6-22-2022

Cooperative Adaptive Cruise Control using V2V Communication and Deep Learning

Haoyang Ke
University of Windsor

Cooperative Adaptive Cruise Control using V2V Communication and Deep Learning

by

Haoyang Ke

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

Cooperative Adaptive Cruise Control using V2V Communication and Deep Learning

by

Haoyang Ke

APPROVED BY:

_____

J. Ahamed
Department of Mechanical, Automotive & Materials Engineering

_____

S. Erfani
Department of Electrical and Computer Engineering

_____

M. Saif, Co-Advisor
Department of Electrical and Computer Engineering

_____

S. Alirezaee, Co-Advisor
Department of Electrical and Computer Engineering

May 24, 2022

Declaration of Co-authorship / Previous Publication

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

Chapters 4 and 5 of this thesis were co-authored with Dr. Alirezaee, Dr. Saif, and Dr. Mozaffari, who provided supervision and guidance during the research and writing process. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes [*1*] original papers that have been previously published/submitted to journals for publication, as follows:

Thesis Chapter	Publication title/full citation	Publication status*
Chapter [4,5]	*H. Ke, S. Mozaffari, S.Alirezaee, and M. Saif, "Cooperative Adaptive Cruise Control using Vehicle-to-Vehicle communication and Deep Learning," 33rd IEEE Intelligent Vehicles Symposium, June 2022.*	*Accepted for publication*

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

This thesis presents a cooperative adaptive cruise control (CACC) system with integrated lidar and vehicle-to-vehicle (V2V) communication. Firstly, an adaptive cruise control system (ACC) is designed for the Q-Car electrical vehicle. Secondly, a CACC system with V2V communication function are designed based on a new algorithm for improving the ACC system traffic capacity performance. Lastly, the CACC agent was trained by Double Deep Q learning (DDQN) and tested. The proposed CACC system improved the stability of the vehicle. Experimental results demonstrate that the CACC system can decrease the average inter-vehicular distance of ACC by 44.74%, with an additional 40.19% when DDQN was utilized. The DDQN system can match the relative distance with the safety distance to have better distance control. In addition to simulation, experimental results on Q-cars have confirmed the same results. By implementing and testing the ACC and CACC system on the ego car, which follows a lead vehicle with an ACC system in a platoon, the CACC system has a better performance both on following the front vehicle speed and minimizing the difference between the safety distance and relative distance. As a result, the traffic capacity of vehicles can be improved. The vehicles communicate with each other through a WiFi module to transmit information with 2 ms latency.

Acknowledgements

Through my study journey, I have received much help, support, and advice. Firstly, I would like to apricate all the support from my co-advisor, Dr. S. Alirezaee, who guided me and provided access to Mechatronics Lav and its facilities which was needed for the experimental part of this study. He supported me mentally and financially and provided his expertise to help me finish my thesis. Furthermore, I would like to thank my co-advisor, Dr. M. Saif, who gave me valuable suggestions during my study. Then, I would like to thank Dr. J. Ahamed and Dr. S.Erfani for participating in my seminar and giving me constructive suggestion on my study.

Additionally, I would like to thank all the help and support from members of the Mechatronics Lab. A special thanks to Dr. S. Mozaffari, who helped me prepare my paper and thesis and was there to discuss any the difficulties that I faced.

I would also like to thank my family, especially my mother, who supported me all the way to finish my studies. Finally, I am grateful to my partner, Yidi Fu, for supporting and staying up with me. She always encouraged me through stressful and challenging periods.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADAS - Advanced Driver-Assistance Systems

ACC – Adaptive Cruise Control

ANFPC - Adaptive Neuro-Fuzzy Predictor-based Control

AGV - Automated Guided Vehicle

BD - BiDirecttional

CACC - Cooperative Adaptive Cruise Control

CTH – Constant Time Headway

CLQR - Constrained Linear Quadratic Regulator

DMPC - Distributed Model Predictive Control

DSRC - Dedicated Short-Range Communication

D2D - Device-to-Device

DDPG-PID - Deep Deterministic Policy Gradient and Proportional-Integral-Derivative

DQN - Deep Q-learning

DP - Dynamic Programming

DNN - Deep Neural Network

LQR - Linear Quadratic Regulator

LTE – Long-Term Evolution

LfD - Learning from Demonstrations

MDP - Makov Decision Process

PF - Predecessor Following

PLF - Predecessor-Leader Following

RL - reinforcement learning

TPF - Two-Predecessors Following

TPLF - Two-Predecessor-Leader Following

TD - Temporal Difference

UAV - Unmanned Aerial Vehicles

V2V – Vehicle-to-Vehicle

V2I – Vehicle-to-Infrastructure

V2P – Vehicle-to-Pedestrian

V2X – Vehicle-to-Everything

VFA - Value Function Approximation

WAVE - Wireless Access in Vehicular Environments

Wi-Fi – Wireless Fidelity

Chapter 1

Introduction

1.1 Objectives and Motivations

This thesis demonstrates the application of a Cooperative Adaptive Cruise Control (CACC) strategy in an actual application. As the electric car market expands to reduce carbon emissions, many countries are encouraging their citizens to switch to electric cars. According to statistics, in 2020, the global electric car stock hit the 10 million marks, a 43% increase over 2019 [1]. In addition, major ride-hailing companies, who provide personal drivers to send their customers to a destination, such as Lyft and Uber, have recently committed to shifting to 100% electrical vehicle fleets by 2030 [2]. This is indicative of the fact that this market has enormous potential, and its period of market growth has just started. New vehicle manufacturers, such as Xiaopeng and Tesla, are building electric cars to occupy the market. Traditional car manufacturers, mainly focusing on fuel engines, are launching their new electric vehicle platform. As an example, take General Motors, which published their new Ultium batteries and a flexible global platform in 2020 [3].

Meanwhile, with the development of autonomous driving and communication technologies (5G), the market offers a product with a combination of communication technologies, safer and more comfortable driving, and more energy efficiency. Furthermore, all the car manufacturers around the world are implementing autonomous driving in their future vehicle products. Nowadays, almost all the new current commercial vehicles offer Advanced Driver-Assistant Systems (ADAS), which mainly include Adaptive Cruise Control (ACC) system [4], Blind Spot Sensing system [5], Lane Change Collision Avoidance systems [6], Automatic Parking system [7], radar, and cameras on board to improve the energy efficiency, driving safety, and customer driving experience.

According to SAE Automation Grading (Figure 1.1), those systems provide an autonomous driving experience between level 2 and level 3 [8], where the vehicle can only sense a part of the environment and perform part of steering, acceleration, and deceleration. By combining these current onboard systems and sensors with communication

technologies, the automation level of vehicles can be upgraded to complete level 3, where the system can monitor the driving environment by itself through the communication modules and onboard sensors to perform all aspects of the dynamic driving tasks. To reach level 3, communication technologies are critical for the situation when vehicles need to communicate in a highly complex environment in an urban area and on highways. Especially, the 5G LTE technology significantly increases the capabilities of self-driving vehicles by communicating with the environment around them, while 4G LTE does not meet the mission-critical communications requirements, such as latency and capacity. Although the 802.11 meets those requirements, the necessary investment in a base station is a major obstacle for governments and manufacturers who are considering the cost and product balance [9]. Finally, the vehicle network is a communication network that includes everything around it, such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), and Vehicle-to-Everything (V2X) [10].

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.1 SAE Automation Grading [8]

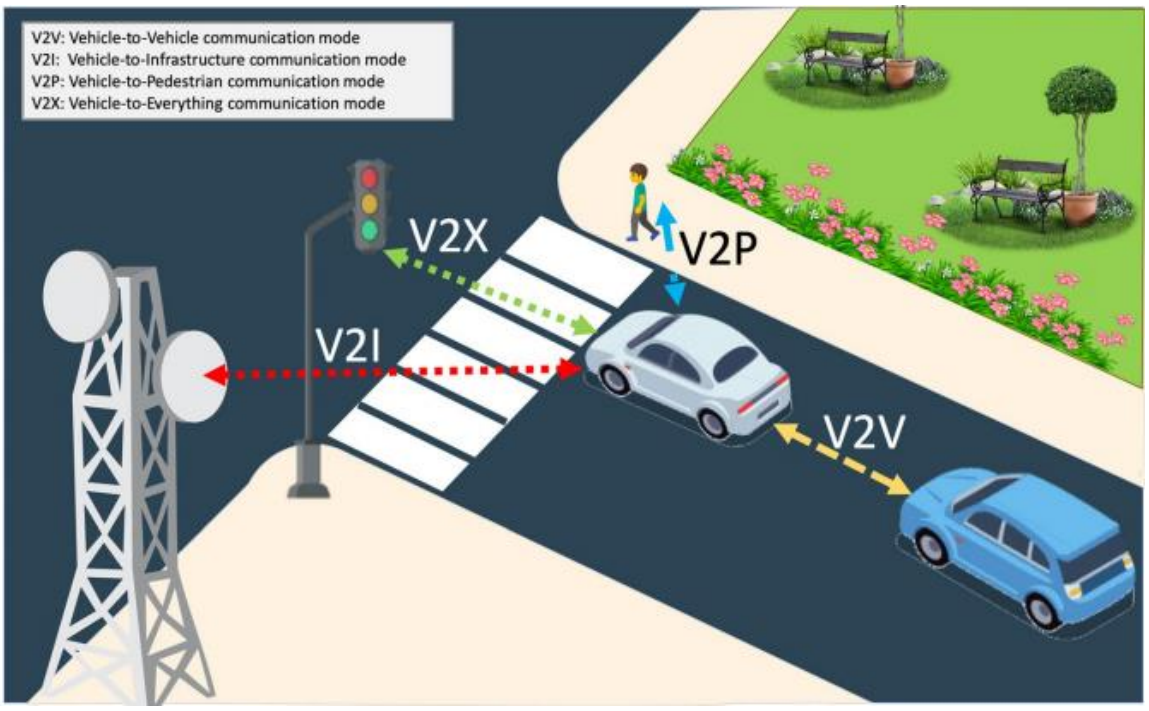

Figure 1.2 Connected vehicles network communication modes [10]

1.2 Problem Definition, Methodology and Challenges

This thesis will focus on implementing and testing a CACC, one of the platooning systems that apply V2V communication to improve vehicles' ability to monitor the environment and performance of their autonomous driving system.

CACC can be considered as an extension ACC system and is a subsystem within ADAS. The most significant part of ADAS is a well-developed and widely applied onboard system that uses onboard sensors and a speed control system to maintain the inter-vehicular distance at its desired distance and compute the desired acceleration (Figure 1.2) [10]. Through decades of study by researchers and automotive manufacturers, CACC safety, traffic flow stability and capacity have been investigated and developed [16] – [20]. However, certain limitations remain. Firstly, its performance is limited by the utilized sensors' sensing range and delay, which requires the vehicles to maintain a conservative inter-vehicular distance to ensure the vehicle's safety. However, the trade-off for this conservativeness is reduced efficiency and stability of the platoon. Additionally, since the ACC system requires many human interventions, traffic congestion is still a significant problem due to unpredictable and uncontrollable human driving behavior. As a result,

additional advanced technology and autonomous intelligence need to be incorporated in the system to address these issues.

Given that the CACC can be considered as a more intelligent and more sophisticated ACC system, it not only provides all the functions of an ACC system, but also, it would heavily rely on V2V communication to meet its additional objectives. In such a scenario, as illustrated in Figure 1.3, the information on the vehicle's status (e.g., position, velocity, steering, accelerations, and decelerations) is shared with other cars through the communication system. Moreover, when necessary, certain emergency alerts can be sent to others. In such a situation, since onboard sensors directly detect this information, there will be more accurate with less latency, and the system can be adapted to more complicated environments with better safety, capacity, and stability [21], [22]. The result would be reduced traffic congestion and improved platooning efficiency through a reduction in latency and inter-vehicular space. Furthermore, by enhancing the platooning efficiency, there can be a saving in fuel consumption with a reduction in the emission of greenhouse gases.

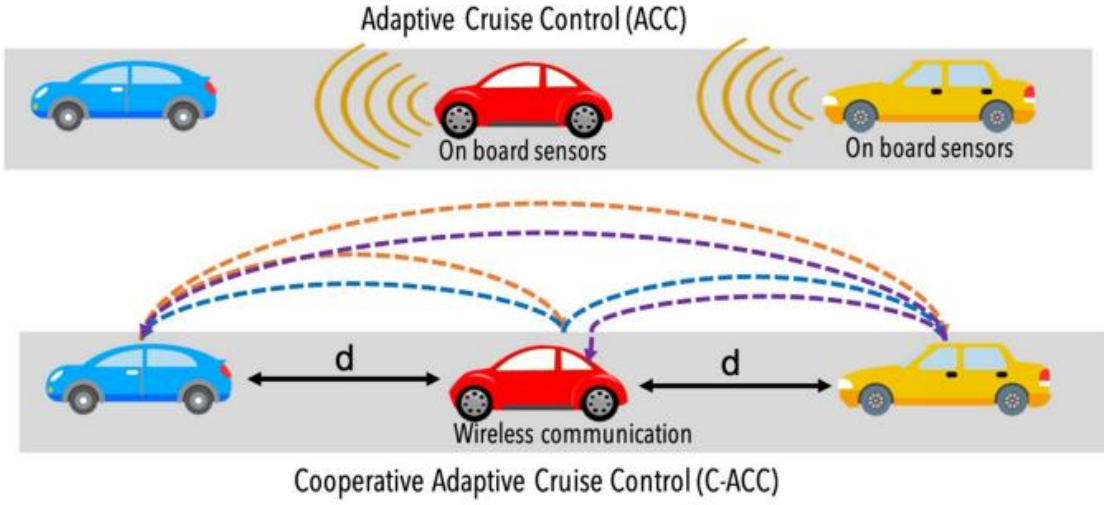

Figure 1.3 ACC and CACC in platooning system [10]

1.3 Contributions

In this thesis, CACC is implemented and tested on two Q-Cars, electrical-driven vehicle models with onboard sensors, GPUs, and wireless communication systems (802.11)

provided by Quanser. Compared with the current work, the main contributions and challenges of this thesis are as follows:

1) Based on lidar sensors and the V2V communication system, a new control algorithm is proposed for control accuracy.
2) Implementing ACC and CACC system on Q-Car models by using MATLAB Simulink.
3) A double Deep-Q-learning module is designed and trained based on the reward function and update policy on the MATLAB Simulink environment.

1.4 Outline of Thesis

The remainder of this thesis is organized as follows. Chapter 2 presents all the recent research in V2V communications, CACC and deep-learning techniques. Chapter 3 includes the theoretical background about reinforcement learning, deep learning and Q-learning. Chapter 4 presents the methods and algorithms that have been applied in this thesis, and all the experimental results are discussed in Chapter 5. Lastly, Chapter 6 presents the conclusions to this thesis and possible future works.

Chapter 2

Literature Review

In 1986, the PATH program was initiated at the University of California Berkeley to study Intelligent Vehicle Highway System for improving traffic [23]. Research studies on the subject have continued ever since, e.g., for some recent work on spacing strategy and algorithm, see [12], [13]. Frederic Serge firstly proposed the definition and assessment of ACC in 1991 [11]. The objective of the ACC system is not only to maintain the vehicle at the preset velocity but also to change the velocity control to the headway time control by controlling the brake and throttle when the front vehicle is too close or too slow [52]. In the ACC study, the selection and design of spacing policy is a significant part of the study. Many spacing policies and breakdowns have been proposed for reliability, safety, and stability, such as constant distance, constant time headway, constant safety headway [14], and constant stability spacing policies [15]. Nowadays, manufacturers are mainly applying the constant time headway (CTH) spacing policy in their ACC systems due to its safety and performance stability [24]. The longitudinal control system of ACC in gasoline vehicles is separated into a hierarchical control strategy consisting of an upper-level controller and a lower-level controller [25]. The upper-level controller is responsible for measuring the desired acceleration and velocity. The lower-level controller is responsible for controlling the throttle and brake to execute the order and give feedback to the upper-level controller.

2.1 Relative Work in CACC

CACC uses a V2V communication module to gather information for the ACC longitudinal controller with less delays than ACC. The first attempt to use V2V communication with IEEE 802.11p in platooning control was reported by Segate [28], who developed an integrated simulator called PLEXE for testing different platooning scenarios. Since the CACC system is the extension of the ACC system, sensors such as lidar or radar, and odometers are also installed on most vehicles with the CACC system. Additionally, the overall structure of the ACC and CACC systems, as illustrated in Figure 2.1, are similar [27]. Both ACC systems and CACC systems have a high-level controller, which

implements the longitudinal control algorithm, and a low-level controller, which implements the engine and break commands by following the instruction from the upper-level controller. Many research studies have concluded that the CACC system has a better performance with more accurate information.

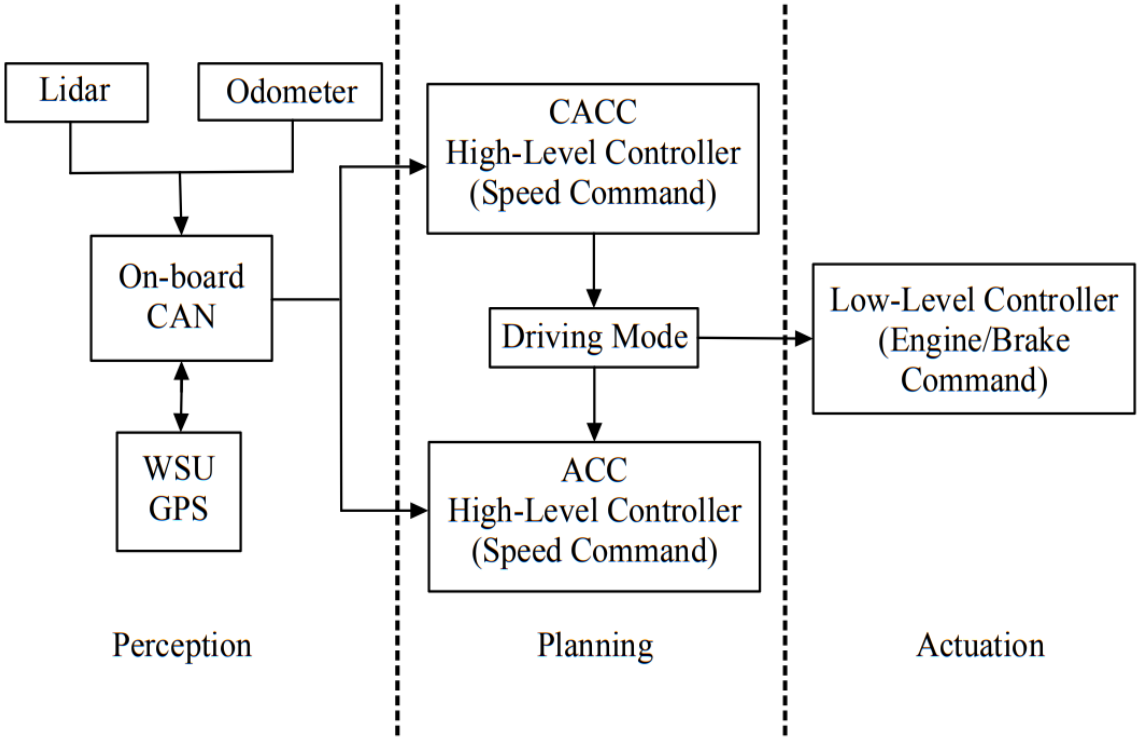

Figure 2.1 System structure of the CACC and ACC system embedded vehicle

To show the benefits that CACC can bring to the traffic flow, the authors in [59] used a traffic-flow simulation model MIXIC, designed to study the impact of the intelligent vehicles on traffic flow and data measured on a four-lane Dutch highway. The simulation is conducted for situations where different penetration rates exist between ACC and CACC vehicles and manually driven vehicles. They demonstrated that traffic-flow stability and efficiency could be enhanced by more CACC vehicles joining the traffic flow, especially in conditions with high-traffic volume. Additionally, authors in [62] developed an eco-CACC system, which can receive signal phasing and timing data using V2I communication. By using those data, each vehicle can arrive at the intersection whenever the last vehicle in

the queue has left. The result demonstrates that the vehicles can save up to 40% fuel when the market penetration rate is 100% using the ECO-CACC system.

Figure 2.2 Vehicle platoon in 2011 GCDC [29]

In recent years, a number of research studies reported successful implementation of CACC. These include Grand Cooperative Driving Challenge (GCDC) [29], as shown in Figure 2.2, California's Partners for Advanced Transit (PATH) [30] and the Safe Road Trains for the Environment project (SARTRE) [31]. In 2011 GCDC, they focused on primary platooning control, such as forming a two-line platoon and maintaining a platoon (Figure 2.2). In 2016 GCDC, there were two challenges: the cooperative platoon merge and the other was cooperative intersection passing. The aim was to present a close-to-reality scenario where cooperative and automated vehicles can perform complicated platooning operations. Even though vehicles in the platoon did not share the same algorithm, the competition still successfully demonstrated that cooperative driving was possible for different kinds of vehicles. The PATH is a comprehensive intelligent transportation system research in many subjects, such as CACC, Truck platoon, and

Cooperative Intersection Collision Avoidance Systems. Their systems simultaneously improved automotive safety and highway capacities. The SARTRE project aimed to minimize the environmental impact and traffic congestion and improve traffic efficiency and comfort for the driver in personal transportation in Europe [31]. They had applied the CACC system in the platooning of vehicles, where the lead vehicle with a professional driver controls all the following vehicles in a platoon. These platoons were estimated to cut 20% of the carbon emissions from private vehicles. In 2014, there was another real-world CACC application. The CACC system was implemented on four production Infiniti M56s vehicles and tested on public roads by authors [32]. Driving vehicles with ACC and CACC demonstrates that the inter-vehicular distances were significantly reduced compared to the commercially available ACC system by taking advantage of wireless communication information.

Additionally, many pieces of research about methodology and testing CACC under different scenarios were published. Lin used an adaptive neuro-fuzzy predictor-based control (ANFPC) with a Takagi-Sugeno fuzzy model to form the CACC system [33]. The Takagi-Sugeno fuzzy model is applied to estimate the preceding vehicle model and to obtain the predicted state sequence of it, and the following vehicle is controlled to maintain the inter-vehicular distance by using ANFPC. Their results from ANFPC are compared with performance under two other control algorithms: linear quadratic regulator (LQR) and constrained linear quadratic regulator (CLQR). Through comparison, they demonstrate that the string stability of vehicle platooning, comfort and fuel efficiency were improved significantly by the ANFPC-based CACC system. In a mixture of human-driven and autonomous vehicles, the platoon can maintain its stability by applying the Linear Quadratic Regulator (LQR) technique to a classic controller [35]. A stochastic, linear model predictive control strategy was present by [61]. They presented a simulation study in CarMaker and MATLAB about using this control strategy in the CACC system such that the fuel consumption was reduced by 11~15%. Moreover, Tugba applied the CACC algorithm to the vehicle platoons using distributed model predictive control (DMPC) based controller to improve the safety of a vehicle platoon [34]. A distributed consensus algorithm and protocol for the CACC system were designed for platoon formation, merging maneuvers, and splitting maneuvers [63].

Meanwhile, with the development of wireless transmission technology, the information can be transmitted to many vehicles simultaneously to control the vehicle platooning. There were many information flow topologies for V2V communication in a platoon, such as Predecessor Following (PF), BiDirecttional (BD), Predecessor-Leader Following (PLF), Two-Predecessors Following (TPF), Two-Predecessor-Leader Following (TPLF) [26] as shown in Figure 2.3. The arrow indicates the data transmission flow between vehicles. Different topologies have different communication requirements, such as data transmission speed, range and latency. In the actual application example, authors in [34] applied PLF and TPLF methods for their study to compare their delay. Additionally, they switched the communication topology from PLF and TPLF to PF and TPF in a scenario where the wireless communication dropout of leader information.

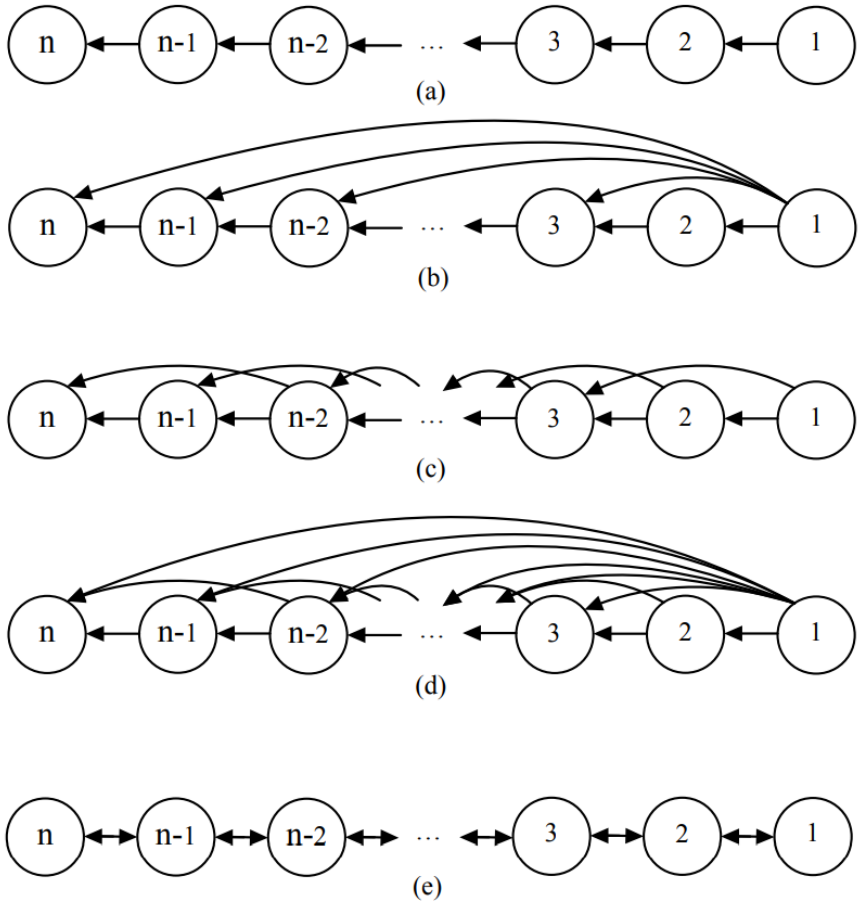

Figure 2.3 Typical information flow topologies: (a)PF, (b)PLF, (c)TPF, (d) TPLF, and (e) BD. [27]

2.2 Relative Work in V2X Communication

As mentioned before, V2X communication is a crucial element for CACC implementation, especially V2V communication. It decides whether vehicles can successfully exchange sufficient information (such as speed, position, direction, and acceleration) with each other. The U.S. National Highway Traffic Safety Administration (NHTSA) suggested Dedicated Short-Range Communication (DSRC) as the standard technology for V2V communication [36]. Due to the development of the technology, some new wireless communication protocols, such as Wireless Access in Vehicular Environments (WAVE), Long-Term Evolution (4G-LTE), and Fifth Generation (5G) technologies, were applied to V2V communication.

The DSRC is a vehicular communication protocol based on the IEEE 802.11p standard. There are many organizations involved in the formulation of DSRC standards, such as the International Organization for Standards (ISO), the European Committee for Standardization (CEN), the European Telecommunication Standards Institution (ETSI), and the Japanese Association of Radio Industries and Business (ARIB). Two other DSRC standards include TC204, which ISO published, and TC278, which CEN has published. Many car manufacturers were already installing IEEE 802.11p equipment in their vehicles.

On top of that, WAVE (Wi-Fi) also uses IEEE 802.11 standard for its physical layer, but it also supports V2I wireless communication by using WAVE Short Message Protocol [37]. Wi-Fi is a wired Ethernet network that uses wireless technology for data transmission. The radio frequency of Wi-Fi is the same as Bluetooth, and as such, the transmission range is limited. As a result, this protocol is mainly applied to the V2I communication, which requires a specific transmission data rate in short-range transmission. However, the Wi-Fi can have access to the World Wide Web, making it easier to access the database worldwide for getting information such as experience samples for deep learning. This worldwide connection may cause the vehicle more vulnerable to attack. In [64], the authors applied Wi-Fi and Zigbee networks for V2V communication in Bangkok, Thailand, and compared their performance under the same condition. Zigbee is another old protocol for short-range wireless communication with a low power consumption design such that its data transmission rate is relatively lower than Wi-Fi. The research demonstrated that the Wi-Fi

could have a 55% successful transmission rate, whereas the Zigbee only has an 8% successful transmission rate. Especially when the vehicle number increase, the Wi-Fi tends to have much better performance than Zigbee.

Nowadays, 4G-LTE is the most used cellular technology supporting Device-to-Device (D2D) to provide an ad hoc network service, which has a better performance on the range and latency than WAVE (Table 2) [38]. Its lowest latency can reach 10 ms. On top of that, it also has a maximum mobility speed of about 350 km/h. However, since it is widely applied to D2D communications, its communication would be easier to be interfered with by other mobile devices, especially in a metropolitan area. Additionally, the proximity discovery delay would be another issue that negatively affects the data transmission performance.

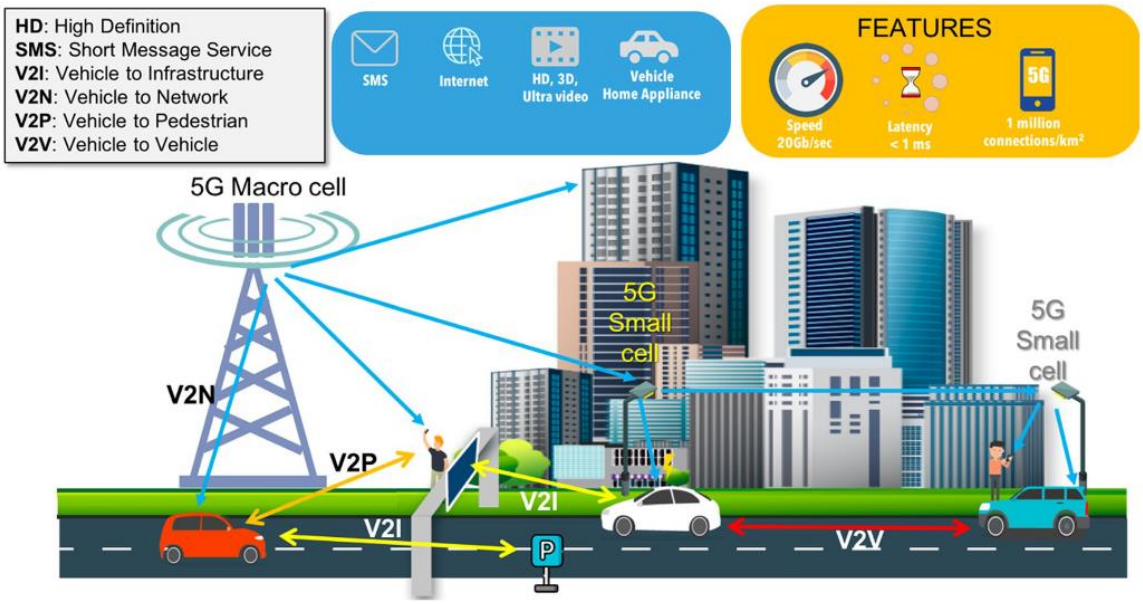

Figure 2.4 5G communication between vehicles to other facilities and devices [10]

The 5G technology is the best commercial wireless communication technology, with the highest data rate and lowest latency [39]. Moreover, it can simultaneously support around 1 million devices every square kilometer to exchange data [10]. As illustrated in Figure 2.4, this communication capability provides a solution for integrated communication between vehicles and other facilities and appliances within the vehicle environment. It provides three different communication modes to communicate with other

devices: device-to-device, device-to-cell, and device-to network. The 5G network slicing makes the end-to-end network possible to meet more specific wireless communication requirements. These three modes provide the communication channel in V2V and V2I and allow the vehicle to access the database from the cloud service. Especially in V2V communications, the 5G technology can be helpful. Its 1 ms latency can meet any real-time communication requirements, and its 1 Gbits/s downloading speed and 10 Mbits/s uploading speed are suitable for any kind of data transmission. Furthermore, it allows image data transformation between vehicles for future applications.

Besides these general data transmission technologies, there are other technologies in real applications. For example, in 2016 GCDC, they used the ITS-G5 specification for the Cooperative ITS, which works on the dedicated 5.9 GHz frequency band [29].

Table 2.1: Wireless Technology Comparison

Communication Technology	**Transferring Data Rate**	**Range**	**Latency**
Dedicate Short-Range Communication (DSRC)	6-27 Mbits/s	Medium	> 5ms
Wireless Access in Vehicular Environments (WAVE)	6-27 Mbits/s	Medium	50-100ms
4G-LTE	10 Mbits/s with peak of 1Gbits/s	Long	10-30 ms
5G	1 Gbits/s with peak of 20 Gbits/s	Long	>1ms

For V2V communication algorithms and models, the authors of [40] developed an application layer handoff method to enable heterogeneous network (Het-Net) communication between Wireless Fidelity (WIFI), DSRC, and LTE. This method increased the range of communication for V2V and V2I, but the latency limited its applications. In [41], the authors implemented a shadow fading model targeting system simulation, which separated the measurement data into three categories: line-of-sight(LOS), obstructed line-of-sight(OLOS) by vehicles, and no-line-of-sight due to buildings. Additionally, when a trusted vehicle cannot be reached, [42] proposed a Lidar-based authentication mechanism to detect surrounding vehicles through onboard sensors. In [34], the authors applied the DMPC algorithm to control a nonlinear vehicular platoon under

bidirectional topologies, demonstrating that the DMPC could process additional information without violating string stability.

2.3 Relative Work in CACC with Machine learning

In recent years, many projects have been developed based on a machine learning approach because it is a powerful tool that can be applied to many domains, such as data mining, imaging processing and cyber security. In an unpredictable environment that vehicle has to face, rather than re-programming the vehicle to be suitable for every kind of scenario, it would be easier if vehicles could learn their task autonomously or semi-autonomously through human interventions. Artificial intelligence is the most popular approach for robots' autonomy, making machine learning one of the most popular areas of study for automation.

There are many approaches and methods in machine learning, such as reinforcement learning, supervised learning, unsupervised learning, and semi-supervised learning [71]. Many were applied to robot systems, such as humanoid robots [72], unmanned aerial vehicles (UAV) [71], remotely underwater vehicles [73] and robotic arms [74]. In the field of intelligent vehicles, there are two kinds of learning approaches that can control the vehicle to make the right decision: learning from demonstrations (LfD) and reinforcement learning (RL) [75]. The LfD is an approach that agents can learn from the examples of human demonstration to predict their output action based on input. Inverse reinforcement learning is one of the learning algorithms representing the LfD methods. However, in RL, the agent learns the policy by trying different actions in different scenarios (state). From their different action and state pairs, the reward can be computed. By maximizing their reward within Makov Decision Process, the agent can predict a policy that can fit their situations.

An adaptive control system that applies reinforcement learning was proposed in 2008 [47]. It uses Monte Carlo Reinforcement learning to develop a longitudinal adaptive control system for a detailed nonlinear longitudinal vehicle model. In [48], the author applied reinforcement Q-learning to the decision-making process during the automatic driving and car following. Recently, [49] proposed a Deep Deterministic Policy Gradient and Proportional-Integral-Derivative (DDPG-PID) controller to automate the PID weight

tuning process with longitudinal tracking control of vehicle platooning. Authors in [50] implemented a deep neural network that generates multimodal predictions of traffic agents around a truck platoon to improve the cut-in performance of trucks. Moreover, a multi-agent RL method was applied to CACC to achieve higher performance and faster convergence [51].

2.4 Chapter Summary

This chapter presented an overview of technologies that have been applied to CACC. Firstly, the development from ACC to CACC was introduced where CACC is a system developed from the ACC system through vehicle-to-vehicle communications. Then, a study about the performance of CACC in both large traffic flow and a single platoon was presented. Many research and projects demonstrated that the vehicle with a CACC system could have better platoon stability and safety with better energy and traffic efficiency than one which uses an ACC system and manual driving. Meanwhile, the methods that have been applied to the CACC system were introduced. Secondly, different wireless communications technologies were compared and discussed for real applications, which is significant for practical CACC applications. Lastly, machine learning approaches were discussed as potential candidates for use in an autonomous vehicle, and certain sample studies in this direction were discussed.

Although some attempts have been directed toward implementing CACC and ACC on cars via DSRC, no research has yet used Wi-Fi in real applications. Although Wi-Fi is not a communication technology that could be applied in a city environment, it is applicable for the transportation of autonomous driving vehicles and Automated Guided Vehicle (AGV) [26] for package distribution within factories' environment. Additionally, the proposed prototype helps demonstrate the data transmission process because other communication technologies, such as 5G can transmit the same amount of data with the same latency as Wi-Fi. This thesis attempts to fill this gap by developing a MATLAB Simulink model on Q-Cars and to test its reliability and stability performance.

Chapter 3

Theoretical Background

The ultimate object of this thesis is to apply deep Q-learning (DQN) to the CACC system for the Q-Car application. Deep Q-learning is one of the deep reinforcement learning algorithms from the machine learning field that apply deep learning models, such as neural networks, to the reinforcement learning (RL) algorithm. This chapter presents some of the most critical underlying structures of reinforcement learning, Q-learning and deep Q-learning.

3.1 Markov Decision Processes

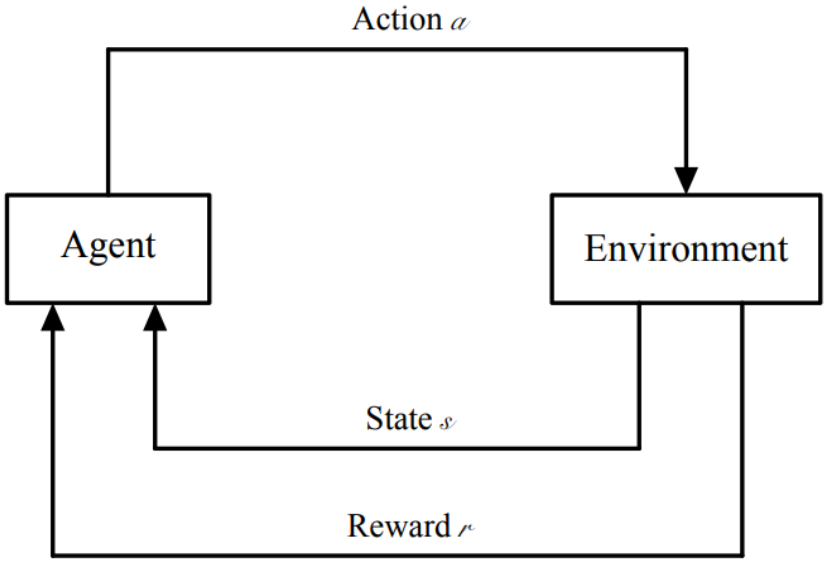

Figure 3.1 Markov Decision Process Model

To understand Q-learning, which is one of the reinforcement learning algorithms, the Markov Decision Processes (MDP), which are foundational in an RL environment, are discussed. MDP is a probabilistic temporal model which includes the agent's and environment's actions and reactions. The object which makes decisions and learns from decisions is called the *agent*. Everything else around it, which can be interacted with continuously and compared with, is called the *environment*. As shown in Figure 3.1, on the

one hand, the agent selects actions and learns from the environment's reaction. On the other hand, the environment takes action from the agent and responds to the agent by presenting new situations. Meanwhile, the environment also gives the agent a specific reward for its actions.

The MDP operates in many discrete time steps, $t = 0, 1, 2, \ldots, n$, and contains the following information:

- A set of finite n states, S, represents the different environments at different times.
- A set of finite n of actions, A, which is executed by the agent.
- A transition probability function, $T(s, a, s')$, represents the state transition probability from action, a, to the next state, s', such that $T(s, a, s') = P(s' \mid s, a)$
- A reward function $r : S \times A \rightarrow \mathbb{R}$, where R (s, a, s') represents the immediate reward in each state that can be used in the next state.
- A discount factor $\gamma \in [0,1]$ represents the importance of each immediate reward for the future.

As a result, the overall MDP consists of tuple $M = (S, A, P, R, \gamma)$. At each time step, t, the agent in the random state of the environment, $s_t \in S$, takes action $a_t \in A$. This action can take the environment to a new state, $s_{t+1} \in S$, at the time of $t + 1$. The probability distribution of the states is given by the transition probability function $T(s_t, a_t, s_{t+1}) = P(s_{t+1}|s_t, a_t)$, where $s_{t+1} = s'$, $s_t = s$, and $a_t = a$. Then, the reward function $R(s_t, a_t, s_{t+1})$ can calculate the reward in the current state s_t. Furthermore, the agent can execute an action a_{t+1} to observe the next state s_{t+1} and reward R_{t+1} which can be used for the next action and repeated during the process.

3.1.1 Policies and Values Function

In reinforcement learning, the agent's action is selected based on either a deterministic or stochastic policy of the action $\pi(s) = a$. In the deterministic stationary policy, the action in a given state always results in the same next state. However, in the stochastic stationary policy, the next state can be varied by the function that matches

different actions in probability distribution to each state. The optimal policy, π^*, is the theory such that the agent can maximize the expected reward by choosing different actions:

$$R = \sum_{t=0}^{\infty} \gamma^t R(s_t) \tag{3.1}$$

The decision network of a finite MDP, as shown in Figure 3.2, demonstrates the flow of the policy. At the very beginning, S_0 is generated by the environment, and an action A_0 is chosen to perform by policy π, where $A_0 = \pi(S_0)$. Then, everything can be transitioned to S_1, by receiving a transition probability $T(S_0, A_0, S_1)$ and a reward $R_1(S_0, A_0, S_1)$. The process can be continued by repeating the steps above, and a sequence of $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$ can be produced by the policy.

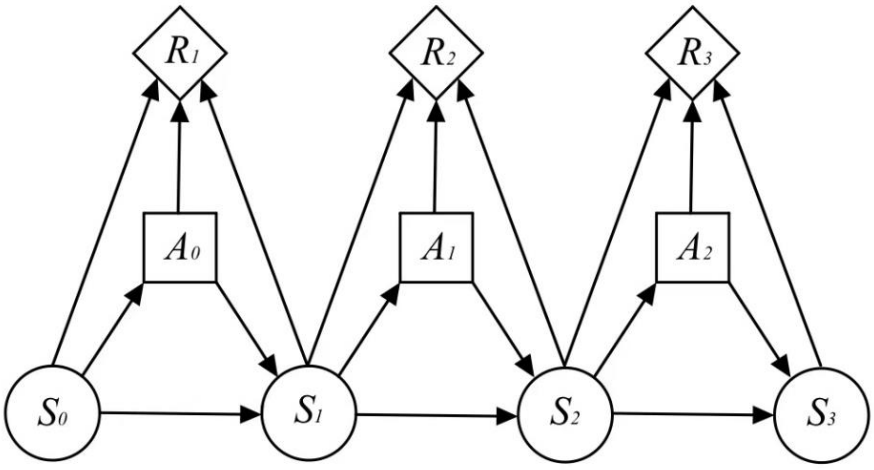

Figure 3.2 Finite MDP's decision network

As mentioned before, in all reinforcement learning applications the value functions, tells the agent how good the action is in the given state. The notion of "how good" here is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return. Of course, the rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular policies [54].

A policy π, is a mapping from each state, $s \in S$, and action, $a \in A$, to the probability $\pi(a|s)$ of taking action a when in state s [54]. As a result, a value function

$v_\pi\,(s)$ can represent the value of the state under the policy. For the MDPs, the state-value function $v_\pi\,(s)$ can be defined as follow:

$$v_\pi\,(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(s_t) \mid s_0 = S\right] \tag{3.2}$$

where $\mathbb{E}_\pi[\cdot]$ is the expected value of a random variable for the policy that the agent follows.

Based on this fundamental equation, the stationary policy's value function can be defined as:

$$\begin{aligned} V^\pi\,(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(S_t) \mid S_0 = s\right] \\ &= \mathbb{E}_\pi\left[R(s_0) + \sum_{t=1}^{\infty}\gamma^t R(S_t) \mid S_0 = s\right] \\ &= R(s) + \mathbb{E}_\pi\left[\sum_{t=1}^{\infty}\gamma^t R(S_t) \mid S_0 = s\right] \end{aligned} \tag{3.3}$$

Based on (3.3), by including the action *a*, state *s*, and transition probability function *T* to the value function of the stationary policy π they follow, the action-value function, $q_\pi\,(s,a)$ can be developed as follow:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t) \mid S_0 = s, A_0 = a\right] \\ &= R(s,a) + \mathbb{E}_\pi\left[\sum_{t=1}^{\infty}\gamma^t R(S_t,A_t) \mid S_0 = s, A_0 = a\right] \\ &= R(s,a) + \gamma\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t) \mid S_0 = s, A_0 = a\right] \\ &= R(s,a) + \gamma\sum_{s_{t+1}} T(s,a,s_{t+1})V^\pi(s_{t+1}) \\ &= R(s,a) + \gamma\sum_{s_{t+1}} P(s_{t+1}|s,a)V^\pi(s_{t+1}) \end{aligned} \tag{3.4}$$

In the finite MDPs, there is always an *optimal policy* π^*, a policy π has a return that is greater or equal to π' for all states. Since all the policies share the same state-value function, the optimal state-value function can be defined as:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \qquad s \in S \tag{3.5}$$

Additionally, when the optimal policy shares the same action-value function, the optimal action-value function can be formed:

$$Q^*(s) = \max_{\pi} Q^{\pi}(s,a), \qquad s \in S\,, a \in A \tag{3.6}$$

Based on the Bellman Equations [54], which demonstrate the value function is an immediate reward with the discounted sum of the future rewards, we have:

$$V^{\pi}\,(s) = R\big(s,\pi(s)\big) + \gamma \sum_{s'} T(s,\pi(s),s')V^{\pi}(s') \tag{3.7}$$

$$Q^{\pi}(s,a) = \; R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^{\pi}(s') \tag{3.8}$$

The optimal value function for the state-value function and action-value function can be arrived at:

$$V^*(s) = \max_{\pi} \left[R\big(s,\pi(s)\big) + \sum_{s_{t+1}} P(s_{t+1}|s,a)V^*(s_{t+1}) \right] \tag{3.9}$$

$$Q^{\pi}(s,a) = \; R(s,a) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s,a) \max_{b \in A} Q^*\,(s_{t+1},b) \tag{3.10}$$

3.2 Dynamic Programming

Dynamic Programming (DP) is a tool that can be used for computing optimal policies in a Markov decision process. The use of dynamic programming is based on the

assumption of a perfect environment. The perfect environment is a finite MDP, which means all the states, actions, and rewards are finite. As a result, DP is classified as a model-based learning algorithm. The object of dynamic programming is to use value functions to help the agent to search for a good policy by bootstrapping the rest of the expected return by the initial estimated value of the value function. Generally, there are two ways to compute the optimal policy: policy iteration and value iteration [54].

3.2.1 Policy Iteration

The idea of policy iteration is that there always is a policy that has a known value for the infinite horizon, and the policy has been incrementally improved by the algorithm. The policy iteration involves two main steps: policy evaluation and policy improvement. For the policy evaluation, the value, V^{π_i}, of the current policy π_i is computed by the value function. Based on the value from the policy evaluation, the new policy π_{i+1} can be computed by the policy improvement step. By repeating these two steps, a sequence of monotonically improving policies and value functions can be formed as [54]:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} V^{\pi_2} \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} V_*$$

where *E* represents the policy evaluation process, and *I* represent the policy improvement process.

By following the sequence, Algorithm 1 demonstrates the procedure of the policy iterations. At the beginning of the policy iteration, the policy is randomly selected. The policy evaluation calculates the action value V^{π_i} of the policy π_i by value functions that satisfy the Bellman Equation (3.7 and 3.8). Then, the new policy π_{i+1} can be formed by the policy improvement process with the equation:

$$\pi_{i+1}(\mathrm{s}) = \underset{a}{\operatorname{argmax}} \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^{\pi}(s') \right] \qquad (3.11)$$

where $\pi_{i+1}(\mathrm{s}) \geq V^{\pi_i}(s)$. As a result, the optimal policy can be found when $\pi_{i+1}(\mathrm{s}) = V^{\pi_i}(s)$.

Algorithm 1: Policy Iteration [54]

1. Initialization:
   $i = 0$,
   $V(s) \in R$ arbitrarily for all $s \in S$
   $\pi(s) \in A(s)$ arbitrarily for all $s \in S$
2. Policy Evaluation
   repeat
      $\Delta \leftarrow 0$
      For each $s \in S$
         $v \leftarrow V^{\pi_i}(s)$
         $V^{\pi_i}(s) = R\big(s, \pi_i(s)\big) + \gamma \sum_{s'} T(s, \pi_i(s), s') V^{\pi_i}(s')$;
         $\Delta \leftarrow \max(\Delta, |v - V^{\pi_i}(s)|)$
   until $\Delta < 0$
3. Policy Improvement
   policy-stable ← *true*
   For each $s \in S$
      $a \leftarrow \pi_i(s)$
      $\pi_{i+1}(s) = \underset{a}{\text{argmax}}[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^{\pi}(s')]$
      If $a \neq \pi_{i+1}(s)$, then *policy-stable* ← *false*
   If *policy-stable*, then stop and return V^{π_i}, and π_{i+1}; else go to 2

3.2.2 Value Iteration

Like the policy iteration, a value iteration is an alternative approach. It maintains optimal value from the beginning of a state *s* if a finite number of steps *k* exists. The advantage of the value iteration is that it can be literate to consider longer episodes for policies. Instead of doing the policy evaluation and policy improvement separately, it estimates the optimal value directly by turning the Bellman optimal equation into a backup rule.

The Bellman equation can be seen as a backup operator because it not only can be applied to an old value function but also can transform it into a new value function. Based on the Bellman equation (3.7 and 3.8), the equation of the Bellman backup operator can be formed as follow:

$$BV\ (s) = \max_{a} \left[R\big(s, \pi(s)\big) + \gamma \sum_{s'} T(s, \pi(s), s') V^{\pi}(s')\right] \qquad (3.12)$$

Algorithm 2: Value Iteration [54]

Initialize array V arbitrarily
repeat
$\Delta \leftarrow 0$
For each $s \in S$
$v \leftarrow V_k(s)$
$V_{k+1}(s) = \max_a [R(s,a) + \gamma \sum_{s'} T(s,a,s')V_k(s')]$;
$\Delta \leftarrow \max(\Delta, |v - (s)|)$
until $\Delta < 0$

Output a deterministic policy, π_{i+1}, such that
$\pi_{k+1}(s) = \operatorname{argmax}_a [R(s,a) + \gamma \sum_{s'} T(s,a,s')V_k(s')]$

Algorithm 2 describes the procedures of the value iteration with the Bellman backup operator, which returns a new value function and improves the value if it is possible in each iteration. Based on the Bellman backup operator, the main function in the second step can be expressed as follow:

$$V_{k+1}(s) = \max_a [R(s,a) + \gamma \sum_{s'} T(s,a,s')V_k(s')] \quad (3.13)$$

which means the value of V_{k+1} for that state is the sum of the best reward from the immediate action and the discounted sum of future reward with the old value function from the previous step, V_k.

In a real application, the operation terminates when there is only a small amount of the changes in the value function, as shown in the last step of Algorithm 2. As a result, the value iteration generates a sequence of value functions as:

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow \cdots \rightarrow V_*$$

3.3 Reinforcement Learning

Machine learning is one of the techniques that people apply to an agent to make it learn from a set of data such that it can solve a given problem that is related to the data. In machine learning, supervised learning is one of the most common algorithms that has been used to solve problems, such as in image classification. During supervised learning, each example has a label, which is provided by supervisors, and the environment can tell the

learner what output is based on its input and labelled examples. However, in reinforcement learning, as one of the machine learning algorithms, there is no such kind of examples for the algorithm to learn from.

Reinforcement learning is a generic framework for representing and solving control tasks, but within this framework, we are free to choose which algorithms we want to apply to a particular control task that can be modelled as MDP [53]. The RL focuses on a bigger picture, which is a problem of an agent for a specific object to interact with a changing environment. In contrast to dynamic programming, which assumes a perfect model of the environment, the RL does not have such an ideal environment available. As a result, reinforcement learning is categorized as a model-free learning algorithm. The RL can optimize the right track to reach the ultimate objects by giving each step reward. By maximizing the reward, the RL algorithm can learn what kind of action helps it to build reward and what kind of action lead to negative reward. As a result, the agent can understand what action it should take in different situations.

The biggest challenge of reinforcement learning is the trade-off between exploration and exploitation [54]. The reinforcement learning agent not only prefers actions that happened in the past and are effective for reward, but it also tries actions that have not been selected in the past. The agent must exploit what it already knows to obtain the reward, but it also must explore to make better action selections in the future [54]. Therefore, the reinforcement learning agent must try many different actions to get a reliable reward. As a result, to balance exploration and exploitation, an ϵ-greedy policy with respect to an action-state value $Q(s, a)$ is generated as:

$$\begin{aligned} \pi(a|s) &= \underset{a}{\operatorname{argmax}}\, Q(s,a)\ \ with\ probablity\ 1-\epsilon+\frac{\epsilon}{|A|} \\ \pi(a|s) &\neq \underset{a}{\operatorname{argmax}}\, Q(s,a)\ \ with\ probablity\ \frac{\epsilon}{|A|} \end{aligned} \tag{3.14}$$

where $|A|$ represents the number of actions.

There are four other significant elements for reinforcement learning besides the environment and the agent: *a policy*, *a value function*, *a reward signal*, and *a model* of the environment [54].

- A *policy* maps the action that an agent takes and the state of the environment when the action takes place. It could be a function, lookup table, and search process.
- A *reward signal* is a goal for the entire RL problem. During each iteration, the environment sends a reward signal to the agent to help the agent understand whether the action is good or bad at that specific state. Moreover, it also helps the agent to maximize the total rewards in the entire process such that the agent can learn which is the right set of actions that the agent should take for the best result.
- A *value function* is a function that helps the agent to cumulate the reward for the long run. The result of the value function directly tells the agent whether a set of actions is good in the long term and whether a set of states are involved. In each state, the agent receives a different reward signal for that particular state from the environment. In the end, all the reward signals will be cumulated together by a value function to come up with a result to determine the overall quality of the set of actions.
- A *model* of the environment defines the reaction behaviour that the environment would have based on each action. The model takes the state and action information and predicts the reaction from the environment. Furthermore, the reaction can be calculated as a reward signal to help the agent to determine the results of its actions.

Reinforcement learning has a very similar process to the MDP process shown in Figure 3.1. In each discrete time step, the agent, which is learning from the environment, and the environment interact with each other. Then, the observation can be collected by the agent from the environment, as well as the state s_t and a reward r_t. According to all this information, the agent takes an action a_t, which changes the environment. Lastly, the agent collects a new observation from the environment, a new state s_{t+1}, and a new reward s_{t+1}.

There are two kinds of methods in reinforcement learning ensuring that the actions selected by the agent are continuous: on-policy and off-policy. The off-policy methods are learning about a policy that has experienced different action selection. However, on-policy methods attempt to improve or evaluate the policy that the environment and agent currently follow, such as the Monte Carlo methods.

3.3.1 Monte Carlo Methods

Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns for episodic MDP tasks [54]. This method can be applied to the policy evaluation of the DP methods before. In the Monte Carlo method, the objective is to estimate $V^{\pi}(s)$ under policy π. Therefore, it defines and updates the $V^{\pi}(s)$ using a sample of the return to approximate an expectation by following policy π as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s] \tag{3.15}$$

where G_t is the discounted sum of the reward in MDP M under the policy as:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \tag{3.16}$$

Then, a simple every-visit Monte Carlo method for non-stationary cases can be generated as:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left(G_{i,t} - V^{\pi}(s) \right) \tag{3.17}$$

Note that α is a step-size parameter that measures the changes from one time step to another time step. Generally, the $\alpha = \frac{1}{N(s)}$ which refers to the case where the chance of every visit is equal, and $N(s)$ is the number of times that a state is visited. The sample episode is represented by $i = s_{i,1} + a_{i,1} + r_{i,1} + s_{i,2} + a_{i,2} + r_{i,2} + \cdots$.

Unlike dynamic programming, it is a model-free method which does not require to have a model of the reward and complete knowledge of the environment. It also does not

require bootstrapping because it is an average return of the sum of all the state-action pairs. Moreover, since it uses samples to make the approximation, the algorithm is not required to go to every state set for each evaluation. As a result, it is more efficient in the small subset of state.

However, there are some limitations to Monte Carlo method. Firstly, it is a high variance estimator which requires a lot of data to reduce the variance because it uses samples to make the approximation, and it does not update $V^{\pi}(s)$ based on other $V^{\pi}(s)$. Additionally, the Monte Carlo methods can only be applied to episodic MDPs, where each episode can be terminated no matter what actions are selected.

3.3.2 Temporal Difference Learning

Temporal Difference (TD) learning, one of the most important learning methods in reinforcement learning, combines Monte Carlo ideas and dynamic programming ideas [54]. It is a model-free learning method because dynamics models or reward models are not required in this method. TD learning not only can bootstraps, which updates estimates as a part of other estimates and approximates the future discounted sum of reward as dynamic programming, but also can sample to approximate the expectation in model-free conditions as the Monte Carlo method does. Additionally, it can be used in episodic or non-episodic settings and immediately updates the estimate of value function after each $M = (s, a, r, s')$ tuple when a new observation is generated. The objective of TD learning is to estimate $V^{\pi}(s)$ for the given episodes under the policy, and all the actions are sampled from the policy. Therefore, the most straightforward TD learning estimation equation can be generated based on the Bellman operator (3.7 and 3.8) and every-visit Monte Carlo method (3.17):

$$V^{\pi}(s) = V^{\pi}(s) + \alpha[r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s)] \tag{3.18}$$

This equation demonstrates that the TD learning updates the value estimate to approximate an expectation by using a sample of the next state and updates the estimation value by bootstrapping. Meanwhile, since TD learning is an estimation function, the TD

error can be generated as the difference between the new estimation and the current estimation value:

$$\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{3.19}$$

where $r_t + \gamma V^{\pi}(s_{t+1})$ is the new estimation value from TD learning and $V^{\pi}(s_t)$ is the current estimation value.

In TD learning, there are two kinds of control methods that represent on-policy control and off-policy control. SARSA [55], which stands for state-action-reward-next-state-next action, is one of the representations of on-policy, model-free control. It updates the state-action value $Q(s_t, a_t)$ from the current state, action, and reward and the next state and action, $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ using equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{3.20}$$

Meanwhile, the policy improvement will be updated based on ϵ-greedy policy:

$$\pi(s_t) = \underset{a}{\operatorname{argmax}}\, Q(s_t, a_t) \; with\; probability\; of\; 1 - \epsilon \tag{3.21}$$

Notice that the updates only happen after every transition from a non-terminal state.

Q-learning is another crucial method that represents off-policy TD control which also is a model-free RL algorithm as well. Instead of choosing a particular next action as SARSA does, the Q-learning is more optimistic by choosing the max action next to estimate future rewards. As a result, SARSA has a better performance where many negative rewards exist, and Q-learning has a better performance during the early convergence. The Q-value estimation equation can be updated as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{3.22}$$

where α is the learning rate, which indicates the updating frequency of the learning process between 0 and 1, and γ is the discount factor, which represents the weight of the future rewards is less than the immediate reward, between 0 and 1. This equation indicates that the optimal action-value function Q^* are directly approximated by the learned action Q. By using substituting the Q-value estimation equation, the algorithm of the Q-learning can be expressed as:

Algorithm 3: Q-learning algorithm [54]

Initialize $Q(s, a)$ arbitrarily, $t = 0$
repeat (for each episode)
    Initialize S
    Repeat (for each step of episode)
        Choose a from s using policy derived from Q (such as ϵ-greedy)
        $Q(s, a) \leftarrow Q(s, a) + \alpha(r_t + \gamma \max_{a'} Q(s', a') - Q(s, a))$
        $s \leftarrow s'$
    until s is terminal
until all episodes end

When the number of times visiting each state goes to infinity, the Q-learning will converge with probability one by bounding the reward $|r_n| \leq R$ and the learning rates $0 \leq \alpha_t \leq 1$ [57]. Meanwhile, the learning rate α_t need to satisfy the Robbins-Munro sequence such that:

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$
$$\sum_{t=1}^{\infty} {\alpha_t}^2 < \infty \tag{3.23}$$

On top of Q-learning, a Double Q-learning is another method using two different Q networks and one of the $Q(s, a)$ as a target for the other, which is very helpful for maximizing the bias issue and saving computation time. Based on the Q-learning algorithm, the double Q-learning algorithm can be formed as:

Algorithm 4: Double Q-learning algorithm [65]

Initialize $Q_1(s,a),\ Q_2(s,a)$ arbitrarily, $t = 0$
repeat (for each episode)
    Initialize S
    Repeat (for each step of episode)
        Choose a from s using policy derived from Q (such as ϵ-greedy)
        if (with 0.5 probability true) then
        $Q_1(s,a) \leftarrow Q_1(s,a) + \alpha(r_t + Q_1(s', arg\max_{a'} Q_2(s',a')) - Q_1(s,a))$
        else
        $Q_2(s,a) \leftarrow Q_2(s,a) + \alpha(r_t + Q_2(s', arg\max_{a'} Q_1(s',a')) - Q_2(s,a))$
        end if
        $s \leftarrow s'$
    until s is terminal
until all episodes end

3.4 Deep Reinforcement Learning

Deep reinforcement learning is one of the learning methods that use the deep learning method to apply to the reinforcement learning algorithm. All the reinforcement learning methods mentioned before using a table that includes all the state-action pairs and values. However, for a huge data subset or complex sensations, it is nearly impossible to have all the information in a table. Besides the issue of the need for memory, generalization, which is the time and data needed to fill the table accurately, is the most significant issue. To solve this issue, value function approximation (VFA) is essential such that policy can still help the agent to make good decisions when it faces a state-action pair that has never occurred before but with many similarities with other learned state-action pairs. In VFA, instead of using a table, a parameterized function is used to represent the state or state-action value function. The approximated value $V^{\pi}(s) \approx \hat{V}(s;w)$ of state s with vector w in the state value function. Meanwhile, the state-action value function can be written as $Q(s,a) \approx \hat{Q}(s,a;w)$. Through value function approximation, it not only reduces the memory that is needed to store but also reduces the computation time and number of experiences for the algorithm to find a good policy.

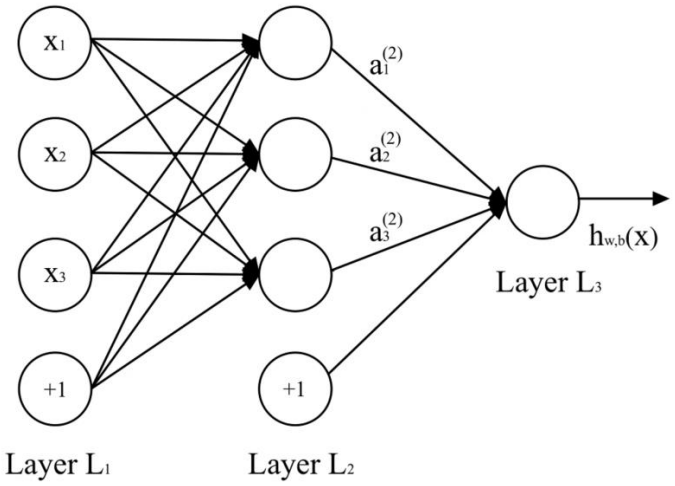

Figure 3.3 Simple neural network structure

The function approximation includes many methods, such as linear feature representations, decision trees, neural networks, and nearest neighbours. The deep neural network (DNN) is now one of the most popular methods for function approximation of reinforcement learning. In the DNN, the function approximator is transferred as a composition of multiple functions. A deep neural network is an artificial neural network with many layers, which induces the model to learn layered representations of input data [53]. According to the example in Figure 3.3, the first layer is the input layer, and the last layer is the output layer. The layers between the first and the last layers are the hidden layers. The connection between two neurons is the weight of the connection strength between neurons. The network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, where $W_{ij}^{(l)}$ represent the parameter associated with the connection between unit *j* in layer *l*, and unit *i* in layer *l+1*, and $b_i^{(l)}$ is the bias associated with unit *i* in layer *l+1* [58]. Additionally, $a_i^{(l)}$ is the activation of unit *i* in layer *l*, where $a_i^{(l)} = x_i$ for the *i*-th input.

For a fixed training set $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)} y^{(m)})\}$ of m training examples, the neural network can be trained by using batch gradient decent. Firstly, the cost function for a single training example (*x*, *y*) can be defined as [58]:

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \tag{3.24}$$

Based on this squared-error cost function, the overall cost function can be defined as:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W,b;x^{(i)}y^{(i)}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ij}^{(l)}\right)^2$$
$$= \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left\|h_{W,b}\left(x^{(i)}\right) - y^{(i)}\right\|^2\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ij}^{(l)}\right)^2 \quad (3.25)$$

The s_l represents the number of nodes in layer l. The first half of $J(W,b)$ equation represents an average sum-of-squares error, and the second half of it is a regularization term (weight decay term) that tends to decrease the magnitude of the weights and helps prevent overfitting [58]. λ is the weight decay parameter which controls the relative importance of two terms.

To train the neural network, firstly, each parameter $W_{ij}^{(l)}$ and each $b_i^{(l)}$ is initialized randomly with a value near zero, and then an optimization algorithm, such as batch gradient descent, is applied to the algorithm. The purpose of random initialization is symmetry breaking, which means to prevent all the hidden layer units learn the same function of the inputs.

One iteration of gradient descent updates the parameter *W*, *b* as follows [58]:

$$W_{ij}^{(l)} := W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b)$$
$$b_i^{(l)} := b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b) \quad (3.26)$$

where α is the learning rate and partial derivatives terms can be computed by using the backpropagation algorithm, which can minimize the $J(W,b)$, as follow [58] :

1. Perform a feedforward pass, computing the activations for layers L_2, L_3, and so on up to the output layer L_{nl}.
2. For each output unit i in layer n_l (the output layer), set

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(l)}) \tag{3.27}$$

3. For $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$
   a) For each node *i* in layer *l*, set

$$\delta_i^{(n_l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(n_l)} \right) f'(z_i^{(l)}) \tag{3.28}$$

4. Compute the desired partial derivatives, which are given as:

$$\begin{aligned} \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) &= a_i^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) &= \delta_i^{(l+1)} \end{aligned} \tag{3.29}$$

By using the Hadamard product, where $a = b \circ c$, then $a_{ij} = b_{ij} \cdot c_{ij}$, the desired partial derivatives for matrix-vectorial operation based on the algorithm above can be computed as:

$$\begin{aligned} \nabla_{W^{(l)}} J(W, b; x, y) &= \delta^{(l+1)} (a^{(l)})^T \\ \nabla_{b^{(l)}} J(W, b; x, y) &= \delta^{(l+1)} \end{aligned} \tag{3.30}$$

As a result, the full gradient descent algorithm can be expressed based on the (3.30). One iteration of batch gradient descent can be implemented as follows [58]:

1. Set $\Delta W^{(l)} := 0, \Delta b^{(l)} := 0$ (matrix/vector of zeros) for all *l*.

2. For i = 1 to m,
    a) Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$.
    b) Set $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$
    c) Set $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$
3. Update the parameters:

$$W^{(l)} := W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$
$$b^{(l)} := b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right] \tag{3.31}$$

where $\Delta W^{(l)}$ is a matrix with the same dimension of $W^{(l)}$, and $\Delta b^{(l)}$ is a vector with the same dimension of $b^{(l)}$.

The neural network can be trained by taking steps of gradient descent to reduce the cost function $J(W, b)$.

3.5 Deep Q-Learning

Deep learning models are a field in machine learning models. Deep neural networks (DNN) that have been applied to this thesis are one of the popular models along with the deep learning models. The reason for choosing this model is that it is the most accurate parametric machine learning model, which has a fixed set of adjustable parameters for a given task [53]. The DNN has the capability to break the complex data into atomic units and handle complexity with units. When Q-learning combines with DNN, it can minimize the mean-square error loss by stochastic gradient descent.

Additionally, it can solve problems that can be caused by other value function approximation methods, such as the correlation between samples and non-stationary targets. These two problems can have a negative effect when the learning agent estimates the right policy. The correlation between samples can make the process very time-consuming because the only way to get the observation is to try the action in different policies. The non-stationary targets would cause the estimated policy to be highly biased

with respect to the optimal policy. The DQN solves these two problems by using experience replay and fixed Q-targets.

In order to remove the correlation between samples, the DQN stores the dataset $D(s, a, r, s')$from prior experience so that it can use the data more than once for training. As a result, the target value for the sample s can be expressed as $r + \gamma \max_{a'} \hat{Q}(s', a'; w)$. By plugging in this target value function, the network weight can be updated by using stochastic gradient descent as:

$$\Delta w = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; w) - \hat{Q}(s, a; w))\nabla_w \hat{Q}(s, a; w) \tag{3.32}$$

Additionally, the stability can be improved by fixing the target weights. In DQN, a set of weights used in the target w^- and weights that are being updated w can be applied together to update the network weights. The target value for the sample s can be expressed as $r + \gamma \max_{a'} \hat{Q}(s', a'; w^-)$. By plugging in this target value function, the network weight can be updated by using stochastic gradient descent as:

$$\Delta w = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; w^-) - \hat{Q}(s, a; w))\nabla_w \hat{Q}(s, a; w) \tag{3.33}$$

As a result, the algorithm of DQN can be written as:

Algorithm 5: Deep Q-learning algorithm [65]

Input $C, \alpha, D = \{\}$, initialize $w, w^- = w$, t = 0
Get initial state s_0
loop
    Sample action a_t given ε-greedy policy for current $\widehat{Q}(s_t, a; w)$
    Observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer *D*
    Sample random minibatch of tuples (s_i, a_i, r_i, s_{i+1}) from *D*
    for j in minibatch do
        if episode terminated at step, $i + 1$ then
            $y_i = r_i$
        else
            $y_i = r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; w^-)$
        end if
        Do gradient descent step on $(y_i - \hat{Q}(s_i, a_i; w^-))^2$ for parameters w: $\Delta w = \alpha(y_i - \hat{Q}(s_i, a_i; w^-))\nabla_w \hat{Q}(s_i, a_i; w)$
    end for
    $t = t + 1$
    i**f** mod (t, C) == 0 then
        $w \leftarrow w^-$
    end if
end loop

More information about how to implement DQN will be introduced in the following chapter.

3.6 Chapter Summary

This chapter introduced the essential knowledge that is needed to understand and implement deep Q-learning. The foundation of all of the techniques discussed is the Markov Decision Process. Moreover, several classical MDP algorithm is introduced as well as the fundamental concepts of reinforcement learning and Q-learning. Lastly, the basic theory, advantages, and disadvantages of Deep RL and DQN were presented.

Chapter 4

Proposed Method

This thesis uses two Q-Cars, as illustrated in Figure 4.1, as models for implementing the CACC system. The Q-Cas is an electrically driven model with onboard CPU and sensors, including Lidar, RGB-D camera, and 360° CSI camera suite. The 2D planar lidar, which the CACC system uses for measuring inter-vehicular distance, supports up to 8000 samples per second, with scanning frequency up to 15Hz, and a sensing range of up to 18m [70]. The 360° CSI camera suite has four 8MP 2D CSI cameras, which can be applied to capture the image for line detection. Meanwhile, for using MATLAB to maneuver the Q-Cars, blocks from the QUANSER Simulink toolbox are used to build the Simulink Model.

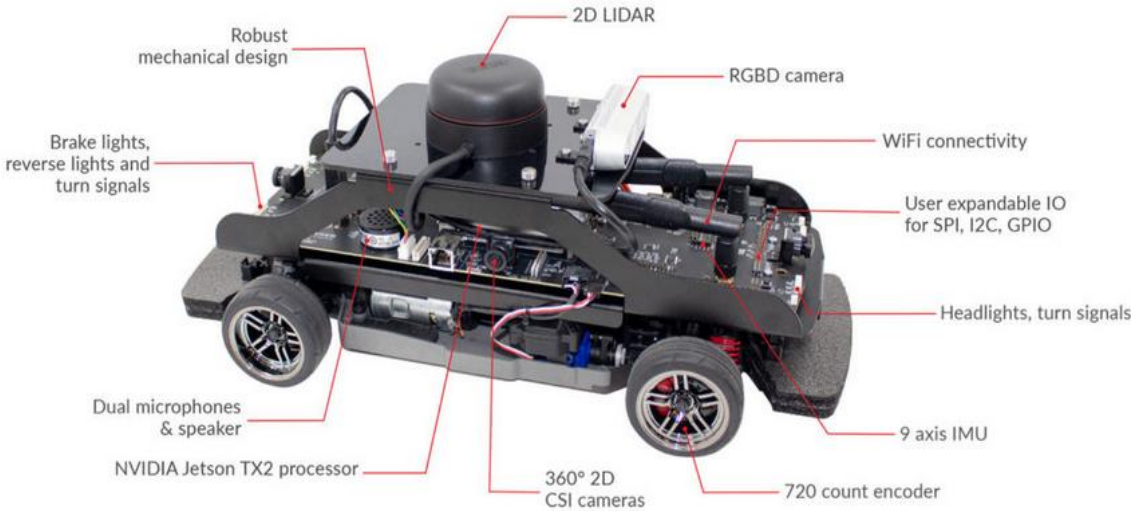

Figure 4.1 Q-Car platform components [70]

In order to present a CACC system in Q-Cars by MATLAB Simulink, an ACC system is designed first. Then, a communication channel is built up for Q-Cas to exchange information such as their speed and inter-vehicular distance. Moreover, a CACC system is designed based on the ACC and communication systems. Lastly, I built a CACC system based on deep Q learning on top of the original CACC system, and the performance of the three systems can be compared together.

4.1 ACC system

The core of an ACC system is its longitudinal control system. The architecture of the longitudinal control system is illustrated in Figure 4.1. In an autonomous vehicle with an ACC system, the ACC controller combines with its vehicle dynamics to perform a prototype of an ACC system. At the beginning of each step, the ACC controller module will read the inter-vehicular distance between the host vehicle and the vehicle in front of it. It will also read the host vehicle's speed through its onboard sensor. Based on these inputs, the ACC Controller will be able to calculate its desired speed and send it to the Vehicles Dynamics module to execute. The objective of the controller is to maintain the inter-vehicular distance as the preset tracking distance. Based on the flow chart, the ACC system of Q-Car can be designed and tested by the MATLAB Simulink.

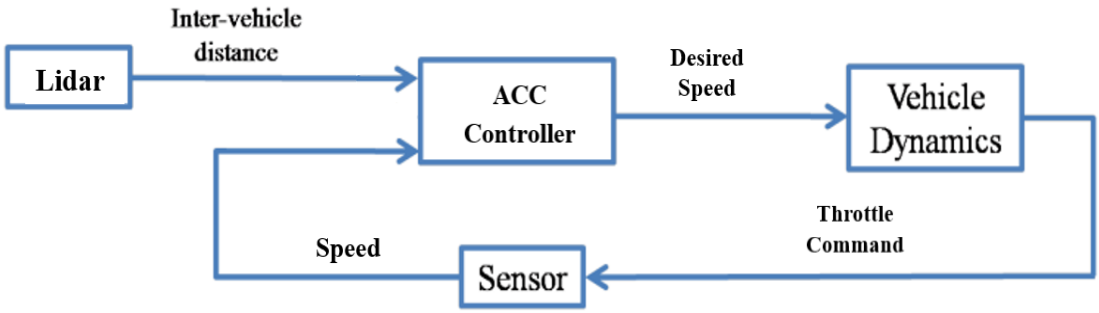

Figure 4.2 Architecture of ACC longitudinal control system

4.1.1 Lidar and line detection

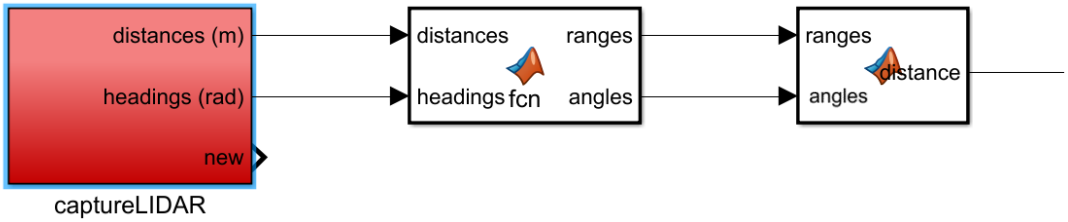

Figure 4.3 Lidar module for distance measurement

The objective of the Lidar system is to measure the inter-vehicular distance between the front car and the ego car, where the front car is the leading car in the platoon and the

ego car is the one that follows it. Figure 4.3 demonstrates the whole system that captures the lidar information around the vehicle and sends the distance information as an input for the ACC controller module. By using the captureLIDAR block from the QUANSER toolbox, the lidar data can be captured and transferred to the computer. The LIDAR on the car scans in a clockwise direction, and its 0 heading is a 270-degree clockwise direction of the actual Q-Cars' heading. To align the reference point of both Q-Car and LIDAR sensors, +270 degrees is added to the lidar sensor's default heading reference, and the heading signal is converted to radians for further calculation. Meanwhile, since the entire ACC system is required to respond to the environment frequently, the distance measurement should be updated frequently as well. Therefore, only 400 samples per revolution are collected to maximize the scanning frequency to 15Hz, which means the lidar can have 15 revolution scans per second.

An extra function block is added to the Lidar system to get the actual inter-vehicular distance. Since the measured distance is the distance between the objects and the lidar sensor, the distance between the front bumper of the Q-Car and the Lidar sensor, which is 0.26m, is reduced from the sensor's measurements to get a more accurate measurement that fits the Q-Car application. Meanwhile, in this function, it takes nine distance measurements, which include the measurements at 0 heading, four measurements beyond 0 heading clockwise, and four measurements beyond 0 headings counterclockwise. This function can find the minimum distance measurements from these measurements and send them as an input for the ACC controller module because the minimum value of these measurements can make the Q-Car operate based on the safest strategy. Additionally, since the lidar will return 0 if it does not sense any object, the 0 value will be excluded from the algorithm.

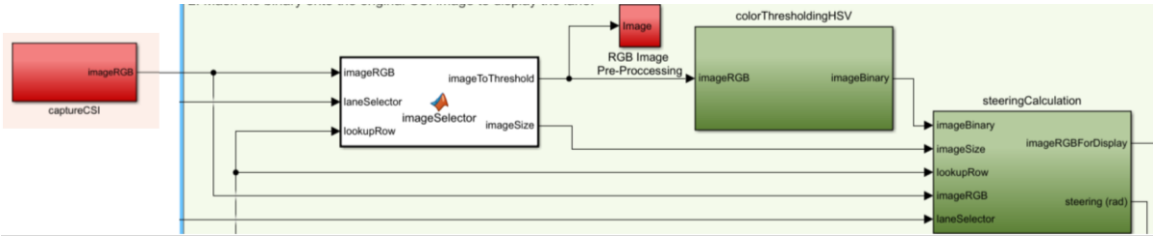

Figure 4.4 Line detection and steering calculation module

As shown in Figure 4.4, the line detection and steering calculation module is an additional module besides the ACC system. The objective of this module is to maintain the vehicle at the center of the road. This module uses four 360° CSI cameras to capture the images on the road. The *colorThresholdingHSV* blocks are responsible for detecting the yellow line in different lighting conditions, and the *steeringCalcuation* will generate the desired steering angle based on the threshold image.

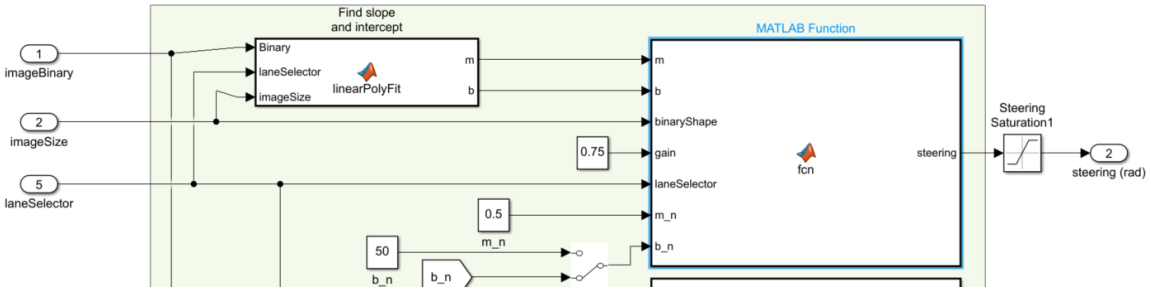

Figure 4.5 Steering calculation module

Figure 4.5 demonstrates the process of the steering calculation. The *linearPolyFit* uses `polyfit` function from MATLAB to generate a slop and y-intercept on the equation:

$$y = mx + b \tag{4.1}$$

where m is the slope, and b is the y-intercept value of the line. The MATLAB function after *linearPolyFit* uses *m* and *b* values to calculate the steering command up to ± 0.5 rad, which is the steering limitation of the Q-Car.

4.1.2 ACC Controller

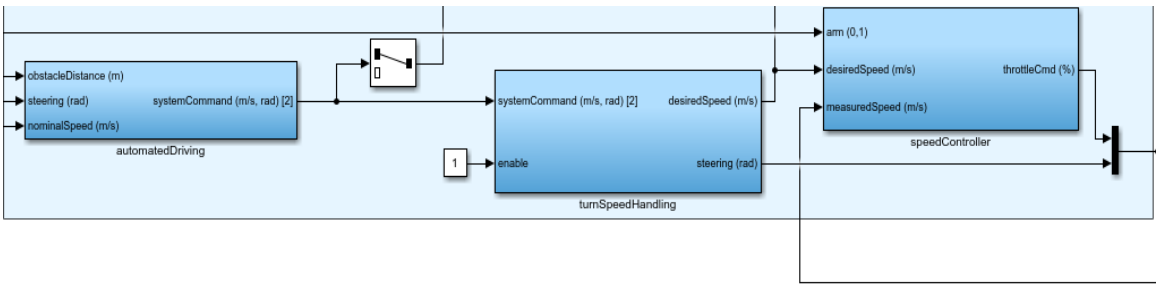

Figure 4.6 Longitudinal control module

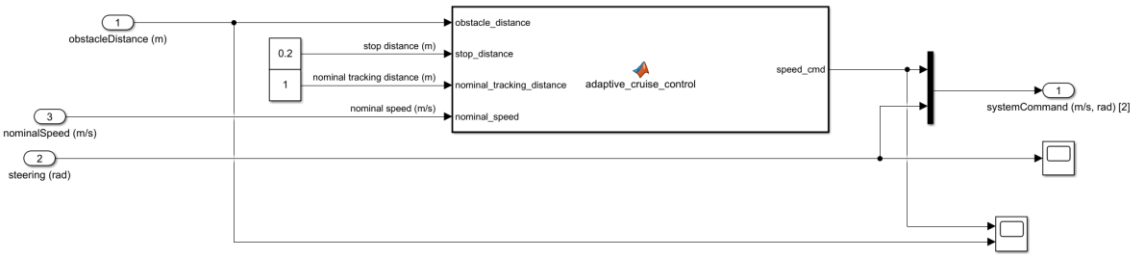

Figure 4.7 ACC controller module

The most common spacing policy applied to the ACC and CACC system by vehicle manufacturers and researchers is the Constant Time Headway (CTH) spacing policy [43], [44]. There are many studies about CACC or ACC systems that applied CTH spacing policy to their system [33], [45], [46], which illustrate its stability and reliability. The desired distance based on CTH is an increasing function of host vehicle speed, which can be expressed as follow:

$$d_r(t) = r + hv(t) \tag{4.2}$$

where $d_r(t)$ is the desired distance, r is the standstill distance, and h is the constant time headway time.

As shown in Figure 4.6 and Figure 4.7, the core module is the *adaptive_cruise_control*, which is included in the *automatedDriving* block. This module will take the obstacle distance measured by the lidar system, preset nominal speed, stop distance, and nominal tracking distance to compute the desired speed as a speed command to send to the *trunSpeedHeading* block. The equation of desired speed is expressed as follows:

$$V_{Desired} = \frac{V_{nominal}}{d_{tracking} - d_{dersired}}\left(d_{obstical} - d_{stop}\right) \tag{4.3}$$

where $V_{nominal}$ is a preset nominal speed which is the desired speed that a vehicle would maintain when there is nothing in front of it, as known as its cruise speed, $d_{tracking}$ is a preset tracking distance such that the speed of the vehicle will decrease when the inter-vehicular

distance is less than it, $d_{desired}$ is the desired distance calculated by equation (4.2). d_{stop} is a preset distance that the vehicle needs to stop, and n is the measured value from the onboard Lidar sensor.

The *turnSpeedHeading* block and *speedContoller* module are two functional modules which are provided by QUANSER. The *turnSpeedHeading* block is responsible for reducing the vehicle speed during the turning process. The Q-Car will go in the vertical direction of the centrifugal force due to the effect of inertia when the speed of the vehicle is too fast during turning. The speed reduction is limited between 50% and the cosine of the steering multiplied by the speed command calculated by the *adaptive_cruise_control* block so that the vehicle will not skid during the turning process. Finally, the desired velocity will be sent to the *speedController* module of the vehicle as input after each iteration. This module is a combination of feedforward and feedback controllers. The throttle command based on the PWM duty cycle and duty cycle bias of the electric motor will be sent to the Vehicles Dynamic module.

4.1.3 Vehicle dynamic module and sensor

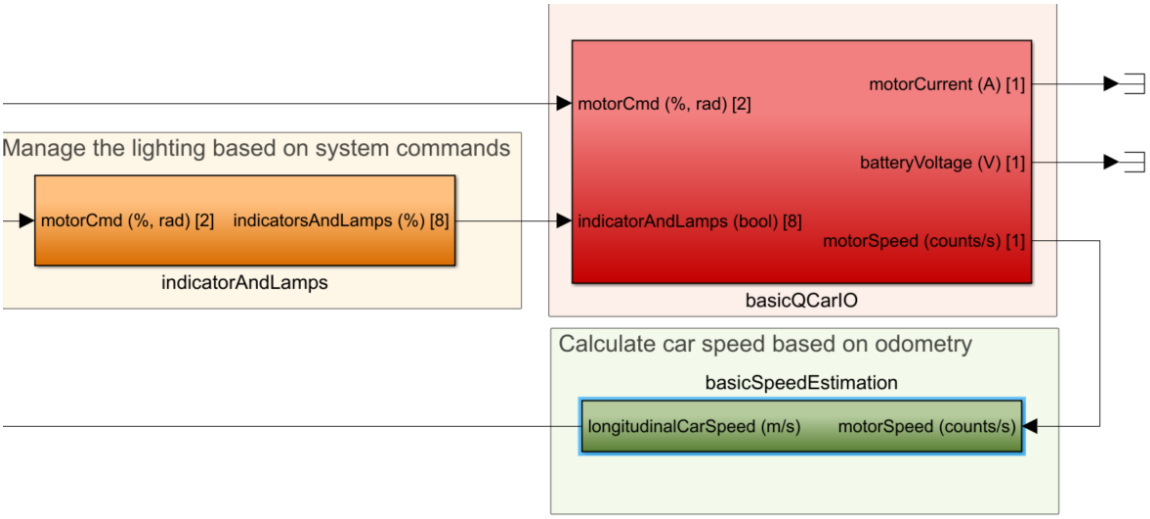

Figure 4.8 Vehicle dynamic module and motor sensor module

In Figure 4.8, the Vehicles Dynamic module and Sensor module of Q-Car, from QUANSER toolbox, are demonstrated. The throttle command, in percentage value, from *speedContorller* block and steering, in radiance value, from *turnSpeedHanding* block are combined and sent to *indicatorAndLamps* and *basicQCarIO* separately. The

indicatorAndLamps block sends a signal to the LED lights on the car during turning or braking based on the motor command. The *baiscQCarIO* is responsible for transferring the throttle command to the motor speed that the Q-Car platform can read.

The sensor module converts the motor speed to longitudinal car speed by using the gear ratios and wheel radius (0.0342m). The motor speed, measured by an encoder, is converted to shaft speed by multiplying the gear ratios. Then, the shaft speed can be computed to longitudinal car speed by using the equation:

$$v = 2 * \pi * r * \Omega \tag{4.4}$$

where r is the wheel radius and Ω is the shaft speed (rots/s).

4.2 CACC

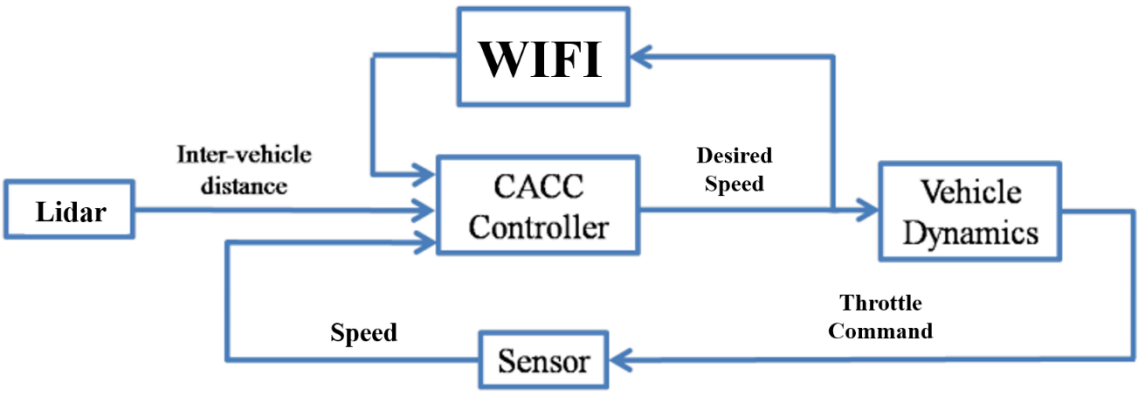

Figure 4.9 Architecture of CACC longitudinal control system

CACC system as an extension of the ACC system, an additional wireless communication module is required, which is responsible for the exchanging information between vehicles. In this case, an onboard Wi-Fi module is responsible for exchanging information. The architecture of the longitudinal control system is illustrated in Figure 4.9. In an autonomous vehicle with a CACC system, its CACC Controller works with its Vehicle Dynamics module to perform a prototype of a CACC system. At the beginning of each step, the Lidar sensor will send the obstacle distance (or inter-vehicular distance) to the CACC Controller. The onboard motor sensor will also send the current speed of the vehicle to it as well. Meanwhile, the Wi-Fi module will receive information from the front

vehicle, such as its speed and the obstacle distance from its lidar sensor. The desired speed can be calculated and sent to the front vehicle with obstacle distance through the Wi-Fi module according to these measured values. At the same time, its Vehicles Dynamics module will receive the command for further execution. As with the ACC system, the controller's objective is to realize and maintain the inter-vehicular distance with the calculated desired speed as well. Therefore, in the MATLAB Simulink platform design, the Lidar, Vehicle Dynamics, and the Sensor module are the same as the ACC system.

4.2.1 Vehicle to vehicle communication

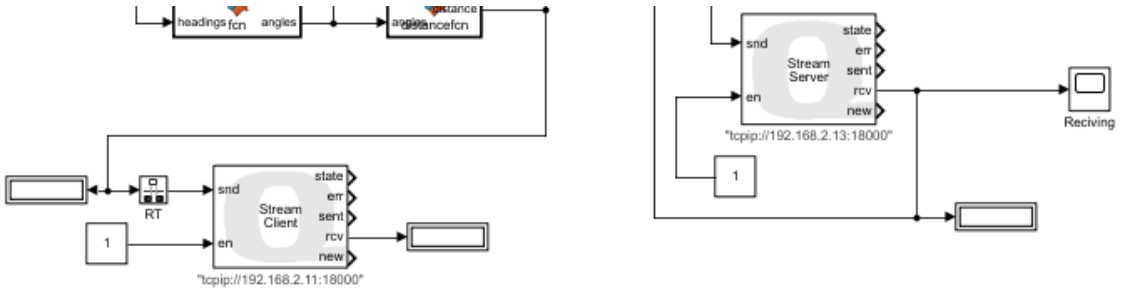

Figure 4.10 V2V communication module

Vehicle to vehicle(V2V) communication is one of the most significant functions of the CACC system. The Q-Car has onboard WIFI chips which support IEEE 802.11 a/b/g/n/ac protocols with dual antennas. Therefore, the V2V communication between Q-Cars can be established through the private router's wireless connection, as shown in Figure 4.10, which supports both 802.11 ac and 802.11 g. The Q-Car can use either the 2.4 GHz or 5GHz band, where the 5GHz band will have a better transmission rate but lower reliability due to the signal degradation. As a result, the 2.4 GHz channel is utilized in the CACC application. In order to send and receive the data through the ethernet, the block provided by QUANSER, which is called a steam client and steam server, is used to establish the communication. These blocks can send and receive data simultaneously and send it to other blocks in the system. The basic CACC system requires obstacle distance and vehicle speed data for further calculation, where the obstacle distance is measured by the onboard lidar sensor, and the vehicle speed is measured and calculated by the onboard motor speed sensor. Therefore, two sets of client and server blocks are used to send and receive these two sets of data, respectively.

4.2.2 CACC Controller

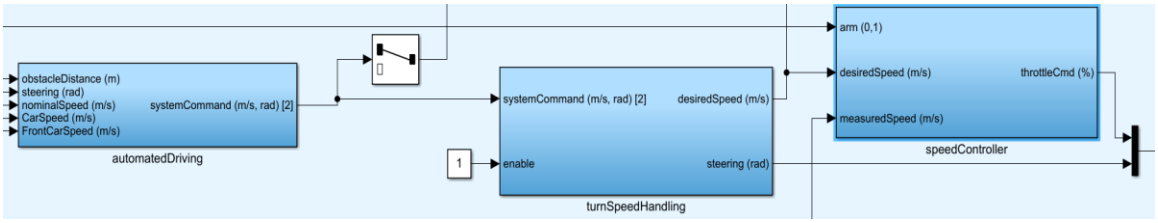

Figure 4.11 CACC longitudinal control module

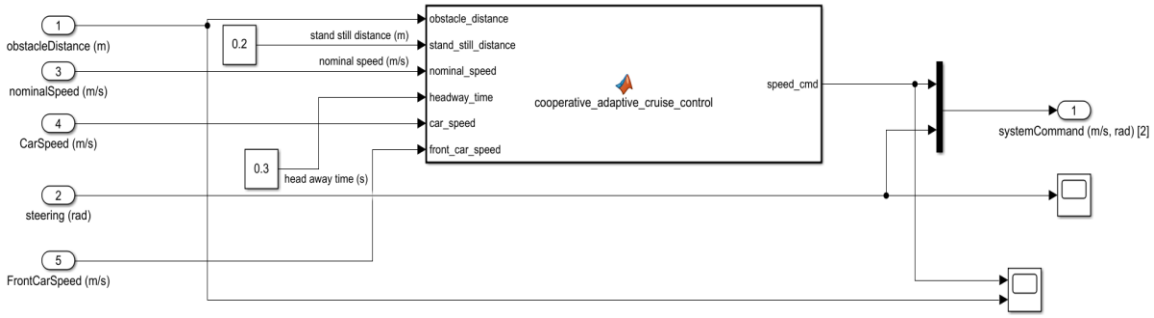

Figure 4.12 CACC controller

As shown in Figure 4.11, the CACC controller is very similar to the ACC controller except for the *automatedDriving* block, which is shown in Figure 4.12. On top of the ACC controller, it takes the host vehicle speed and front vehicle speed as two additional inputs to compute the desired speed of the host vehicle and sends it to the *turnSpeedHeading* block, where the front vehicles speed is directly transformed to the host vehicle through the V2V communication module and the host vehicle's speed is measure by its onboard motor speed sensor. The speed control algorithm not only involves the speed of the front and host vehicle but also is computed based on a new spacing policy.

The real-time front car and host car's speed can be used to compute a speed coefficient based on their value as follow:

$$\omega = V_{front} - V_{ego} \tag{4.5}$$

where V_{front} and V_{ego} are the speed of the car in front and the host car respectively.

The CACC system also applied CTH for distance control. Based on equation (4.2) and (4.3), a new equation of desired speed for Q-Car application is expressed as follow:

$$V_{Desired} = V_{nominal} \frac{(d_{obstical} - \mathrm{r})}{h * \omega} \tag{4.6}$$

where $V_{nominal}$ is a preset nominal speed which is the desired speed that a vehicle would maintain when there is nothing in front of it, as known as its cruise speed, $d_{obstical}$ is the measured value from the onboard Lidar sensor, *r* is the standstill distance, *h* is the constant time headway time, which is set to 0.3s in my case, ω is a car speed coefficient based on the real-time front and host vehicle speed. Additionally, the follower vehicle would follow the same speed as the front vehicle while the inter-vehicular distance is within $\pm$10% range of the desired distance or the ω equals zero.

Like the ACC system, the desired speed of the CACC system, that computed by *automatedDriving* module, will be sent to the *tunrSpeedHanding* module to finalize a final desired speed that is compromised with steering. Then, the *turnSpeedHanding* module will send the finalized desired Speed to the *speedController* to finalize a throttle command to the Vehicle Dynamics module to execute.

4.3 CACC with Deep Q-learning

The Q-learning is one of the RL algorithms that I choose for the CACC control task on the Q-Car. The reason why the RL algorithm is applied to this application is that in each step of the CACC problem, it is not known what the right decision is. However, by using the RL and giving each step rewards, it is possible to still come up and optimize the right track to reach the ultimate objective, which is to maintain the longitudinal distance between the vehicles. Q-learning is an off-policy model-free RL method that can handle stochastic problems. Q-learning algorithm is based on the Markov Decision Process (MDP), where a set of states *(S)* describes the position of the vehicle, a set of actions *(A)* indicates the acceleration or deacceleration action of the vehicle, and a set of observations *(O)* describes the inter-vehicular distance that is measured by lidar. A stochastic policy $\pi\,(s_t, a_t) = O_t \times A_t$, is used to select actions to produce the next state. Then, the reward function *r :*

$S \times A \rightarrow \mathbb{R}$ can be obtained from the state and vehicle's action. Based on the states, actions, observations and reward, the update of state-action value can be defined as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t) \right] \tag{4.7}$$

where α is the learning rate and can be set between 0 and 1. r_t is the reward, and γ is the discount factor that shows the weight of the previous reward.

4.3.1 State and Action Space

Since Q-learning is based on the MDP, the state and action space should be defined first. In the state space, three state variables are defined as follows:

$$H = \frac{S_{lead} - S_{ego}}{V_{ego}} \tag{4.8}$$

where H is the headway time which defines the inter-vehicular distance in the time domain. S_{ego} and S_{lead} are the position of the following vehicle and leading vehicle, respectively, and V_{ego} is the velocity of the following vehicle. Instead of measuring the inter-vehicular distance directly from their position, transforming the distance difference into time difference helps the ego car (the car follows the front car in a platoon) measure the inter-vehicular distance based on its current velocity.

$$V_{diff} = V_{lead} - V_{ego} \tag{4.9}$$

where V_{diff} is the velocity difference between the front and the following vehicle, combining it with the headway time can demonstrate whether the vehicle is close or far away from the previous state. Based on the velocity difference, the difference between the displacement of two vehicles, which illustrates if the ego car is getting closer or getting far away from the front car, can be computed by integration:

$$\dot{d_{diff}} = V_{diff} \tag{4.10}$$

The third variable in state-space is the front car acceleration:

$$S = \{H, d_{diff}, a_{lead}\} \tag{4.11}$$

Similarly, the action space can be defined with three actions: acceleration (AC), deacceleration (DAC), and no operation (NO). Acceleration and deacceleration are specified by two different levels, respectively: AC_a, AC_s, DAC_a, and DAC_s, such that the vehicle can have different strategies, which are aggressive and smooth, to respond to the action of the front vehicle to make a smoother control. The aggressive strategy has a larger acceleration or deacceleration compared to the soft strategy. Eventually, the action space can be described as follow:

$$A = \{AC, DA, NO\} \tag{4.12}$$

$$A = \{AC_a, AC_s, DA_a, DA_s, NO\} \tag{4.13}$$

4.3.2 Reward Function

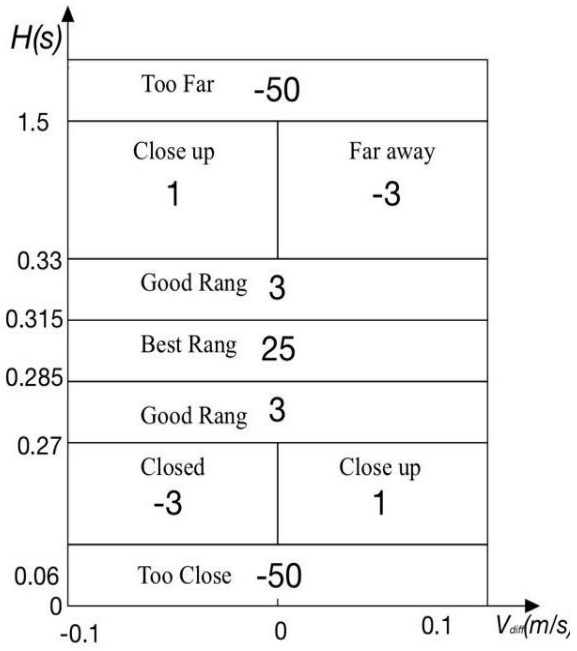

Figure 4.13 Reward function

After defining state and action spaces, the reward function should be determined to evaluate the performance of each action in the action space. Based on the state space and constant time headway time value in our case, the reward was designed as shown in Figure 4.13. The objective of the reward function is to force the agent to stick to the safety distance as close as possible. Positive reward values are given to the agent's actions at a safe inter-vehicular distance.

Different reward values are assigned to the different headway time ranges based on the ratio of constant headway time values, and they are also given to the positive and negative V_{diff} in different headway time zones. The pre-defined headway time is 0.3 seconds, which is ten times smaller than the pre-defined headway time in real-world (3 s), because the length of the Q-Car is 0.425 m, which is around ten times smaller than a general Sedan in the real world (4 m). A larger positive reward is given to the Q-Car when it enters the zone that is within $\pm$ 10% of the headway time, and more rewards are given when the Q-Car enters a more accurate zone, which is within $\pm$ 5% of the headway time. On the contrary, a negative reward is given when the inter-vehicular distance is too close or too far away from the safety distance. The negative reward is also given when the ego car is faster than the lead car and the relative distance is smaller than the safety distance, or when the ego car is slower than the lead car and the relative distance is larger than the safety distance. Additionally, an extra punishment is given to the situation where the inter-vehicular distance is out of sensing range and smaller than the standstill distance, as shown in (4.14).

$$r = -500 \quad d_{obstical} > 5 \; or \; d_{obstical} < 0.2 \tag{4.14}$$

4.3.3 Neural Network Design

In reinforcement Q-learning, a Q-table was applied to store all the states and action values. In this complicated case, the size of the Q-table will be too large to store and search. Therefore, a three-layer neural network with an additional input layer was designed for this application (Figure 4.14). The neural network includes an input layer, an output layer, and two hidden layers. Hidden layers have 120 neurons in each layer, which is a fully connected layer with the rectified linear unit (ReLU) activation function.

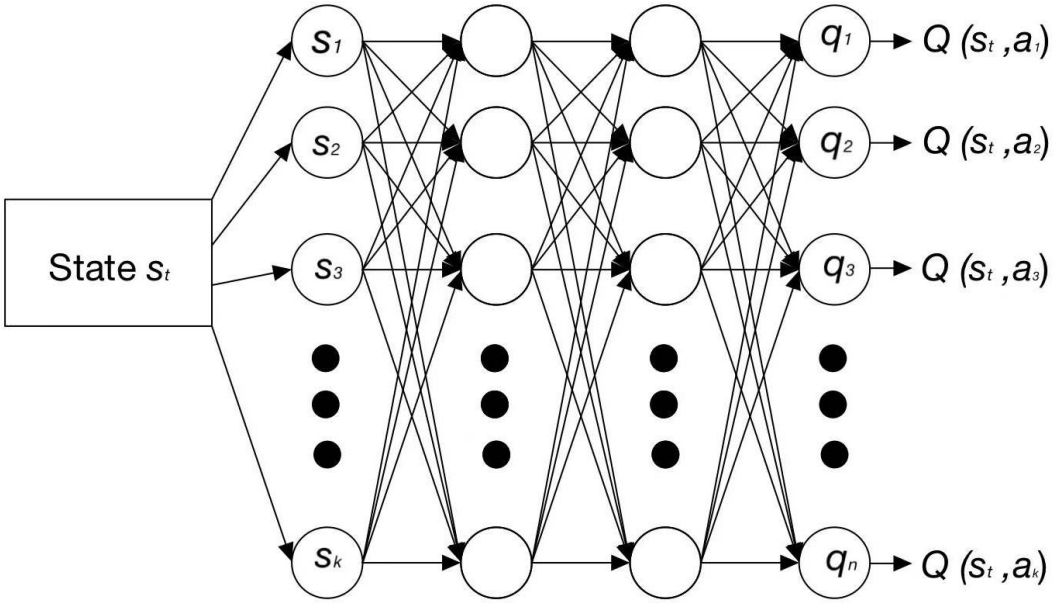

Figure 4.14 Four-layer neural network structure

4.3.4 Training Model

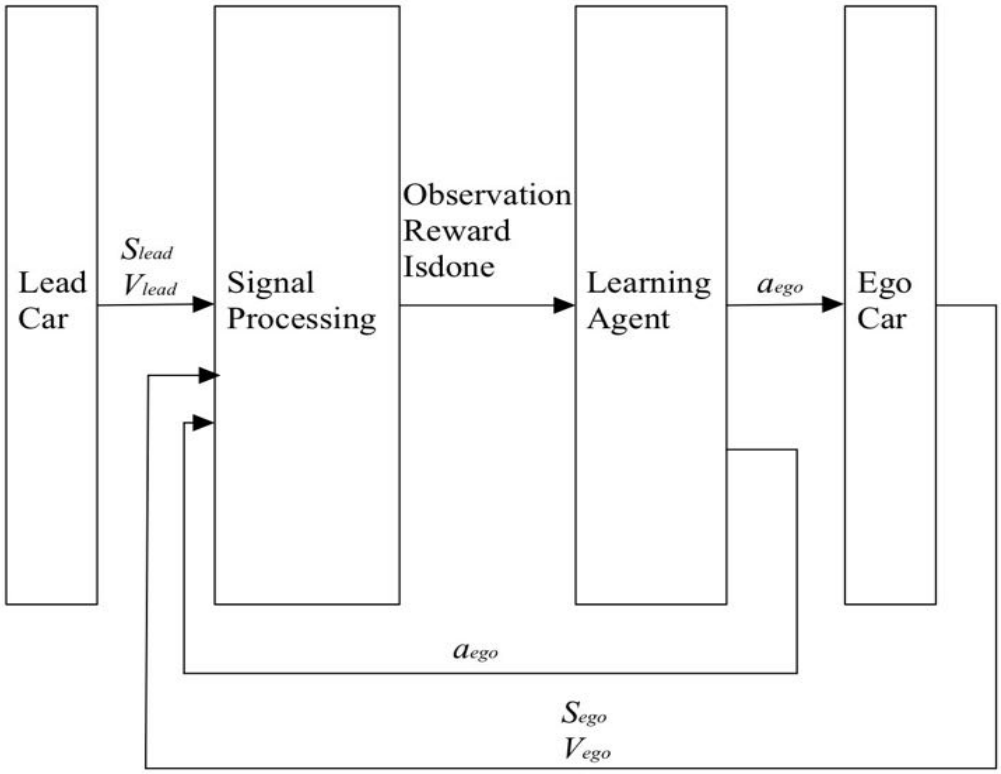

Figure 4.15 The framework of the training environment

A DQN training environment is set up on MATLAB Simulink (Figure 4.16) based on (Figure 4.15). In each state iteration, the lead car sends its position and acceleration to the signal processing module, where the Observation, Reward and Isdone results can be computed. The Learning Agent receives those data, updates the $\hat{Q}(s,a;w)$ and select an action according to the stochastic policy. The action selected from the action space will be sent to the ego car and Signal Processing module. Meanwhile, the learning agent is expected to adjust ego car speed according to the front vehicle speed variation to maintain a safe distance. To this end, the MATLAB Reinforcement Learning toolbox has been used. Lastly, the ego car module executes the action and sends its position and velocity to the Signal Processing Module.

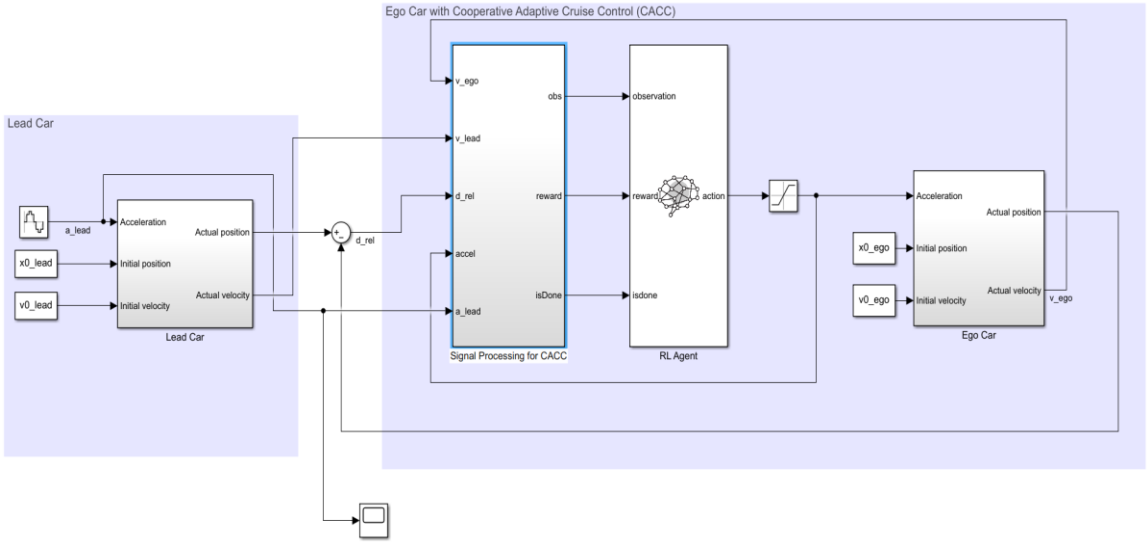

Figure 4.16 Training environment in MATLAB Simulink

x0_lead and *x0_ego* represent the initial position of the lead car and ego car, respectively. By taking the difference between these two values, the inter-vehicular distance can be calculated. In the training process, the initial position for the lead car is randomly generated between 10 and 11 meters, and the initial position for the ego car is set to 9.6 meters. Therefore, in each episode, the initial inter-vehicular distance would be randomly generated between 0.4 m to 1.4 m to cover more different scenarios that the agent can learn from. Similarly, the *v0_lead* and *v0_ego* represent the initial speed of the lead car and ego car, respectively. The term *a_lead* represents the acceleration of the lead car.

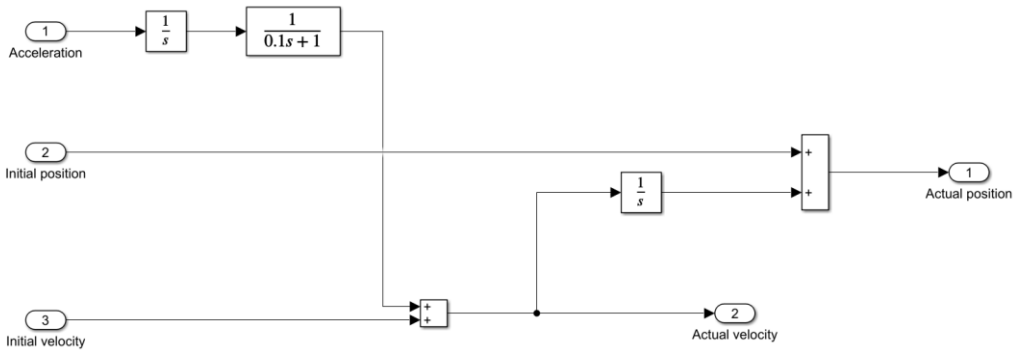

Figure 4.17 Lead car and ego car model

As shown in Figure 4.17 and Figure 4.18, the lead and ego cars are designed based on a vehicle kinematics bicycle model to simulate their acceleration, velocity, and positions [66]:

$$\begin{aligned} \dot{x} &= v\cos(\psi+\beta) \\ \dot{y} &= v\sin(\psi+\beta) \\ \dot{\psi} &= \frac{v}{l_r}\sin(\beta) \\ \dot{v} &= a \\ \beta &= \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta_f)\right) \end{aligned} \tag{4.15}$$

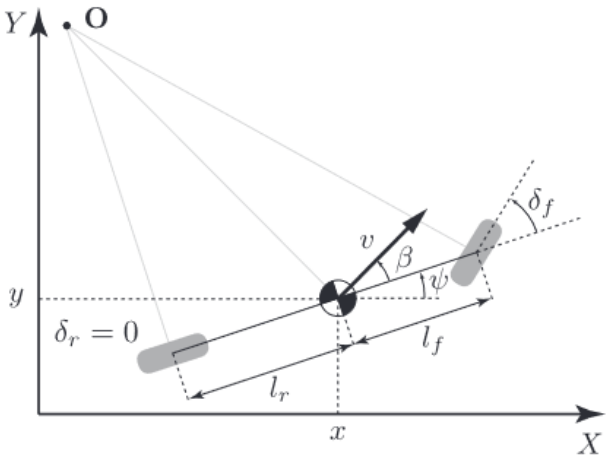

Figure 4.18 Kinematics bicycle model [66]

where the *x* and *y* are the positions of the center of mass of the vehicle in the frame of (*X*, *Y*) in Figure 4.18, ψ is the initial heading of the vehicle, β is the angle of the current velocity with respect to the longitudinal axis of the car, and a is the acceleration in the same direction of velocity. The variables l_r and l_f are the distance from the center of mass of the vehicle to the rear axles and front axels, respectively. δ_f and δ_r are the steering angles of the front and rear wheel, respectively. Since the Q-Car application simulation only considers the longitudinal control of the vehicle moving straight forward, the ψ, β, and *x* are zero, and the equation of the kinematics bicycle model can be written as:

$$\begin{aligned} \dot{y} &= v \\ \dot{v} &= a \end{aligned} \tag{4.16}$$

The acceleration of the lead car is assumed to be in the form of a sine wave with 0.03 amplitude and 30 seconds wavelength, which is the environment in that the front vehicle is moving. Additionally, a transform function is applied to represent the execution delay between the higher-level control and the lower-level control of the vehicle:

$$H(S) = \frac{V(S)}{A(S)} = \frac{1}{0.1s^2 + s} \tag{4.17}$$

The Signal Processing module (Figure 4.19) is designed to calculate the observations, including the relative distance and headway time between two vehicles, the signal flag to ending a set of states, and the reward value based on the reward function. This module is mainly combined with four different modules for this purpose: *Distance Calculation*, *Velocity Difference*, *IsDone*, and *Reward function*. The *v_ego* and *v_lead* are the velocity of the ego car and lead car, respectively. *v_set* is the preset cruising speed the ego car would reach if there were no other objects in front of it. The term *d_rel* is the relative distance between the ego car and the lead car, which can be measured by lidar sensors in the real application. *d_err* represents the difference between safety distance and actual relative distance, and *v_err* represents the velocity difference between the ego car and lead car.

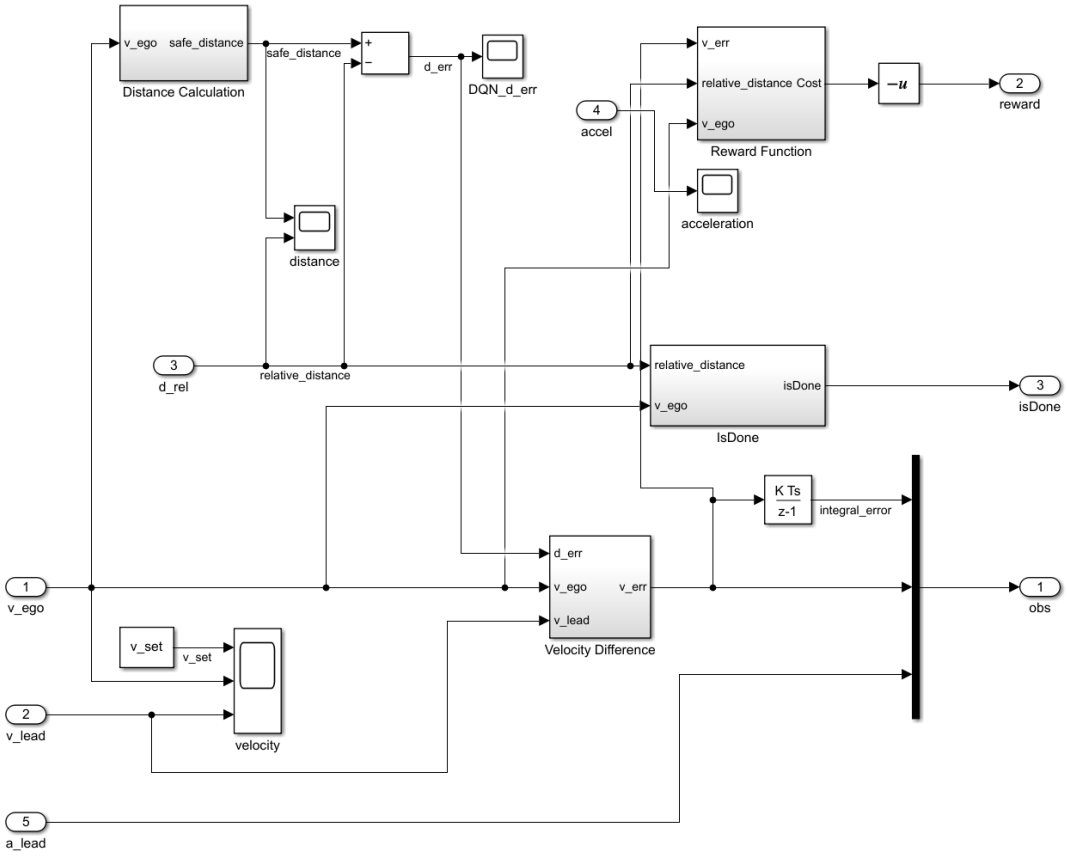

Figure 4.19 Signal Processing module for DDQN CACC

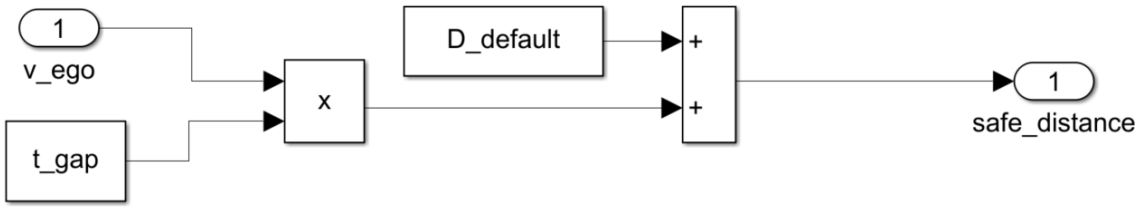

Figure 4.20 Safety distance calculation by MATLAB Simulink

The safety distance is calculated by the *Distance Calculate* module (Figure 4.20), which uses the CTH policy and satisfies equation (4.2), where the *t_gap* is the preset constant time headway time (0.3s), and *D_default* is the preset standstill distance (0.3 m).

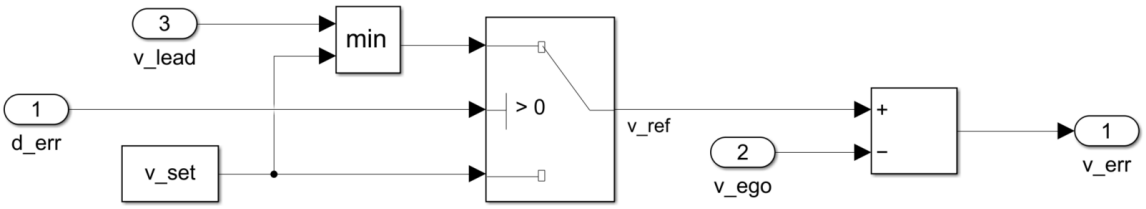

Figure 4.21 Velocity Difference module

The *Velocity Difference* module (Figure 4.21) is responsible for calculating the velocity difference, *v_err*, between the lead car and ego car and sending the result to the observation space, IsDone and Reward Function Module. To simulate the condition where there is no object in front of the ego car, the preset cursing speed, *v_set*, can replace the *v_lead* as its desired speed through this module.

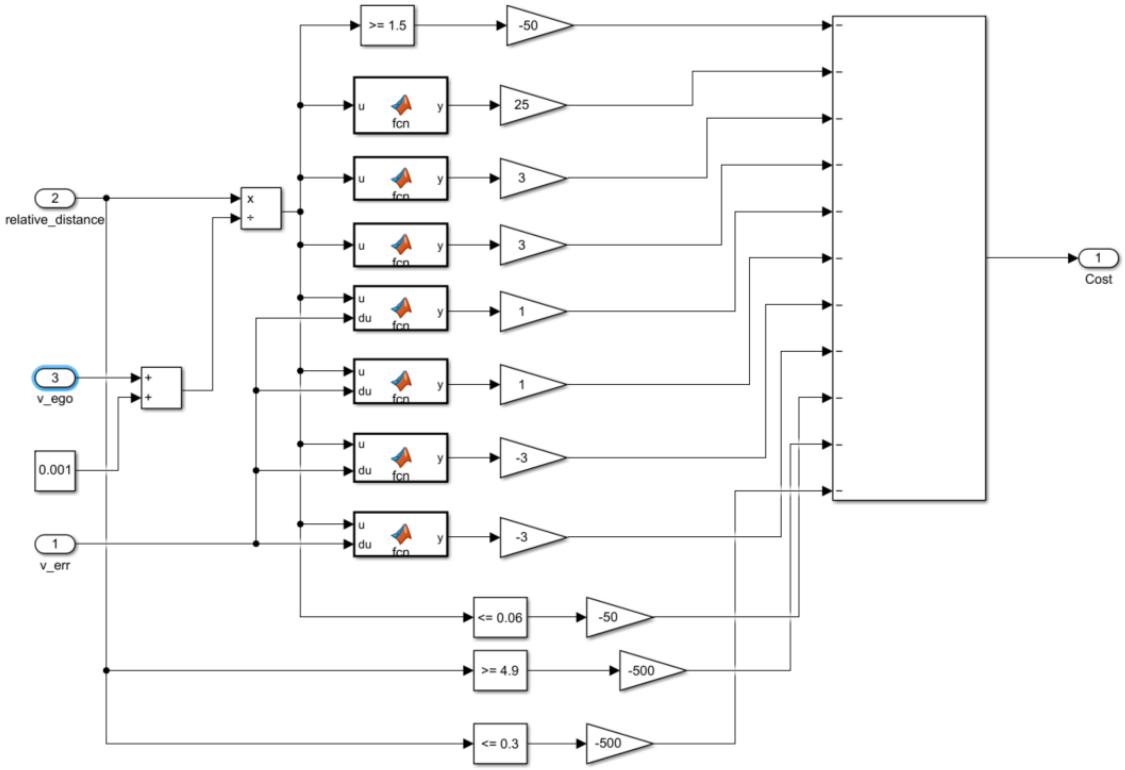

Figure 4.22 Reward Function by MATLAB Simulink

The *Reward Function* module (Figure 4.22) is implemented based on the reward function, which is designed in the Reward Function section of this chapter by using the MATLAB Simulink function block. The extra punishments are given by the equation (4.14), when the headway time is too short, which might lead to crushing, or it is too large that would be out of the vehicle's sensing range.

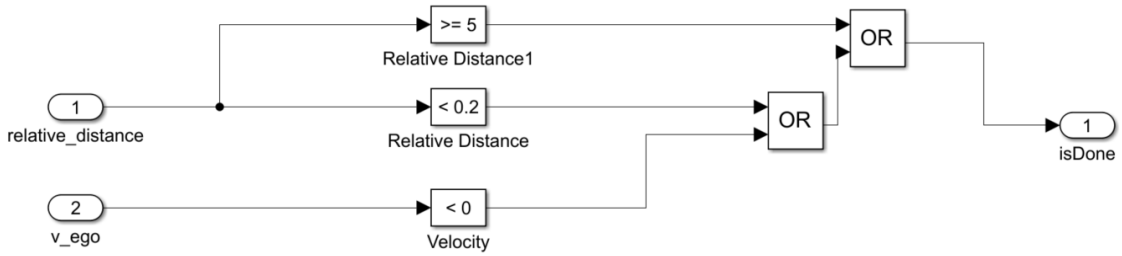

Figure 4.23 isDone signal module

Additionally, the isDone value is calculated by using *IsDone* module (Figure 4.23) for the learning agent to understand when a set of training should be terminated. This module demonstrates that episodes would be terminated when the relative distance is too small or too big, and the velocity of the ego car becomes a negative value, which means the ego car starts to move backward. Moreover, it can reduce the unnecessary training episodes that are too far away from the right policy.

4.3.5 Training Algorithm with Double Deep Q-learning (DDQN)

The training algorithm is another important aspect of training because its efficiency relies on the efficiency of using state-action pairs and their Q-values. Some of the new research [67] demonstrate that the DDQN can have a better performance by finding better policies and obtaining new state-of-the-art results. The double DQN is a learning algorithm combined with DNN and a double Q-learning algorithm without requiring additional networks and parameters. Instead of using a function:

$$y_i = r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; w^-)$$
$$w: \Delta w = \alpha(y_i - \hat{Q}(s_i, a_i; w^-))\nabla_w \hat{Q}(s_i, a_i; w) \quad (4.18)$$

The double DQN replaced y_i as:

$$y_i = r_i + \gamma\, \hat{Q}(\underset{a'}{\text{argmax}}\, \hat{Q}(s_{i+1}, a'; w); w^-) \tag{4.19}$$

As a result, the DDQN algorithm that can be applied to train the Q-Car model can be written as:

Algorithm 6: Double Deep Q-learning algorithm

Input $C, \alpha, D = \{\}$, initialize $w, w^- = w$, t = 0
Get initial state s_0
loop
  Sample action a_t given ε-greedy policy for current $\widehat{Q}(s_t, a; w)$
  Observe reward r_t and next state s_{t+1}
  Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer *D*
  Sample random minibatch of tuples (s_i, a_i, r_i, s_{i+1}) from *D*
  for j in minibatch do
    if episode terminated at step, $i + 1$ then
      $y_i = r_i$
    else
      $y_i = r_i + \gamma\, \hat{Q}(\underset{a'}{\text{argmax}}\, \hat{Q}(s_{i+1}, a'; w); w^-)$
    end if
    Do gradient descent step on $(y_i - \hat{Q}(s_i, a_i; w^-))^2$ for parameters w: $\Delta w = \alpha(y_i - \hat{Q}(s_i, a_i; w^-))\nabla_w \hat{Q}(s_i, a_i; w)$
  end for
  $t = t + 1$
  i**f** mod (t, C) == 0 then
    $w \leftarrow w^-$
  end if
end loop

4.4 Chapter Summary

This chapter presented all the implementation methods and procedures used in this thesis. Firstly, the hardware of the Q-Car model was introduced with its technical specifications. Then, an ACC model and algorithm were presented and designed using MATLAB Simulink. Additionally, a CACC model was built based on the ACC model with an extra communication module and a different algorithm. Lastly, a training model for Q-Car was established by using the MATLAB Simulink Reinforcement Learning Toolbox

with a designed neural network, and a DDQN training algorithm for this model was presented in the end.

Chapter 5

Experimental Results

Vehicle models with essential sensors are necessary for the implementation and testing. The objective of this section is to compare the performance of the ACC and CACC system on Q-Cars and to compare the performance of the ACC, CACC and CACC with Double DQN through MATLAB simulation. The simulation result is introduced first because the simulation results can give a good indication of the performance of the system in an actual experimental application. In the simulation part, the experimental setup for simulating the vehicle's longitudinal control by using the ACC system is presented. Next, the result for the ACC simulation is also presented. Then, a simulation model of the vehicle using the CACC system is established with the results. Lastly, the simulation for the CACC system using DDQN is formed by using the same environment as the training model, and the result is discussed.

5.1 Simulation Model and Results

For the consistency of simulation results in three different systems, the initial parameters are set as and are maintained in the experiments. The initial velocity of the ego car and lead car were set to 0.5 m/s and 0.6 m/s, respectively. Also, the initial inter-vehicular distance, which is the difference between their actual position, was set to 0.55 m. The standstill distance, which is the distance needed to be maintained by the ego car when its speed is zero, is set to 0.3 m. To make the simulation closer to the reality, where the vehicle movement involves in both acceleration and deacceleration in the tour, the acceleration of the lead car is assumed to occur by a step-changed sine wave with 0.03 amplitude and 30 seconds wavelength, which is to represent the environment in which the front vehicle is moving. This acceleration wave generator is very similar to the one that other researchers applied for real vehicle and traffic flow simulation for other CACC algorithms [68]. All the simulations use the kinematic bicycle model (4.16), which is a widely applied model for trajectory planning, such as in [67], to simulate the longitudinal changes of two vehicles.

5.1.1 Simulation for ACC system

As shown in Figure 5.1, the ACC simulation module is similar to the learning module built for the DDQN. Figure 5.2 demonstrates the Signal Processing module, which generates the *Safety distance* for the *Desired distance*, *Obstacle Distance*, *Nominal Speed* and *Car Speed* of the ego car as the input of the ACC controller function. Meanwhile, all the simulation results are plotted within this module. The ACC controller function is designed as the ACC controller from the previous chapter (4.3).

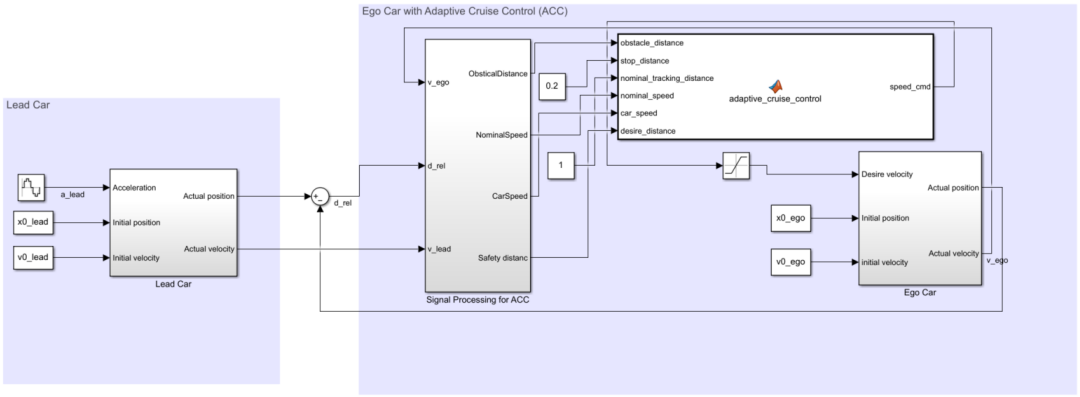

Figure 5.1 Simulation Model for ACC system in MATLAB Simulink

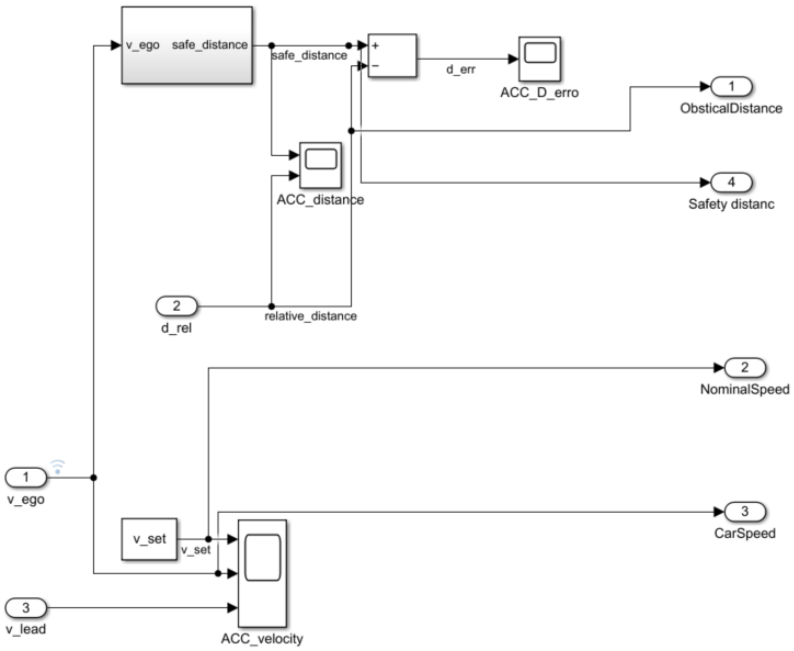

Figure 5.2 Signal Processing for ACC

The simulation results for the ego car with the ACC system and lead car's speed, inter-vehicular distance and distance error are demonstrated in Figure 5.3, Figure 5.4, and Figure 5.5, respectively.

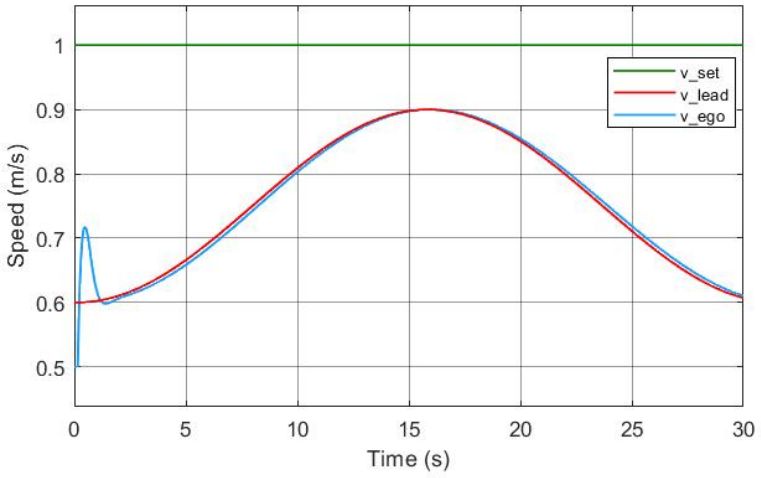

Figure 5.3 Speed measurements for ACC simulation

In Figure 5.3, the speed of the lead car and ego car, where *v_lead* is the speed of the car leading the platoon, and the *v_ego* is the speed of the ego car, which is trying to maintain a safe distance with the lead car is displayed. It demonstrated here that the ego car with the ACC system could follow of the lead car with the same speed only after a transient period of only 1.4 seconds where there is a speed overshoot, and afterward, it will maintain the lead car speed for the remainder of the tour.

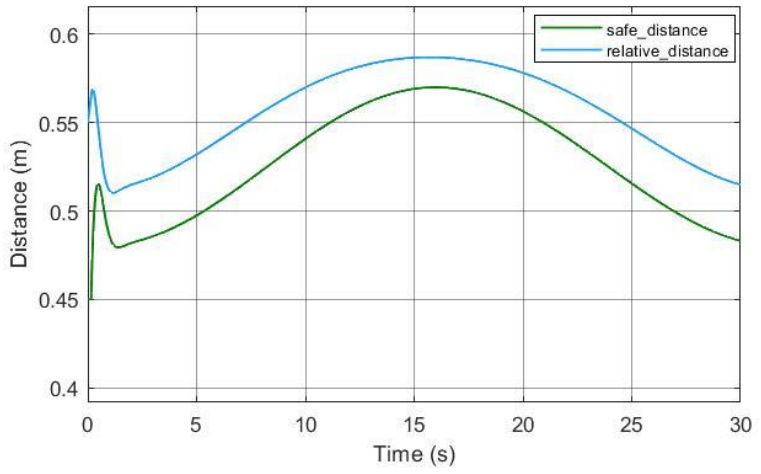

Figure 5.4 Safety distance and relative distance for ACC simulation

In Figure 5.4, inter-vehicular distance (*relative_distance*) is plotted with the safety distance (*safe_distance*) calculated by the *Signal processing* module. This figure shows that the ego car with the ACC system is able to maintain the inter-vehicular distance by following a similar waveform with the changing of the safety distance and keeping the inter-vehicular distance always greater than the safety distance. However, since there is always a gap between the relative distance and safety distance, it is not that efficient with respect to distance maintenance.

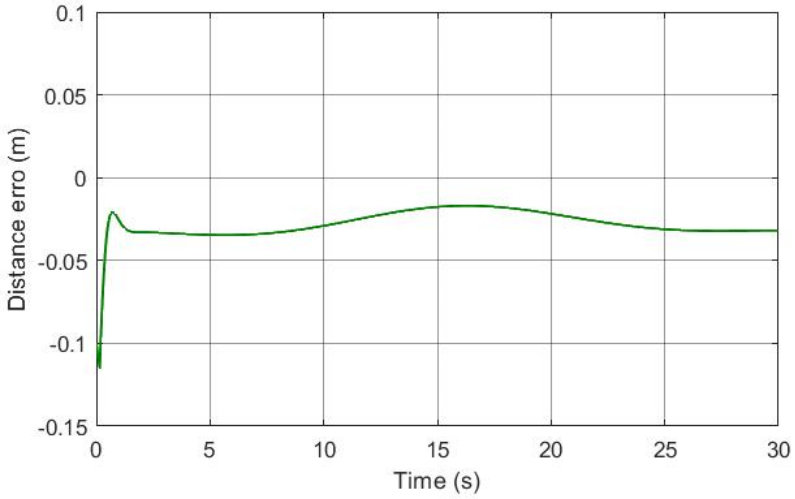

Figure 5.5 Distance error for ACC simulation

The plot in Figure 5.5 shows the distance error, which is the difference between the relative distance and safety distance. Notice that all the data in this plot is considered as absolute value for future comparison. This plot demonstrates the performance of maintaining the inter-vehicular distance for the ego car with the ACC system. It indicates that the distance between two vehicles varies between -0.1150 and -0.0169m. As a result, its peak-to-peak difference is 0.1319 m. However, it never reaches zero, which means that the relative distance is never equal to the safety distance during the whole tour.

5.1.2 Simulation of CACC system

As shown in Figure 5.6, the CACC simulation module is similar to the ACC simulation module, except for the *Signal Processing* module and the control algorithm. Besides the function provided in the ACC system, the *Signal Processing* module for the CACC system, in Figure 5.7, delivers the *FrontCarSpeed*, which simulates the direct data

transformation from the lead car to the ego car as the input of the CACC controller as well. Meanwhile, all the simulation results are plotted within this module. The CACC controller function is designed as the CACC controller from the previous chapter.

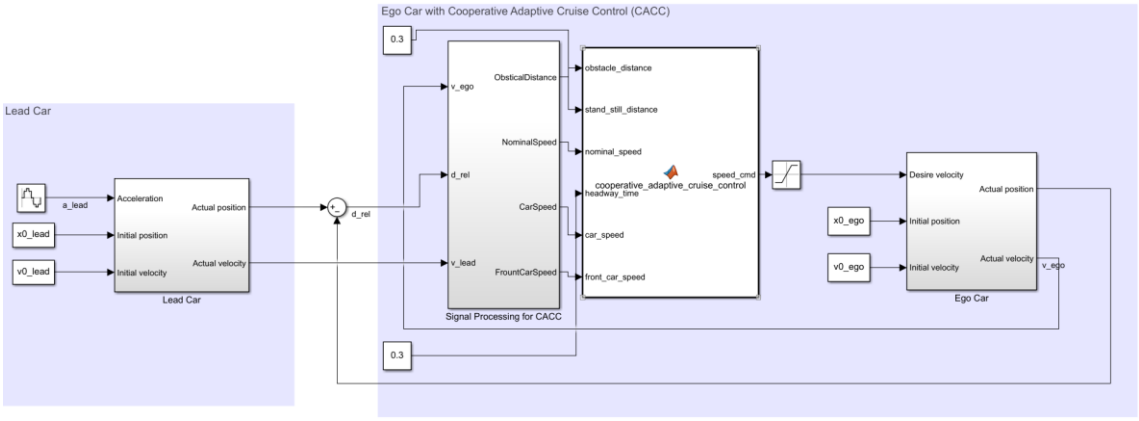

Figure 5.6 Simulation model for CACC system in MATLAB Simulink

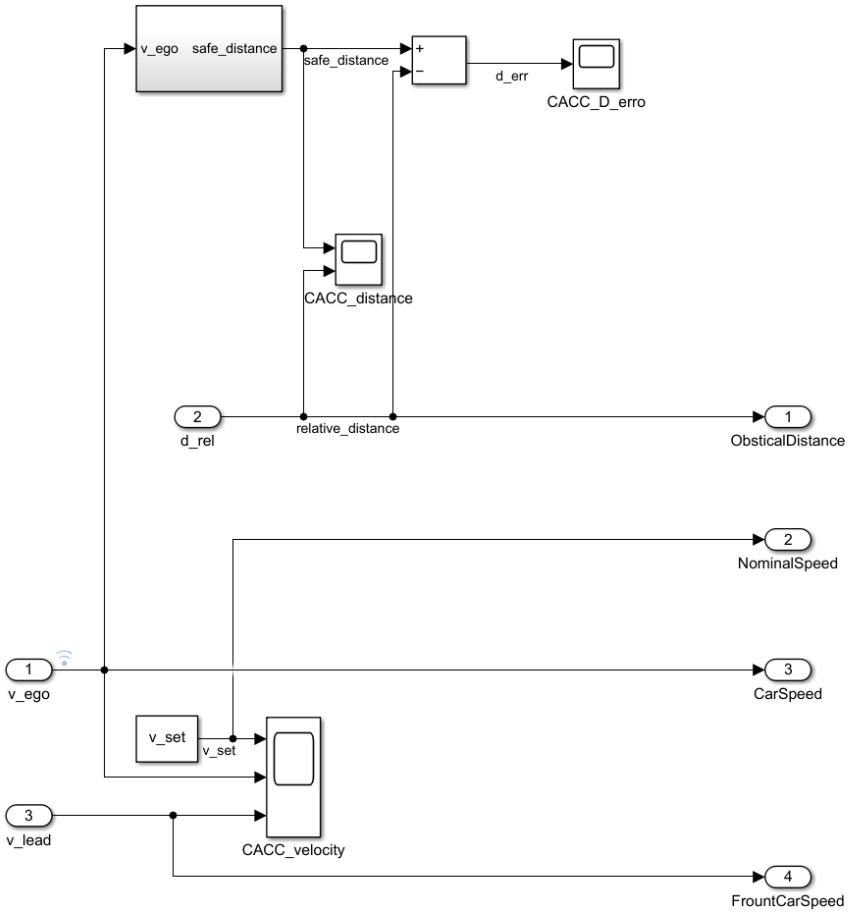

Figure 5.7 Signal Processing module for CACC system

The simulation results for the ego car with the CACC system and lead car's speed, inter-vehicular distance and distance error are demonstrated in Figure 5.8, Figure 5.9, and Figure 5.10, respectively.

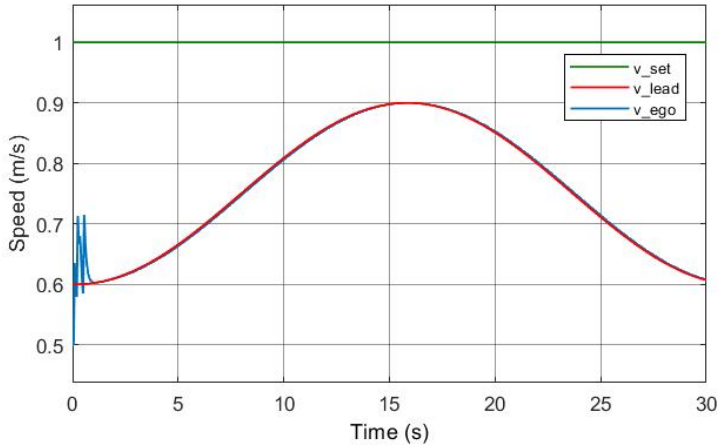

Figure 5.8 Speed measurements for CACC simulation

In Figure 5.8, similar to the ACC system plots, the speed of the lead car and ego car are displayed while all the parameters measured the same variable as in the ACC system simulation. It is clearly demonstrated that the ego car with the CACC system could follow the same speed as the lead car with 0.9 seconds of transient time and thereafter maintains the speed during the remainder of the tour. The mismatch during the transient is due to the fact that initially, the vehicle speed is not high, and the controller is trying to find the best algorithm to follow the lead car.

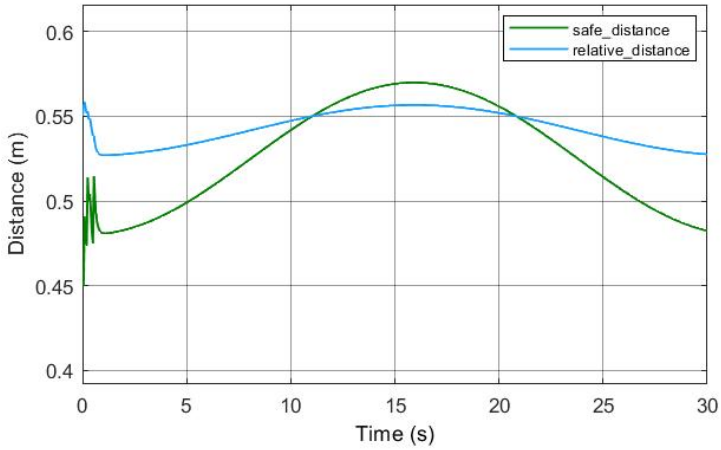

Figure 5.9 Safety distance and relative distance for CACC simulation

The plot in Figure 5.9 also shows that the ego car with the CACC system also is able to maintain the inter-vehicular distance by following a similar waveform with the changing of the safety distance. There are two intersections between two lines, which indicates that the CACC system's relative distance sometimes is smaller than the safety distance. However, the gap between the relative distance and safety distance is smaller than the one with the ACC system.

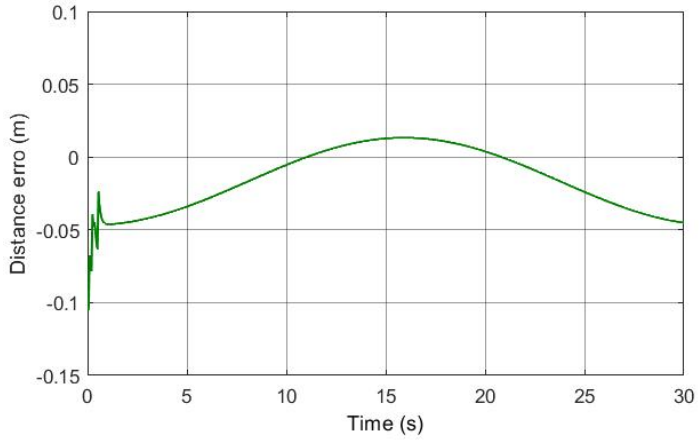

Figure 5.10 Distance error for CACC simulation

The distance error plot in Figure 5.10 for the CACC system shows the performance of maintaining the inter-vehicular distance for the ego car with the CACC system. It demonstrates that the distance between two vehicles varies between -0.1050 m and 0.0013m. As a result, its peak-to-peak difference is 0.1037 m. However, it only has two intersections with zero, which means that relative distance barely equals the safety distance during the whole tour.

5.1.3 Simulation for CACC system with DDQN

The simulation model for the CACC system with DDQN is the same model with a similar setup that is applied for the training (Figure 4.16). After the training, all the Q values are stored in the neural network within the RL leaning agent from the MATLAB Reinforcement Learning toolbox. Then simulation is based on running the preset parameters through the model without the learning process.

The simulation results for the ego car with the DDQN CACC system and lead car's speed, inter-vehicular distance and distance error are illustrated in Figure 5.11, Figure 5.12, and Figure 5.13, respectively.

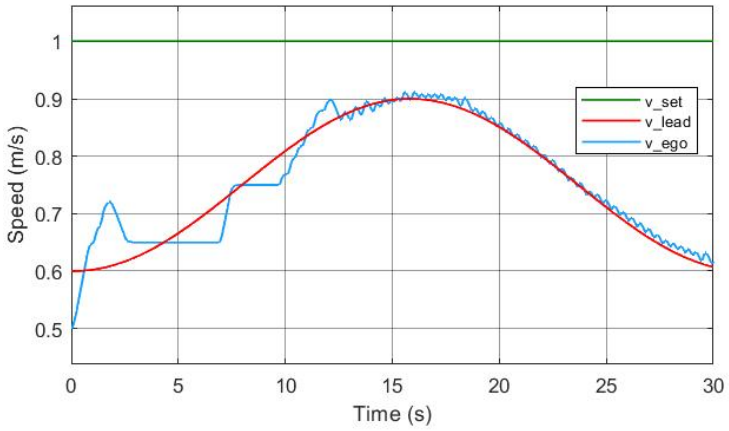

Figure 5.11 Speed measurements for CACC with DDQN simulation

In Figure 5.11, the speed of the lead car and ego car and all the parameters measured the same variable as in the ACC system, and the CACC simulation is presented. The figure illustrates demonstrated that the ego car with the CACC system could follow the same speed as the lead car with 12.8 seconds of transient time, and thereafter it can maintain the speed of the lead car for the remainder of the tour.

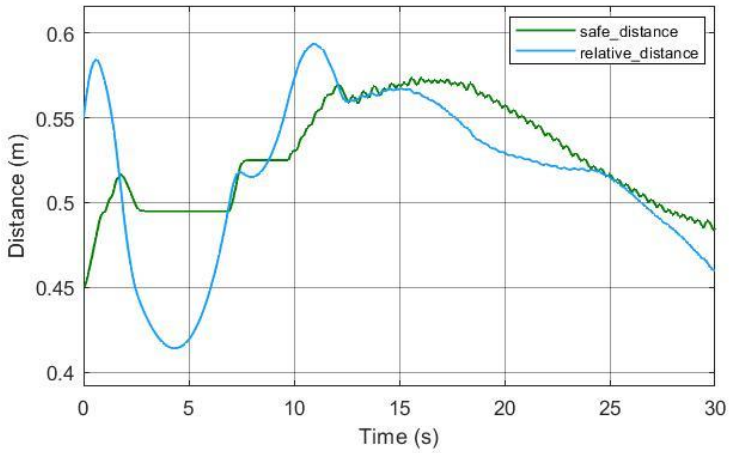

Figure 5.12 Safety distance and relative distance for CACC with DDQN simulation

The plot in Figure 5.12 also shows that the ego car with the CACC system also can maintain the inter-vehicular distance by following a similar waveform with the changing

of the safety distance. These two lines have more overlapping than the ACC and the CACC system, which indicates that the DDQN CACC system's ability to maintain the relative distance with that of the safety distance is better than those methods.

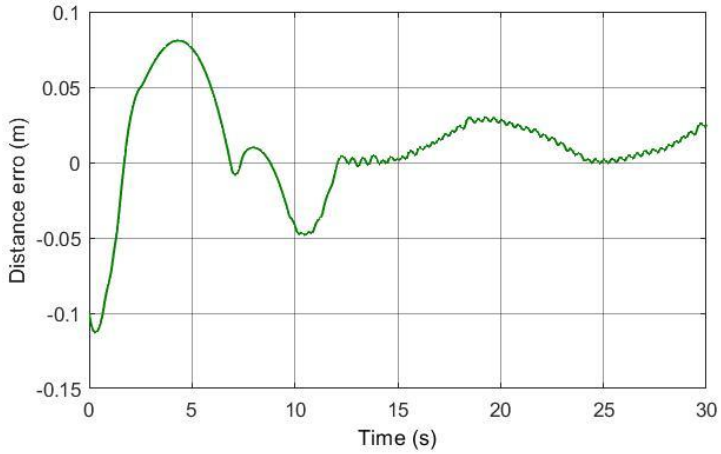

Figure 5.13 Distance error for CACC with DDQN simulation

The distance error plot in Figure 5.13 for the CACC system with DDQN shows the performance of maintaining the inter-vehicular distance for the ego car with the DDQN CACC system. It demonstrates that the distance between two vehicles varies between 0.08087 m and -0.1128m. As a result, its peak-to-peak difference is 0.1935 m.

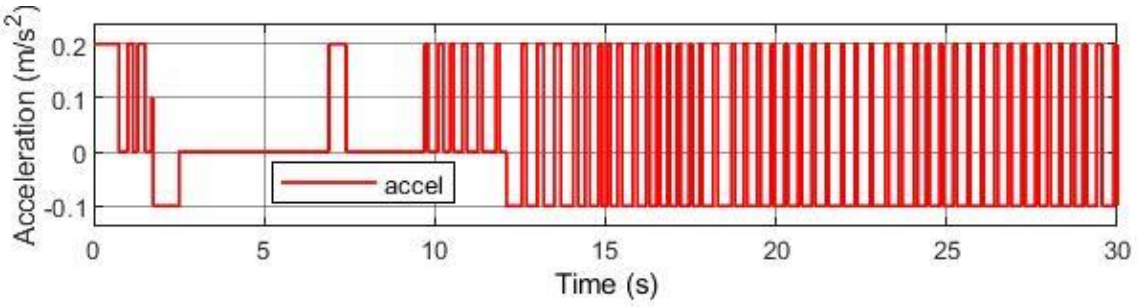

Figure 5.14 Acceleration of Ego Car using CACC with DDQN

Figure 5.14 shows the acceleration profile of the ego car throughout the time of interest. Based on the designed action space (4.13), actions include AC_a , AC_s, DAC_a, DAC_s, and *NO* are signed with acceleration value as 0.2 m/s^2, 0.1 m/s^2, -0.05 m/s^2, -0.1 m/s^2, and 0 m/s^2, respectively. This plot demonstrates all the actions involved in finding the best policy for the system.

Table 5.1 The simulation result for distance error

Methods	Absolute Median (m)	Absolute Average (m)	Peak-to-peak (m)
ACC	2.897e-2	2.832e-2	1.319e-1
CACC	1.470e-2	1.565e-2	1.037e-1
CACC with DDQN	8.70e-3	9.36e-3	1.935e-1

According to the measurements of distance error (see Table 5.1), the average distance error from the ACC system, the CACC system, and the CACC system with DQN are 0.02832 m, 0.01565 m, and 0.00936 m, respectively. The smaller the error is, the better performance in the scenario of maintaining the relative distance as the safety distance. Therefore, the CACC system improves the performance of distance maintenance by 44.74% from the ACC system. Additionally, the CACC system with DDQN further enhances the performance of the CACC system by 40.19%. The median value comparison is another way to compute the performance. The median value of distance error from the ACC system, the CACC system, and the CACC system with DQN are 0.02897 m, 0.01470 m, and 0.00870 m, respectively. Therefore, the CACC system improves the performance of distance maintenance by 49.25% from the ACC system.

Additionally, the CACC system with DDQN further enhances the performance of the CACC system by 40.81%. In both comparisons, the CACC with DDQN has better performance in maintaining the safety distance by matching the relative distance to it. Even though the DDQN agent requires around 35 hours to be trained, depending on the hardware setup, the system is still the best choice when safety and efficiency are essential for the vehicles. Additionally, according to the speed plot for all three systems, the ACC and CACC system has a smoother speed profile than the CACC system with DDQN. The reason is that both the ACC and CACC control algorithm computes the desired velocity directly and send it to the vehicle dynamics, but the CACC with DDQN has to calculate the acceleration as an action for sending to the vehicle dynamics. Therefore, the ACC and CACC system have a better performance in tracking the vehicle speed of the front vehicle, especially the CACC system.

In summary, according to the simulation result for all three systems. The CACC system and the ACC system are good at tracking the vehicle speed, but the CACC system with DDQN has a better performance in distance control and tracking to improve the traffic capacity by reducing the average difference between safety distance and relative distance.

5.2 Real Application Setup and Results

This research study uses two Quanser's Q-Cars as models for implementing both systems. The Q-Cas is an electrically driven model with onboard CPU and sensors, including Lidar, RGB-D camera, and 360° CSI camera suite. Additionally, Q-Cars can implement the module built by MATLAB Simulink models, and it has libraries to drive all the sensors and actuators. Therefore, both autonomous driving systems are constructed and implemented on Q-Cars using MATLAB Simulink. Additionally, the onboard Lidar sensor is set to 10 Hz sampling rate, and the CSI cameras are applied for the line detection to keep the vehicle in the centre of the road.

In the experiment, in order to restore the actual traffic situation and maintain the same status as the simulation, the preset value of Q-cars is proportional to a regular sedan. Since the length of a Q-car is 40cm and the size of a typical sedan is around 4 m, the standstill distance for equation (4.2) is set to 20 cm to represent the average standstill distance for a sedan is 2 to 3 m. Additionally, the constant time headway time for equation (4.2) is set to 0.3s to represent the 3s constant time headway time in real traffic situations. Lastly, the initial speed for both cars is set to zero, and the initial relative distance is set at around 0.1 m.

The front car can be driven manually or with ACC automated driving system, and the follower car can use ACC or CACC automated driving system to follow the front car. In image 5.14, both cars are driving on a mini road, which simulates the car on a real road. In order to compare the performance of the two systems, both systems are tested with the same car on the same route. Firstly, the lead Q-Car is using the ACC system to compute a simple autonomous driving condition. Then, the ego Q-Car is tested with the ACC system and CACC system onboard, respectively.

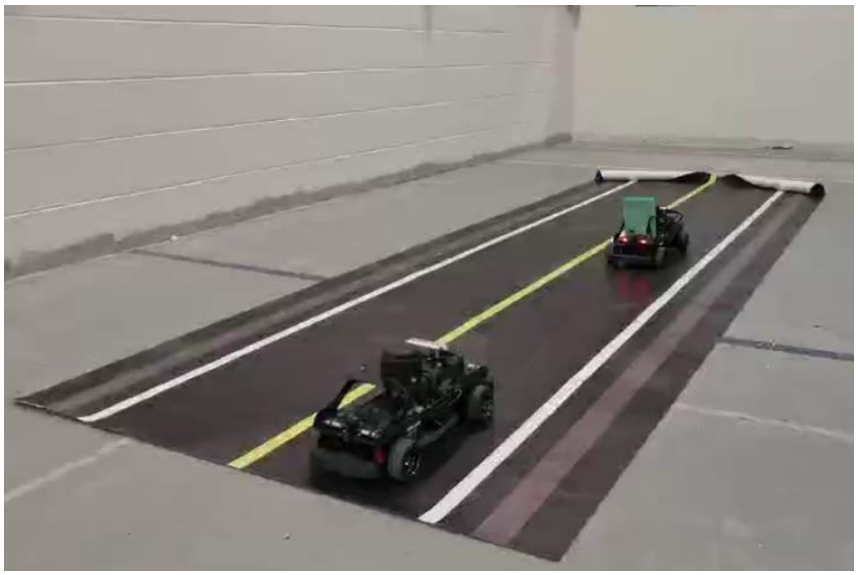

Figure 5.15 Experimental setup with two Q-Cars on a mini road

5.2.1 Experimental Result for ACC System

For the ACC system implementation on Q-Car, Figure 5.16 presents the speed of the lead car and ego car. Figure 5.17 demonstrates and distance error between the relative distance and safety distance.

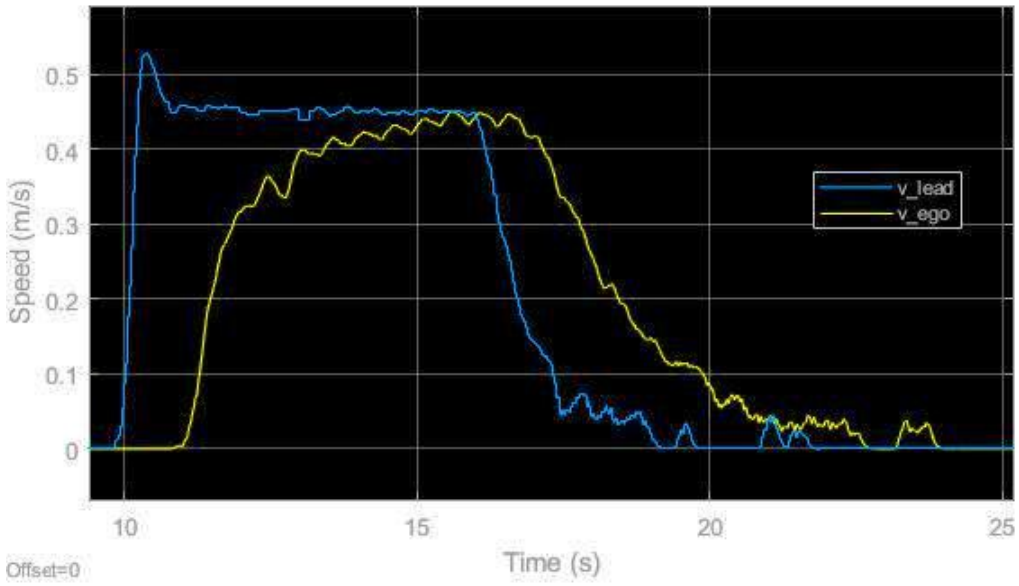

Figure 5.16 Speed measurements for ACC experimental result

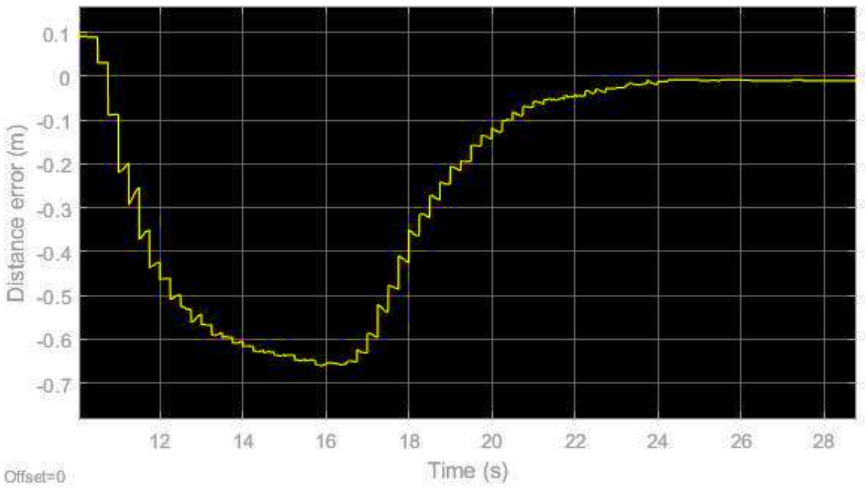

Figure 5.17 Distance error for ACC experimental result

Since both the lead car and ego car uses the ACC system to control the distance, their speed plot has a similar shape with transmission and execution delay. The distance error (relative distance subtract the safety distance) between two vehicles is between 0.0914 m to -0.6604 m.

5.2.2 Experimental Result for CACC System

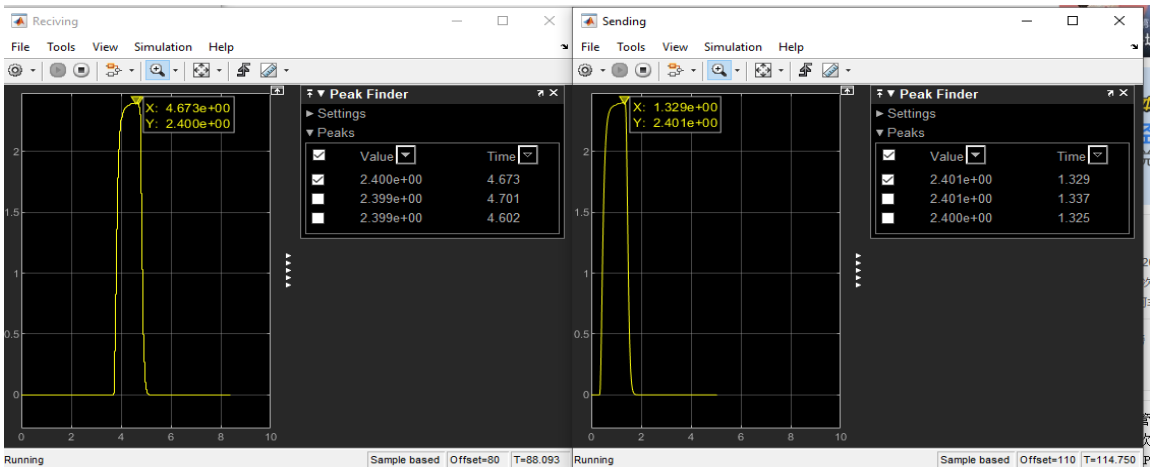

Figure 5.18 Experimental result for transmission delay testing

For the CACC system in the Q-Car application, the V2V communication is done by connecting both cars to a local area network. In the Simulink model, the V2V communication is established using the *Stream* block from Quanser's library. Before implementing the CACC system in the Q-Car, the V2V communication module was tested

for its latency. As Figure 5.18, the delay of data transmission is 2 ms, which is calculated by the time difference of the speed's peak value.

For the CACC system implementation on Q-Car, Figure 5.19 presents the speed of the lead and ego car. Figure 5.20 demonstrates and distance error between the relative distance and safety distance. The variance is mainly due to the fact that lead and ego cars use different systems to control the distance. The distance error (relative distance subtracted from the safety distance) between two vehicles is between 0.07764 m to -0.2331 m.

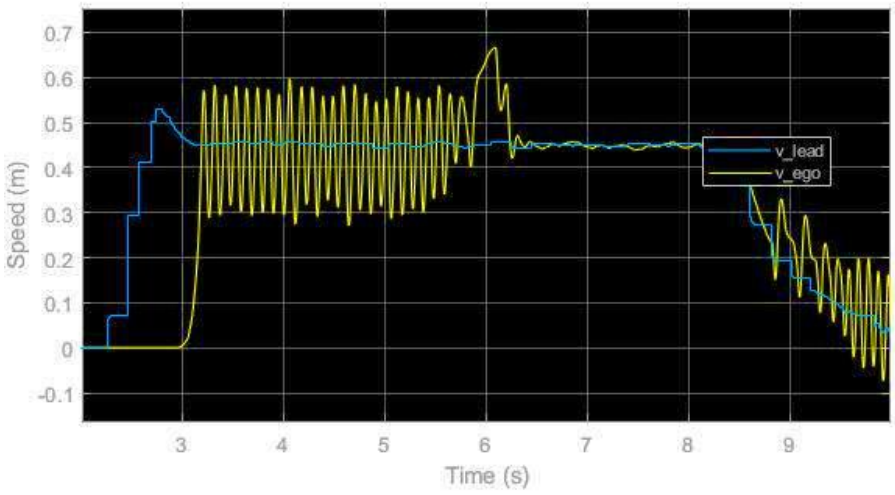

Figure 5.19 Speed measurements for CACC experimental result

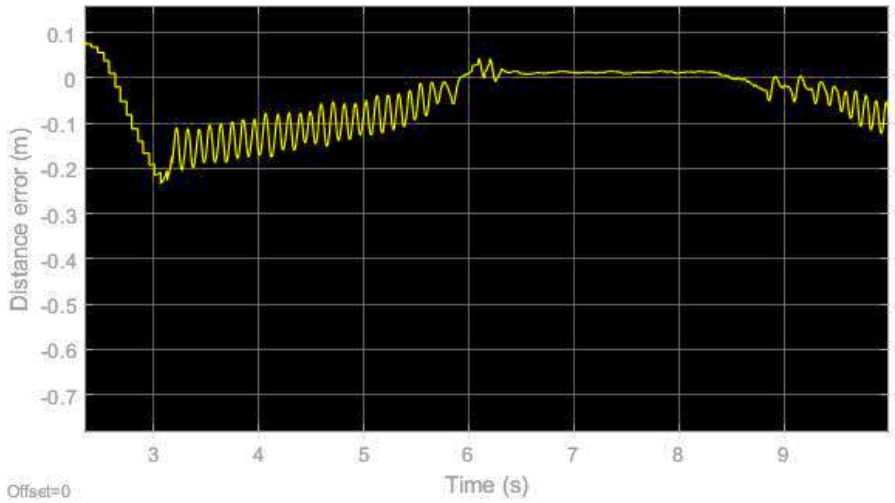

Figure 5.20 Distance error for ACC experimental result

The simulation result for the two cars' speed and distance error for ACC and CACC are illustrated in Figures 5.21, 5.22, 5.23, 5.24. During this simulation, the acceleration of the lead car is assumed to be a step-changed sine wave with 0.06 amplitude, 30 seconds wavelength, with the initial distance of 0.1 m, and initial speeds of both cars set to zero, both the ACC and CACC systems have a great deal of variances when they are operating in the low-speed condition. Additionally, due to this scenario, many commercial vehicles embedded with the ACC system have the lowest speed limit for enabling the cruise control system, such as 30 km/h for VOLVO XC90 [69].

This simulation result matches the results in the Q-Car actual application. Firstly, the CACC system is more variant than the ACC system, but the CACC has a better performance on following the front car speed and maintaining the relative distance to the safety distance as closer as possible.

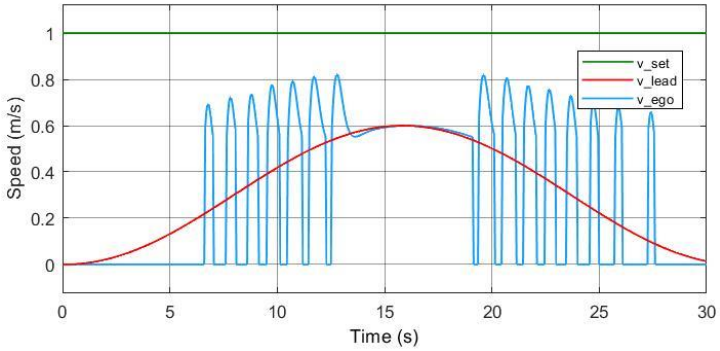

Figure 5.21 Simulation of speed measurements for ACC system in low-speed condition

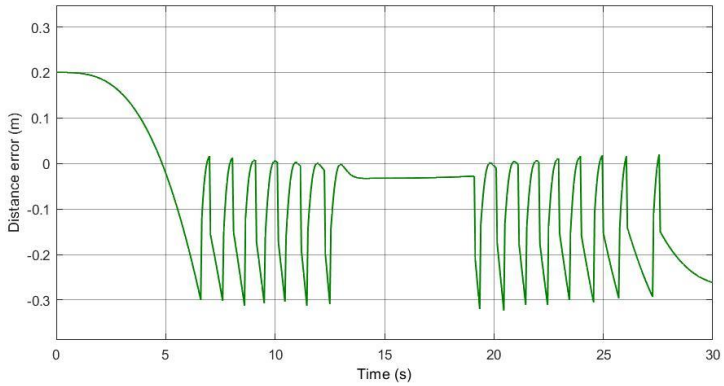

Figure 5.22 Simulation of distance error for ACC system in low-speed condition

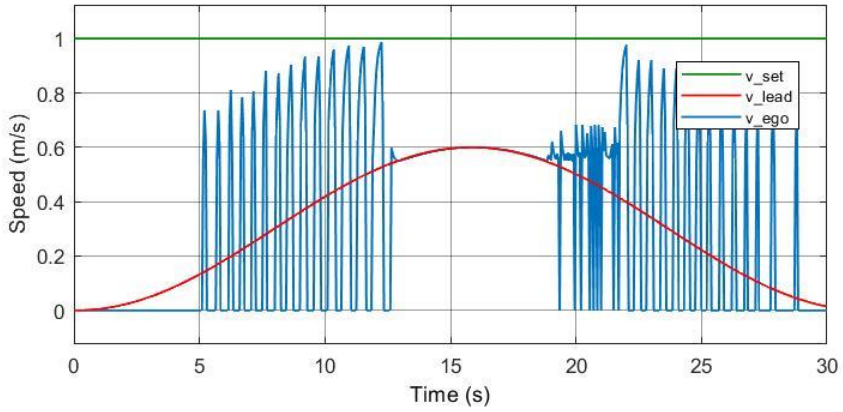

Figure 5.23 Simulation of speed measurements for CACC system in low-speed condition

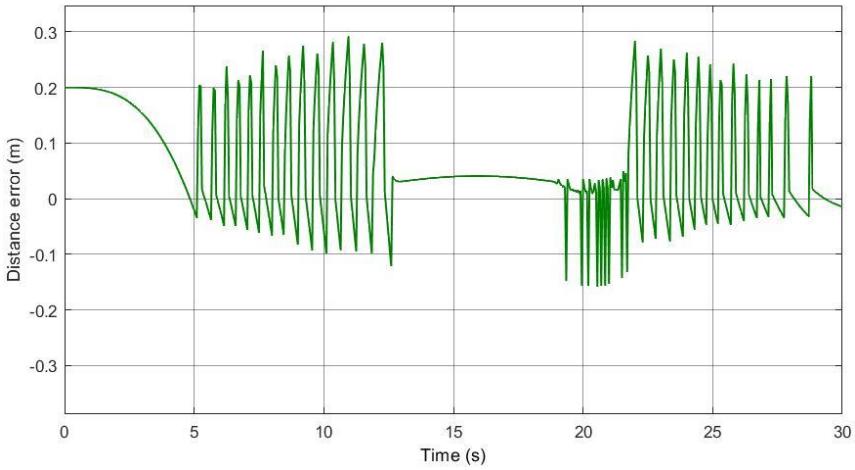

Figure 5.24 Simulation of distance error for CACC system in low-speed condition

Table 5.2 The experimental result for distance error in each system

Methods	Absolute Median (m)	Absolute Average (m)	Peak-to-peak (m)
ACC	2.167e-1	2.869e-1	7.518e-1
CACC	2.600e-2	4.799e-2	3.108e-1

According to Table 5.2 above, it is hard to compare the performance of the two strategies accurately because the operating conditions cannot be precisely the same in a real-world application. Nevertheless, it still demonstrates that the ego car with a CACC

system which gathers information directly from the lead car has a better performance on distance control. In [49], authors have reached a similar conclusion when applying the DDPG-PID controller, which is a deep learning based PID controller, to a platoon of the vehicle using CACC. They conclude that the method involves deep learning to improve 60.94% of the performance of the distance error control by comparing the performance of the conventional PID controller.

5.3 Chapter Summary

This chapter presented the simulation result for the ego car using the ACC, the CACC, and the CACC with DDQN. Additionally, the experimental result by implementing both ACC and CACC systems on the actual Q-Cars is presented. Firstly, based on the simulation, in a vehicle platoon, the capacity would be increased by using both CACC and CACC with the DDQN method compared to the ACC system. The CACC system has the best performance in following the speed of the front vehicle, and the CACC with DDQN can minimize the difference between the safety distance and relative distance to increase the traffic capacity as well as traffic efficiency. Then, the experimental results conclude in the same way as the simulation results, that the CACC system has a better performance both on the following the front vehicle speed and minimizing the difference between the safety distance and relative distance.

Chapter 6

Conclusion and Future Work

6.1 Contributions

In this thesis, a CACC system on two Q-Cars with a new control algorithm and DDQN was built and tested based on the lidar sensor and V2V information exchanges. A training model and simulation environment were built by using MATLAB Simulink. Furthermore, the CACC and ACC system was implemented on two Q-Cars, which is a real mockup of an autonomous vehicle, to test its performance on traffic capacity. This work aims to demonstrate the benefits of the CACC system and the possibility of applying Wi-Fi in indoor and short-range V2V communication. Although WiFi has been used to V2V and V2I communication before, it is not an ideal communication technology in real-world applications. However, it still can demonstrate the data transmission process of AGV and other automated vehicles navigating in indoor and small areas. The challenge of this thesis is to compute an algorithm for increasing the traffic capacity and maintaining the safety requirements.

6.2 Summary

Firstly, by researching the relative field of the CACC, V2V communication, and CACC with deep learning, it was concluded that the CACC system has a considerable advantage compared to the ACC system in traffic capacity, traffic efficiency,energy efficiency, safety, customer comfort, and platooning control. These studies were completed in both real applications and simulations. Many algorithms, policies and V2V communication technology were applied to the CACC system to make it a hot field to study. Additionally, they also indicate that the CACC has a strong place in the future of the autonomous driving of intelligent vehicles with the development of communication technologies.

Deep learning and reinforcement learning theories were developed from the Markov Decision Process, which is the foundation of reinforcement learning, followed by relative learning algorithms in reinforcement learning, such as temporal difference learning,

SARSA, and Q-learning. Meanwhile, one of the most popular deep learning algorithms, the neural network, was introduced. By combining the theories of both reinforcement learning and deep learning, theories and algorithm of deep Q-Learning and double deep Q-learning that had been applied to this thesis were introduced and explained with their advantages and disadvantages. The most important advantage of using DDQN instead of DQN is to reduce the computation time to improve the efficiency of using state-action pairs.

From all these pieces of research and theories, the methodology to face those challenges was chosen. Firstly, the algorithm of the ACC system was computed， and the ACC module was designed and implemented on the Q-Car by using a lidar sensor for distance measurements and MATLAB Simulink. Then, based on the ACC module, the CACC system was designed and implemented on the Q-Car with an additional wireless communications module (WiFi). Lastly, the training module for CACC with DDQN was developed based on the bicycle model, which is a widely applied model for the vehicle's trajectory planning simulation, proper state-space action, reward functions and neural networks. Meanwhile, the training algorithm was implemented in the model by using the Reinforcement Toolbox in MATLAB Simulink.

Lastly, in the experimental result, the performance of the CACC system with a new control algorithm and the CACC system with DDQN was compared to the performance of the ACC system by simulation. The acceleration input for the lead car is generated as a step-changed sine wave, which also has been applied to the simulation of platoon control in many studies. The results demonstrate that the CACC system maintains the distance 44.74 % better than the ACC system by collecting more accurate information through low latency wireless communication. Additionally, the CACC system with DDQN improves the performance of maintaining distance by an additional 40.19%. The CACC system can reduce the inter-vehicular distance and respond more quickly to the front vehicle's action with accurate information.

Meanwhile, the DDQN system can match the relative distance with the safety distance to have better distance control. As a result, the traffic capacity of vehicles can be improved. On top of the simulation, the experiments on Q-cars confirmed the same results. By implementing and testing the ACC and CACC system on the ego car, which follows a

lead car with an ACC system, the CACC system has a better performance both on following the front vehicle speed and minimizing the difference between the safety distance and relative distance.

6.3 Future Work

6.3.1 Wireless communication Security

Implementing an additional wireless communication module makes the vehicle more vulnerable to attack. There are slight changes in the acceleration when the algorithm is combined with multiple vehicle speeds. Information safety during the communication process should also be considered in the future for massive public applications.

6.3.2 Wireless communication Technology

Nowadays, 5G communication protocol is developing rapidly with faster and more stable connections and transmission capacity. This thesis did not apply this technology because wireless communication is not the main objective of this thesis. However, the issue should be explored

6.3.3 V2X communication-based Autonomous Driving

As mentioned before, V2V is only a part of the V2X communication. By vehicle communicating with more elements around it, such as infrastructures and mobile devices, it can gather more accurate information for it to sense the environment around it. As a result, a level 4 or level 5 autonomous driving can be reached more safely and efficiently.

6.3.4 Platoon Forming and Control

The work in this thesis could be explored further by expanding the current two-car situation into a platoon scenario involving more vehicles, further development and evaluation of the CACC control policy will be required. This thesis did not extend to the multiple cars problem due to the limitations on the availability of the hardware.

6.3.5 Other Deep Learning Approaches for CACC

Recently, many research studies have focused on different deep learning algorithms to improve the performance of CACC, such as control, policy evaluation, and wireless

communication. New algorithms can be developed, and other deep learning methods for the CACC system can be compared to the existing ones in future work.

References

[1] Iea, "Trends and developments in Electric Vehicle Markets – Global EV outlook 2021 – analysis," *IEA*. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets. [Accessed: 15-Sep-2021].

[2] K. Stricker, T. Wendt, W. Stark, M. Gottfredson, R. Tsang, and M. Schallehn, "Electric and autonomous vehicles: The future is now," *Bain*, 29-Oct-2020. [Online]. Available: https://www.bain.com/insights/electric-and-autonomous-vehicles-the-future-is-now/. [Accessed: 15-Sep-2021].

[3] "GM reveals new Ultium batteries and a flexible global platform to rapidly grow its EV portfolio," *media.gm.com*, 04-Mar-2020. [Online]. Available: https://media.gm.com/media/us/en/gm/home.detail.html/content/Pages/news/us/en/2020/mar/0304-ev.html. [Accessed: 15-Sep-2021].

[4] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143-153, Sept. 2003, doi: 10.1109/TITS.2003.821292.

[5] Bin-Feng Lin et al., "Integrating Appearance and Edge Features for Sedan Vehicle Detection in the Blind-Spot Area," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 737-747, June 2012, doi: 10.1109/TITS.2011.2182649.

[6] H. Muslim and M. Itoh, "Effects of Human Understanding of Automation Abilities on Driver Performance and Acceptance of Lane Change Collision Avoidance Systems," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2014-2024, June 2019, doi: 10.1109/TITS.2018.2856099.

[7] Jin Xu, Guang Chen and Ming Xie, "Vision-guided automatic parking for smart car," *Proceedings of the IEEE Intelligent Vehicles Symposium 2000* (Cat. No.00TH8511), 2000, pp. 725-730, doi: 10.1109/IVS.2000.898435.

[8] "SAE J3016 automated-driving graphic," *SAE International*, 15-May-2020. [Online]. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic. [Accessed: 15-Sep-2021].

[9] sight Technologies. (2018). *How 5G will influence autonomous driving systems, white paper*. Retrieved August 30, 2019

[10] S. Zeadally, J. Guerrero, and J. Contreras, "A tutorial survey on vehicle-to-vehicle communications," *Telecommunication Systems*, vol. 73, no. 3, pp. 469–489, 2019.

[11] F. S. Brogua, "Cooperative Driving: Basic Concepts and a First Assessment of 'Intelligent Cruise Control' Strategies," 1991.

[12] L. Yang, J. Mao, K. Liu, J. Du and J. Liu, "An Adaptive Cruise Control Method Based on Improved Variable Time Headway Strategy and Particle Swarm Optimization Algorithm," in *IEEE Access*, vol. 8, pp. 168333-168343, 2020, doi: 10.1109/ACCESS.2020.3023179.

[13] Z. S. Jiang, H. H. Zhang, and B. Yang, "An improved variable time headway strategy for ACC," *Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence* - RICAI 2019, 2019.

[14] D. MacKinnon, "High capacity personal rapid transit system developments," in *IEEE Transactions on Vehicular Technology*, vol. 24, no. 1, pp. 8-14, Feb. 1975, doi: 10.1109/T-VT.1975.23591.

[15] Junmin Wang and R. Rajamani, "Should adaptive cruise-control systems be designed to maintain a constant time gap between vehicles?," in *IEEE Transactions on Vehicular Technology*, vol. 53, no. 5, pp. 1480-1490, Sept. 2004, doi: 10.1109/TVT.2004.832386.

[16] J. Zhao, M. Oya and A. El Kamel, "A safety spacing policy and its impact on highway traffic flow," *2009 IEEE Intelligent Vehicles Symposium*, 2009, pp. 960-965, doi: 10.1109/IVS.2009.5164410.

[17] R. Rajamani and Chunyu Zhu, "Semi-autonomous adaptive cruise control systems," Proceedings of *the 1999 American Control Conference* (Cat. No. 99CH36251), 1999, pp. 1491-1495 vol.2, doi: 10.1109/ACC.1999.783618.

[18] J. Piao and M. McDonald, "Advanced Driver Assistance Systems from autonomous to cooperative approach," *Transport Reviews*, vol. 28, no. 5, pp. 659–684, 2008.

[19] L.-hua Luo, H. Liu, P. Li, and H. Wang, "Model predictive control for adaptive cruise control with multi-objectives: Comfort, fuel-economy, safety and car-following," *Journal of Zhejiang University SCIENCE A*, vol. 11, no. 3, pp. 191–201, 2010.

[20] G. Marsden, M. McDonald, and M. Brackstone, "Towards an understanding of adaptive cruise control," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 33–51, 2001.

[21] C. L. Melson, M. W. Levin, B. E. Hammit, and S. D. Boyles, "Dynamic traffic assignment of Cooperative Adaptive Cruise Control," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 114–133, 2018.

[22] M. A. S. Kamal, M. Mukai, J. Murata and T. Kawabe, "Ecological Vehicle Control on Roads With Up-Down Slopes," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 783-794, Sept. 2011, doi: 10.1109/TITS.2011.2112648.

[23] M. Michaelian and F. Browand, "Quantifying platoon fuel savings: 1999 Field Experiments," *SAE Technical Paper Series*, 2001.

[24] D. SWAROOP, J. K. HEDRICK, C. C. CHIEN, and P. IOANNOU, "A comparison of spacing and headway control laws for automatically controlled vehicles1," *Vehicle System Dynamics*, vol. 23, no. 1, pp. 597–625, 1994.

[25] R. Rajamani, *Vehicle Dynamics and control*. New York: Springer Science, 2006.

[26] J. Ploeg, A. F. Serrarens, and G. J. Heijenk, "Connect & Drive: Design and Evaluation of Cooperative Adaptive Cruise Control for congestion reduction," *Journal of Modern Transportation*, vol. 19, no. 3, pp. 207–213, 2011.

[27] Z. Wang, G. Wu and M. J. Barth, "A Review on Cooperative Adaptive Cruise Control (CACC) Systems: Architectures, Controls, and Applications," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2884-2891, doi: 10.1109/ITSC.2018.8569947.

[28] M. Segata, F. Dressler, R. Lo Cigno, and M. Gerla, "A simulation tool for automated platooning in mixed highway scenarios," *Proceedings of the 18th annual international conference on Mobile computing and networking* - Mobicom '12, 2012.

[29] "The Grand Cooperative Driving Challenge 2016: Boosting the introduction of Cooperative Automated Vehicles," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/document/7553038. [Accessed: 2-Oct-2021].

[30] "California Partners for Advanced Transportation Technology," *Home | California Partners for Advanced Transportation Technology*. [Online]. Available: https://path.berkeley.edu/. [Accessed: 02-Oct-2021].

[31] "Verdict media limited," *Verdict Traffic*. [Online]. Available: https://www.roadtraffic-technology.com/projects/the-sartre-project/. [Accessed: 02-Oct-2021].

[32] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe and M. Nakamura, "Cooperative Adaptive Cruise Control in Real Traffic Situations," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296-305, Feb. 2014, doi: 10.1109/TITS.2013.2278494.

[33] Y. Lin and H. L. T. Nguyen, "Adaptive Neuro-Fuzzy Predictor-Based Control for Cooperative Adaptive Cruise Control System," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1054-1063, March 2020, doi: 10.1109/TITS.2019.2901498.

[34] T. Tapli and M. Akar, "Cooperative Adaptive Cruise Control Algorithms for Vehicular Platoons Based on Distributed Model Predictive Control," *2020 IEEE 16th International Workshop on Advanced Motion Control (AMC)*, Kristiansand, Norway, 2020, pp. 305-310, doi: 10.1109/AMC44022.2020.9244429.

[35] C. Anayor, W. Gao and A. Odekunle, "Cooperative Adaptive Cruise Control of A Mixture of Human-driven and Autonomous Vehicles," *SoutheastCon 2018*, 2018, pp. 1-3, doi: 10.1109/SECON.2018.8479268.

[36] "FMVSS No. 150 vehicle-to-vehicle communication technology ..." [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/v2v_pria_12-12-16_clean.pdf. [Accessed: 02-Oct-2021].

[37] B.-young Kang, B. JeongKyu, W.-C. Seo, Y. EunJu, and D.-W. Seo, "Performance analysis of wave communication under high-speed driving," *ICT Express*, vol. 3, no. 4, pp. 171–177, 2017.

[38] G. Fodor et al., "Design aspects of network assisted device-to-device communications," in *IEEE Communications Magazine*, vol. 50, no. 3, pp. 170-177, March 2012, doi: 10.1109/MCOM.2012.6163598.

[39] T. Fisher, "What does 5G mean and how fast is it?," *Lifewire*, 1-Oct-2021. [Online]. Available: https://www.lifewire.com/5g-wireless-4155905. [Accessed: 02-Oct-2021].

[40] K. C. Dey, A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin, "Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network – performance evaluation," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 168–184, 2016.

[41] T. Abbas, K. Sjöberg, J. Karedal, and F. Tufvesson, "A measurement based shadow fading model for vehicle-to-vehicle network simulations," *International Journal of Antennas and Propagation*, 14-Jun-2015. [Online]. Available: https://doi.org/10.1155/2015/190607. [Accessed: 28-Oct-2021].

[42] K. Lim and K. M. Tuladhar, "LIDAR: Lidar Information based Dynamic V2V Authentication for Roadside Infrastructure-less Vehicular Networks," *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019, pp. 1-6, doi: 10.1109/CCNC.2019.8651684.

[43] Jing Zhou and Huei Peng, "Range policy of adaptive cruise control vehicles for improved flow stability and string stability," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 229-237, June 2005, doi: 10.1109/TITS.2005.848359.

[44] J. Wang and R. Rajamani, "The impact of adaptive cruise control systems on highway safety and traffic flow," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 218, no. 2, pp. 111–130, 2004.

[45] G. Marsden, M. McDonald, and M. Brackstone, "Towards an understanding of adaptive cruise control," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 33–51, 2001.

[46] J. Ploeg, A. F. Serrarens, and G. J. Heijenk, "Connect & Drive: Design and Evaluation of Cooperative Adaptive Cruise Control for congestion reduction," *Journal of Modern Transportation*, vol. 19, no. 3, pp. 207–213, 2011.

[47] L. Ng, C. M. Clark, and J. P. Huissoon, "Reinforcement learning of Adaptive Longitudinal Vehicle Control for dynamic collaborative driving," *2008 IEEE Intelligent Vehicles Symposium*, 2008.

[48] Z. Gao, T. Sun, and H. Xiao, "Decision-making method for vehicle longitudinal automatic driving based on reinforcement Q-learning," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, p. 172988141985318, 2019.

[49] J. Yang, X. Liu, S. Liu, D. Chu, L. Lu, and C. Wu, "Longitudinal tracking control of vehicle platooning using DDPG-based PID," *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2020.

[50] S. P. Douglass, S. Martin, A. Jennings, H. Chen and D. M. Bevly, "Deep Learned Multi-Modal Traffic Agent Predictions for Truck Platooning Cut-Ins," *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 688-697, doi: 10.1109/PLANS46316.2020.9109809.

[51] A. Peake, J. McCalmon, B. Raiford, T. Liu and S. Alqahtani, "Multi-Agent Reinforcement Learning for Cooperative Adaptive Cruise Control," *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020, pp. 15-22, doi: 10.1109/ICTAI50040.2020.00013.

[52] L. Xiao and F. Gao, "A comprehensive review of the development of Adaptive Cruise Control Systems," *Vehicle System Dynamics*, vol. 48, no. 10, pp. 1167–1192, 2010.

[53] B. Brown and A. Zai, *Deep reinforcement learning in action*. Shelter Island, NewYork: Manning Publications Co., 2020.

[54] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, Massachusetts: MIT Press Ltd, 2018.

[55] G. A. Rummery and M. Niranjan, *On-line Q-learning using Connectionist Systems*. Cambridge, England: Cambridge University, Engineering Dept., 1994.

[56] C. J. Cornish and H. Watkins, "(PDF) learning from delayed rewards - researchgate." [Online]. Available: https://www.researchgate.net/publication/33784417_Learning_From_Delayed_Rewards. [Accessed: 21-Mar-2022].

[57] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[58] A. Ng, “Sparse autoencoder - stanford university,” *CS294A Lecture notes*, 2011. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf. [Accessed: 30-Mar-2022].

[59] B. van Arem, C. J. G. van Driel and R. Visser, "The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429-436, Dec. 2006, doi: 10.1109/TITS.2006.884615.

[60] J. Madiba, P. A. Owolawi and T. Mapayi, "Wi-Fi Enabled Speech Automated Guided Vehicle using Android and NodeMCU," *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2019, pp. 1-4, doi: 10.1109/IMITEC45504.2019.9015846.

[61] D. Moser, H. Waschl, H. Kirchsteiger, R. Schmied and L. del Re, "Cooperative adaptive cruise control applying stochastic linear model predictive control strategies," *2015 European Control Conference (ECC)*, 2015, pp. 3383-3388, doi: 10.1109/ECC.2015.7331057.

[62] H. Yang, H. Rakha and M. V. Ala, "Eco-Cooperative Adaptive Cruise Control at Signalized Intersections Considering Queue Effects," in *IEEE Transactions on Intelligent*

Transportation Systems, vol. 18, no. 6, pp. 1575-1585, June 2017, doi: 10.1109/TITS.2016.2613740.

[63] Z. Wang, G. Wu, and M. J. Barth, "Developing a distributed consensus-based cooperative adaptive cruise control system for heterogeneous vehicles with predecessor following topology," *Journal of Advanced Transportation*, vol. 2017, pp. 1–16, 2017.

[64] P. Eamsomboon, P. Keeratiwintakorn and C. Mitrpant, "The performance of Wi-Fi and Zigbee networks for inter-vehicle communication in Bangkok metropolitan area," *2008 8th International Conference on ITS Telecommunications*, 2008, pp. 408-411, doi: 10.1109/ITST.2008.4740296.

[65] E. Brunskill, "Lecture 6: CNNs and Deep Q Learning," *CS234: Reinforcement Learning Winter 2022*, 2022. [Online]. Available: https://web.stanford.edu/class/cs234/. [Accessed: 03-May-2022].

[66] J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094-1099, doi: 10.1109/IVS.2015.7225830.

[67] P. Polack, F. Altché, B. d'Andréa-Novel and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812-818, doi: 10.1109/IVS.2017.7995816.

[68] Y. Liu and W. Wang, "A safety reinforced cooperative adaptive cruise control strategy accounting for dynamic vehicle-to-vehicle communication failure," *Sensors*, vol. 21, no. 18, p. 6158, Sep. 2021.

[69] Volvo Cars, "Changing Adaptive Cruise Control (ACC) speed: Adaptive Cruise Control (ACC): Driver support: XC90 twin engine 2017: Volvo Support," *Volvo Cars*, 2020. [Online]. Available: https://www.volvocars.com/en-ca/support/manuals/xc90-twin-engine/2016w46/driver-support/adaptive-cruise-control-acc/changing-adaptive-cruise-control-acc-speed. [Accessed: 07-May-2022].

[70] "Courseware & Resources," *Quanser*, 27-Jan-2020. [Online]. Available: https://www.quanser.com/courseware-resources/?fwp_resource_types=manuals&fwp_resource_related_products=9706. [Accessed: 27-Oct-2021].

[71] P. S. Bithas, E. T. Michailidis, N. Nomikos, D. Vouyioukas, and A. G. Kanatas, "A survey on machine-learning techniques for UAV-based communications," *Sensors*, vol. 19, no. 23, p. 5170, 2019.

[72] D. Katic and M. Vukobratovic, "Reinforcement learning algorithms in Humanoid Robotics," *Humanoid Robots: New Developments*, 2007.

[73] K. Katija et al., "Visual tracking of deepwater animals using machine learning-controlled robotic underwater vehicles," *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 859-868, doi: 10.1109/WACV48630.2021.00090.

[74] Y. hao Li, Q. jiang Lei, C. P. Cheng, G. Zhang, W. Wang, and Z. Xu, “A review: Machine learning on robotic grasping,” *Eleventh International Conference on Machine Vision (ICMV 2018)*, 2019.

[75] L. C. Cobo, K. Subramanian, C. L. Isbell, A. D. Lanterman, and A. L. Thomaz, “Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains,” *Artificial Intelligence*, vol. 216, pp. 103–128, 2014.

Vita Auctoris

NAME:	Haoyang Ke
PLACE OF BIRTH:	Haikou, Hainan, China
YEAR OF BIRTH:	1997
EDUCATION:	Maple Leaf International School, Chongqing, Chongqing, 2015
	University of Windsor, B.A.Sc., Windsor, ON, 2019
	University of Windsor, M.A.Sc, Windsor, ON, 2022