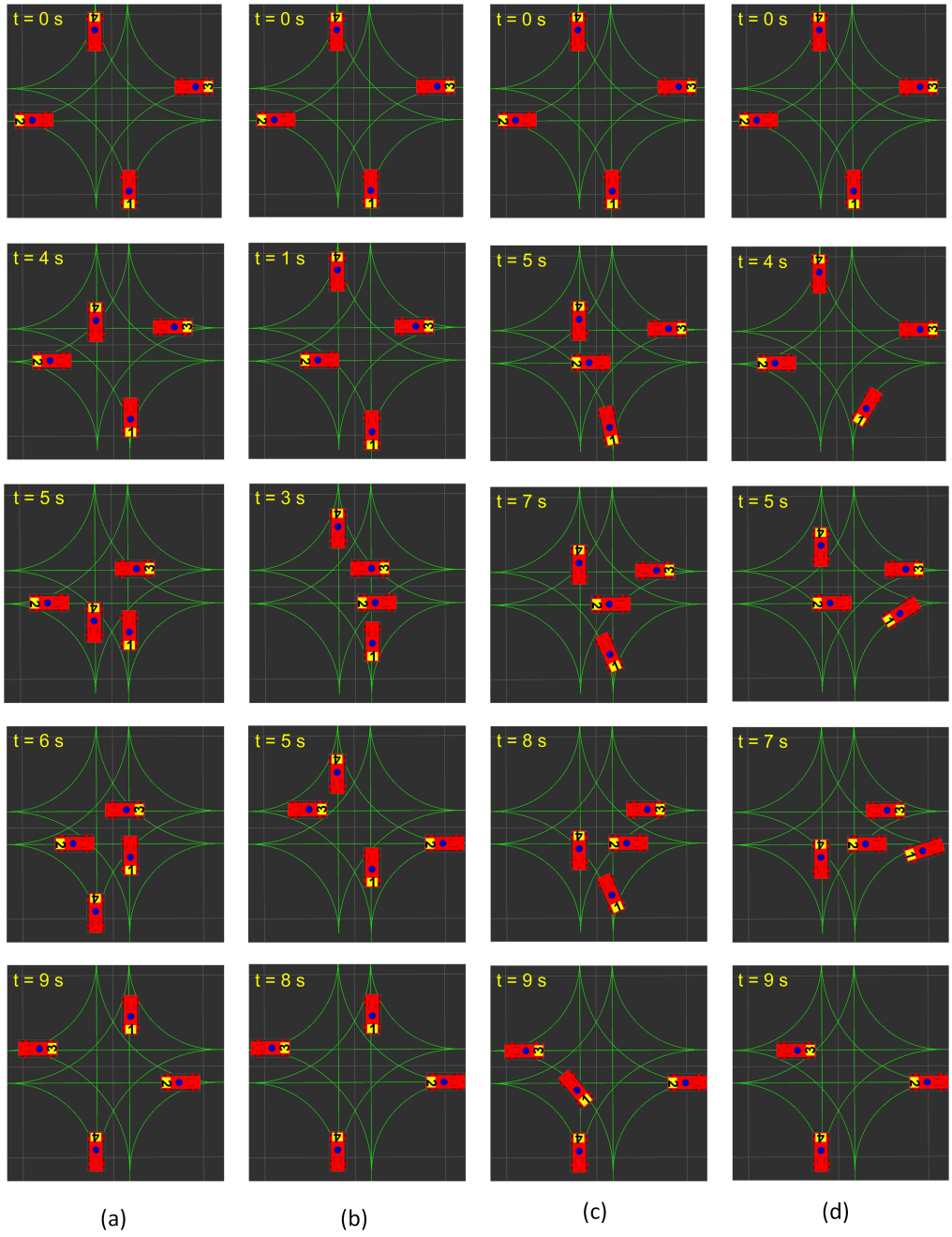



Figure 3.4: 10 sec motion state of car 1 to car 4 shown in the column under (a) policy [l1, l2, l2, l2] with car 1 going straight; (b) policy [l1, l2, l2, l1] with car 1 going straight; (c) policy [l1, l2, l1, l1] with car 1 turning left; (d) policy [l2, l1, l1, l1] with car 1 turning right. All surrounding vehicles are going straight. The numbers represent the car IDs

# 4 Enhancing Reinforcement Learning via Human Demonstration

This chapter will introduce a comprehensive solution for achieving human-like AD through DRL. The challenges of low sample efficiency, partial observability, and sim-to-real transfer in DRL-driven AD will be discussed. A novel algorithm called Deep Recurrent Q-learning from demonstration algorithm (DRQfD) is proposed for lane-changing decision-making, mitigating low sample efficiency in DRL and improving generalization in IL. Additionally, a twin high-fidelity simulator based on ROS-Gazebo is developed to bridge the sim-to-real gap, incorporating LiDAR sensing, model training, and evaluations. This chapter will also cover some other challenging problems in DRL, including generalization ability in real world scenarios, and multi-objective optimization in DRL training.

## 4.1 Problem Formulation

In lane-changing scenarios, vehicles must be able to adjust their actions to fit into the dynamic traffic environment safely and efficiently. Given the unknown driving behavior and intentions of the surrounding vehicles, the decision-making problem of AV in lane-changing scenarios can be modeled as a POMDP [78]. To train driving policies, this chapter proposes a hierarchical decision-making architecture, as shown in Figure 4.1, which is

mainly divided into two levels. The first is the high-level decision-making module achieved by *Global DQN Network*, which is about transitioning between discrete actions including car-following, changing the lane to the left, and changing the lane to the right subject to the following conditions:

- Safety - releasing restrictions on lane-changing action under the condition that there will be no collision with surrounding vehicles.
- Travel efficiency - navigating the ego vehicle to the target lane where the car can drive faster.

In other words, this module should be able to give AV a strategic task to perform at the current stage after observing the environment.

Next, once the car-following command is generated from the high-level decision-making block, *Local DQN Network* will serve as a tactical module to maintain a safe distance and speed with the vehicle in front of AV while considering travel speed, comfort, and reducing energy consumption. Trajectory generation is a non-learning-based tactical module. It will generate a smooth and comfortable trajectory using a fifth-order polynomial according to its current lane and speed, as well as the target lane and target speed when getting a switching lane command from the high-level module. However, the outputs from the *Local DQN Network* and trajectory generation block are still at the command level, such as the desired acceleration, velocity, position, and steering angle. Ideal driving behavior requires precise control performance, which can be guaranteed by implementing a low-level control module consisting of throttle control and steering control. In the following sections, we will introduce how to implement the above modules in detail.

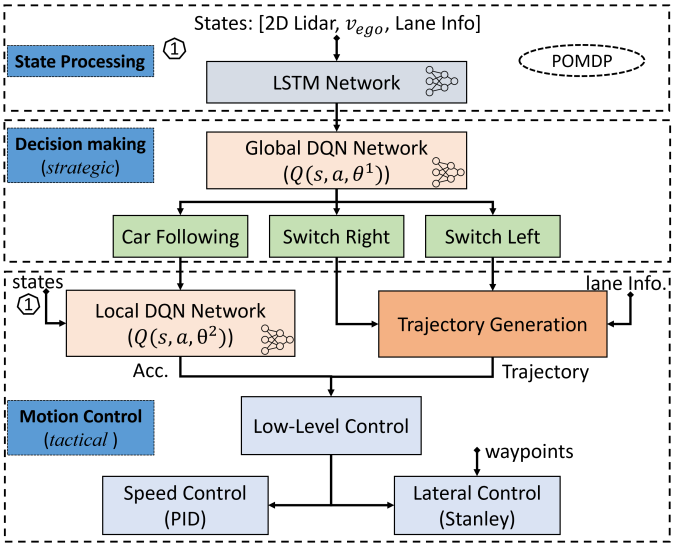



Figure 4.1: Hierarchical decision-making framework.

### 4.1.1 Observation Information

When driving a car, we make decisions primarily based on the information captured by our eyes. However, the driving policies of the surrounding vehicles are unknown to us, and we have to observe them for a period of time to approximately estimate their future motions. Therefore, in our study, we prohibited the use of accurate speed and position information from other vehicles. All observations were obtained from onboard 2D LiDAR plus the ego vehicle speed. It is worth noting that the method proposed in this chapter can also be extended to 3D LiDAR, but requires more computational power.

The LSTM network is used to implicitly learn and predict the driving behavior of surrounding vehicles by feeding the past multiple frames of point clouds concatenated with ego speed and lane information to the DRL. It can help reduce the ambiguity of LiDAR-based end-to-end AD to stable training processing of DRL. The scanning frequency is set to 12 Hz with 1160 points/scan. In Figure 2.3b, the left image offers BEV of a simulated environment, showcasing LiDAR scans of nearby vehicles as seen from the ego vehicle

("1"). Other numerical labels identify surrounding vehicles. On the right, the hardware testing environment is depicted, featuring the ego vehicle positioned in the middle lane and represented by a blue triangular frame. The corresponding point clouds generated by the ego LiDAR hardware are presented in this figure. Surrounding vehicles are differentiated by dashed circles in various colors.

### 4.1.2 Reward Function and Action Space

To avoid manually designing a large number of predefined driving behaviors like rule-based methods, we adopt a coarse-grained reward function to encourage RL to learn near-optimal driving behaviors through exploration. A punishment mechanism is added to prevent the car from learning unsafe driving behaviors based on human common knowledge.

#### 4.1.2.1 Car-Following Policy

According to the US101 data [70], the intervehicle distance is mainly maintained between 11 m and 25 m above 50% of the time. The distribution of intervehicle distance can be found in section 4.3. Because the size ratio of the car used in our study and a sedan car is 1:10. Therefore, the values for both the distance and the acceleration will be divided by 10 to define our reward function and action space.

The stage reward function for car-following is defined as

$$R = w_1\phi_1 + w_2\phi_2 + w_3\phi_3 + w_4\phi_4 + w_5\phi_5 \tag{4.1}$$

where $\phi_i$ represent indicator variables and $w_i$ are the weight variables where $i \in [1, 5]$. The values of weights corresponding to each factor are shown in Table 4.1. If a collision occurs, indicator variable $\phi_1$ equals to 1, otherwise 0. According to the distance ($D$) between the ego vehicle and the car in front of ego car (to simplify the description, we denote this vehicle as the leader vehicle), three zones are defined: unsafe zone ($D \in [0.5, 1.1]$ m),

interaction zone ($D \in [1.1, 2.5]$ m), and safe zone ($D \in [2.5, +\infty]$ m). When $\phi_2$ =1, it indicates that the leader vehicle is in its unsafe zone, and is 0 otherwise. When $\phi_3$=1, which indicates that the leader car is in its interaction area. Ego car is encouraged to maintain a relative velocity of 0 m/s to the leader vehicle. When $\phi_4 = 1$, the leader car is outside the interaction area, and we encourage the ego vehicle to keep the speed above the average speed. The fifth term, $\phi_5$ is introduced to consider energy consumption (or unnecessary driving actions) and equals to 2 if the action is a hard acceleration or a hard deceleration, 1 if the action is acceleration or deceleration, and 0 if the action is maintain. The action space of the car-following policy consists of five actions: maintaining the current speed (0 m/s$^2$); hard acceleration (1.2 m/s$^2$); hard deceleration ($-1.2$ m/s$^2$); acceleration (0.6 m/s$^2$); deceleration ($-0.6$ m/s$^2$). This discrete representation of the action space is based on the distribution of vehicle accelerations obtained by processing the real traffic data [78,79], which is recognized as a reasonable approximation to the set of human drivers' actions in highway traffic. It should be pointed out that we set speed saturation during training and evaluation, thus, the AV is only allowed to drive within the maximum speed. When the AV reaches the maximum speed, it cannot further obtain additional rewards through increasing speed. Therefore, the AV will finally learn to maintain its highest speed (by choosing 0 m/s$^2$) when there is no leader vehicle in the current lane by the trade-off between travel efficiency and energy consumption factors.

#### 4.1.2.2 Lane-Changing Policy

For the high-level module, we expect the ego car to learn a more efficient driving behavior similar to that of an experienced driver. That is, the ego car will take any opportunity to navigate to the target lane where it can gain a higher speed under the premise that it will be collision-free. Therefore, we list some unsafe driving behaviors and encouraged the

Table 4.1: Setting of reward function

| **Conditions** | **Values** |
|---|---|
| Collision violation ($w_1$) | -1000 |
| Unsafe zone violation ($w_2$) | $-(\frac{1.1}{D})^3$ |
| Relative speed ($w_3$) | $-2 \cdot \lvert v_{ego} - v_{front} \rvert$ |
| Travel efficiency ($w_4$) | $v_{ego} - (v_{max} + v_{min})/2$ |
| Energy consumption ($w_5$) | -0.6 |

Note: $v_{max} = 1.5$ m/s, and $v_{min} = 0.0$ m/s.

car to change lanes without triggering penalties. To obtain a positive reward during the training process, the ego car needs to avoid the following unsafe lane-changing behaviors:

(i) The penalty is -1 for lane-changing, if there's no leader vehicle in the current lane within the LiDAR detection range.

(ii) The penalty is -1.5, if the car in front in the current lane is farther than the car ahead in the target lane.

(iii) The penalty is -5, if a collision occurs during lane-changing.

The action space of the high-level module consists of {switching left, switching right, car-following}. To simplify the training scenario, we assume that the speed of the ego vehicle before and after the lane-changing remains the same. The logic behind this assumption is that the ego vehicle first generates a trajectory to a target lane where it can gain higher speed, and then accomplishes the acceleration goal in the target lane by using a trained car-following policy. Trajectories are generated based on the fifth-order polynomial under the following boundary conditions: (1) the target speed of the trajectory is the 'same' as the current speed, and (2) the target position of the trajectory is the location in the target lane right 'behind' its leading vehicle in the original lane to achieve a safe lane-changing.

It should be noted that to obtain more flexible trajectories, we can modify the boundary

conditions of the fifth-order polynomial by expanding the action space of the high-level module such as {switching left faster, switching right faster, switching left, switching right, Car-following}, where the constraints of polynomial trajectory for switching lanes faster are (1) the target speed is 'higher' than the current speed, and (2) the target position of the trajectory is the location in the target lane 'parallel' to the leading vehicle in the original lane.

## 4.2 Deep Recurrent Q-learning from Demonstrations

Given the high-fidelity simulator introduced in the Section 2.2.2, we can manually collect human demonstrations and automatically score the expert's action at a certain state according to the pre-designed reward function. To enhance the training efficiency of DRL, we propose a DRQfD framework, which involves using a small set of demonstrations to pre-train DRL network, followed by IL to guide its early exploration. The overall structure of the algorithm is shown in Figure 4.2. Given the LiDAR-based perception scheme and the POMDP, RD3RQN PER [57,80] becomes an ideal choice for our DRL algorithm. The parameter setting of algorithm is provided in Table 4.2.

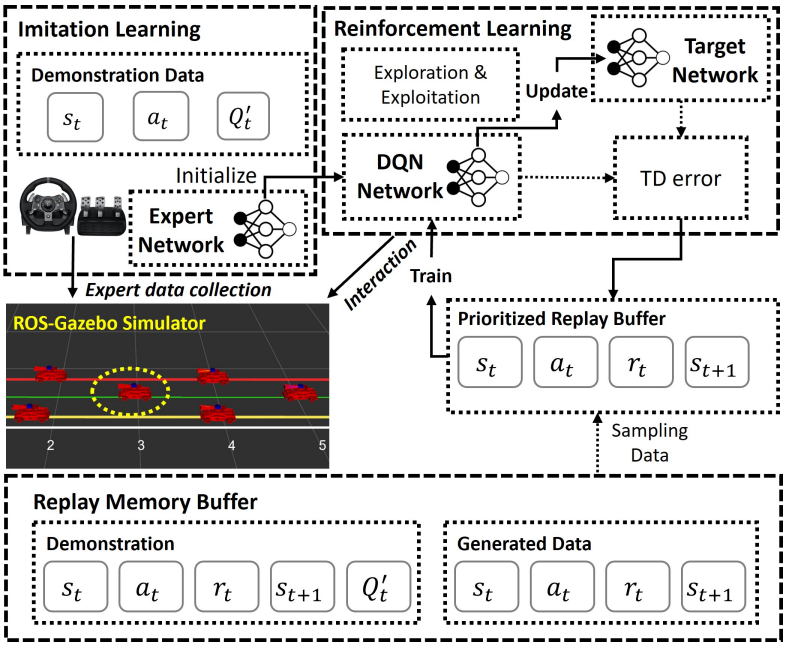



Figure 4.2: Algorithm structure

### 4.2.1 Pretrained DRL Network

The pre-training phase aims to teach the vehicle to imitate the demonstrator while also satisfying the Bellman equation for its value function, which can then be updated through TD error during interaction with the environment. To get pretrained DRL Network, the learning vehicle updates the network by sampling mini-batches from demonstration data and applying four losses. These include 1-step and n-step double Q-learning losses to ensure the network satisfies the Bellman equation, large margin classification loss to enforce the value of the demonstrator's action, and L2 regularization loss to prevent over-fitting on the small demonstration dataset. The details of the loss function implementation can be found in [49].

### 4.2.2 IL-Based Guidance Policy

After completing the pre-training phase, a hybrid policy combining $\epsilon$-greedy exploration and Guidance policy, is used to increase the probability of the vehicle taking the correct actions during early exploration through the IL-Based Guidance policy. During the $\epsilon$-greedy exploration process, learning vehicle has a certain probability of taking actions generated by IL, which needs to be adjusted to balance exploration and exploitation to mitigate the bias introduced by using an IL policy on the replay buffer. It should be noted that both DRL and IL use LSTM network to process a fixed number of past sequential observations as input.

### 4.2.3 Experience Replay

During the DRL training, the learning vehicle interacts with surrounding vehicles in the simulator, generating its own data and adding it to the replay buffer $\mathcal{D}^{\text{replay}}$. The expert demonstration data will be permanently stored in the experience replay buffer $\mathcal{D}^{\text{expert}}$ and assigned a constant value in PER to ensure that the data is sampled during the training process.

### 4.2.4 IL Training

In order to avoid increasing human workload, both pretrained DRL network and IL Guidance policy mentioned earlier are trained on a small amount of demonstration data collected by a human player using our simulator. To ensure a fair comparison of the performance between DRL policy and IL Guidance policy in Section 4.3, we increase the amount of demonstration data to the same number as DRL training episodes. For both the IL Guidance policy and the IL Baseline policy, we performed supervised classification of the demonstrator's actions using a cross-entropy loss, with the same network architecture used by

DRL. Additionally, we still used L2 regularization loss to prevent overfitting of the model.

Table 4.2: Parameter settings

| **Parameters** | **Values** |
|---|---|
| Discount factor ($\gamma$) | 0.95 |
| Learning rate | 0.001 |
| Starting (ending) value of $\epsilon$ greedy policy | 1 (0.01) |
| Number of car-following actions | 5 |
| Number of high-level module actions | 3 |
| Size of replay memory ($\mathcal{D}^{\text{replay}}$) | 10000 |
| Size of expert memory ($\mathcal{D}^{\text{expert}}$) | 2000 |
| Number of steps to update the target network | 100 |
| Mini-batch size | 32 |
| $e$ | 0.00001 |
| Priority ($\alpha$) | 0.6 |
| Adjusting the deviation ($\beta$) | 0.4 |
| N-step returns (steps) | 10 |

### 4.2.5 Simulation to Real World

For DRL training, a high-fidelity simulator is developed based on ROS-Gazebo. To reduce the sim-to-real gap, Domain Randomization is adopted to improve the robustness and generalization ability of DRL policies by randomizing the dynamic properties of the Source Domain. In this work, key factors for closing the sim-to-real gap include perception, environment complexity, vehicle dynamics, and interaction features. Regarding the perception, we adopt a LiDAR-based end-to-end scheme, since the LiDAR is insensitive to different lighting conditions. Thanks to the Unified Robotics Description Format (URDF) setting supported by ROS, we can keep the dynamics and sensor data of the virtual car as close as possible to the physical car. We set our simulator parameters based on the data obtained from system identification of physical cars, such as the physical properties

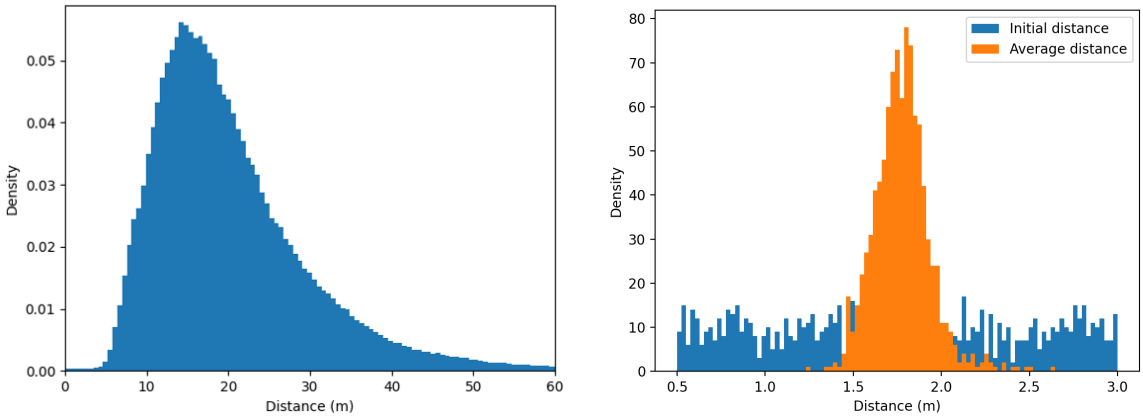
(e.g. mass, inertia, geometry) and LiDAR parameters (e.g. sampling frequency, noise, detection range). Real-world scenarios are more complex than the simulated environment, which could also affect the performance of DRL models. As we conduct hardware tests indoors, in order to avoid the influence of irrelevant indoor obstacles on the model, we limit the maximum detection distance of LiDAR to 2.7 m and apply a mask function to set the point cloud data beyond the detection range to 0. In addition, during the training and hardware evaluation phases, point cloud data and ego speed are normalized based on their maximum value settings before feeding into the DRL algorithm. Finally, to make the trained model generalize well in the test scenarios, we encourage the vehicle to experience as many interaction scenarios as possible during training by randomizing dynamic properties including the position and velocity of surrounding vehicles. Training scenarios and reward functions can be effectively designed by referring to the distribution of human driving characteristics (e.g., inter-vehicle distance and driving action) in real traffic data, as described in the Section 4.3 and the Section 2.2.4.1.

## 4.3 Experimental Validation

### 4.3.1 Implementation in Car-Following Scenarios

The distribution of the distance to the leader vehicle in car-following scenarios maintained by human drivers is shown in Figure 4.3a. Since the size ratio of our scaled model to a real vehicle is 1:10, the distance value considered in both training and testing environment is 10 times smaller than that in the real world. The simulated environment consists of six scaled cars and three 100-m road segments. One car is the ego vehicle, and the others are surrounding vehicles that adopt rule-based driving policy. To train and evaluate the Adaptive Cruise Control (ACC) using the scaled model, we set the initial intervehicle

distance to obey the uniform distribution between 0.5 m and 3.0 m with maximum LiDAR detection of 2.7 m. The maximum speed limit of surrounding vehicles in each episode obeys uniform distribution between 0.5 m/s and 1.5 m/s. The maximum speed limit of the ego vehicle is 1.5 m/s. The trained model was tested on 1000 randomly generated scenarios. Testing results indicate that the ego vehicle can always maintain a safer driving distance with the leader car when performing the car-following task. The distance between the ego vehicle and the leader vehicle follows a normal distribution with $\mu = 1.85$ and $\sigma^2 = 0.8$ m, as shown in Figure 4.3b. In Figure 4.3b, the blue bars represent the distribution of initial distance between the ego vehicle and the leader car, and the orange bars represent the distribution of average intervehicle distance in each episode, which is similar to the distribution of intervehicle distance in the real traffic data, especially in the detection range, as shown in the Figure 4.3a.




(a) US-101 data set

(b) Testing results

Figure 4.3: Distribution of distance to the car in front.

### 4.3.2 Pretraining and Baseline Comparison

To benchmark the performance of the DRL-based high-level policy, human player policy and IL-based policies are considered in this work as baselines. All policies use our trained car-following policy to achieve the ACC on the current lane. We randomly generated 100 evaluation scenarios, and the tested vehicle randomly selected a lane to start the test (0: left lane; 1: middle lane; and 2: right lane). The maximum speed of the ego vehicle is set to 1.5 m/s, while the speed limit for other vehicles in each episode is uniformly distributed between 0.5 to 0.8 m/s. In addition, the longitudinal distance between surrounding vehicles and the ego vehicle follows a uniform distribution between 1.0 to 3.0 m. All vehicles are located in front of the tested vehicle and each lane is guaranteed to have at least one vehicle.

We had a human player drive the tested vehicle in simulated environments about 100 episodes using joystick. Each episode was played either until the driving task terminated or exceeded 50 seconds. During the human player driving, we collected the vehicle's laser scan concatenated with ego speed and lane number, actions, rewards, and terminations. This data serves two purposes. First, pretrained DRL network and the IL Guidance policy are trained on this small dataset. Second, IL Guidance policy will also be regarded as one of the baselines for later comparison with DRQfD policy. We noticed that the data related to the car-following scenarios is about 20 times that of the data for changing the lane to left or right. Therefore, the pre-trained model converges more easily to a local optimal policy (i.e., following strategy). To ensure a fair comparison between the RL policy and the IL Baseline policy, we increased the demonstration data to 700 episodes (equivalent to 28,000 steps, close to the total training steps of the RL algorithm). To avoid overfitting the model, we balanced the number of three types of scenarios in the training dataset.

### 4.3.3 Naturalistic Human Driving Data Set

To further evaluate the generalization ability of the DRL policy, we test the DRL-based high-level model on real traffic data. We randomly selected 100 vehicles with lane-changing behaviors from real traffic data containing 3000 vehicles for comparison. The data was recorded by eight cameras for 10 minutes of all vehicle information in the US101 road segment with a length of 640 m. To train the lane-changing policy, we reconstructed the road section in the simulator as shown in Figure 2.6.

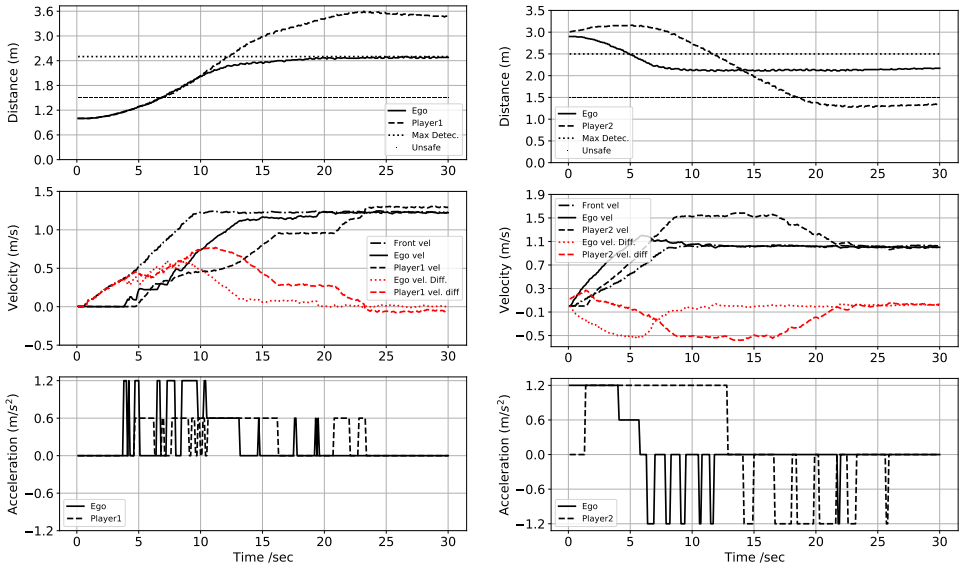
Since our scaled car is 10 times smaller than real vehicles, we need to scale the traffic data proportionally in order to replay real traffic data for testing in the simulation environment. Considering that the speed limit of our scaled car during training was set to 1.5 m/s (equivalent to 15 m/s in real scenario), therefore, we remove all cases with vehicle speeds above 15 m/s from the real traffic data. During the testing, if the trained policy selects the same lane-changing action as the human driver in a given scenario, we consider that the driving behavior of the current case is successfully modeled. Due to the presence of noise in the original data, the trajectories of surrounding vehicles are fitted by a fifth-order polynomial, as introduced in Section 2.2.2. See Section 2.2.4.1 for details about preprocessing real traffic data and replaying it in the simulator to test the trained policy.

## 4.4 Implementation Results

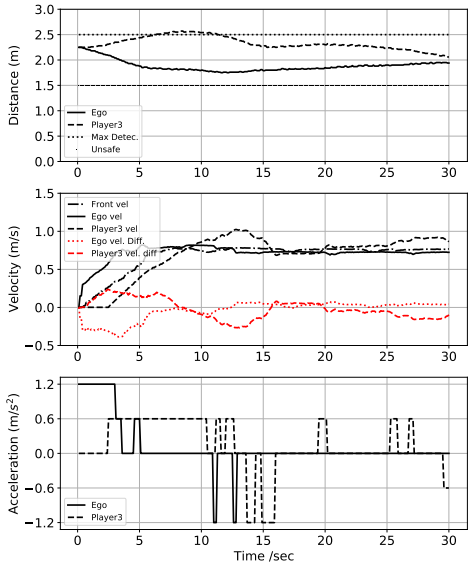
### 4.4.1 Robustness Analysis of Car-Following Policy

To verify the driving performance of the trained ACC policy, we compared it with human drivers. In the comparison experiments, we selected three male drivers (aged 25-35) from different countries, including China, Canada, and India, representing different driving styles and preferences. In each group of comparisons, human players control the speed of the car

using a joystick, and the trained policy was also tested in the same traffic scenarios. All human players and the trained policy use the same observation information to ensure the fairness of the comparison. Human drivers watch the speed information of the controlled car as well as the point cloud data detected by the LiDAR to obtain the position information of the vehicle in front. For example, if the intervehicle distance is out of the maximum detection range, players cannot observe the point cloud information of the front vehicle in the simulator. In each group of tests, we considered 30 cases where the leading vehicle, in the beginning, is located in the unsafe zone (gap<1.5 m), the interaction zone (1.5 m< gap <2.7 m), and the safe zone (gap>2.7 m) of the controlled vehicle respectively. The maximum speed limit of the leading vehicle is set to change within the range of [0.5, 1.5] m/s. In each episode, the simulator will randomly select a pair of parameters (initial distance, max speed) from list D = [0.8, 1.0, 1.25, 1.4, 1.5, 2, 2.25, 2.7, 2.9, 3.0] m and list $v$= [1.0, 0.75, 1.2] m/s. The evaluation includes safety, comfort, and energy consumption. The last two factors are reflected by the change of the vehicle's acceleration. To save fuel consumption, drivers usually try to use minimum effort to achieve the desired behaviors. In other words, the less often the driver uses deceleration actions, the better. The order of each action with respect to comfort is as follows: maintain the current speed>acceleration (or deceleration)>hard acceleration (or hard deceleration). In order to improve safety, the car-following policy needs to be able to maintain a stable distance from the leading vehicle and maintain a relative speed of around 0 with it.




(a) (b)




(c)

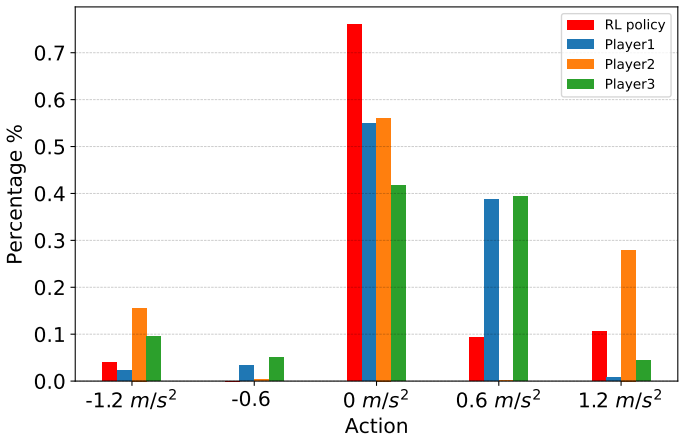
Figure 4.4: Comparison of learned policy with human players on features of intervehicle distance, relative speed and actions: (a) learned policy vs. player 1 (unsafe zone) with 1.25 m/s speed limit of the leader car; (b) learned policy vs. player 2 (safe zone) with 1.0 m/s speed limit of the leader car; (c) learned policy vs. player 3 (interaction zone) with 0.75 m/s speed limit of the leader.

Table 4.3: Comparison with human players

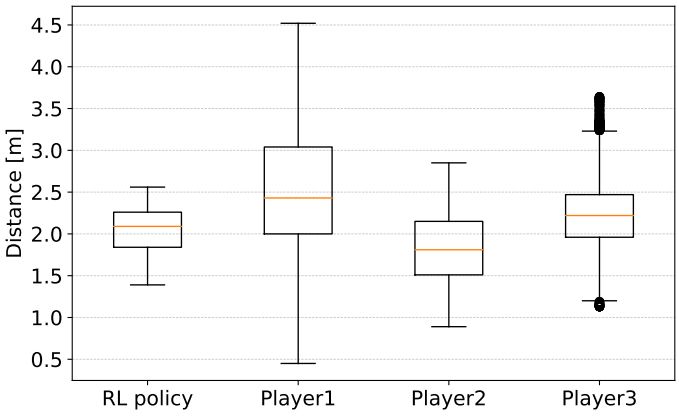
| **Policy** | **Rewards** | **Unsafe rate** | $\Delta d$ **(m)** | $\Delta v$ **(m/s)** |
|---|---|---|---|---|
| RL policy | **-235.54** | **0.0%** | 2.06 | **0.105** |
| Player 1 | -427.08 | 6.7% | 2.51 | 0.248 |
| Player 2 | -488.31 | 63.3% | 1.82 | 0.238 |
| Player 3 | -408.13 | 3.3% | 2.23 | 0.257 |

During the test, we use the reward function defined in Section 4.1.2.1 to score the driving behaviors. The action frequency is 16 Hz, and the maximum number of steps is 300. We counted the average score for each group of tests, the probability of violating the unsafe zone, the distance between the ego car and the leader, and the speed difference separately. Table 4.3 shows that the trained car-following policy outperforms human players in the above aspects. To understand why the trained policy scored high, we analyzed the recorded data further. We selected three representative sets of comparative data respectively, as shown in Figure 4.4. In Figure 4.4a, the initial distance is 1.0 m, and the maximum speed of the leader is 1.25 m/s. Both the DRL policy and player 1 choose to maintain the current speed of 0 to avoid collision with the leader. Compared to the performance of player 1, the trained policy can adjust actions quicker to maintain a stable distance and to get a smaller speed difference with the leader. From the speed curve, we can find that player 1 focuses more on comfort, and the DRL policy is more on travel efficiency. In Figure 4.4b, the leader is out of the LiDAR detection range at the beginning, and both the DRL policy and player 2 choose to accelerate. As the distance decreases, the leader begins to enter the detection range of the LiDAR. However, player 2 cannot take actions as quickly and effectively as the DRL policy to maintain a safe interactive distance with the leader. Finally, the car controlled by player 2 enters the unsafe area, thereby increasing the risk of collision with the leader. According to Figure 4.4c, we can see that both the DRL

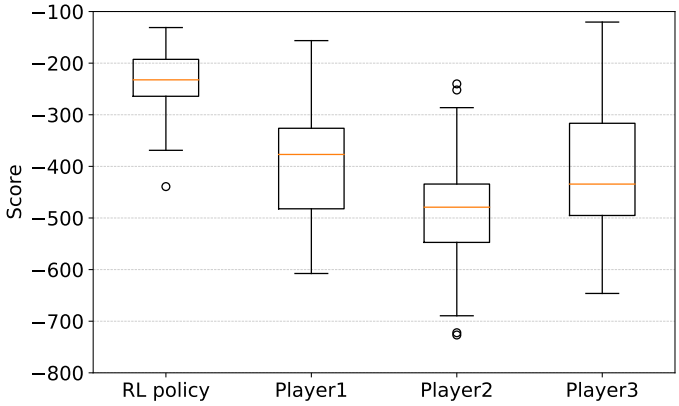
policy and player 3 can effectively take actions to maintain a safe interaction distance with the leader, and take as few "deceleration" actions as possible to save energy consumption. To compare the robustness of different driving behaviors during testing, we performed a statistical analysis of the recorded data. From Figure 4.5a, we can see that compared to the other three human players, the DRL policy uses more "maintain" actions and uses less "deceleration" action to reduce energy consumption. Figure 4.5b shows that the average distance between the ego vehicle and the leader is 2.06 m and the fluctuation range is 1.2 m smaller than other three groups of tests. According to Figure 4.5c, the average score of the DRL policy in the 30 groups of tests is -235.54, and the fluctuation of the scores is significantly smaller than that of human players which proves that the trained car-follow policy is more robust. Although the discrete actions adopted in this chapter is different from the continuous action space in real world, we can conclude from the above comparison that the learning-based driving policy can perform the same or even better than human drivers on some specific driving tasks. Unlike human drivers, the machine will not be fatigued due to repetitive tasks which leads to unsafe-driving in humans.




(a) Distribution of actions




(b) Distribution of distances to leader




(c) Distribution of scores

Figure 4.5: Comparison of driving preference and robustness of different policies (a) action distribution of different policies; (b) head-way space distribution of different policies; (c) score distribution of different policies.

### 4.4.2 Training Results

To maintain the conciseness of this section, we demonstrate the results of utilizing proposed DRQfD framework to enhance the training efficiency of the DRL algorithm for the high-level policy. We also compare it with vanilla D3QN PER and IL Guidance policy, as shown in Figure 4.6. The red dashed line in the figure represents the performance of the Guidance

policy trained by the IL algorithm on a small dataset, which is used to guide DRL training. Due to the small data size, Guidance policy falls into a local minimum of choosing the car-following policy most of the time. On average, IL Guidance policy obtains an accumulated reward of around 2.75/episode, demonstrating a lower travel efficiency. The blue curve represents the training process using the DRQfD framework. We can see that the proposed framework can effectively help the DRL algorithm to converge quickly to the optimal policy level, saving about 30% of training episodes. The orange line represents the training curve of the vanilla D3RQN PER algorithm. We cut off the training curve at the termination of the DRQfD training. However, as the number of training episodes increases, we found that the vanilla D3RQN PER algorithm can also converge to the level of the DRQfD algorithm. Therefore, we can conclude that the pre-trained DRL network and the IL Guidance policy trained on a small amount of expert demonstration can effectively help to improve the learning efficiency of DRL. In the next section, we will compare the DRL policy with the IL and human player policies in randomly generated scenarios.

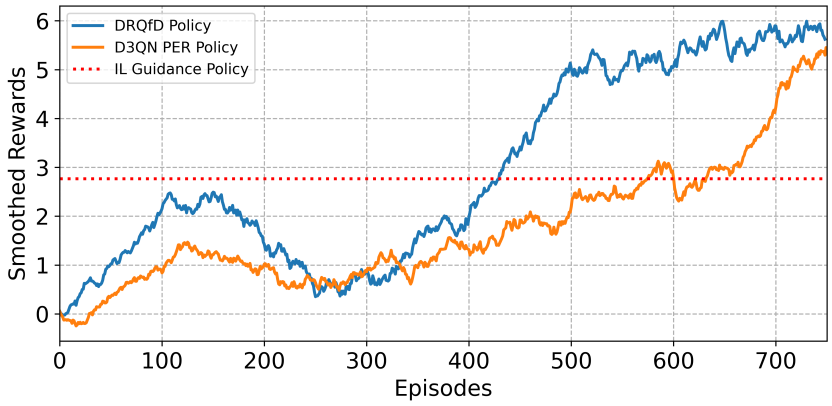



Figure 4.6: Training processes of different learning methods in the lane-changing scenarios.

Table 4.4: Baseline comparison results

| **Methods** | **Collision** | $\Delta d$ | $v_{avg}$ **(m/s)** | **Invalid** $a$ |
|---|---|---|---|---|
| Human Player | **5%** | 42.5 | 0.85 | **0.03%** |
| DRQfD | 7% | **44.5** | **0.89** | 0.27% |
| IL Baseline | 20% | 38.5 | 0.77 | 1.06% |
| IL Guidance | 13% | 36.0 | 0.72 | 0.88% |

### 4.4.3 Comparison with Imitation Learning

Since the Vanilla D3QN PER and our DRQfD algorithms eventually converge to the same level, for the sake of simplicity, we only select one model to represent the DRL policy. We compare the performance of DRL policy with a human player, IL Guidance policy, and IL Baseline policy in the same scenarios. We evaluate the driving safety of each policy based on the collision rate. The driving distance and average speed are used to evaluate the traffic efficiency. Invalid action rate is the proportion of the total number of actions taken by the ego vehicle during the entire testing, in which either the ego changed lanes despite no vehicles being present in front of it, or the lane-changing action caused the ego to leave the road. To provide consistent observation information for the human player and other trained policies, similar to the car-following scenario introduced earlier, we only provided the human player with the visualized point cloud of surrounding vehicles through RViz, without displaying information about vehicles out of the LiDAR detection range. The test results are shown in Table 4.4. The human player has lower collision rate and invalid action rate than all trained policies, indicating ideal driving behavior. Although the DRL policy has slightly higher values than the human driving policy in these two metrics, it is significantly better than the IL policy. This result is reasonable, as even though we provided the IL policy with the same amount of data as the DRL training, the demonstration data only includes successful driving scenarios. It may not generalize well to

new and unseen scenarios during testing. We also analyzed the reasons why human players had collisions. This was mainly due to the fact that in order to be compatible with the size of the indoor validation environment, we set the detection range of LiDAR to 2.7m. Therefore, all collision scenarios of human players occurred when there were no displayed vehicles in front of the target lane before the lane-changing. However, when the human drivers choose to change lanes to get a higher speed, vehicles suddenly appeared in front of the ego in the target lane, leading to collisions. The above reasons also apply to other learning-based policies that experienced collisions in the same scenarios. This problem can be solved by increasing the maximum detection range of the LiDAR. Regarding the evaluation of driving efficiency, we found that the DRL policy performed the best in terms of driving efficiency, with an average driving speed of 0.89 m/s, higher than the other driving policies. Additionally, this speed was higher than the maximum driving speed setting of surrounding vehicles (0.8 m/s), indicating that the DRL policy can effectively take lane-changing actions to improve driving speed. To evaluate the similarity between different learning-based policies and human driving behavior, Figure 4.7 shows the action distribution of each driving policy in the same 100 test scenarios. Overall, the DRL policy is closest to human driving behavior with a slightly higher proportion of lane-changing actions than human players, which explains why DRL achieves higher travel efficiency. However, based on Figure 4.7 and Table 4.4, we can see that the generalization ability of the IL Baseline policy is unsatisfactory, showing a higher collision rate and invalid action rate. The red bar represents the IL Guidance policy used to guide DRL early exploration. Due to the small amount of data, IL Guidance policy falls into a local minimum of choosing car-following policy most of the time. The proportion of lane-changing actions taken by IL Guidance policy is significantly lower than that of the other policies. We can conclude that our DRL-driven policy outperforms IL-based policy in terms of driving safety, travel

efficiency, and human likeness.

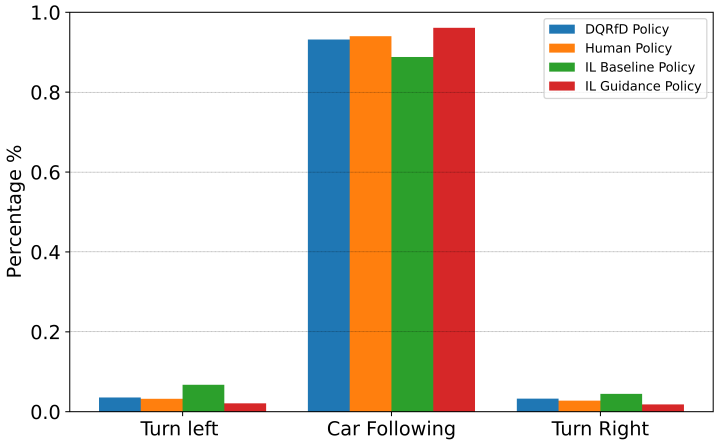



Figure 4.7: Distribution of high-level actions.

### 4.4.4 Test Results in Real Traffic Data

According to the previous section, we demonstrated that the DRL policy outperforms the IL policy. In this section, we aim to validate whether the DRL policy can choose lane-changing actions consistent with human drivers in the given scenarios. The primary factors considered by a vehicle during lane-changing decision-making involve relational attributes such as position and speed relative to other vehicles in the surrounding environment. As explained in Section 2.2.2, we scaled the boundary conditions of surrounding vehicles based on real traffic data to generate trajectories for replaying in the testing environments. After testing 100 lane-changing scenarios, the modeling success rate under the scaled speed settings is above 81%. The remaining cases, which involve collision or not selecting a lane-changing action, are considered as modeling failures. Here, we take three cases from the 100 testing scenarios as examples to demonstrate the driving performance of the trained policy. The position and speed information of surrounding vehicles as well as the testing duration for each case are shown in Table 4.8.

Taking vehicle 953 as first example, the first two values of each set of data in the second row of Table 4.8 indicate the initial and final scale positions of surrounding vehicles. From Figure 4.9a, we can see that 959 (blue), 950 (cyan), 945 (red) are the vehicles in front of the tested vehicle (953, black) at the very beginning, of which the 945 (red) is the farthest from the tested vehicle (black). The first values of the fourth and fifth set of data in the second row of the table are smaller than initial position of ego car ($p_y^0 = 11.31$ m), indicating that 966 (yellow) and 957 (green) are the vehicles behind the tested vehicle (black). The last two values of each set of data in the table are the initial and final scale speeds of surrounding vehicles. We can see that the initial speed of the vehicle 950 (cyan) ahead of tested vehicle in the current lane and the rear vehicle 957 (green) in the target lane are smaller than the initial speed of tested vehicle ($v_y^0 = 1.25$), indicating that their current speeds are lower than the ego vehicle speed while the current speed of vehicle 945 (red) in the target lane is faster than the tested vehicle. The trajectories and speeds of surrounding vehicles extracted from real traffic data for case 1 are shown in Figure 4.9a. Figure 4.9b shows the testing result in the simulated environment, and we can find that speed profiles of surrounding vehicles in the simulated environment have the same trend as the real traffic data. The ego vehicle chooses to change the lane to the right similar to the decision made by human driver. Therefore, the driving behavior in this scene is considered to be successfully modeled. Similarly, the other two examples are about real scenarios where the tested vehicles change lane to the left from the middle lane and the bottom lane respectively as show in Figure 4.9c and 4.9e. Tested vehicles all made decisions to change the lane same as human drivers as shown in Figure 4.9d and Figure 4.9f. It should be noted that the reward function designed for high-level policy focuses more on driving safety and travel efficiency under different driving conditions rather than fitting the true trajectories of lane-changing vehicles, therefore, we have excluded the speed curve of tested vehicles

Table 4.8: Boundary Conditions of Surrounding Vehicles

| $(p_y^1, p_y'^1, v_y^1, v_y'^1)$ | $(p_y^2, p_y'^2, v_y^2, v_y'^2)$ | $(p_y^3, p_y'^3, v_y^3, v_y'^3)$ | $(p_y^4, p_y'^4, v_y^4, v_y'^4)$ | $(p_y^5, p_y'^5, v_y^5, v_y'^5)$ | $\Delta t$ |
|---|---|---|---|---|---|
| **Ego IDs = 953**, $(p_y^0, v_y^0)$ = (11.31, 1.25); Surroundings = 959, 950, 945, 966, 957 | | | | | |
| (12.27, 21.45, 1.22, 0.91) | (13.62, 23.29, 1.07, 1.07) | (14.67, 24.87, 1.39, 0.90) | (10.18, 20.05, 1.24, 1.06) | (10.93, 21.14, 1.11, 1.11) | 8.6 |
| **Ego IDs = 1054**, $(p_y^0, v_y^0)$ = (17.10, 0.47); Surroundings =1053, 1049, 1061, 1058, 1056 | | | | | |
| (18.24, 22.64, 0.31, 0.92) | (18.22, 24.17, 0.69, 1.09) | (17.93, 20.35, 0.35, 0.69) | (17.02, 20.08, 0.15, 0.76) | (15.77, 20.97, 0.56, 0.91) | 6.9 |
| **Ego IDs = 2610**, $(p_y^0, v_y^0)$ = (12.90, 0.92); Surroundings = 2603, 2608, 2605, 2621, 2612 | | | | | |
| (14.66, 20.15, 1.21, 1.20) | (14.17, 19.57, 0.93, 1.06) | (14.47, 20.30, 1.07, 1.22) | (10.64, 15.11, 0.74, 0.93) | (12.56, 18.39, 1.06, 1.19) | 5.2 |

Note: boundary conditions of vehicle $i$: $(p_y^i, p_y'^i, v_y^i, v_y'^i)$ = (Init. pos., Final pos., Init. speed, Final speed); Testing duration: $\Delta t$.

Figure 4.8: Boundary conditions of surrounding vehicles.

from test results to avoid confusion. More details about tests can be found in video [1].

### 4.4.5 Hardware Implementation

To verify the performance of sim-to-real transfer, in this section, we select three scenarios with four cars to show the lane-changing behavior in real world. Both the car-following model and the lane-changing model are loaded to the cars and conducted continuous testing for 5 hours. The size of the physical car and the virtual car in the simulator are completely identical. Each car is equipped with an Nvidia Jetson-TX2 GPU and a 2D LiDAR. The decision frequency is 12 Hz and the control frequency is 100 Hz. Although there are only four physical cars available for use, we select the testing scenarios where the surrounding vehicles can directly influence decision-making made by the ego car. The surrounding vehicles are set to execute the trained car-following policy only during testing, and the ego car adopts the complete policy proposed in this chapter. To facilitate continuous testing, we designed an elliptical three-lane scene as shown in Figure 4.10 where the outer lane is lane 3 and the inner lane is lane 1. All cars move counter-clockwise. We use motion capture system to get the ground truth of positions and velocities. From Figure 4.10a, we can see that the ego car is currently in the lane 3 where the speed of car 1 (yellow) in front

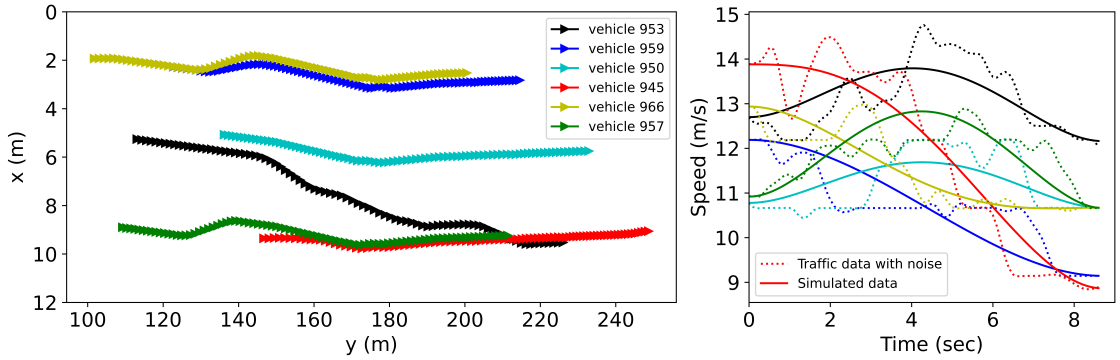
[1] Experimental Video can be viewed at `https://youtu.be/Svp2S10aSB8`

is lower than the speed of car 3 (red) in the target lane. And the gap between car 2 (blue) and car 3 (red) in the target lane is safe enough for the ego car to complete lane-changing. According to Figure 4.10b, the ego car is in the lane 2 while the speed of car 3 (red) in the current lane is the slowest. In addition, car 1 (yellow) in the left lane is faster and farther away from the ego car than car 2 (blue) in the lane 3, therefore, the ego car chooses to overtake car 3 (red) from lane 1 in this case. From Figure 4.10c, the ego car is in the lane 2, and the gap between car 3 (red) in the lane 1 and car 2 (blue) in the current lane of the ego car is smaller than that of car 1 (yellow) in the lane 3. Therefore, the ego car chooses to overtake the car 2 (blue) in front from lane 3. Refer to the video for other test scenarios. From the above examples, we can see that the policies learned from simulator can be directly used on the hardware. The performance of the ego car in the real scene is consistent with the driving behaviour in the simulated environment, which proves the feasibility of the method proposed in this chapter.

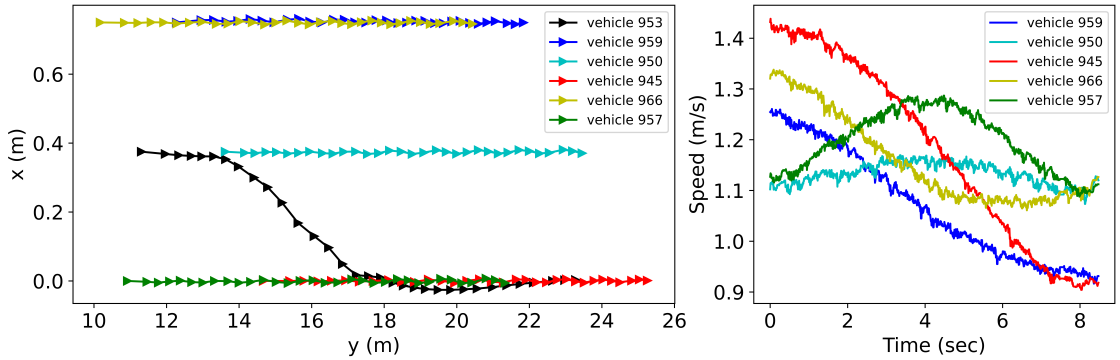
## 4.5 Summary and Discussion

In this work, we systematically studied the application of a DRL method in lane-changing scenarios from three aspects: learning efficiency, partial observability, and sim-to-real transfer. Specifically, we first propose a new DRQfD algorithm, which has three advantages: (i) improving the learning efficiency of DRL; (ii) improving the generalization ability of IL; (iii) solving the POMDP problem in decision-making of AD through the LSTM network, thereby stabilizing the training process. Secondly, a hierarchical decision-making framework training car-following policy and high-level policy separately in an end-to-end manner is proposed to address the multi-objective problem in lane-changing scenarios. Third, to validate the effectiveness of our method, we build a high-fidelity simulation platform based on ROS-Gazebo for training and evaluating of different driving policies. According to the

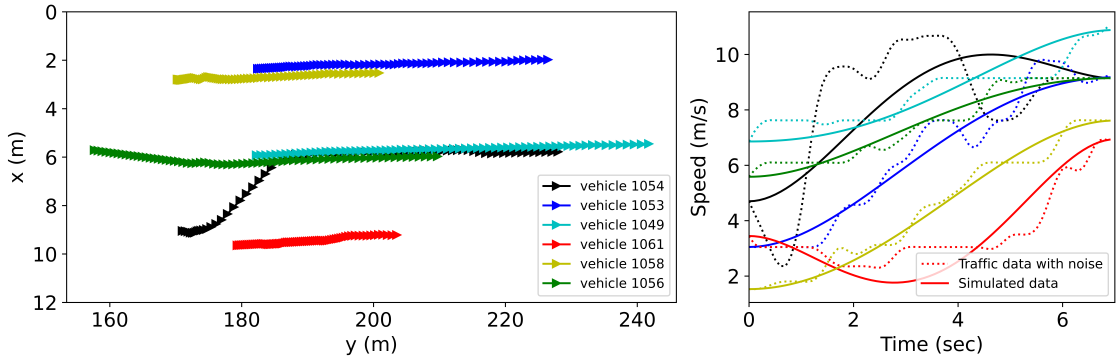
testing results in car-following scenarios, the DRL-driven policy is capable of performing as well as or even better than human drivers in safety and energy consumption. Regarding the high-level lane-changing policy, we compared our DRQfD method with a pure DRL method showing about 30% improvement in learning efficiency. Qualitative and quantitative comparisons between our model and IL baseline are also conducted. The experimental results show that our proposed method outperforms IL in terms of safety, travel efficiency, and human likeness. To further validate the generalization ability of our model, we test the model on real traffic data, demonstrating a successful modeling rate of 81%. Finally, we load the trained model onto our hardware platform for evaluation, which exhibits consistent behaviors with the simulation. As a result, the overall decision-making framework proposed in this work exhibits great potential to enhance the practical application of DRL-driven AD.



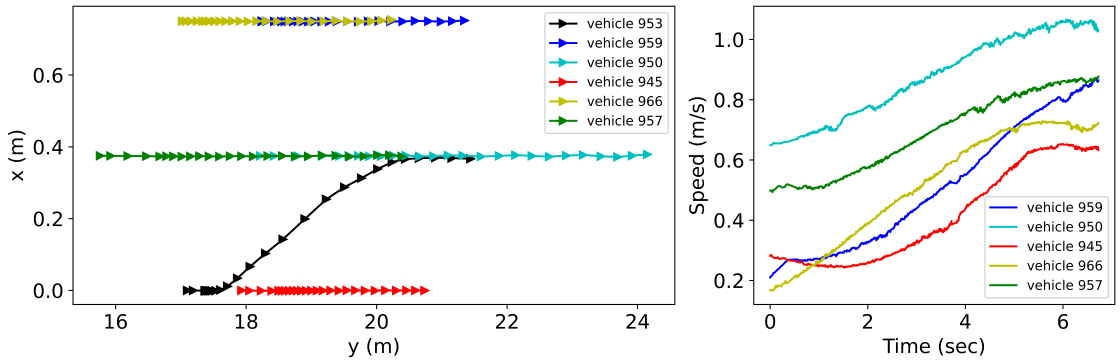

(a) Real traffic data (ID: 953)



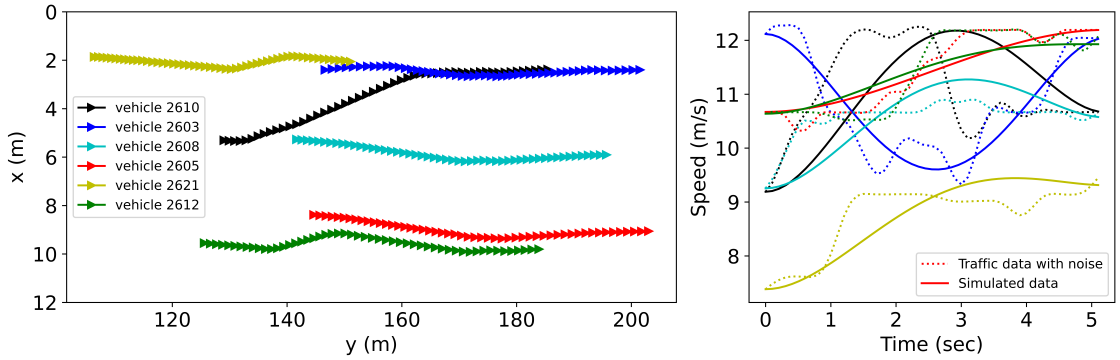

(b) Test results of 953 on scaled data



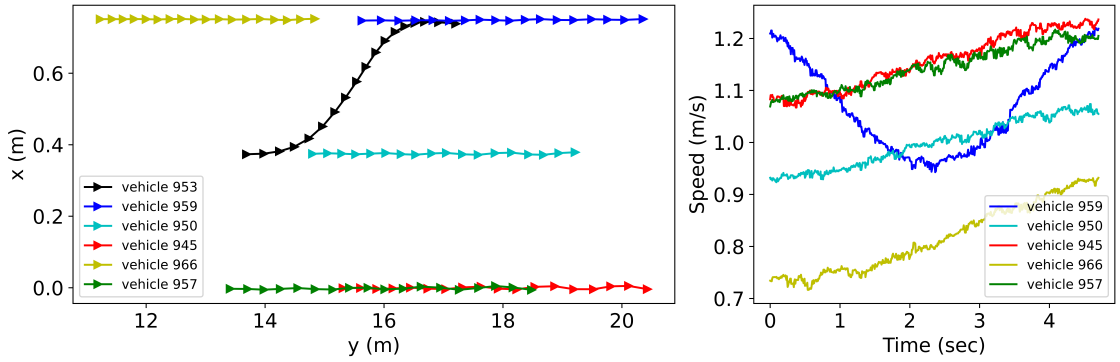

(c) Real traffic data (ID:1054)




(d) Test results of 1054 on scaled data

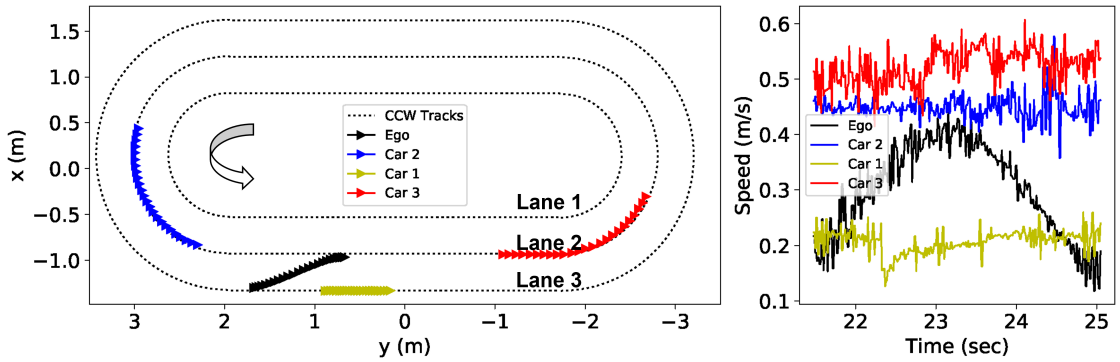



(e) Real traffic data (ID:2610)

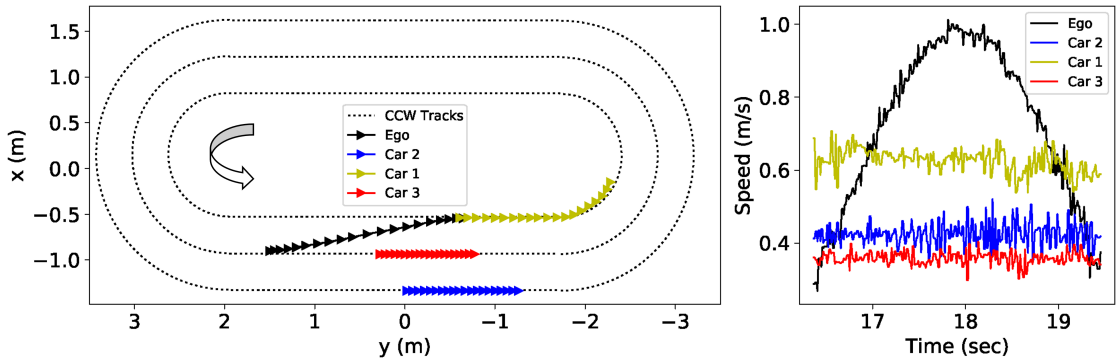



(f) Test results of 1054 on scaled data

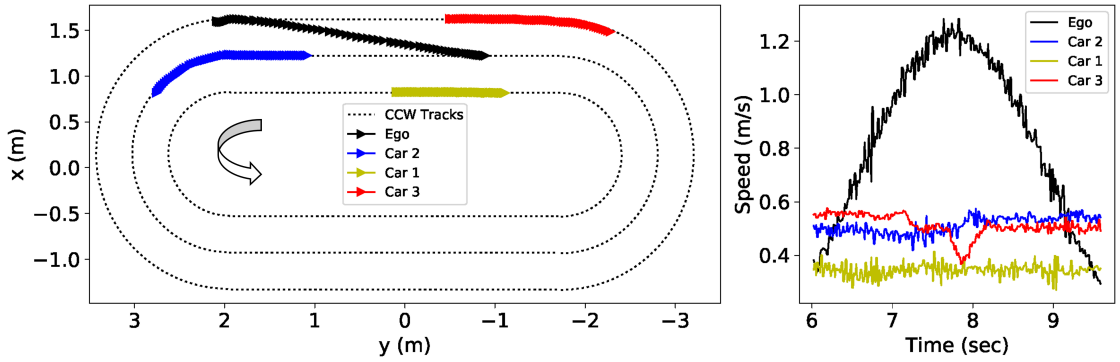
Figure 4.9: Comparing driving behavior of DRL policy with human driver policy under simulated environment: (a), (c), (d) the trajectory and velocity information extracted from US101 with lane-changing vehicle of 953, 1054, and 2610; (b), (d), (f) are the testing results of DRL policy with replaying the trajectories of surrounding vehicles on scaled data.




(a)




(b)




(c)

Figure 4.10: Testing results of the hardware implement in different scenarios (a) changing the lane to the left from lane 3; (b) changing the lane to the left from lane 2; (c) changing the lane to the right from lane 2.

# 5 Scalable Game-Theoretic Decision-Making for AVs

The previous chapters have focused on model-free optimization algorithms for decision-making of AVs. These algorithms allow DRL vehicles to learn driving policies offline through interaction with the environment, addressing real-time decision-making challenges in complex scenarios. This chapter focuses on model-based optimization algorithm, aiming to improve scalability and to reduce the computational complexity of the algorithm. This chapter introduces a novel approach to address the challenge of sharing the road with other road users with different driving behaviors. The proposed method focuses on resolving traffic conflicts in unprotected left-turn scenarios by designing a robust adaptive game-theoretic decision-making algorithm. This algorithm incorporates receding horizon optimization, level-k game theory, and a switching directed graph to ensure scalability. Considering the potential mismatch between predicted driver types in level-k theory and actual driver behavior, the chapter presents a novel solution, which estimates the driver types of surrounding vehicles based on interactions and adapts AV's driving strategy accordingly to improve safety and driving efficiency. Additionally, the computational complexity of the algorithm is significantly reduced by incorporating switching interaction graph into the adaptive level-k framework, disconnecting the ego vehicle from nearby vehicles that do not impact its driving behavior. The feasibility, effectiveness, and real-time implementation of

the proposed algorithm will be validated on both hardware and the ROS-Gazebo platform.

## 5.1 Problem Formulation

There are 12 possible paths for AVs at a four-way single-lane unsignalized intersection as shown in Figure 5.1. One vehicle is chosen as the ego vehicle and the others are classified as opponent vehicles with different reasoning levels. Following [16], we define the roles of AVs and potential traffic conflicts. Each vehicle is seen as an independent decision-maker and can be divided into four categories based on their potential conflicts: Host vehicles (HV), leading vehicle (LV), interactive vehicle (IV), and other vehicles (OV). At an unsignalized intersection, there are three types of driving conflicts determined by the moving trajectories and speeds of AVs.

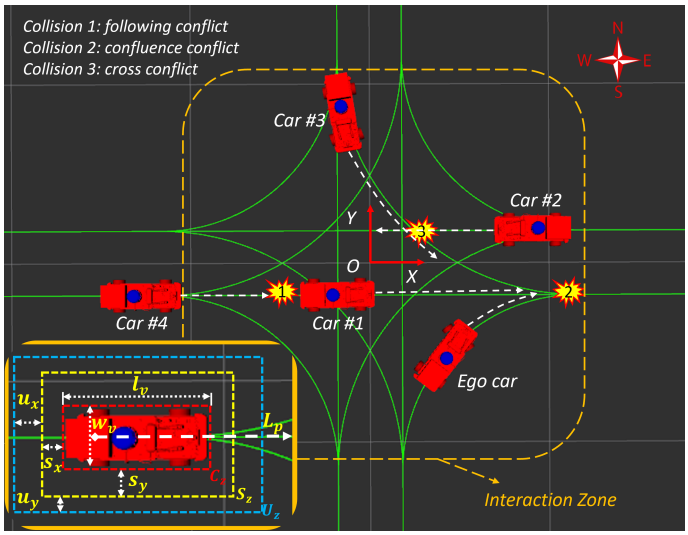



Figure 5.1: Unsignalized intersection scenario

- **Following Conflict** arises when a HV is traveling on the same path as a LV at a higher speed.

- **Confluence Conflict** occurs when two vehicles traveling on different paths merge into the same lane. If an IV passes the collision point before HV, the Confluence Conflict becomes a Following Conflict.
- **Cross Conflict** happens when two vehicles traveling on different paths with an intersection point are heading in different directions

The collision points highlighted in Figure 5.1 demonstrate traffic conflicts faced by AVs at intersections. Interactions among AVs are depicted through a switching directed graph, which will be further explained later. To define the Laplacian matrix of the graph, we refer to the AV experiencing Confluence Conflicts or Cross Conflicts with the HV as an IV. Vehicles that do not impact the actions taken by HVs are considered as OVs. The designations of LV, IV, and OV are subject to change according to the traffic states of AVs in real time.

To resolve the traffic conflicts of AVs at unsignalized intersections, we propose a scalable adaptive game-theoretic decision-making framework, which consists of two modules: modeling, and decision-making, as illustrated in Figure 5.2. First, the interaction topology of HV is established according to the traffic states obtained from perception system. Since the driving aggressiveness of IV has significant effects on their driving behaviors, HV must account for the driver type of IV during the decision-making process. Specifically, HV uses the kinematic model to predict the future actions of IV based on the level-k game theory following a receding horizon strategy to find its own optimal actions. Then, HV's belief on the driver type of IV is updated by comparing the actual actions taken by IV and corresponding predictions made by HV. Finally, the generated speed command will be executed by the controllers. In this work, PID and pure pursuit controllers are used to achieve the longitudinal and lateral control, respectively.

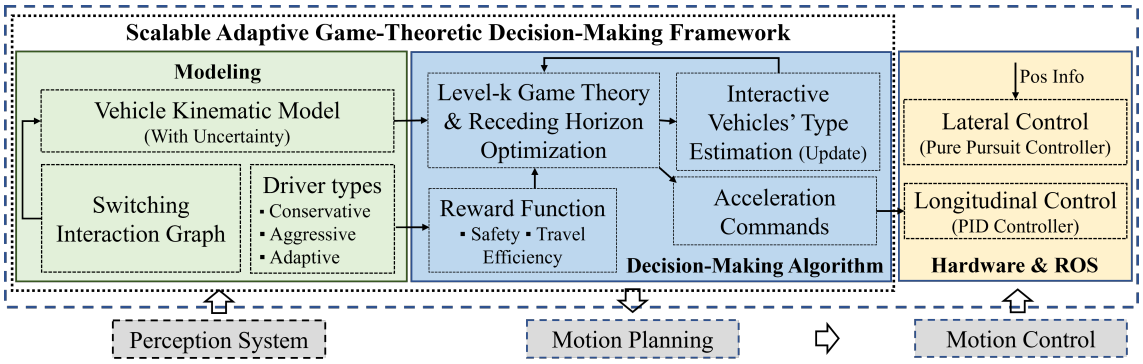



Figure 5.2: Game-theoretic decision-making framework

## 5.2 Vehicle Dynamic Model

Kinematic bicycle model is commonly used to design decision-making algorithms for AVs [20] denoted by Eq. (5.1):

$$x(t+1) = x(t) + v(t)\cos(\psi(t) + \beta(t))\Delta t + u_x(t) \tag{5.1a}$$

$$y(t+1) = y(t) + v(t)\sin(\psi(t) + \beta(t))\Delta t + u_y(t) \tag{5.1b}$$

$$\psi(t+1) = \psi(t) + \frac{v(t)}{l_r}\sin(\beta(t))\Delta t \tag{5.1c}$$

$$v(t+1) = v(t) + a(t)\Delta t \tag{5.1d}$$

$$\beta(t) = \arctan\left(\frac{l_r}{l_r + l_f}\tan\left(\delta(t)\right)\right) \tag{5.1e}$$

$$\delta(t) = \arctan\left(\frac{2L\sin\alpha(t)}{ld}\right) \tag{5.1f}$$

The pair of $(x(t), y(t))$ represents the center of gravity's coordinate position at time $t$, while $v(t)$, $\psi(t)$, and $\beta(t)$ denote the longitudinal speed, yaw angle, and slip angle relative to the vehicle's longitudinal axis, respectively. $a(t)$ represents the longitudinal acceleration. $\delta(t)$ denotes the steering angle. The distances of vehicle's center of mass to the front and rear axles are denoted by $l_f$ and $l_r$. To account for uncertainties resulting from a simplified

model and unknown driving models, we use $u_x(t)$ and $u_y(t)$ to represent the position uncertainties in both longitudinal and lateral directions. $\Delta t$ represents the sampling period. Since we focused on optimizing acceleration, control of lateral movements have delegated to the pure pursuit controller by incorporating Eq. (5.1f). The look-ahead distance, denoted as $ld$, represents the distance between the vehicle's rear axles and a specified point on a desired path, which is calculated using the parameter $k_v v(t)$. The angle between the vehicle's body heading and the look-ahead line, as defined by the center of the rear axles and the target point, is known as $\alpha$.

## 5.3 Scalable Decision-Making Algorithm with Level-k Theory

### 5.3.1 Graph Theory Notions

The interactions among vehicles in the intersections are denoted by switching directed graphs. Let $\left\{\mathcal{G}_k = \left(\mathcal{V}, \mathcal{E}_k, \mathcal{W}_k = \left[w_{ij}^k\right]\right) \mid k \in \mathcal{P}\right\}$ be the set of all possible graphs with $\mathcal{P} = \{1, ..., Q\}$ where $Q > 1$ is an integer. $\mathcal{V} = \{Car1, Ego, Car2, ..., CarN\}$ is a set of $N+1$ nodes in $\mathcal{G}_k$ and $\mathcal{E}_k \subset \mathcal{V} \times \mathcal{V}$ represents the set of edges. $\mathcal{W}_k = \left[w_{ij}^k\right]$ denotes the weighted adjacency matrix, where $w_{ij}^k$ is the weight of the directed edge $(j, i)$ and $w_{ij}^k > 0$ if $(j, i) \in \mathcal{E}_k$; $w_{ij}^k = 0$ otherwise. Let $w_{ii}^k \equiv 0, \forall i \in \mathcal{V}$. The Laplacian matrix of $\mathcal{G}_k$ is defined as $\mathbf{L}_k = \text{diag}\left\{\Delta_1^k, ..., \Delta_N^k\right\} - \mathcal{W}_k$, where $\Delta_i^k = \sum_{j=1}^N w_{ij}^k$ is the in-degree of node $i = 1, ..., N$ [2]. Given any graph $\mathcal{G}$, $\mathcal{V}(\mathcal{G}), \mathcal{E}(\mathcal{G})$, and $\mathbf{L}(\mathcal{G})$ are represented its node set, edge set, and Laplacian matrix, respectively [81, 82].

The topology of the interaction graph is updated in real time according to the traffic conflicts defined in Section 5.1. A sequence of waypoints representing the future path of HV at time $t$ intersects with that of another vehicle leading to a Cross Conflict or a

Confluence Conflict, such vehicle is classified as an IV. The length of the future path of vehicle $l$ is defined by

$$\mathcal{L}\left(p_t[l]\right) = \sum_{j=0}^{\hat{\mathbf{N}}-1} |\hat{\mathbf{p}}_{t+j+1} - \hat{\mathbf{p}}_{t+j}|, \tag{5.2}$$

where $\hat{\mathbf{p}}_{t+j}$ represents the coordinates of the next $j^{th}$ waypoint starting from the center point of its rear axle. $\hat{\mathbf{N}}$ is the total number of waypoints in its future path, $p_t[l]$, of vehicle $l$.

Then the edge between the HV and IV will be denoted by a double arrow, which represents an interaction among them. For the Following Conflict, the edge between HV and the other vehicle will be denoted by a single arrow pointing to the HV, which represents a collision avoidance task for HV and no interaction among them. Taking Figure 5.1 as an example, in traditional level-k based decision-making algorithms, the interaction topology between vehicles can be represented by Figure 5.3(a), which could limit the scalability and real-time implementation of algorithms due to its strongly connected property. However, some vehicles that do not affect the actions taken by HV should be removed from the interaction graph for reducing computational burden purposes. Switching directed graph can help to effectively simplify the interactions between vehicles as shown in Figure 5.3(b) with solid edges. Therefore, the ego only needs to account for Car 1 instead of all of them in this case.

**Remark 1**: Set $\hat{\mathbf{D}}[l]$ represents the group of vehicles for which HV needs to account for their potential actions when making decisions, while set $\hat{\mathbf{O}}[l]$ includes the vehicles that pose collision risk or are in a Following Conflict situation with HV.

### 5.3.2 Action Set and Running Reward for Decision-Making

To resolve driving conflicts, we assume that a vehicle has a finite set of acceleration levels to choose to adjust its speeds along the desired path at each time step, i.e., $a(t) \in A =$

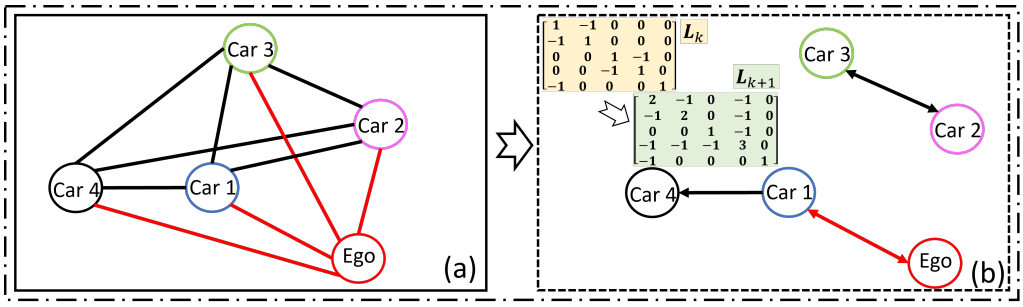



Figure 5.3: Interaction graph: (a) undirected graph (strongly connected); (b) directed graph

$\left\{a^1, \cdots, a^{\mathcal{M}}\right\}, \forall t$. The acceleration to be applied to the vehicle at each step is decided according to the optimization of a reward function described as follows.

The cumulative reward is given by

$$\mathcal{R}\left(\gamma_t\right) = \sum_{j=0}^{N-1} \lambda^j R_{t+j}. \tag{5.3}$$

An optimal action sequence of HV with prediction horizon $N$, $\gamma_t^* = \left\{a_t^*, a_{t+1}^*, \cdots, a_{t+N-1}^*\right\}$, can be obtained by maximizing above cumulative reward given by Eq. (5.3) following a receding horizon strategy. Since the HV only executes the first element of the action sequence at each time step, and repeats for every control cycle, it provides a certain degree of inherent robustness to uncertainties because of the feedback loop [83].

The stage reward is defined as:

$$R_{t+j} = w_1\phi_{t+j}^{(1)} + w_2\phi_{t+j}^{(2)} + w_3\phi_{t+j}^{(3)} + w_4\phi_{t+j}^{(4)} + w_5\phi_{t+j}^{(5)} \tag{5.4}$$

where $\phi_{t+j}^{(k)}$ is $k^{th}$ indicator variable for a specific driving feature at prediction step $j$. $\omega_k > 0$ is the weight of the corresponding factor. More features could be added to Eq. (5.4) for more driving preferences of HV.

There are four approximations of vehicle perceptions, i.e., Collision zone ($C_z$), Safe zone ($S_z$), Uncertainty zone ($U_z$), and the length of future path ($\mathcal{L}\left(p\right)$), which will be used

to define the reward function. $C_z$ is red dashed rectangular with the length of $l_v$ m and the width of $w_v$ m; the $S_z$ is yellow dashed rectangular with the length of ($l_v$+2$s_x$) m and the width of ($w_v$+2$s_y$) m; $U_z$ is blue dashed rectangular with the length of ($l_v$+2$s_x$+2$u_x$) m and the width of ($w_v$+2$s_y$+2$u_y$) m, in which $s_x, s_y, u_x, u_y \geq 0$, as shown in Figure 5.1.

According to the observations defined above, driving features of the reward function are characterized by:

- **Interaction Status** determined by the Cross/Confluence Conflict between HV and IV. If intersection of their future paths is detected $\phi_t^{(1)} = J_{ttc}$; and 0 otherwise.

$$\phi_t^{(1)} = \sum_{i=1}^{n} \begin{cases} -J_{ttc}, & p_t[l] \cap p_t[i], \forall i \in \hat{\mathbf{D}}[l] \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

$$\begin{aligned} J_{ttc} &= 1/\left(\left(\Delta T_{(i,l)}\right)^2 + \varepsilon\right); \Delta T_{(i,l)} = T_{\hat{\mathbf{c}}}[l] - T_{\hat{\mathbf{c}}}[i] \\ T_{\hat{\mathbf{c}}}[l] &= \Delta s^{\hat{\mathbf{c}}}[l]/v_l; T_{\hat{\mathbf{c}}}[i] = \Delta s^{\hat{\mathbf{c}}}[i]/v_i \end{aligned} \tag{5.6}$$

  where $p_t[i]$ represents the sequence of waypoints of the $i$th vehicle at time $t$. And the HV is denoted by $l$. We assume that the cross point between future paths is $\hat{\mathbf{c}}$. The time for vehicle $l$ and vehicle $i$ to reach the cross point $\hat{\mathbf{c}}$ at current velocity from their current positions can be expressed as $T_{\hat{\mathbf{c}}}[l]$ and $T_{\hat{\mathbf{c}}}[i]$, respectively. The closer $\Delta T_{(i,l)}$ is to zero, the greater the risk of collision between them.

- **Collision status**: If an overlap, representing a vehicle collision, between the $C_z$ of HV and that of any other cars is detected then $\phi_t^{(2)}$ = -1; and 0 otherwise.

$$\phi_t^{(2)} = \sum_{i=1}^{n} \begin{cases} -1, & \mathcal{C}_t[l] \cap \mathcal{C}_t[i], \forall i \in \hat{\mathbf{O}}[l] \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

- **Safe zone violation status**: If an overlap between the $S_z$ of HV and that of any other cars is detected then $\phi_t^{(3)}$ = -1; and 0 otherwise.

$$\phi_t^{(3)} = \sum_{i=1}^{n} \begin{cases} -1, & \mathcal{S}_t[l] \cap \mathcal{S}_t[i], \forall i \in \hat{\mathbf{O}}[l] \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

- **Uncertainty zone violation status**: If an overlap between the $U_z$ of HV and that of any other cars is detected then $\phi_t^{(4)}$ = -1; and 0 otherwise.

$$\phi_t^{(4)} = \sum_{i=1}^{n} \begin{cases} -1, & \mathcal{U}_t[l] \cap \mathcal{U}_t[i], \forall i \in \hat{\mathbf{O}}[l] \\ 0, & \text{otherwise} \end{cases} \tag{5.9}$$

- **Travel efficiency**: $\phi_t^{(5)}$ is to encourage vehicles to pass the intersection efficiently, which is described by

$$\phi_t^{(5)} = -\left| v_t - v^{\text{ref}} \right| \tag{5.10}$$

where reference speed $v^{\text{ref}}$ is typically chosen as the legislated speed limit of the traffic scenario.

The calibration of model-based controllers is challenging [20], however, intuitively, we prioritize driving safely over travel efficiency when tuning the weights of these factors in the reward function. Therefore, the tuning parameters are chosen as,

$$w_2 > w_3, w_4 > w_1 > w_5 \tag{5.11}$$

### 5.3.3 Robust Adaptive Game-Theoretic Decision-Making

#### 5.3.3.1 Level-k Decision-Making

To model multi-vehicle interactions, all driving features except $\phi^{(5)}$ in the stage reward function (5.4) are utilized to reveal the interactive behaviors, depending on the states of vehicles. A sequence of action of a HV is represented by $\gamma_t[l]$. The action sequence of $i^{th}$ IVs with reasoning depth 'k' predicted by HV is denoted by $\gamma_t^{(k)}[i]$. These actions are used to calculate the cumulated reward in Eq. (5.4) according to the corresponding traffic states at prediction steps $j \in \mathbb{Z}_{[0,N-1]}$, denoted by

$$\begin{aligned} s_{t+j} = & \big[x_{t+j}[1], y_{t+j}[1], \psi_{t+j}[1], v_{t+j}[1], \\ & \cdots, x_{t+j}[n], y_{t+j}[n], \psi_{t+j}[n], v_{t+j}[n]\big]^T. \end{aligned} \tag{5.12}$$

**Remark 2**: It should be noted that $s_{t+j}$ is the state information of selected vehicles, represented by the $\mathbf{L}_{\sigma(t)}$ of interaction graph in the set of $\mathcal{O}[l] = \hat{\mathbf{D}}[l] \oplus \hat{\mathbf{O}}[l]$, that affect the action of HV instead of accounting for interactions among all vehicles in the unsignalized intersections.

Inspired by skilled human drivers (HDs), they could navigate complex unsignalized intersections effectively due to: (1) driving difficulty being dependent on the number of interacting vehicles rather than the total number present; (2) prioritization of driving tasks based on conflict types with surrounding vehicles; and (3) personalized driving preferences that experienced HDs utilize to estimate and predict others' behaviors, leading to optimal actions for safety and efficiency.

The level-k theory adopted in this chapter exactly works in the way mentioned above. Specifically, level-'k' represents the reasoning level of decision-makers starting from non-strategic policy, level-0, that usually takes action to achieve its goal without accounting for the interactions between itself and other agents. However, decision-makers above this level of reasoning will assume that all the other agents are level-(k-1). They will predict the future actions taken by IV based on such an assumption and take their own optimal actions accordingly.

In this chapter, level-0 HV, representing an aggressive driver, mainly behaves as a collision avoidance strategy in static environments with a shorter length of future path and zero uncertainty to others. A level-1 HV will assume that all IVs are drivers with level-0 reasoning, therefore, responds to them cautiously. The length of its future path and the size of the uncertainty zone around IVs will always be the maximum values from the perspective of HV. Similarly, further higher levels can be defined. In this work, only level-0 and level-1 drivers are considered due to the similarity between level-0 and level-2 [17, 20]. However, this algorithm can be extended to higher levels at the expense of

increased computational complexity. Once the level-0 is defined, the action calculation for finding level-k action, $k \geq 1$, in the case of N-vehicle interactions is represented by the following form:

$$\begin{aligned} R_{t+j}^{(k)}[l] = R_{t+j}\Big(\gamma_{t+j}^{(k)}[l] \mid s_0, \gamma_t^{(k)}[l], \gamma_{t+1}^{(k)}[l], \cdots, \gamma_{t+j-1}^{(k)}[l], \\ \gamma_t^{(k-1)}[i], \gamma_{t+1}^{(k-1)}[i], \cdots, \gamma_{t+j-1}^{(k-1)}[i]\Big)\, i \in \hat{\mathbf{D}}[l], \end{aligned} \tag{5.13}$$

and its cumulative reward is

$$\mathcal{R}^{(k)}\left(\gamma_t^{(k)}[l]\right) = \sum_{j=0}^{N-1} \lambda^j R_{t+j}^{(k)}[l] \tag{5.14}$$

#### 5.3.3.2 Robust Adaptive Decison-Making

The assumption that level-k HV always interacts with IVs with level-(k-1) reasoning level is unrealistic. Mismatch between the (k-1) assumption and actual driver type may lead to unsafe action selection and reduce the driving safety. Intuitively, the interactions of level-k versus level-k could lead to unexpected driving behavior, i.e., congestion or collision [40]. Inspired by HDs, HV should also be capable to assess the driver type of each IV via interactions. The *trust* (or belief) of HV on the driving model of each IV should also be updated in real time by comparing the actual action applied by each IV, $\gamma[i](t)$, and corresponding predictions for a level-k driver, $\gamma_t^{(k)}[i]$, made by HV, which can be represented as a continuous parameter between level-0 and level-1. The HV's *trust*, $T_{HV}^{K=k^*}[i](t)$, can be represented by a probability that the $i$-th IVs can be modeled as model $k$ driver, given by

$$k^* = \arg\min_{k \in \{0,1\}} \left\| \gamma[i](t) - \gamma_t^{(k)}[i] \right\| \tag{5.15a}$$

$$\tilde{T}_{HV}^{(K=k^*)}[i]_l(t) = T_{HV}^{(K=k^*)}[i](t-1) + \Delta P \tag{5.15b}$$

$$T_{HV}^{(K=k)}[i](t) = \frac{\tilde{T}_{HV}^{(K=k)}[i](t)}{\sum_{k'=0}^{1} \tilde{T}_{HV}^{(K=k')}[i](t)}, \forall k \in \{0,1\}, \tag{5.15c}$$

where the rate of increment of the *trust* is denoted by $\Delta P$, which is a positive constant. Since there may exist some scenarios in which the actions taken by drivers are the same regardless of the driver types, the probability of each driver type remains the same. Otherwise, the HV's *trust* in driver model $k^*$ that matches the actual action best increases by $\Delta P$. Then, the probability distribution is normalized by Eq. (5.15c). An algorithm flowchart is provided to help illustrate the decision-making process of above level-0, level-1, and adaptive policies, see Figure 5.4.

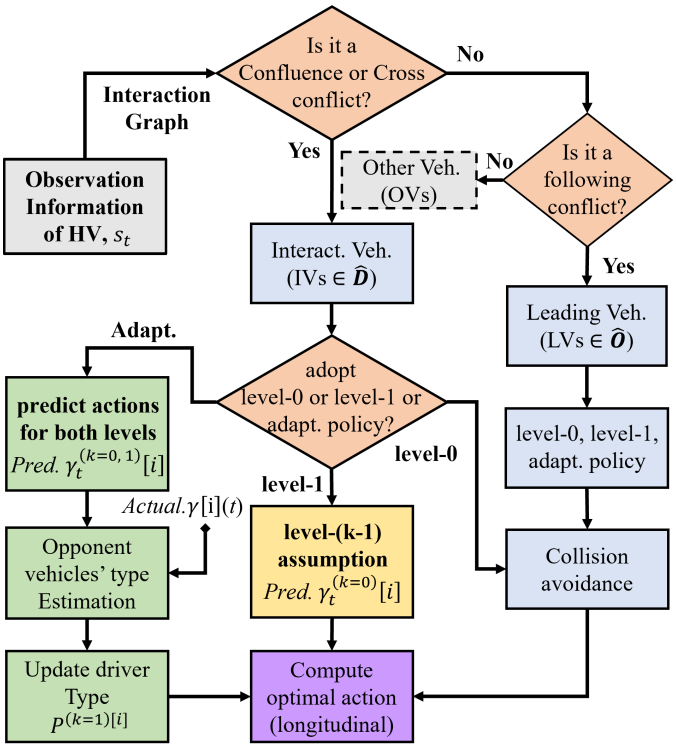



Figure 5.4: Algorithm flowchart

The adaptive decision-making approach proposed in this chapter is based on the multi-model strategy, it finds an optimal action sequence for HV at each time step according to the estimated driver model of each IV. The expected accumulated reward of an action

sequence is calculated by

$$\mathcal{R}_P\left(\gamma_t[l]\right) = \sum_{k=0}^{1} T_{HV}^{(K=k)}[i](t)\mathcal{R}^{(k)}\left(\gamma_t^{(k)}[l]\right), \forall i \in \mathcal{D}[l] \tag{5.16}$$

To improve the robustness of the algorithm, two sources of modeling errors are considered during the decision-making process of HV. Specifically, a simplified kinematic model could introduce the model uncertainty to the decision-making process, which is denoted by $u^{(m)} = (u_x^{(m)}, u_y^{(m)}) \in U_m$, and unknown driver type could lead to an interaction uncertainty denoted by $u^{(d)} = (u_x^{(d)}, u_y^{(d)}) \in U_d$. To deal with the uncertainties from the simplified kinematic model, the safe margin of the safety zone is used to compensate for the model uncertainty, $u^{(m)}$, which is a mismatch in the position between the actual position of IV and the corresponding prediction by HV, the values of which will not be changed all the time. The uncertainty zone of a vehicle is also a rectangle area that subsumes the safe zone of the car with margins on both longitudinal sides and lateral sides to handle the interaction uncertainty. To realize the adaptive scheme, the size of it is modified according to the HV's trust in the driver type of each IV. The uncertainty set could be denoted by

$$\mathcal{U}_{HV}[i](t) = \mathcal{U}_m \oplus T_{HV}^{(K=0)}[i](t)\mathcal{U}_d \tag{5.17}$$

To ensure safety, the probability that the driver type of IV can be modeled as level-0 is set to 1 initially in Eq. (5.15), which is reasonable since HV should behave cautiously when it does not have too much interaction data with IV at the beginning.

In addition, unlike the highway scene where cars are only moving in the longitudinal direction, HV should be capable to resolve lateral conflicts including Cross Conflict and Confluence Conflict in unsignalized intersections, therefore, the length of its future path should be modified according to the value of Trust in the model of each IV as well, given by

$$\hat{\mathcal{L}}\left(p_t[l]\right) = \mathcal{L}\left(p_t[l]\right) T_{HV}^{(K=0)}[i](t) \tag{5.18}$$

The driving feature about interaction conflict status will be calculated based on the length of it. To find the optimal action sequence, $\hat{\gamma}[l]$, it essentially solves an optimization problem by maximizing the expected cumulative reward Eq. (5.16), given by

$$\begin{aligned} \hat{\gamma}_t[l] = \arg \max_{\gamma_t[l] \in \mathcal{A}} \min_{u_{t+j} \in \mathcal{U}_l[i](t)} \mathcal{R}_P \left(\gamma_t[l]\right) \\ \text{s. t. } \forall j \in \mathbb{Z}_{0,N-1}, [5.1a - 5.1f], \forall i \in \mathcal{O} \end{aligned} \tag{5.19}$$

A decision tree approach is used for searching an optimal action sequence at each time step by enumerating all possible combinations of discrete actions. Then, the first element of $\hat{\gamma}_t[l]$ is applied to the vehicle, and the cycle continues.

## 5.4 Experimental Validation

To evaluate the performance of our algorithm in terms of adaptability, computational complexity, real-time performance, and scalability, we analyze test results on both hardware platform and high-fidelity simulator, including switching interaction graph, travel efficiency, drivers' type estimation, computational load and its comparison with a traditional method.

### 5.4.1 Performance of Adaptive Policy on Hardware

To experimentally validate the effectiveness of the scalable adaptive game-theoretic decision-making algorithm (Adpt.), four AVs in indoor environment unsignalized intersection are utilized, see Figure 2.7a. More details about our hardware platform can be found in the Section 2.2.3.

Due to space limitations, unprotected left turn is chosen as the primary testing scenario since it is the most challenging case among intersection-related problems [44], where all vehicles navigate intersections based on their local observation. In each test, different

Table 5.1: Policy setting at unprotected left turn scenarios

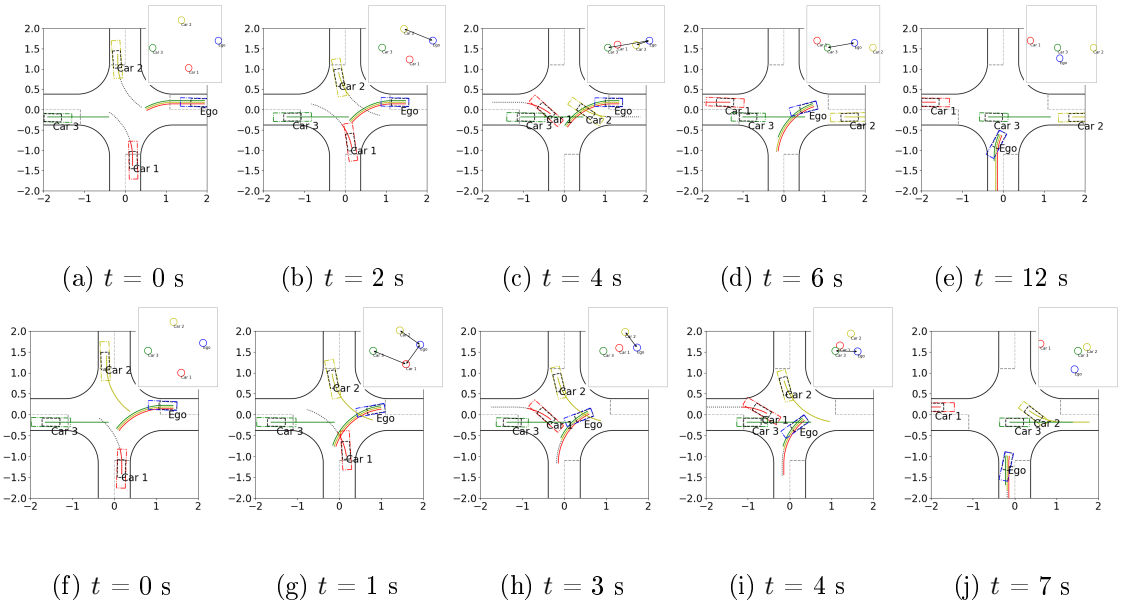
| Case | Car1 | Ego | Car2 | Car3 | Car4 | Car5 | Car6 |
|---|---|---|---|---|---|---|---|
| **1** | L0(l) | Adpt.(l) | L0(l) | L1(s) | ∅ | ∅ | ∅ |
| **2** | L0(l) | Adpt.(l) | L1(l) | L1(s) | ∅ | ∅ | ∅ |
| **3** | L0(l) | Adpt.(l) | L1(l) | L1(s) | L0(l) | L1(r) | ∅ |
| **4** | L0(l) | Adpt.(l) | L1(l) | L0(r) | L0(l) | L1(r) | L1(s) |

Note: L0: level0; L1: level1; Turn left (l), right (r); straight (s).

driving strategies were assigned to vehicles as shown in Table 5.1, where the level-0 policy can be regarded as an aggressive driver (Aggr.) while level-1 is a conservative driver (Consrv.). These driving policies are unknown to each other. Ego adopts the proposed adaptive driving policy that allows it to navigate through an unsignalized intersection where surrounding vehicles exhibit varying degrees of aggressive driving behaviour safely and efficiently via interaction[1].

The decision-making result of Case 1 is shown in the first row of Figure 5.5, where the dotted line in grey with fixed length indicates the future path of each vehicle, which is used to define the driving conflicts, i.e., Cross, Confluence, and Following conflicts. Solid lines in different colors in front of all vehicles except ego can be regarded as the ground truth of the driver type of each vehicle, which is used to calculate stage rewards at each time step. The length of which will not be changed until the next round of tests. The length of the line segment represents the aggressiveness of a driver. The more conservative the driving behaviour, the longer the line segment. The maximum length is equal to that of the gray dotted line. For ego car, there are multiple line segments in front of it. And the number of lines depends on the number of IVs. The bounding box of each vehicle in different colors represents the uncertainty zone from the perspective of ego. Both the

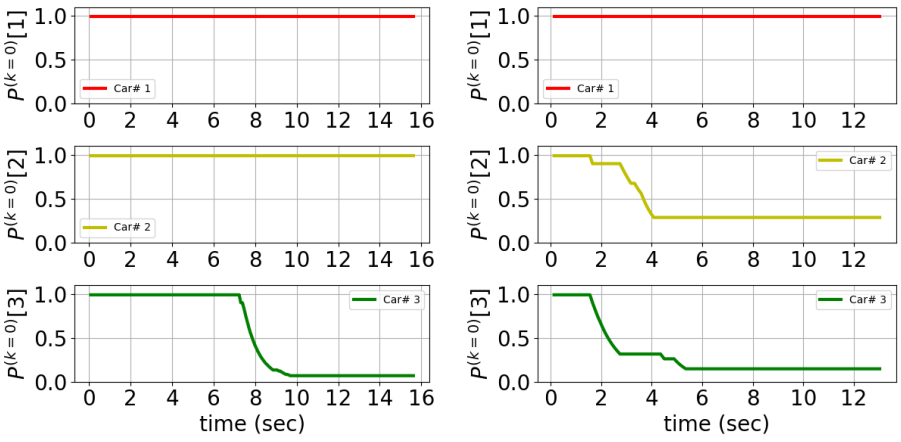
[1]More testing scenarios, algorithm parameters, and experimental settings can be found at `https://youtu.be/q6vKrjqHD54`




(a) $t = 0$ s (b) $t = 2$ s (c) $t = 4$ s (d) $t = 6$ s (e) $t = 12$ s

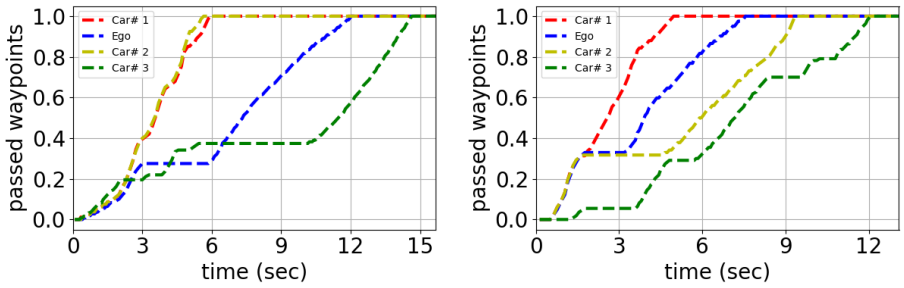
(f) $t = 0$ s (g) $t = 1$ s (h) $t = 3$ s (i) $t = 4$ s (j) $t = 7$ s

Figure 5.5: Adaptive decision-making results of ego vehicle in unprotected left turn scenarios ($a$-$e$) aggress: car 1, car 2; consrv.: car 3; ($f$-$j$) aggress: car 1; consrv.: car 2, car 3



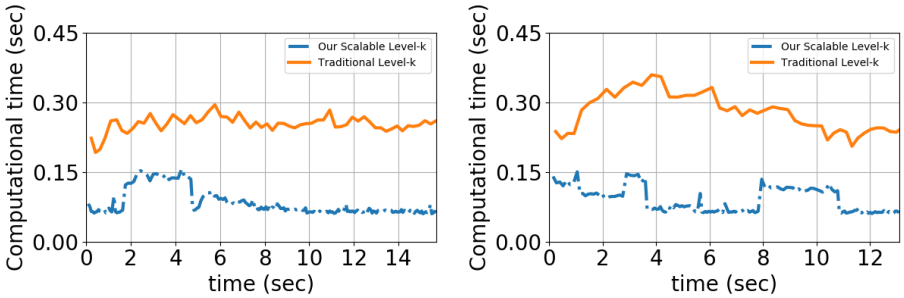

(a) 2 Aggr. & 1 Consrv. Drivers (b) 1 Aggr. & 2 Consrv. Drivers

Figure 5.6: Driver models identification history of ego vehicle




(a) 2 Aggr. & 1 Consrv. Drivers (b) 1 Aggr. & 2 Consrv. Drivers

Figure 5.7: Travel efficiency of four vehicles




(a) 2 Aggr. & 1 Consrv. drivers (b) 1 Aggr. & 2 Consrv. drivers

Figure 5.8: Computational time

length of each line segment in front of the ego car and the size of the bounding box of each IV have a linear dependence relation with the probability that a specific car can be modeled as level-0 driver. Intuitively, the higher the confidence that a certain vehicle can be modeled as an aggressive driver, the larger the reserved safety interaction space would be. In the beginning, the length of all line segments of ego and the size of the bounding boxes of IVs are the maximum. This is because ego initially assumes that the driving types of surrounding vehicles are all level-0. Ego would behave cautiously due to the lack of interactive data. From 0 s to 4 s, car 1 and car 2 start to accelerate and choose to pass through the intersection first while car 3 and ego choose to yield the right of the way

to them. At t = 6 s, ego car is interacting with car 3. The length of the green line of ego indicating the probability that car 3 can be modeled as level-0 starts reducing. This is because car 3 chooses to continue to yield the right of way to the ego, therefore, car 3 finally is regarded as a level-1 driver after interactions instead of level-0. However, the length of the solid red line and yellow line in front of ego car does not change during the interactions, which means that the driver types of car 1 and car 2 identified by ego are level-0. The driver model identification history of ego car in case 1 can be also found in Figure 5.6a.

The travel efficiency of vehicles at the intersection can be represented by Figure 5.7a, in which the number of waypoints for each path is normalized. The y-axis represents the completion progress of vehicles on their path, and the time corresponding to the progress of 1 can be used to indicate the order in which each vehicle exits the intersection. In Case 1, aggressive vehicles, car 1 and car 2, are the first to pass through the intersection, followed by ego vehicle. And the conservative vehicle of car 3 is the last to exit the intersection.

In Case 2, level-0 policy is assigned to car 1 while level-1 policy is assigned to car 2 and car 3. According to the second row of Figure 5.5, all cars choose to stop and let car 1 pass the intersection before t = 3 s. Then car 2 and car 3 continue to wait until ego passed the potential collision points. After that car 2 choose to move since it is closer to the exit point of the intersection. Two level-1 drivers are successfully identified by ego according to Figure 5.6b. The order of exiting the intersection is car 1, ego, car 2, and car 3, respectively as shown in Figure 5.7b. Due to space limitations, more test scenarios and model parameter settings can be found in the video.

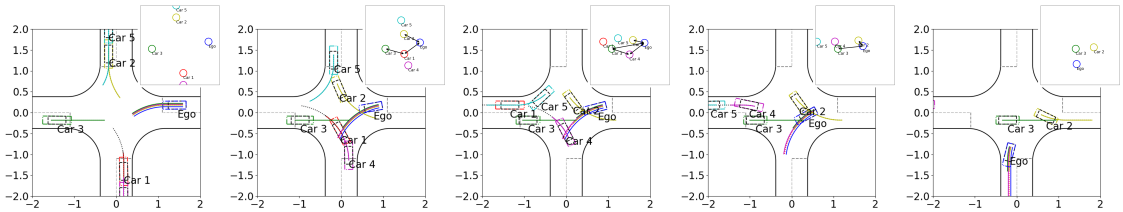
The computational load mainly comes from solving the optimal actions for the HV following receding horizon optimization while considering all possible driving behaviours of the IVs, which is mainly affected by the number of IVs. The switching interaction graphs

of the ego vehicle are generated according to its local observation and corresponding trust as shown in Figure 5.5. In such four-vehicle intersection scenarios, the proposed algorithm effectively simplifies the interaction relationship between the ego car and surrounding vehicles, thereby reducing the computational cost of game-theoretic decision-making algorithm. Compared with the traditional level-k decision-making algorithm [20], the computational time of our algorithm is reduced by around 50% on average as shown in Figure 5.8. The advantage of the proposed algorithm in reducing computational complexity is more obvious in more complex multi-vehicle scenarios, which will be discussed in the next section using our high-fidelity platform.

### 5.4.2 Scalability and Computational Complexity

Developing a high-fidelity simulator for verifying the AV system is of great significance for improving the driving safety of AVs and saving costs of road tests. In this work, we developed a high-fidelity testing platform for the unsignalized intersection scenarios with multiple vehicles using ROS-Gazebo, which supports dynamic simulation, sensor data acquisition, and customized traffic environments helping to narrow the gap between the simulation and the real world, see Figure 2.7b in the Section 2.2.3.

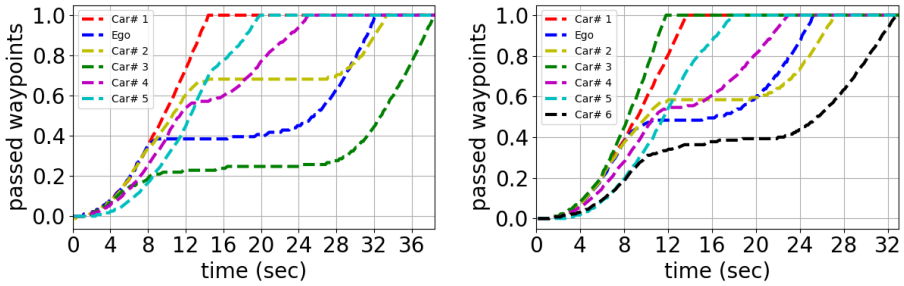
To validate the scalability and computational efficiency of the proposed algorithm, more complex scenarios are considered in this section, where 6 or 7 vehicles are navigating the intersection at the same time with different combinations of aggressiveness as shown in Table 5.1. Case 3 is a six-car scenario, where the ego car adopts an adaptive policy while the rest vehicles use the level-1 strategy except car 1 and car 4 using level-0. In Case 4, an additional car 6 is added behind car 3 and goes straight with a level-1 driving strategy. The other vehicles are basically the same as Case 3 except the car 3 turning right with a level-0 strategy. According to Figure 5.9, the ego car passed the intersections successfully




(a) $t = 0$ s (b) $t = 10$ s (c) $t = 16$ s (d) $t = 25$ s (e) $t = 33$ s

(f) $t = 0$ s (g) $t = 7$ s (h) $t = 15$ s (i) $t = 17$ s (j) $t = 27$ s

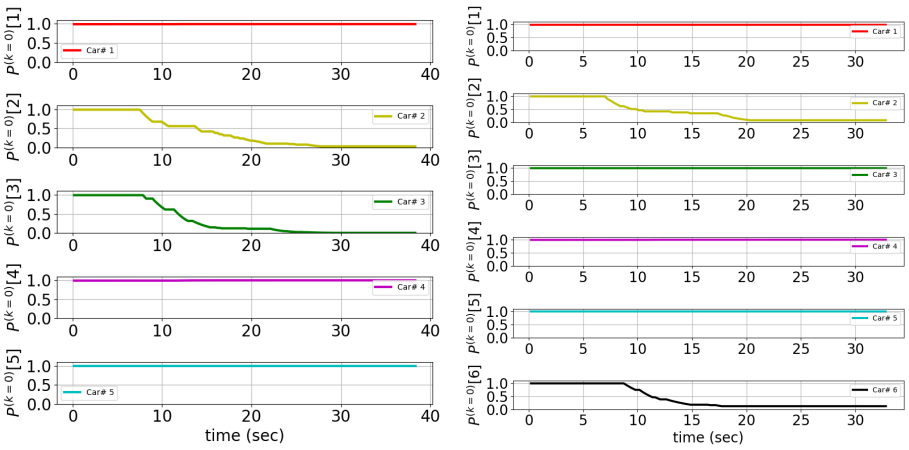
Figure 5.9: Adaptive decision-making results of ego vehicle in unprotected left turn scenarios ($a$-$e$) consrv.: car 2, car 3, and car 5; aggr.: car 1 and car 4; ($f$-$j$) consrv.: car 2, car 5, and car 6; aggr.: car 1, car 3, and, car 4




(a) Six-car Scenario (b) Seven-car Scenario

Figure 5.10: Travel efficiency

without any collisions with other cars. The orders of exiting the intersections for these two cases are shown in Figure 5.10. Similar to the previous four-car scenarios, vehicles with aggressive policy pass through the intersection before vehicles with conservative policy. And ego car adjusted its driving strategy adaptively based on the aggressiveness of IVs



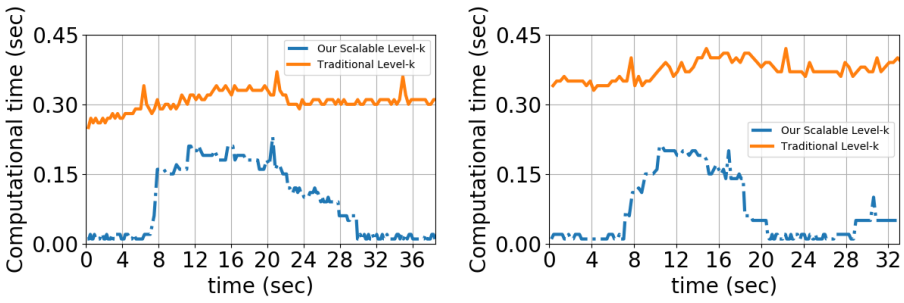

(a) 2 Aggr. & 3 Consrv. Drivers (b) 3 Aggr. & 3 Consrv. Drivers

Figure 5.11: Driver models identification history of ego vehicle

while guaranteeing both traffic efficiency and driving safety. According to Figure 5.11, ego has successfully identified that level-1 policy is assigned to car 2 and car 3 in Case 3 and assigned to car 2 and car 6 in Case 4. It should be noted that although car 5 adopts a level-1 policy in both scenarios, ego car does not update its trust level for car 5 as a level-0 driver. This is because car 5 turns right in both cases, according to the switching interaction graph provided in Figure 5.9, there is no interaction between the ego car and car 5. Therefore, the ego's trust level for car 5 as a level-0 driver remains at its initial value.

The reduction in computational complexity in this work is mainly attributed to the introduction of a switching interaction topology mechanism. In the following, we will use the seven-vehicle scenario to demonstrate how this algorithm can effectively reduce computation time. At $t = 0$ s, ego's future path has no intersections with that of surrounding vehicles, resulting in no interaction behavior and allowing the algorithm to run at its fastest state, close to 0.03 s. However, at $t = 7$ s, ego vehicle models car 1 and car 2 as interactive vehicles because their future paths intersected. Additionally, car 6 was also considered an

interactive vehicle because the driving behaviour of car 1 is going to be influenced by the car 6, indirectly affecting ego's decision-making. The interaction topology between ego and the surrounding vehicles can be seen in the upper right corner of each figure, see Figure 5.9 (g). Therefore, the computation time rapidly increases in Figure 5.12 (b), peaking at around 0.17 seconds. As the vehicles proceed, the interaction topology changes, and by $t = 17$ s, only car 2 is affecting ego's actions, leading to a decrease in computation time. After $t = 27$ s, ego successfully passes through the intersection, terminating all interaction behaviour and returning to its fastest-running state. In contrast, for the traditional algorithm, ego needs to consider the behaviour of all vehicles, resulting in a computation time of approximately 0.4 s.




(a) Six-car Scenario (b) Seven-car Scenario

Figure 5.12: Computational time

In summary, the computational time of our algorithm does not increase with the number of vehicles compared with the traditional algorithm, as shown in Figure 5.12. The computational load is reduced by 55% in six-car scenario and reduced by 63% in seven-car scenario on average, where the blue dotted line represents the computational efficiency of the traditional level-k algorithm and the yellow line represents that of our algorithm. This is because the proposed algorithm can effectively cut off the relationship between ego

and other vehicles that do not affect its decision-making directly, thereby improving the scalability and performance in real-time implementation. We remark that more vehicles could be added to the simulator at the expense of increasing the computational burden due to the physics calculation of the Gazebo.

## 5.5 Summary and Discussion

Based on game-theory and interaction graph, a novel scalable decision-making framework for AVs is proposed for resolving driving conflicts at unsignalized intersections. The aggressiveness of drivers and uncertainties arising due to the simplified model are taken into account in the decision-making framework. In the payoff function design of decision-making, multiple driving features are considered including driving safety, travel efficiency, and driving aggressiveness. To reduce the inherent computational complexity of level-k game theory, the concept of switching directed graphs is incorporated into the adaptive decision-making framework. Finally, the algorithm is verified on both self-driving car hardware and a high-fidelity simulator with multiple vehicles. According to the testing results, it can be conducted that the proposed algorithm makes robust adaptive decisions for AVs, meanwhile, the performance of the algorithm in terms of interpretability, computational efficiency, and scalability can be guaranteed. Our future work will focus on the real time implementation of the proposed method in continuous action space. Game Theory-Model Predicted Control-Deep Reinforcement Learning hybrid approach could further boost the performance of proposed algorithm in computational complexity and safety.

# 6 Conclusion and Future Work

## 6.1 Conclusions

To guarantee safe and efficient functioning of AVs, that share the road with other road users, it is crucial to equip them with robust decision-making and control systems. These systems must reliably adapt and respond to interactions with other road users. This dissertation tackles pressing issues in the development and trustworthy validation of these systems, approaching the challenges from the following distinct perspectives.

Firstly, an efficient and stable DRL algorithm has been introduced as a tool in this dissertation to achieve the long-term goal optimization of a AV in partially observable driving environments. Based on this algorithm, a generic algorithmic framework for multi-sensor fusion has been proposed, enabling end-to-end decision-making in AD. To validate the various decision-making algorithms proposed in this dissertation, a comprehensive digital twin platform has been developed, facilitating end-to-end RL training in diverse traffic scenarios, sim-to-real transfer, and algorithm verification across different environments.

Secondly, an adaptive game-theoretic decision-making strategy utilizing DRL has been proposed for AVs navigating in a multi-agent traffic scenario. The interactions between vehicles have been modeled using a level-k game-theoretic framework. The ego vehicle potentially estimates the aggressiveness of other vehicles in real-time based on sensor data

and adapts its behaviour accordingly. Simulation and hardware tests demonstrate that the proposed model exhibits reasonable behaviour in traffic situations. The framework is suggested for use in calibrating, validating, and verifying AD systems. The proposed model can also be extended to model vehicle interactions in various traffic scenarios without loss of generality, such as highway merging and parking lots, with modifications to road layouts and geometries.

In addition, decision-making in lane-changing scenarios with DRL is discussed, focusing on improving learning efficiency, solving POMDP, and achieving sim-to-real transfer. A novel DRQfD algorithm and a hierarchical decision-making framework have been introduced. Results indicate that the RL-driven policy performs competitively with or surpasses human drivers in safety and energy consumption. The DRQfD method improves learning efficiency compared to pure DRL for high-level lane-changing policy. Comprehensive comparisons with IL reveal superior safety, travel efficiency, and human-likeness. Generalization tests on real traffic data show success. And, hardware platform evaluation confirms consistent behaviours with simulation, suggesting significant potential for practical application in DRL-driven AVs.

Finally, a novel scalable decision-making framework has been proposed for AVs at unsignalized intersections, integrated with game theory and interaction graphs. The framework considers the aggressiveness of drivers and uncertainties in a simplified model. The payoff function incorporates multiple driving features such as safety, travel efficiency, and aggressiveness. To address computational complexity, the adaptive decision-making framework incorporates switching directed graphs. The algorithm is validated on both hardware and a high-fidelity simulator, demonstrating robust adaptive decisions for AVs with interpretability, computational efficiency, and scalability.

## 6.2 Future Work

In this dissertation, the integration of MPC and DRL with game theory have been discussed in terms of optimizing long-term objectives for addressing the complex interaction problems faced by AVs. While the results are promising, we still identified several issues that require further exploration. The following are potential future research directions to enhance this work.

Firstly, in MPC, there is a trade off between computational complexity and the performance of the decision module (dependent on the prediction horizon of the future trajectories of surrounding vehicles). Specifically, if the prediction time is too short, the driving policy exhibits myopic behaviour, focusing solely on maximizing immediate returns, with the advantage of lower computational complexity. Conversely, with a longer prediction time, the driving policy aims for maximizing long-term cumulative rewards, aligning with human driving behaviour, but significantly increasing computational complexity and compromising real-time performance. Given the advantage of DRL algorithms in optimizing long-term cumulative rewards, it intuitively aids in addressing the computational complexity of MPC. However, effectively combining MPC and DRL for real-time optimization of long-term objectives and improving the learning efficiency of DRL is a meaningful research direction [84, 85]. This direction presents several challenges: (1) MPC optimizes based on a finite prediction horizon, and its solution may be locally optimal rather than globally optimal. Improving performance based on imperfect policies (e.g., MPC) is meaningful, especially for agents with lifelong learning capabilities [86, 87]. (2) DRL algorithms involve unnecessary exploration during interaction with the environment. Effectively guiding DRL to explore the environment and improving training stability are essential in future works. (3) Although building high-fidelity simulators can assist in transferring trained models to

real-world scenarios, it cannot completely eliminate the gap between simulation and reality. Investigating whether improvements in DRL efficiency and exploration could enable direct hardware training is an intriguing research direction [88].

Additionally, to enhance the safety of AVs, accurate prediction of the driving trajectories of surrounding vehicles is crucial for planning the optimal trajectory of the ego vehicle. Most current work on vehicle trajectory prediction focuses on short-term motion prediction [89], neglecting the impact of interactive behaviour between vehicles on future trajectories. Furthermore, as vehicles interact, they continually update their understanding of each other's driving preferences, making accurate identifying multimodality of driving behaviour of surrounding vehicles crucial [90,91]. Some recent works [92] attempt to design the decoder part using game theory in Transformer-based models for autonomous driving prediction and planning. This approach effectively considers the multimodal characteristics of driving strategies when predicting the trajectories of surrounding vehicles, showing promising driving performance. Another approach involves modeling the evolution of a complex dynamic driving environment through a world model [93–95]. Using the input of historical driving information (e.g., vision-based observational data) and employing AGI technique [96–98], this approach generates future driving scene videos, offering the potential for end-to-end autonomous driving.

# References


[1] T. Inagaki and T. B. Sheridan, "A critique of the sae conditional driving automation definition, and analyses of options for improvement," *Cognition, Technology & Work*, vol. 21, pp. 569–578, 2019.

[2] R. Bishop, "A survey of intelligent vehicle applications worldwide," in *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No. 00TH8511)*. IEEE, 2000, pp. 25–30.

[3] M. Yuan, J. Shan, and H. Schofield, "Scalable game-theoretic decision-making for self-driving cars at unsignalized intersections," *IEEE Transactions on Industrial Electronics*, 2023.

[4] M. Yuan, J. Shan, and K. Mi, "From naturalistic traffic data to learning-based driving policy: A sim-to-real study," *IEEE Transactions on Vehicular Technology*, 2023.

[5] A. Shariff, J.-F. Bonnefon, and I. Rahwan, "How safe is safe enough? psychological mechanisms underlying extreme safety demands for self-driving cars," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103069, 2021.

[6] R. Sparrow and M. Howard, "When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 206–215, 2017.

[7] O.-R. A. D. O. Committee, *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE international, 2021.

[8] A. Efrati, "Waymo riders describe experiences on the road," *URL https://www. theinformation. com/articles/waymo-riders-describe-experiences-on-the-road.(Accessed on 09/22/2020)*, 2020.

[9] G. Li, Y. Yang, *et al.*, "Risk assessment based collision avoidance decision-making for autonomous vehicles in multi-scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 122, p. 102820, 2021.

[10] G. Li *et al.*, "Deep learning approaches on pedestrian detection in hazy weather," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 10, p. 8889, 2019.

[11] H. Wang, Y. Huang, A. Khajepour, Y. Zhang, Y. Rasekhipour, and D. Cao, "Crash mitigation in motion planning for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3313–3323, 2019.

[12] M. Bosnak and I. Skrjanc, "Efficient time-to-collision estimation for a braking supervision system with lidar," in *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, 2017, pp. 1–6. DOI: 10.1109/CYBConf.2017.7985775

[13] S. Noh, "Decision-making framework for autonomous driving at road intersections: Safeguarding against collision, overly conservative behavior, and violation vehicles," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3275–3286, 2018.

[14] P. Hang, C. Lv, Y. Xing, C. Huang, *et al.*, "Human-like decision making for autonomous driving: A noncooperative game theoretic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2076–2087, 2020.

[15] H. Qi *et al.*, "Online inference of lane changing events for connected and automated vehicle applications with analytical logistic diffusion stochastic differential equation," *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103874, 2022.

[16] P. Hang, C. Huang, Z. Hu, and C. Lv, "Driving conflict resolution of autonomous vehicles at unsignalized intersections: A differential game approach," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5136–5146, 2022.

[17] N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz, "Game theoretic modeling of vehicle interactions at unsignalized intersections and application to autonomous vehicle control," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 3215–3220.

[18] M. Garzón and A. Spalanzani, "Game theoretic decision making based on real sensor data for autonomous vehicles' maneuvers in high traffic," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5378–5384.

[19] N. Li, Y. Yao, I. Kolmanovsky, E. Atkins, and A. R. Girard, "Game-theoretic modeling of multi-vehicle interactions at uncontrolled intersections," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[20] G. S. Sankar and K. Han, "Adaptive robust game-theoretic decision making strategy for autonomous vehicles in highway," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 484–14 493, 2020.

[21] R. Tian, S. Li, N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz, "Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 321–326.

[22] R. Tian, N. Li, I. Kolmanovsky, Y. Yildiz, and A. R. Girard, "Game-theoretic modeling of traffic in unsignalized intersection network for autonomous vehicle control verification and validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2211–2226, 2022.

[23] W. Wang *et al.*, "An intelligent lane-changing behavior prediction and decision-making strategy for an autonomous vehicle," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 3, pp. 2927–2937, 2021.

[24] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Springer, 2018, pp. 621–635.

[25] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[26] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta reinforcement learning for sim-to-real domain adaptation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2725–2731.

[27] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in

*2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[29] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[30] E. Candela, L. Parada, L. Marques, T.-A. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8814–8820.

[31] H. Shi, G. Liu, K. Zhang, Z. Zhou, and J. Wang, "Marl sim2real transfer: Merging physical reality with digital virtuality in metaverse," *IEEE Transactions on Systems, Man, and Cybernetics*, 2022.

[32] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in Neural Information Processing Systems*, vol. 1, 1988.

[33] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey on imitation learning techniques for end-to-end autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[34] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, "Off-road obstacle avoidance through end-to-end learning," *Advances in Neural Information Processing Systems*, vol. 18, 2005.

[35] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[36] J. Zhou, R. Wang, X. Liu, Y. Jiang, S. Jiang, J. Tao, J. Miao, and S. Song, "Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1450–1457.

[37] P. Cai, S. Wang, H. Wang, and M. Liu, "Carl-Lead: Lidar-based end-to-end autonomous driving with contrastive deep reinforcement learning," *arXiv preprint arXiv:2109.08473*, 2021.

[38] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, "Stabilization approaches for reinforcement learning-based end-to-end autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4740–4750, 2020.

[39] R. Cimurs, I. H. Suh, and J. H. Lee, "Goal-driven autonomous exploration through deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 730–737, 2021.

[40] M. Yuan, J. Shan, and K. Mi, "Deep reinforcement learning based game-theoretic decision-making for autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 818–825, 2021.

[41] G. Li, S. Li, *et al.*, "Continuous decision-making for autonomous driving at intersections using deep deterministic policy gradient," *IET Intelligent Transport Systems*, vol. 16, no. 12, pp. 1669–1681, 2022.

[42] D. Chen, L. Jiang, Y. Wang, and Z. Li, "Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 4355–4361.

[43] J. Wu, Z. Huang, Z. Hu, and C. Lv, "Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving," *Engineering*, vol. 21, pp. 75–91, 2023.

[44] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[45] J. Wu, Z. Huang, W. Huang, and C. Lv, "Prioritized experience-based reinforcement learning with human guidance for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[46] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[47] J. Wu, Z. Huang, C. Huang, Z. Hu, P. Hang, Y. Xing, and C. Lv, "Human-in-the-loop deep reinforcement learning with application to autonomous driving," *arXiv preprint arXiv:2104.07246*, 2021.

[48] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 921–928.

[49] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep Q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[50] T. Shi, P. Wang, X. Cheng, C.-Y. Chan, and D. Huang, "Driving decision and control for automated lane change behavior based on deep reinforcement learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2895–2900.

[51] K. Rezaee, P. Yadmellat, M. S. Nosrati, E. A. Abolfathi, M. Elmahgiubi, and J. Luo, "Multi-lane cruising using hierarchical planning and reinforcement learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1800–1806.

[52] V. A. Banks, K. L. Plant, and N. A. Stanton, "Driver error or designer error: Using the perceptual cycle model to explore the circumstances surrounding the fatal tesla crash on 7th may 2016," *Safety Science*, vol. 108, pp. 278–285, 2018.

[53] M. Yuan, J. Shan, and I. K. Sethi, "Game-theoretic decision-making for autonomous driving vehicles," in *Autonomous Vehicles and Systems: A Technological and Societal Perspective*. CRC Press, 2024, pp. 269–301.

[54] M. Yuan and J. Shan, "Learning adaptive cruise control for autonomous vehicles using end-to-end deep reinforcement learning," in *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2023, pp. 1–6.

[55] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[56] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[57] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[58] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1995–2003.

[59] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[60] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[61] S. Hagedorn, M. Hallgarten, M. Stoll, and A. Condurache, "Rethinking integration of prediction and planning in deep learning-based automated driving systems: a review," *arXiv preprint arXiv:2308.05731*, 2023.

[62] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 982–988.

[63] X. Jia, Y. Gao, L. Chen, J. Yan, P. L. Liu, and H. Li, "Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7953–7963.

[64] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki, "Multi-task learning with attention for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2902–2911.

[65] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li, "Think Twice before Driving: Towards scalable decoders for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 983–21 994.

[66] J. Wu, Y. Zhou, H. Yang, Z. Huang, and C. Lv, "Human-guided reinforcement learning with sim-to-real transfer for autonomous navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[67] X. He, J. Wu, Z. Huang, Z. Hu, J. Wang, A. Sangiovanni-Vincentelli, and C. Lv, "Fear-neuro-inspired reinforcement learning for safe autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[68] L. Wang, J. Liu, H. Shao, W. Wang, R. Chen, Y. Liu, and S. L. Waslander, "Efficient reinforcement learning for autonomous driving with parameterized skills and priors," *arXiv preprint arXiv:2305.04412*, 2023.

[69] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "NuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[70] V. Punzo, M. T. Borzacchiello, and B. Ciuffo, "On the assessment of vehicle trajectory data accuracy and application to the next generation SIMulation (NGSIM) program data," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1243–1262, 2011.

[71] M. Montanino and V. Punzo, "Making NGSIM data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction," *Transportation Research Record*, vol. 2390, no. 1, pp. 99–111, 2013.

[72] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transportation Research Part B: Methodological*, vol. 80, pp. 82–106, 2015.

[73] H. Alkomy and J. Shan, "Vibration reduction of a quadrotor with a cable-suspended payload using polynomial trajectories," *Nonlinear Dynamics*, vol. 104, no. 4, pp. 3713–3735, 2021.

[74] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[75] D. E. Rivera, M. Morari, and S. Skogestad, "Internal model control: PID controller design," *Industrial & Engineering Chemistry Process Design and Development*, vol. 25, no. 1, pp. 252–265, 1986.

[76] G. Gigerenzer and R. Selten, *Bounded rationality: The adaptive toolbox*. MIT press, 2002.

[77] Y. Wen, Y. Yang, R. Luo, and J. Wang, "Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning," *arXiv preprint arXiv:1901.09216*, 2019.

[78] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "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, pp. 1782–1797, 2017.

[79] B. M. Albaba and Y. Yildiz, "Modeling cyber-physical human systems via an interplay between reinforcement learning and game theory," *Annual Reviews in Control*, vol. 48, pp. 1–21, 2019.

[80] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[81] J. Qin, Q. Ma, Yu, *et al.*, "On synchronization of dynamical systems over directed switching topologies: An algebraic and geometric perspective," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5083–5098, 2020.

[82] H. Wang and J. Shan, "Fully distributed event-triggered formation control for multiple quadrotors," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 12, pp. 12 566–12 575, 2023. DOI: 10.1109/TIE.2023.3239870

[83] P. O. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[84] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," *arXiv preprint arXiv:2306.09852*, 2023.

[85] D. Sun, A. Jamshidnejad, and B. De Schutter, "A novel framework combining mpc and deep reinforcement learning with application to freeway traffic control," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[86] H. Hu, Z. Zhang, K. Nakamura, A. Bajcsy, and J. F. Fisac, "Deception Game: Closing the safety-learning loop in interactive robot autonomy," in *Conference on Robot Learning*. PMLR, 2023, pp. 3830–3850.

[87] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac, "ISAACS: Iterative soft adversarial actor-critic for safety," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 90–103.

[88] J. Luo, P. Dong, Y. Zhai, Y. Ma, and S. Levine, "RLIF: Interactive imitation learning as reinforcement learning," *arXiv preprint arXiv:2311.12996*, 2023.

[89] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2095–2104.

[90] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2605–2611.

[91] J. D. Ortega, N. Kose, P. Cañas, M.-A. Chao, A. Unnervik, M. Nieto, O. Otaegui, and L. Salgado, "DMD: A large-scale multi-modal driver monitoring dataset for attention and alertness analysis," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*. Springer, 2020, pp. 387–405.

[92] Z. Huang, H. Liu, and C. Lv, "GameFormer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving," *arXiv preprint arXiv:2303.05760*, 2023.

[93] X. Wang, Z. Zhu, G. Huang, X. Chen, and J. Lu, "Drivedreamer: Towards real-world-driven world models for autonomous driving," *arXiv preprint arXiv:2309.09777*, 2023.

[94] G. Rossolini, F. Nesti, G. D'Amico, S. Nair, A. Biondi, and G. Buttazzo, "On the real-world adversarial robustness of real-time semantic segmentation models for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[95] G. Zhao, X. Wang, Z. Zhu, X. Chen, G. Huang, X. Bao, and X. Wang, "DriveDreamer-2: Llm-enhanced world models for diverse driving video generation," *arXiv preprint arXiv:2403.06845*, 2024.

[96] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li, "LMDrive: Closed-loop end-to-end driving with large language models," *arXiv preprint arXiv:2312.07488*, 2023.

[97] C. Cui, Y. Ma, X. Cao, W. Ye, Y. Zhou, K. Liang, J. Chen, J. Lu, Z. Yang, K.-D. Liao, *et al.*, "A survey on multimodal large language models for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 958–979.

[98] D. Fu, X. Li, L. Wen, M. Dou, P. Cai, B. Shi, and Y. Qiao, "Drive like a human: Rethinking autonomous driving with large language models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 910–919.