EXTRACTION OF INFORMATION FROM AI DESIGN DECISIONS - THEORETICAL AND PRACTICAL CASE STUDY

by

NASHWAN JACOB SEBI

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2024

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Ka C. Cheok, Ph.D., Chair
Hongwei Qu, Ph.D.
Nasim Nizamoddini, Ph.D.
Aycil Cesmelioglu, Ph.D.
Sami Oweis, Ph.D.

To my intercessor, the Virgin Mary, the Seat of Divine Wisdom

ACKNOWLEDGMENTS

With all gratitude, I would like to express my deep appreciation to my esteemed Ph.D. advisory committee for their invaluable guidance during my Ph.D. journey.

Dr. Ka C Cheok deepened my understanding of control engineering, mechatronics, Artificial Intelligence, and other engineering fundamentals, equipping me with a wealth of knowledge. Dr. Hongwei Qu always made himself available for me to answer any question relevant to the thesis, especially in the application section. Dr. Aycil Cesmelioglu refreshed my mathematical skills and their application in engineering thinking. Dr. Nasim Nizamoddini generously helped me complete the tasks as intended with her notices and comments about the artificial intelligence part of the thesis. Thanks to Dr. Sami Oweis for his invaluable support in reviewing the write-up of the proposal and the dissertation and for his input during the long journey of my Ph.D.

Lastly, I am endlessly grateful to my parents (Jacob and Aleigh), my wife (Rania Sebi), my children (Kristelle, Natalie, and Jaden), my siblings, especially my brother, Dr. Laith Jacob, and all my in-laws for all their unwavering support and encouragement throughout my overseas study. Their belief in me has been an invaluable motivation source and made this goal achievable.

Nashwan Jacob Sebi

ABSTRACT

EXTRACTION OF INFORMATION FROM AI DESIGN DECISIONS - THEORETICAL AND PRACTICAL CASE STUDY

by

Nashwan Jacob Sebi

Adviser: Ka C. Cheok, Ph.D.

In recent years, the emergence of Artificial Intelligence (AI) has revolutionized numerous fields [1], such as the healthcare and automotive industries, and the technologies that use AI have become part of our daily routines. This surge in AI use in different fields has raised many issues because of a lack of transparency regarding the black-box nature of the AI algorithms and, specifically, how the neurons of the neural networks in the AI algorithms interact to conclude outcomes [2]. This problem is especially notable in the automotive industry, where AI vehicles increasingly integrate AI into their systems, especially the Advanced Driver Assistance Systems (ADAS), to improve safety and driver convenience. For engineers, consumers, and regulators, it's essential to understand the reasoning behind AI decision-making for many reasons. Accountability, among many other reasons, is considered significant in this context; for example, who will be responsible for accidents when they happen? Is it the person behind the steering wheel or the car maker? [3]. Because of those reasons, the significance of revealing the AI decision-making process by extracting information from AI's decision-

making algorithms cannot be overstated, especially in the context of ADAS and, more specifically, in Rear-End Collision Avoidance (RECA) systems.

This thesis is broken down into several sections:

1. We trained a reinforcement learning agent to operate a Rear-End Collision Avoidance (RECA) System using the Reinforcement Learning toolbox of MATLAB [4]. We used the code and Simulink model of the example of the Collision Avoidance System (Train DDPG Agent for Adaptive Cruise Control) in the Reinforcement Learning toolbox in MATLAB [5] and the car dynamics of the example of the Adaptive Cruise Control System Using Model Predictive Control in the Model Predictive Control Toolbox in MATLAB [6]. Then, we modified that example's code and Simulink model to fit my application.
2. We developed MATLAB code to extract the trained agent from the Reinforcement Learning (RL) algorithm.
3. We verified the extracted RL agent by simulating it using Simulink and then without the RL agent to confirm its necessity for the RECA system.
4. We formulated the mathematical relationship between the trained agent's inputs and output by fitting it to a Linear Regression Model (LRM) using a MATLAB function called fitlm [7].
5. We verified the LRM by simulating it using Simulink.
6. We used two scaled robotic cars to verify the LRM to confirm the reliability of the extracted information.

7. We analyzed the LRM using the State Space Control technique to determine the basis function and better understand the foundation control law of the RL algorithm.
8. In the last part, we verified the basis functions formula by simulating it using Simulink and then simulating it by excluding some of the basis functions as an extra step to confirm the necessity of all the basis functions in the set.

This in-depth theoretical and practical analysis is a significant step towards unraveling the ambiguity of AI algorithms in different fields, especially in the automotive field, by shedding light on the black box of AI algorithms and complex decision-making processes of AI.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv

ABSTRACT v

LIST OF FIGURES xiv

LIST OF ABBREVIATIONS xviii

CHAPTER ONE
INTRODUCTION 1

1.1 Artificial Intelligence 1

1.2 Artificial Intelligence is a Black Box 1

1.3 Explainable Artificial Intelligence (XAI) 2

1.3.1 Techniques of XAI 2

1.3.1.1 Function Approximation 2

1.4 Artificial Intelligence in the Automotive Industry 3

1.4.1 Advanced Diver Assistance Systems (ADAS) 3

1.4.1.1 Collision Avoidance Systems 5

CHAPTER TWO
STATEMENT OF PROBLEM 6

2.1 Reinforcement Learning (RL) 7

2.2 Using RL in The Rear-End Collision Avoidance (RECA) 7

2.3 Verifying the Trained Agent Black Box Using Simulink 7

2.4 Using XAI to Interpret the Inner Working (Black Box) of the AI Algorithm 7

2.4.1 Using Function Approximation as an XAI Technique 8

TABLE OF CONTENTS—Continued

2.5 Verifying the Fitted Model Using Simulink 9

2.6 Verifying the LRM Using Scaled Robotic Cars 9

2.7 Control and Basis Functions Analysis of the Extracted Information 11

CHAPTER THREE
REINFORCEMENT LEARNING WITH APPLICATION TO REAR-END COLLISION AVOIDANCE 12

3.1 Introduction 12

3.2 Reinforcement Learning (RL) 13

3.2.1 Key Components of Reinforcement Learning 13

3.3 Deep Deterministic Policy Gradient (DDPG) Algorithm 14

3.4 Rear-End Collision Avoidance 15

3.4.1 Currently Available Collision Avoidance Systems 15

3.5 Proposed Rear-End Collision Avoidance (RECA) System 17

3.6 Methodology of the Suggested RECA System 18

3.6.1 The Agent of the Proposed RECA System 18

3.6.2 The Environment of the Proposed RECA System 19

3.6.3 The States (Observations) of the Proposed RECA System 21

3.6.4 The Action of the Proposed RECA System 21

3.6.5 The Reward Function of the Proposed RECA System 22

3.6.5.1 Penalty Term 23

3.6.5.2 Positive Rewards 23

TABLE OF CONTENTS—Continued

3.6.6 The Goal of the Proposed RECA Reward Function 24

3.7 Configuration of the Suggested RECA System 25

3.8 Dynamics of RLA of the Suggested RECA System 25

3.9 Training the RL Agent of the Proposed RECA System 26

3.10 Extracting and Simulating the Trained RL Agent of the RECA System 29

3.11 The Results of Simulating the Trained RL Agent of the RECA System 31

3.12 The Results of Simulating the RECA System Without the RL Agent 33

CHAPTER FOUR
EXTRACTION OF INFORMATION FROM AI DESIGN DECISIONS 36

4.1 Extracting Information from the Trained RL Agent Black Box of the RECA 37

4.1.1 The Inputs and Output of the Trained RL Agent Black Box 37

4.1.2 AI Black Box and Info Extractor 39

4.2 Linear Regression Model & Basis Functions 40

4.2.1 Linear Regression Model with MSE Coefficients 41

4.3 Fit Linear Regression Model for the RECA System 42

4.4 Simulating the Extracted Linear Regression Model (LRM) 43

4.5 Simulation Results of the LRM of the RECA System 44

TABLE OF CONTENTS—Continued

CHAPTER FIVE
EXPERIMENTAL VERIFICATION OF THE EXTRACTED INFORMATION FOR THE RECA SYSTEM 47

5.1 Introduction 47

5.2 Hardware Used in the Experiments 47

5.2.1 Rear Car 48

5.2.2 Ego Car 49

5.3 Experiments Track 51

5.4 Software and Algorithm Used in the Experiment 52

5.4.1 Rear Car Algorithm 52

5.4.1.1 Rear Car Algorithm Flowchart 53

5.4.2 Ego Car Algorithm (Using the Extracted LRM) 53

5.4.2.1 Measuring, Calibrating, and Filtering Distance 54

5.4.2.2 Calculating Rear Car Speed 55

5.4.2.3 Calculating Rear Car Acceleration 56

5.4.2.4 Measuring Ego Car Speed 57

5.4.2.5 Calculating the Acceleration Equation of Ego Car 57

5.4.2.6 Ego Car Algorithm Flowchart 58

5.5 Experimental Results 58

5.5.1 Top View Camera Capture Both Cars and the Track 59

5.5.2 Camera Mounted on the Back of the Ego Car to Video the Rear Car 60

5.5.3 Camera Mounted on the Front of the Rear Car to Video the Ego Car 62

CHAPTER SIX
BASIS FUNCTIONS AND CONTROL ANALYSIS 65

6.1 Rear-End Collision Avoidance Control Theory Analysis 65

6.2 Rear-End Collision Avoidance Dynamics 66

6.2.1 Ego Car Dynamics 67

6.2.2 Rear Car Dynamics 67

6.3 Relative Distance and Speed Between the Ego and Rear Vehicles 68

6.4 Dynamics of Separation in the Time Domain 69

6.5 Control Analysis 71

6.5.1 First Set of Basis Functions for the LRM 71

6.5.2 Second Set of Basis Functions for the LRM 72

6.6 Simulating the LRM with the Basis Functions Only 73

6.7 Results of Simulating the LRM With Basis Functions 74

6.8 Analysis of Closed-Loop Stability 75

6.9 Simulating the RECA System Excluding D_{Rel} from the Basis Functions 76

TABLE OF CONTENTS—Continued

CHAPTER SEVEN
CONTRIBUTION, RECOMMENDATION, CONCLUSION
AND FUTURE WORK 79

7.1 Contribution 79

7.2 Recommendation 80

7.3 Conclusion 81

7.4 Future Work 83

REFERENCES 84

LIST OF FIGURES

Figure 1.1	AI perceived as a black box	1
Figure 2.1	Function Approximation method as an XAI approach	9
Figure 2.2	Sensors used in the ego and rear car	10
Figure 2.3	Shot of the experiments' setup	10
Figure 3.1	Five major components of the reinforcement learning (RL) algorithm	14
Figure 3.2	Forward Collision Avoidance systems, Rear is Ego car	15
Figure 3.3	RECA system using AEB or RCTW to help with parking	16
Figure 3.4	Dependent RECA System, Rear, and Front both are Ego Cars	17
Figure 3.5	Proposed RECA system, the front is the ego car	17
Figure 3.6	The RL agent goal is to maintain the relative distance $\geq$ the safe distance	19
Figure 3.7	The input/output of the RL agent of the proposed RECA system	19
Figure 3.8	Simulink blocks of the dynamics of the rear car	20
Figure 3.9	Simulink blocks of the dynamics of the ego car	20
Figure 3.10	The action (output) of the RL agent of the RECA system	22
Figure 3.11	Inputs/output Simulink diagram of the RL agent of the RECA system	24
Figure 3.12	The suggested RECA system configuration	25
Figure 3.13	Dynamics of the RLA of the proposed RECA system	26
Figure 3.14	Computers used to train the RECA RL agent	27
Figure 3.15	Example of training processes of the RECA system with 2000 episodes	28

LIST OF FIGURES—Continued

Figure 3.16	Simulink model of the training of the RECA system	29
Figure 3.17	Simulink model of the simulator of the RL agent of the RECA system	30
Figure 3.18	Inputs (observations)/output of the trained RL agent of the RECA system	30
Figure 3.19	Speeds of Ego and Rear cars of the trained RL agent of the RECA system	31
Figure 3.20	Speeds of Ego and Rear cars of the trained RL agent (Zoomed-in shot)	31
Figure 3.21	Relative distance of the trained RL agent of the RECA system	32
Figure 3.22	A snippet captured from the RECA trained RL agent simulation	33
Figure 3.23	RECA system when the RL agent is inactive	34
Figure 3.24	Speeds of Ego and Rear cars when the RL agent is inactive	34
Figure 3.25	Relative distance when the RL agent is inactive (without RL agent)	35
Figure 3.26	Snippet of the collision between the two cars when the RL agent is inactive	35
Figure 4.1	Saving input and output data of RL agent in a .mat file	37
Figure 4.2	Trained RL agent inputs and output plots	38
Figure 4.3	Trained RL agent inputs and output plots (zoomed-in)	38
Figure 4.4	Tuning an Info Extractor to emulate an AI block box (AIBB)	39
Figure 4.5	Simulink model of the simulation of the LRM of the RECA system	43
Figure 4.6	Inputs/output of the Simulink function of the RECA LRM	43
Figure 4.7	The function calculates the output of the LRM of the RECA	44

Figure 4.8	Relative distance of the LRM of the RECA system	44
Figure 4.9	Speeds of ego and rear cars of the LRM of the RECA system	45
Figure 4.10	Snippet of the animation of the LRM	45
Figure 5.1	Differential drive scaled robotic cars used in the experiments	46
Figure 5.2	Rear car with the plastic reflector and ultrasonic sensor	48
Figure 5.3	Ego car with the ToF ranging sensor	49
Figure 5.4	Track and webcam used in the experiments	51
Figure 5.5	Rear car algorithm flowchart	52
Figure 5.6	Ego car algorithm flowchart	57
Figure 5.7	Webcam on a tripod to record footage of the track with both cars (top view)	58
Figure 5.8	Scheme of videoing the track along with the two cars (top-view)	58
Figure 5.9	Some pictures from a video recorded by the top-view camera	59
Figure 5.10	Webcam is installed on the back of the ego car to video the rear car	60
Figure 5.11	Scheme of ego car's webcam videoing the rear car (top-view)	60
Figure 5.12	Some shots captured by the camera of the ego car	61
Figure 5.13	Webcam mounted to the rear car captures the front car	61
Figure 5.14	Scheme of rear car's webcam videoing the ego car (top-view)	62
Figure 5.15	Some shots captured by the rear car camera	62
Figure 6.1	The Dynamics of the RECA System	64

LIST OF FIGURES—Continued

Figure 6.2	The block diagram of the extended State Space Model	71
Figure 6.3	Simulink model of the first set of the basis functions of the LRM	73
Figure 6.4	Ego and rear car speeds of the 1st set of the LRM basis functions	71
Figure 6.5	Relative distance of the 1st set of the LRM basis functions	72
Figure 6.6	Simulink model of the RECA system excluding D_{Rel} from basis functions	76
Figure 6.7	Speeds of Ego and Rear cars when excluding D_{Rel} from basis functions	77
Figure 6.8	Relative distance when excluding D_{Rel} from basis functions	78
Figure 6.9	Snippet from the animation when excluding D_{Rel} from basis functions	78
Figure 7.1	Forward and Rear-Ends Collision Avoidance	86

LIST OF ABBREVIATIONS

LiDAR	Light Detection and Ranging
AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
ADAS	Advanced Driver Assistance Systems
AVs	Autonomous Vehicles
RECA	Rear-End Collision Avoidance
REC	Rear-End Collision
FCA	Forward Collision Avoidance
RCTW	Rear Cross-Traffic Warning
RL	Reinforcement Learning
RLA	Reinforcement Learning Algorithm
DDPG	Deep Deterministic Policy Gradient
ToF	Time of Flight
AEB	Automatic Emergency Braking
DARPA	Defense Advanced Research Projects Agency
IE	Info Extractor
AIBB	AI Black Box
LR	Linear Regression
MSE	Mean Squared Error
LSE	Least Squared Error

LPF	Low-Pass Filter
IR	Infrared
LED	Light Emitting Diode
V_{Ego}	Ego Car Speed
V_{Rear}	Rear Car Speed
V_{Rel}	Relative Speed
A_{Rel}	Relative Acceleration
D_{Rel}	Relative Distance
A_{Rear}	Rear Car Acceleration
E_{Acc}	Ego Car Acceleration
LRM	Linear Regression Model
TCS	Traction Control Systems
ESC	Electronic Stability Control
ABS	Anti-lock Braking System
ACC	Adaptive Cruise Control
ML	Machine Learning
BFA	Basis Function Analysis
TRLA	Trained Reinforcement Learning Agent
CTA	Control Theory Analysis
TF	Transfer Function

SS	State Space
SSE	State Space Equation
SSM	State Space Model
SSC	State Space Control
OCD	Optimal Control Design
LTI	linear Time-Invariant
LDW	Lane Departure Warning

CHAPTER ONE

INTRODUCTION

1.1 Artificial Intelligence

Artificial Intelligence (AI) is a revolutionary concept that enables machines to mimic human intelligence [8]. It has already become an integral part of our daily lives, enhancing various sectors such as healthcare, finance, retail, education, entertainment, agriculture, energy, and even the automotive industry [9].

1.2 Artificial Intelligence is a Black Box

While AI is already used in many applications and has moved ahead in various fields, the interactions among the neurons of neural networks of AI techniques are highly nonlinear, complex, and unclear. The need for more transparency and insight into how decisions are made inside neural networks raises many issues, such as ethical, accountability, fairness, and regulatory issues among stakeholders, including end users, policymakers, and regulators. This unclarity of how AI produces certain decisions often contributes to the perception of AI as a "black box." see Figure 1.1.

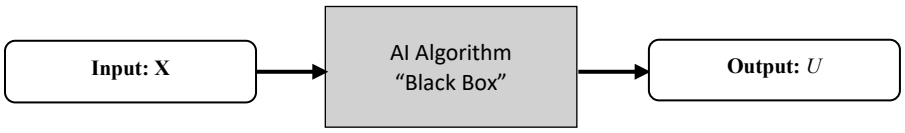

Figure 1.1 AI perceived as a black box

1.3 Explainable Artificial Intelligence (XAI)

The uncertainty and doubts about the inner workings of AI algorithms incentivized the stakeholders in various domains, especially in sensitive fields such as healthcare, finance, and automotive, to demand transparency and explanations for AI decisions. Responding to these concerns, techniques, and methodologies emerged in the second decade of the 21st century that aimed to clarify the reasoning behind the decision-making of AI systems. These techniques are called Explainable Artificial Intelligence (XAI). Essentially, XAI seeks to bridge the gap between the "black box" nature of many AI algorithms and the need for transparency and accountability in decision-making processes [10].

1.3.1 Techniques of XAI

Here are some examples of Explainable AI (XAI) techniques and methodologies:

1. Function Approximation: This technique provides information about the overall performance of the model and the individual contributions of each feature (input) in predicting the output.
2. Feature Importance Analysis: This technique identifies which features or variables are the most influential in a model's predictions.
3. Counterfactual Explanations: Counterfactual explanations provide insights into how changes in input variables would affect the model's predictions.
4. Surrogate Models: Surrogate models approximate the behavior of the original model and provide insights into how it makes predictions.

1.3.1.1 Function Approximation Function approximators, such as linear regression (LR), are crucial in modeling relationships between input and output variables

in various domains. This method approximates complex functions by fitting simpler models to observed data. Linear regression, for instance, approximates functions by assuming a linear relationship between the input variables and the output [11]. Through their ability to generalize data patterns, function approximators are powerful tools for understanding and modeling real-world phenomena, driving advancements in research and industry applications.

1.4 Artificial Intelligence in the Automotive Industry

Artificial intelligence is becoming more potent in the automotive industry, enhancing features like Advanced Driver Assistance Systems (ADAS) and overall vehicle performance. From ADAS to AVs, the benefits of AI in the automotive industry are countless [12]. In today's world, everything around is smart, such as TVs, tablets, computers, and phones, which leads consumers to want their cars to be as clever as these smart devices. Indeed, with the help of AI, today's modern vehicles are intelligent in many aspects.

1.4.1 Advanced Driver-Assistance Systems (ADAS)

Advanced Driver Assistance Systems (ADAS) are technological features that are vital to ensuring a safer, efficient, and convenient driving experience. ADAS primarily focuses on collision avoidance technologies, such as Forward Collision Avoidance (FCA), Lane Departure Warning (LDW), and blind-spot applications; ADAS also covers driver aids, such as night vision, driver alertness, and adaptive cruise control, among many other technologies that are classified under ADAS that are available in today's cars. We can classify ADAS technologies into two major classes: passive, where the technology alerts the driver to act, such as lane departure warning, and active, where the

technology acts without human intervention, such as lane-keeping and centering technology. The more active (automated) ADAS features integrated into the vehicles, the closer we get to fully Autonomous Vehicles (AVs).

ADAS has a long history that started over 120 years ago and has been enhanced over the decades since then. Windshield wipers, turn signals, and rear-view mirrors were safety features added to vehicles in the early 20th century that would later become part of ADAS [13]. The first modern ADAS technology developed in the 1960s was Antilock Braking Systems (ABS) [14]. ABS prevents wheels from locking up during braking, thus allowing the driver to maintain steering control. ABS was invented in 1971 by Mario Palazzetti (known as 'Mister ABS') and became popular in many 1970s cars [15]. Automakers introduced various ADAS features to improve safety and convenience throughout the late 20th century and into the early 2000s, such as Traction Control Systems (TCS), Electronic Stability Control (ESC), Adaptive Cruise Control (ACC), and Automatic Emergency Brake (AEB) [16]. In the 21st century, ADAS capabilities rapidly expand, “driven by advancements in computing power, sensor technology, and artificial intelligence” [17]. ADAS systems highly depend on sensors to monitor the surrounding environment and detect potential hazards. Developing cost-effective and reliable sensor technologies, such as radar, LiDAR, ultrasonic, and cameras, revolutionized ADAS capabilities [18].

The concept of autonomous driving has been a long-term goal in the automotive industry. Research institutions, universities, and companies have worked on autonomous vehicle technology for decades [19]. While fully autonomous vehicles have yet to become mainstream, advancements in sensor technology, computing power, and artificial

intelligence have brought the industry closer to this goal. As technology continues to evolve, ADAS will likely play an even more significant role in shaping the future of transportation and ensuring safety [20].

1.4.1.1 Collision Avoidance Systems

Collision Avoidance systems are considered essential technologies integrated into modern cars [21]. These technologies were developed to prevent or mitigate these collisions, and they use different sensors, such as LiDAR, radar, camera, and ultrasonic [22].

Most collision avoidance systems available in today's cars are Forward Collision Avoidance (FCA) systems designed to be installed in the rear vehicles to avoid colliding with the front car in a forward motion (both cars moving forward) [23]. Other collision avoidance systems are RECA systems designed to avoid colliding with vehicles behind in reverse motion, such as Rear Cross-Traffic Warning (RCTW) systems [24], usually used in self-parking and parking assist systems [25]. Some researchers suggested Rear-end Collision avoidance (RECA) systems in forward motion. Still, those suggested designs conditioned that all cars be equipped with sensors and communicating devices to alert each other when the separation distance drops below the safe distance [26]. This kind of system didn't see the light because it was impractical, requiring all the cars to have these same sets of communication devices and sensors.

CHAPTER TWO

STATEMENT OF PROBLEM

Despite AI being approved as a powerful tool that could replace or integrate human rules in domains such as health, finance, customer service, education, automotive, and entertainment [27-32], the strategy of these algorithms that use neural networks, which could be constructed from many layers with many neurons, is unclear [33]. This ambiguity regarding how these algorithms process data and produce decisions is because the neurons of these networks show highly non-linear interaction among them [34], making it very challenging even for computer scientists to interpret. The success that systems that use AI showed didn't wave the users' and stakeholders' need to understand the underlying rules and strategies these systems use to make decisions so they can trust [35]. For example, doctors who use an AI tool to decide on the best treatment for their patient's needs, the doctors need to understand why this tool comes up with its recommendation about the treatment so the doctors can trust and be ready to answer patients' questions if they have concerns about those tools. That example could be applied to other fields as well. Considering the increase in our modern lives' dependence on AI, blind trust in AI technologies could lead to severe and unexpected consequences in wide spans [36], especially in critical fields like health and finance [37]. The need for ways to interpret these algorithms is increasing not just to avoid those undesired impacts but also for legal, ethical, transparency, and risk management reasons [38-42].

2.1 Reinforcement Learning (RL)

This thesis's case study uses the Reinforcement Learning (RL). Reinforcement Learning is an AI technique used to solve control problems [43] by training an agent who learns through a series of interactions with the environment [44]. Through each interaction, the agent observes some states from the environment, takes action and receives feedback from the environment as a reward or penalty. After many interactions, the agent modifies and tunes its policy and learns that some actions in given states are more rewarding than others, reinforcing these actions [45].

2.2 Using RL in The Rear-End Collision Avoidance (RECA)

We used the Deep Deterministic Policy Gradients (DDPG) RL algorithm [46] to implement a Rear-End Collision Avoidance System (RECA). We have two cars in this scenario: the Ego (front) and the Rear car. We trained an RL agent to operate the RECA system in the ego car to avoid colliding with the rear car when both cars are in forward motion. The RL agent was able to handle the task as intended; however, it was still a black box in terms of the algorithm used to train it. Next, the task was extracting the trained RL agent as a stand-alone object.

2.3 Verifying the Trained Agent Black Box Using Simulink

Using Simulink of MATLAB, the reliability of the trained agent black box was verified with different scenarios, covering different initial positions and speeds for both the ego and the rear cars.

2.4 Using XAI to Interpret the Inner Working (Black Box) of the AI Algorithm

An explainable AI (XAI) method is used to demystify the ambiguity of the AI algorithm that is perceived as a black box [47]. The trained RL agent operating the Rear-

End Collision Avoidance System (RECA) system in the ego (lead) car achieved the task as intended by keeping a safe distance from the rear car; however, the agent's inner algorithm is still a black box. The RL agent of the RECA receives five states (features) from the environment as inputs and generates an action, which is the accelerating force of the ego car as output. The nature of the interactions among these states leading to specific output is not explicit. To unravel the ambiguity of the RL agent algorithm, we could fit the system to a Linear Regression Model (LRM) using fitlm (MATLAB function), which stands for Fit Linear Model.

2.4.1 Using Function Approximation as an XAI Technique

Function approximation, the XAI method used in this thesis, provides transparent and understandable explanations of the AI core's inner workings to clear the users' doubts and concerns. Investigating the basis functions that model the AI algorithm is crucial because they interpret the reasoning behind AI decisions using regular mathematics. This method provides information about the system's overall performance and individual feature contribution. The simplest form of these approximated functions could be represented with an output (y), which is provided as a function of the inputs ($x_1, x_2, x_3, \dots, x_n$) is shown in equation (2.1).

$$y = k_1 + f(a_1x_1, a_2x_2, a_3x_3, \dots, a_nx_n) \tag{2.1}$$

The inputs are usually called the states, k, is constant (bias). Figure 2.1 shows the block diagram of the function approximation diagram.

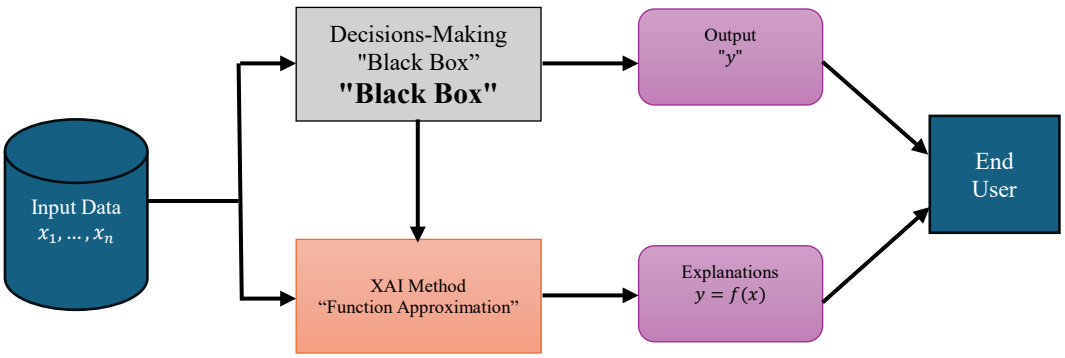

Figure 2.1 Function Approximation method as an XAI approach

2.5 Verifying the Fitted Model Using Simulink

Using Simulink of MATLAB, the accuracy of the fitted linear regression model (LRM) was verified with different scenarios, such as different initial positions and speeds for both the ego and the rear cars, to test the ego car's performance under various scenarios.

2.6 Verifying the LRM Using Scaled Robotic Cars

Verifying the linear regression model (LRM) was taken to the next level using two scaled robotic cars for the rear and ego cars. A thirty-five-foot-long tack was prepared in my house's basement for the experiments. The two robotic cars, rear and ego, were equipped with all the sensors and microcontrollers required to complete their tasks, see Figure 2.2, which shows the ego and ear cars and some of the sensors and IR reflector of the rear car to reflect better the light of the Tof ranging sensor of the ego car. Figure 2.3 depicts the experiments' setup.

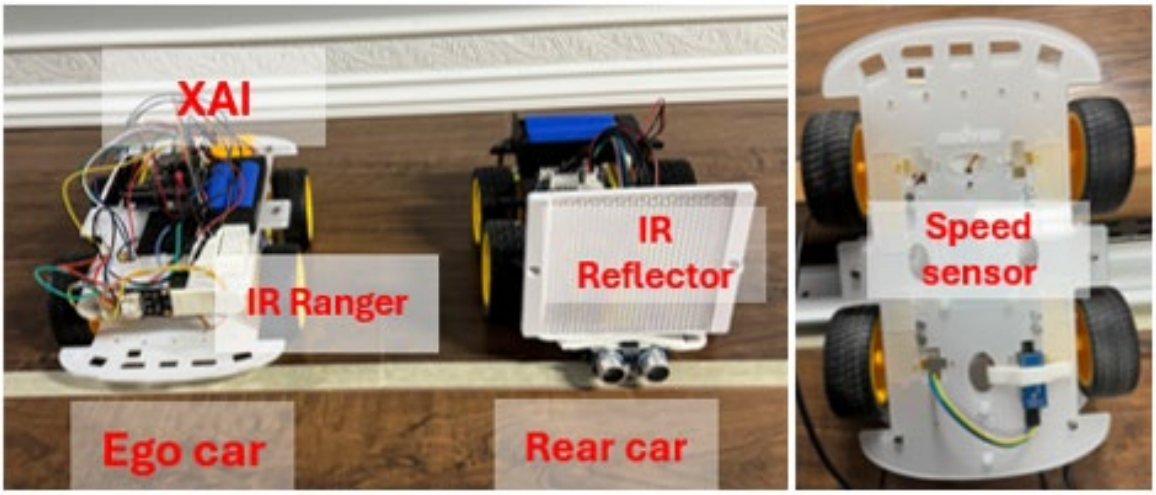

Figure 2.2 Sensors used in the ego and rear car

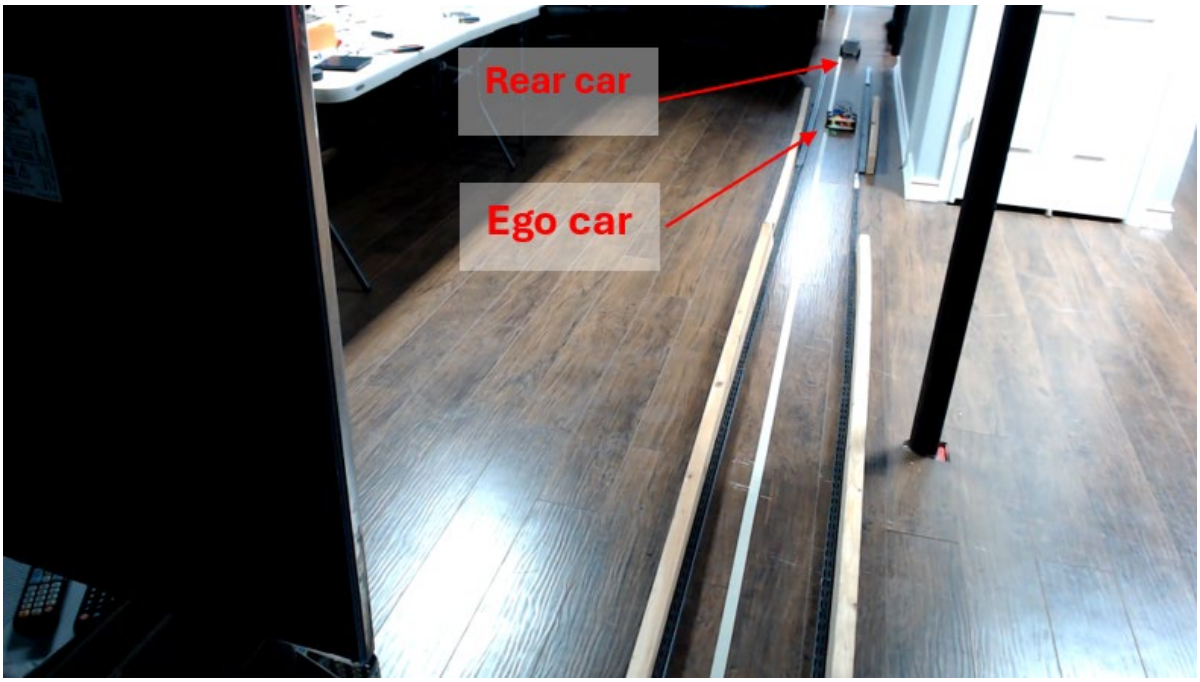

Figure 2.3 Shot of the experiments' setup

2.7 Control and Basis Functions Analysis of Extracted Information

In chapter 6 of the dissertation, using a state-space control, we analyzed the Rear-End Collision Avoidance System to reveal the basis (essential) functions that represent the system's most influential features. Then, we analyzed the stability of the linear egression model with basis functions only using the state-space control model (Counterfactual Explanations). This analysis provides a deep understanding of the system's nature and explains the reasons why the system worked as intended. Based on our initial experience with our system, we trained the RL agent using five states (features), which we thought were necessary for the agent to make decisions. These five states were Ego car speed (V_{Ego}), Rear car Speed (V_{Rear}), Relative Speed (V_{Rel}), Relative Distance (D_{Rel}), and the Relative Acceleration (A_{Rel}). The basis functions and stability analysis helped us specify the fitted model's most dominant and least influential states. The analysis suggested only three dominant states: V_{Ego}, V_{Rear}, and D_{Rel}; one redundant state, V_{Rel}; and one influential state, A_{Rel}. We simulated the XAI model with the basis functions to verify our control and stability analysis. Then, we excluded the influential basis functions from the linear regression model to see the impact on the system.

CHAPTER THREE

REINFORCEMENT LEARNING WITH APPLICATION TO REAR-END COLLISION AVOIDANCE

3.1 Introduction

A rear-end collision occurs when a car collides with the vehicle ahead from behind. Such accidents occur at traffic signals, stop signs, or in heavy traffic conditions. While most rear-end collisions are low-speed incidents, they can also happen on highways. Rear-end accidents typically involve two vehicles, sometimes leading to a domino effect, impacting multiple cars [48].

This case study of this thesis investigates the potential of using reinforcement learning (RL) methods to train an agent to operate a Rear-End Collision Avoidance (RECA) system. This thesis focuses on a scenario involving two vehicles, Ego (front) and Rear cars. The RL agent is to be integrated into the Ego car to prevent collisions with the Rear car while both are moving forward.

By using a reinforcement learning algorithm, the RECA system can learn from its interactions with the environment, adapt to changing scenarios, and make intelligent decisions to mitigate collisions effectively [49]. This chapter highlights different aspects of the RL technique. Then, it explains the RL algorithm we modified to operate as a RECA system for the front (Ego) car to mitigate collisions with the rear car when both move forward.

3.2 Reinforcement Learning (RL)

Reinforcement learning is a Machine Learning (ML) technique [50] for solving control problems. RL learns from interacting with the environment and receiving feedback (reward). It reinforces fruitful actions that result in better rewards [51]. That's where the name Reinforcement originated because it reinforces the actions that return higher rewards [52].

3.2.1 Key Components of Reinforcement Learning

There are five key components of Reinforcement Learning.

1. **Agent:** is the entity that makes decisions and learns from the environment through interactions [53].
2. **Environment:** the space where the agent operates [54].
3. **Rewards:** are scalar feedback signals provided by the environment to the agent [54].
4. **States (observations):** are the scenarios or situations the agent encounters [55].
5. **Actions:** are an RL agent's possible moves or decisions within its environment [56].

The block diagram of RL [57] is shown in Figure 3.1 below. There are two main types of RL policies: deterministic and stochastic. The key difference between deterministic and stochastic policies is how they choose actions. A deterministic policy always picks just one action for each situation (state). A stochastic policy chooses from a probability distribution over actions for each state [58].

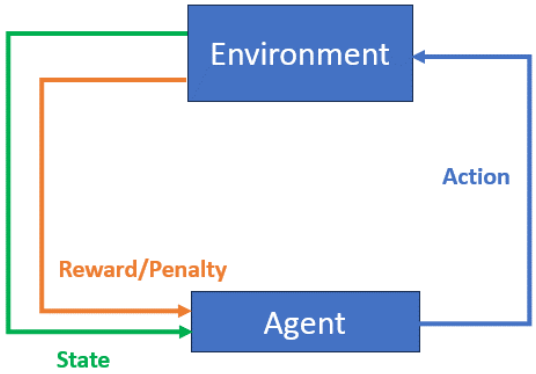

Figure 3.1 Five major components of the reinforcement learning (RL) algorithm [57]

3.3 Deep Deterministic Policy Gradient (DDPG) Algorithm

This thesis investigates the application of the Deep Deterministic Policy Gradient (DDPG) algorithm in the context of rear collision avoidance [49]. The DDPG algorithm operates as a model-free, online, and off-policy [59] method within the reinforcement learning domain. An agent utilizing DDPG maximizes the expected cumulative reward over the long term [46]. DDPG is an off-policy algorithm that learns from experience; therefore, it is well-suited to my application because the environment is a continuous action space where actions are not limited to a finite set of options. We use the Reinforcement Learning Toolbox of MATLAB, which provides many functions, and Simulink blocks for training policies using reinforcement learning algorithms, including DDPG and many other algorithms [4]. DDPG is considered a hidden part from the user. In our case, the front (Ego) hosts the RL algorithm (DDPG), which learns from interactions with the environment to maintain a safe separating distance from the rear car when both move forward.

3.4 Rear-End Collision Avoidance

Rear-end collision avoidance is very critical for road safety. Rear-end collisions (RECs) are considered one of the most common types of road crashes. According to the National Highway Traffic Safety Administration, RECs comprise approximately 23 percent of all motor vehicle crashes. This kind of vehicle accident generates bodily injury and millions of dollars in property damage and losses yearly [60]. Utilizing RCEA technology can create a safer and more secure road environment for everyone by helping to prevent accidents, mitigating deaths and injuries [48], protecting vulnerable road users [61], and enhancing driver comfort [62].

3.4.1 Currently Available Collision Avoidance Systems

Researchers have covered the subject of collision avoidance extensively. However, in most cases, the system is incorporated into the rear car, focusing on preventing collisions with the vehicles in front [63]; this system is called forward collision avoidance (FCA), as illustrated in Figure 3.2. Studies show that vehicles with forward collision (front-end) warning and automatic emergency braking significantly reduced forward collisions [64]. The rear car hosts the FCA system, so it is the ego car (which hosts the sensory system) in this kind of system.

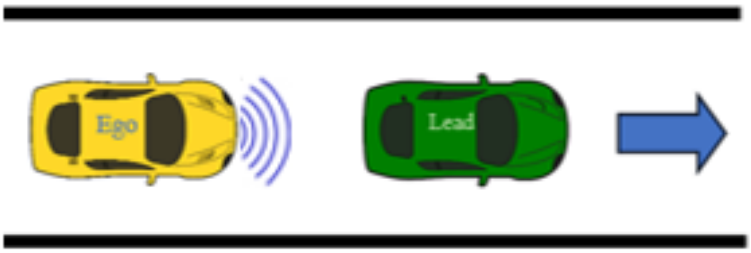

Figure 3.2 Forward Collision Avoidance systems, Rear is the Ego car

While other kinds of RECA systems are installed in the front cars, designed to prevent collisions with vehicles behind them, they are typically only discussed in Reverse Automatic Emergency Brake (AEB) systems and Rear Cross-Traffic Warning (RCTW) systems [65]. These systems are often used to help with parking or self-parking technologies, as illustrated in Figure 3.3. These systems often use AEB-rear systems.

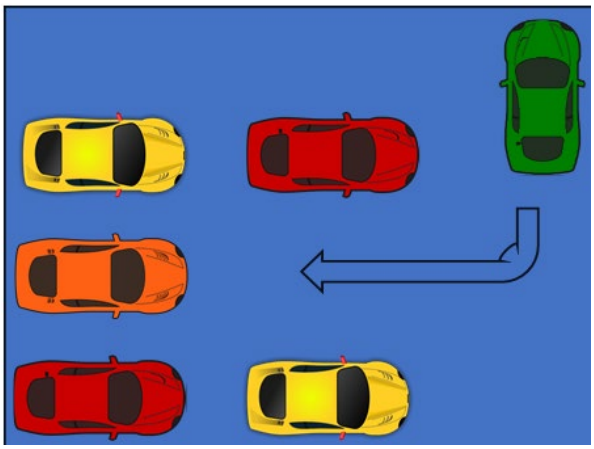

Figure 3.3 RECA system using AEB or RCTW to help with parking

Other collision systems are considered dependent systems, as both cars must communicate using compatible communication transceivers integrated into both vehicles [26]. These systems are impractical as it is hard to guarantee that all road vehicles have compatible communication components (transmitters and receivers). Plus, that would be costly as both cars should be equipped with sensors and other communication devices; in this case, both are ego cars, see Figure 3.4. In addition to the above reasons, these systems are impractical because any system faults in any of the cars may increase the chances of the vehicles colliding.

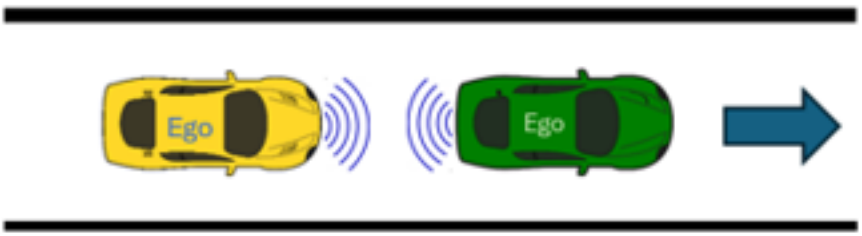

Figure 3.4 Dependent RECA System, Rear, and Front both are Ego Cars

3.5 Proposed Rear-End Collision Avoidance (RECA) System

In this thesis, we introduced an RECA system that uses an RL algorithm tailored for integration into front (Ego) cars to mitigate collisions with rear vehicles when both are in forward motion, as illustrated in Figure 3.5. The ego is the front car. It's critical to mention here that in our RECA system, we assume this system would only work if all the following conditions are satisfied: We have only two cars, a single-lane road, no car is in front of the ego car (there is no limit on the ego car's speed), and lane change is unavailable (lane change is not allowed).

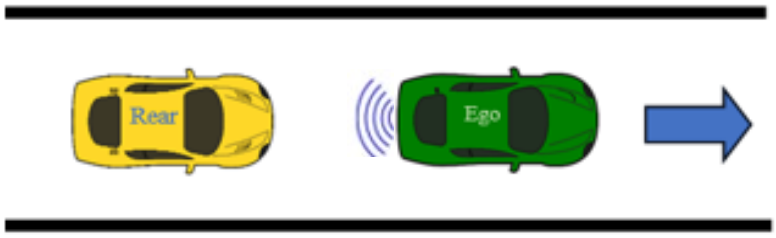

Figure 3.5 Proposed RECA system, the front is the ego car

The suggested RECA system is novel because it is independent and implemented only in the front car. It will avoid colliding with the rear vehicle in forward motion. The

outcomes of this research are expected to contribute to the development of advanced independent Rear-End Collision Avoidance (RECA) systems in front cars that will enhance road safety, reduce accidents, and mitigate deaths and injuries caused by Rear-End Collisions (RECs).

3.6 Methodology of the Suggested RECA System

The methodology of training an RL agent as a decision maker installed in the ego (front) to avoid colliding with the rear car involves using the "Train DDPG Agent for Adaptive Cruise Control" example [5], found in the MATLAB Reinforcement Learning toolbox [4]. That example was an excellent starting point because it already has a setup of two cars with their car dynamics. However, we modified that example to fit our application; the modification covered the MATLAB code, the Simulink model, changing the number of states, and crafting a new reward function. The five components of the RL algorithm were the first step in this process that we needed to determine.

3.6.1 The Agent of the Proposed RECA System

The agent is the decision-making entity. The ego (front) car hosts the trained RL agent of the RECA system. The agent gets rewarded or penalized based on the outcome of their action in each given state. Based on this exploration mechanism, the agent will learn how to maintain a safe distance from the rear car by reinforcing the actions that return higher rewards (keeping a safe distance). We used a Deep Deterministic Policy gradient (DDPG) based agent, which fits our application because it is particularly well-suited for continuous and high dimensional action spaces. The trained RL agent's tasks are depicted in Figure 3.6. The trained RL agent in the ego car task is to avoid colliding with the rear car by keeping a relative distance $\geq$ the safe distance ($D_{rel} \geq$ 10 m).

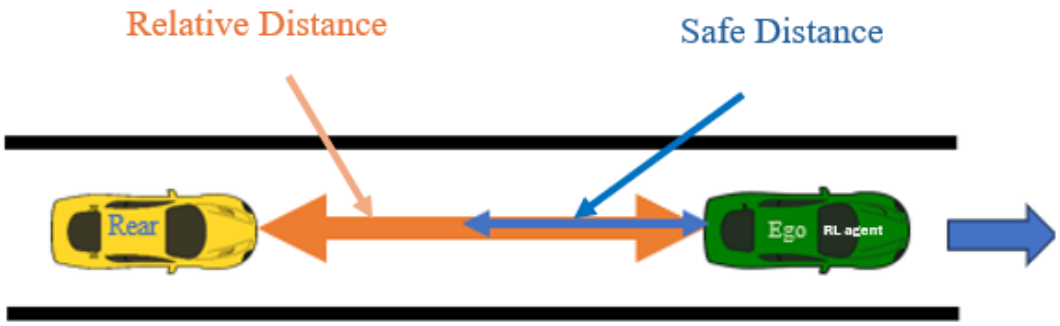

Figure 3.6 The RL agent goal is to maintain the relative distance ≥ the safe distance

The agent of the suggested Rear-End Collision Avoidance (RECA) takes five states ($V_{Ego}, V_{Rear}, V_{Rel}, D_{Rel}, A_{Rel}$) and produces an action that is the accelerating force for the ego car, as shown in Figure 3.7.

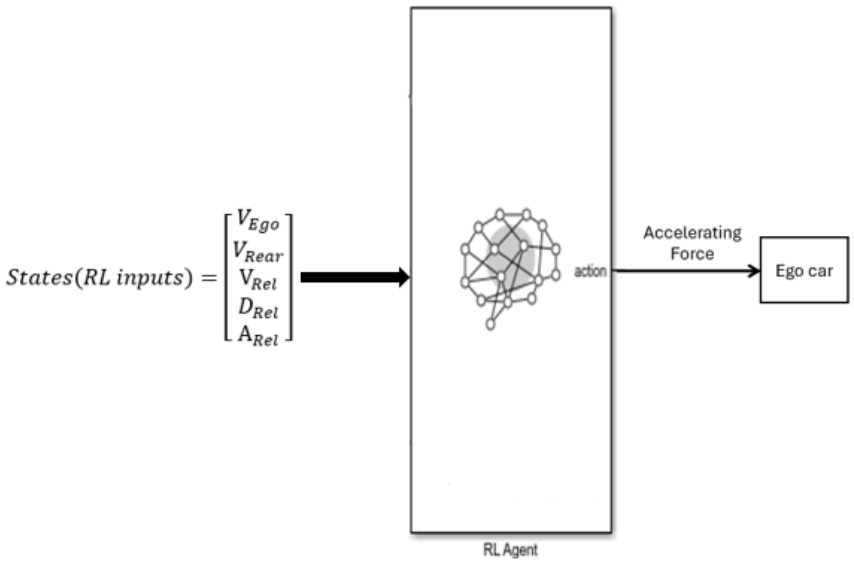

Figure 3.7 The input/output of the RL agent of the proposed RECA system

3.6.2 The Environment of the Proposed RECA System

The environment is the space where the agent operates. In our case, the environment consists of two cars traveling on a single-lane road. Besides the road

elements, the RECA environment consists of two vehicles, the ego and the rear. We imported the same environment as the Train DDPG Agent for Adaptive Cruise Control example [6] with some changes, such as changing the number of observations from three to five, to fit our application. Figures 3.8 and 3.9 depict the Simulink blocks of the dynamics of the rear and ego cars, respectively.

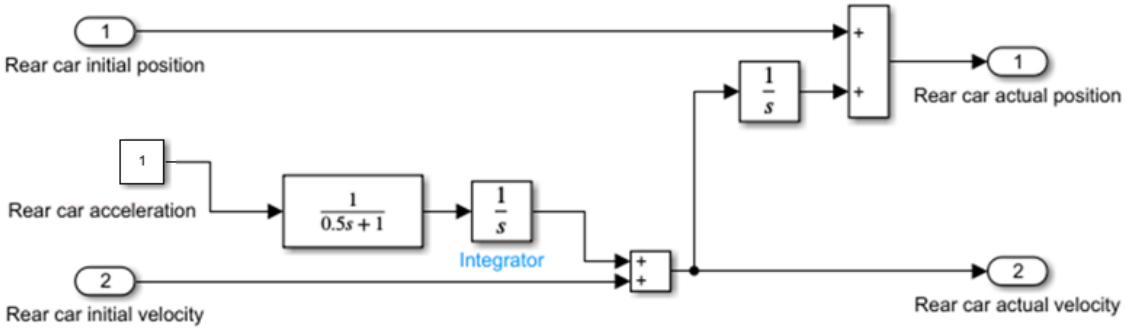

Figure 3.8 Simulink blocks of the dynamics of the rear car

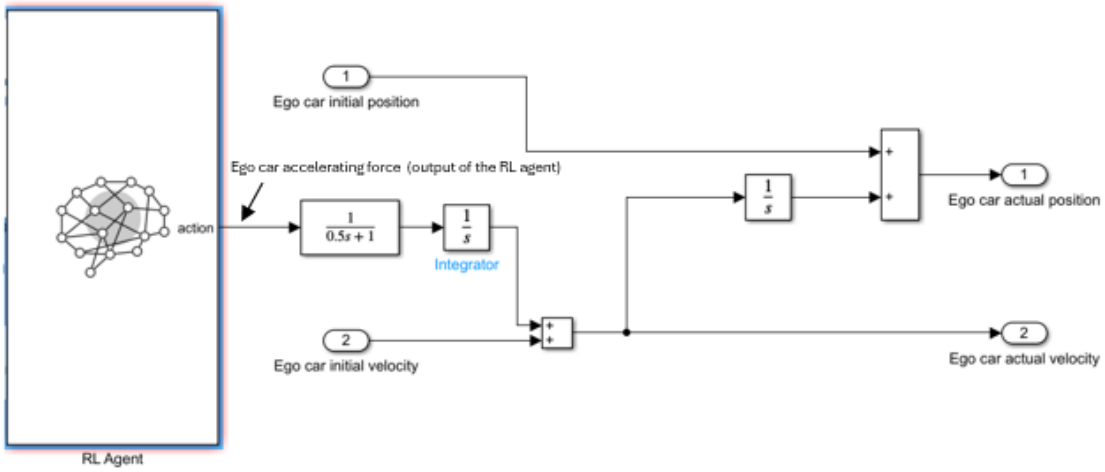

Figure 3.9 Simulink blocks of the dynamics of the ego car

3.6.3 The States (Observations) of the Proposed RECA System

The states are the scenarios or situations that the agent encounters due to the actions taken by the RL agent. Picking the right states is crucial for the RL algorithm, as these states are the most influential factors in training the RL agent to implement the expected tasks. For the RL agent of the RECA system installed in the front car, it is vital to have continuous access to useful information to learn which actions will be well rewarded by the environment. The states for the RECA system are the speed of the Ego car (V_{Ego}), speed of the Rear car (V_{Rear}), relative speed (V_{Rel}), relative distance (D_{Rel}), and relative acceleration (A_{Rel}) as shown in equation (3.1) above.

3.6.4 The Action of the Proposed RECA System

The RL agent takes an action for every given state (observation) from the environment. In our application, the action is the accelerating force of the ego car, tuned during the exploration stage of the training process by experimenting with many scenarios and isolating the actions that return higher rewards from poorly rewarded ones. These experiments are called episodes, and each episode refers to a series of interactions between an agent and its environment, starting from the initial state to the ending (reaching the end of simulation duration, which is 180 seconds in my application) or to a predetermined ending state (collision between the rear and ego car take place). In each episode, based on the rewards received, the agent updates its policy and fine-tunes it. The underlying concept the trained agent of the RECA system uses to produce its actions, which is the accelerating force fed to the dynamics of the ego car, is a black box. Figure 3.10 shows the Simulink block diagram of the RL agent action (accelerating force) of the RECA system driving the dynamics of the ego car.

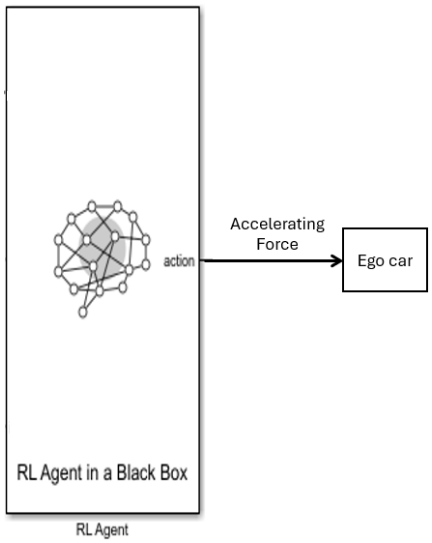

Figure 3.10 The action (output) of the RL agent of the RECA system

3.6.5 The Reward Function of the Proposed RECA System

The rewards in the RL algorithm are a scalar feedback signal provided by the environment to the agent. This feedback is crucial for the RL agent to learn which actions are more rewarding and which are detrimental. Concerning our application of the RECA system, if the actions taken by the RL agent led the Ego car to avoid colliding with the rear car by maintaining at least a safe distance (10 meters) or higher $D_{rel} \geq$ 10 m, the agent will be rewarded with a high score, but if $D_{Rel} < safe\ distance$ the agent may receive a lower score or a significant negative reward for a collision when $D_{Rel} < Zero\ or\ V_{Ego} < Zero$. The reward function that we formulated for the RECA system is shown in equation (3.3):

$$r_t = -(0.7\ V_{Rel_t}^2 + A_{Rel_t}^2) + G_t + K_t \tag{3.3}$$

- V_{Rel_t} is the relative velocity between the vehicles (how fast their distance changes).
- A_{Rel_t} is the relative acceleration (how fast the relative velocity is changing)
- G_t, K_t.are two complementary values that foster actions aimed at maximizing rewards, G_t = 5, if the relative distance $D_{Rel_t} \geq$ 10 meters ; K_t =10, if there is no colliding ($D_{Rel} > Zero\ and\ V_{Ego} > Zero$); Otherwise, they equal Zero.

Below is the breakdown of the reward function for the RECA RL algorithm. Here's a step-by-step explanation:

3.6.5.1 **Penalty Term:** $-(0.7\ V_{Rel_t}^2 + A_{Rel_t}^2)$

This term penalizes the agent based on the relative velocity and acceleration:

- $V_{Rel_t}^2$: Squared relative velocity. A higher relative velocity (i.e., the vehicles are moving toward each other quickly) results in a higher penalty.
- $A_{Rel_t}^2$: Squared relative acceleration. A higher relative acceleration (i.e., the rate at which the relative velocity changes) also produces a higher penalty.

The coefficients (0.7 for velocity and 1 for acceleration) scale the penalties for these factors. This penalization encourages the vehicle to maintain a lower relative velocity and smoother relative acceleration, which helps avoid collisions.

3.6.5.2 **Positive Rewards:** $+ G_t + K_t$

G_t, K_t are two complementary values that foster actions aimed at maximizing rewards, G_t = 5 if the relative distance $D_{Rel_t} \geq$ 10 meters ; K_t =10 if there is no colliding $D_{Rel} > Zero\ and\ V_{Ego} > Zero$; otherwise, they equal to Zero.

3.6.6 The Goal of the Proposed RECA Reward Function

The reward function of the suggested RECA system is designed to:

1. Penalize dangerous situations: Penalize high relative velocities and accelerations to discourage the vehicle from approaching too quickly or making abrupt changes in speed.
2. Reward safe behavior: Provide rewards for maintaining a safe distance and avoiding collisions.

The combination of penalties and rewards guides the reinforcement learning algorithm to learn behaviors that maximize safety and minimize collision risk. Figure 3.11 shows the RL agent's inputs and output block diagram in the Simulink model.

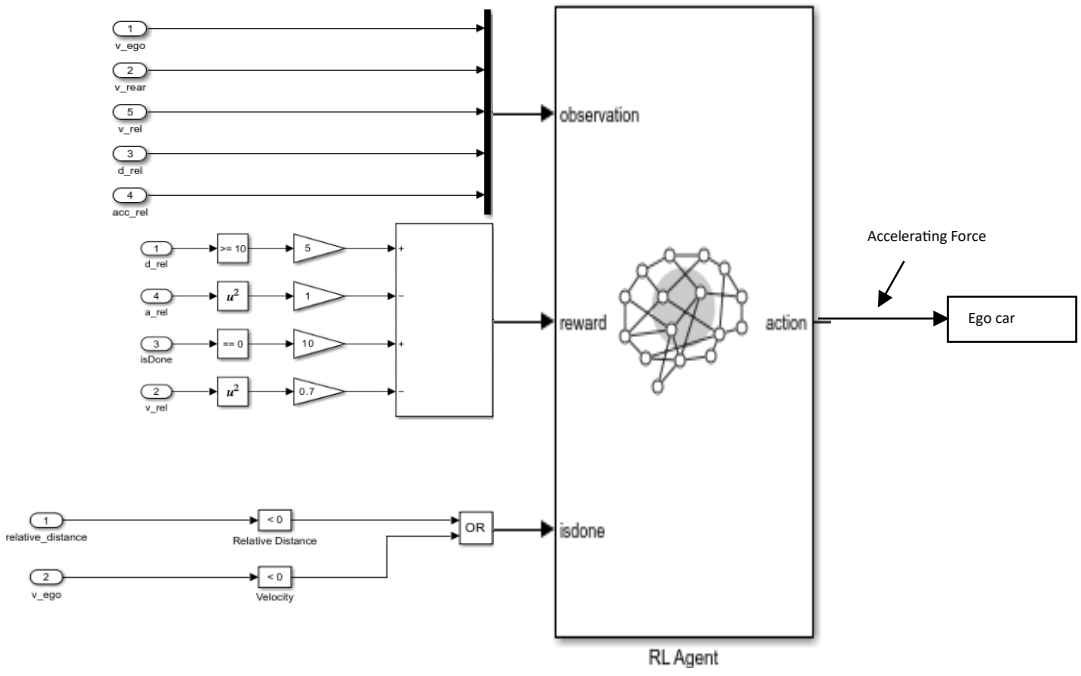

Figure 3.11 Inputs/output Simulink diagram of the RL agent of the RECA system

3.7 Configuration of the Suggested RECA System

Figure 3.12 below shows the configuration of the two cars in the RECA system. It displays the inputs of the RECA system, the five states (observations): ego car speed, rear car speed, relative speed, relative distance, and relative acceleration. The ego car's accelerating force (action) is the RECA system's RL agent output. It is worth remembering that forward motion, the absence of a car in front of the ego (front) car, and the absence of lane change are the considerations of my (RECA) system.

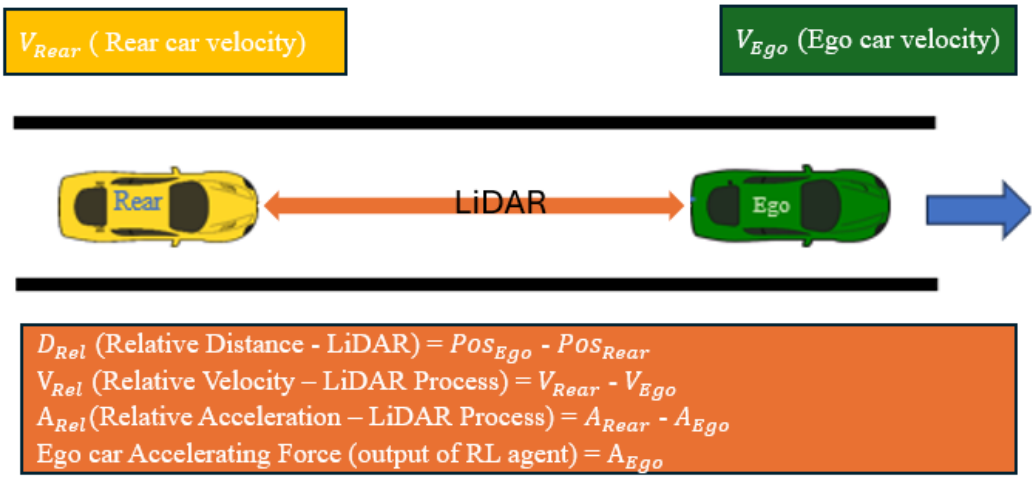

Figure 3.12 The suggested RECA system configuration

3.8 Dynamics of RLA of the Suggested RECA System

Figure 3.13 shows the scheme of formulating the ego and rear cars' dynamics with Reinforcement Learning (RL). The inputs (observations) to the RL agent are coded with orange. The RL algorithm (RLA) output, which is the ego car's input (accelerating force), is green, and the rest of the scheme is black. This scheme traces how we calculated the

states of the RECA system using both the rear and the ego car dynamics. This scheme clearly shows the RL agent algorithm is a black box.

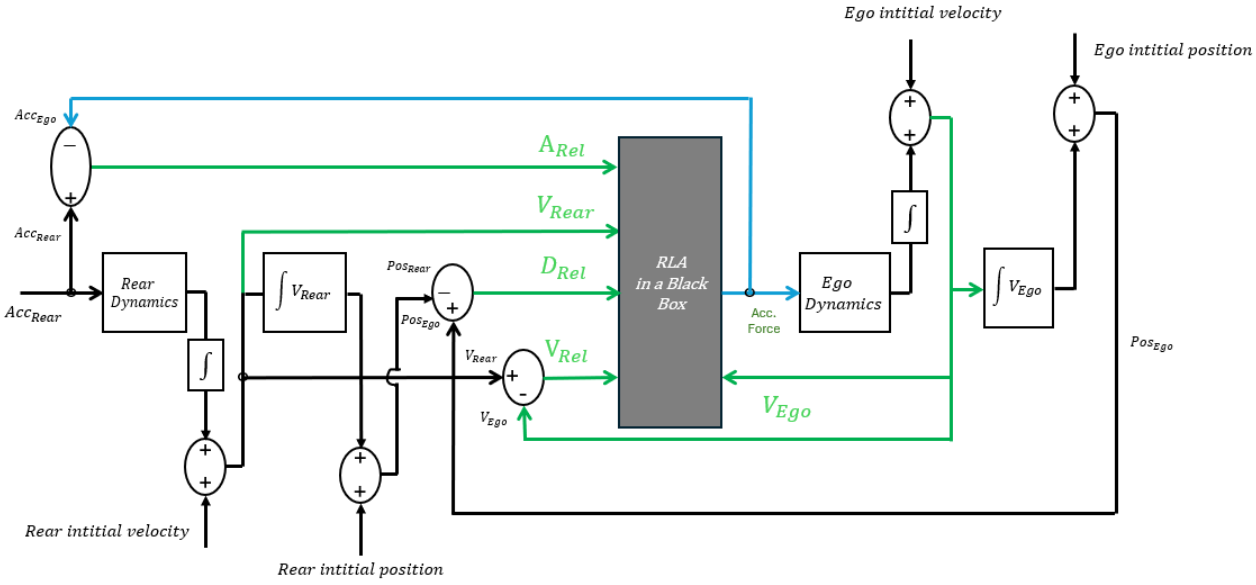

Figure 3.13 Dynamics of the RLA of the proposed RECA system

3.9 Training the RL Agent of the Proposed RECA System

Using and modifying the "Train DDPG Agent for Adaptive Cruise Control" example [5], found in the MATLAB Reinforcement Learning toolbox [4]. the next step was setting up the reinforcement learning algorithm for the RECA system using MATLAB R2023b, Reinforcement Learning Toolbox, and Simulink. To handle the computationally demanding tasks, we employed computers outfitted with multi-core GPUs adept at swiftly managing and modifying memory to boost processing speed. Furthermore, to optimize time efficiency, these two machines, Dell (Alienware m17 R4) and MSI (GE65 Raider 95F), were deployed to execute various training algorithms concurrently. Figure 3.14 shows the two computers used to train the RECA RL agent.

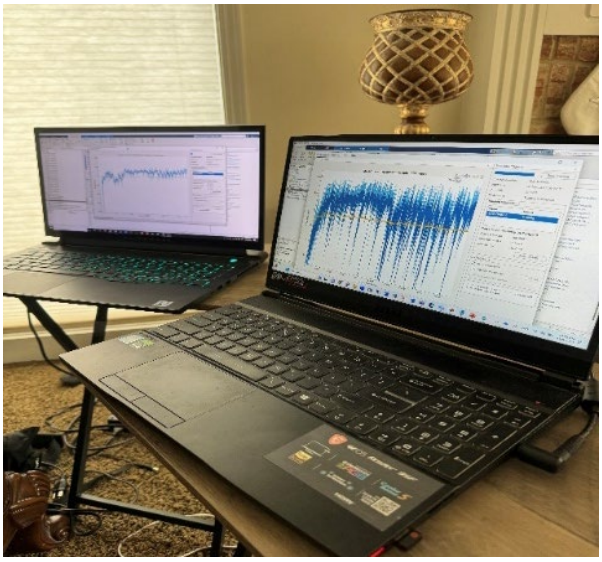

Figure 3.14 Computers used to train the RECA RL agent

Setting and fine-tuning the hyperparameters, such as the maximum number of episodes and the learning rate of the RL algorithm, is crucial. Choosing the appropriate maximum episode number and the learning rate affect the quality of the training process and the time required for each training session, which may sometimes take over 48 hours, even with computers equipped with relatively advanced multi-core GPUs. Besides the hyperparameters, visualizing the training process in real-time could be very important as it may indicate how the training is going, so in some cases, especially when the training is not going well for any reason, observing the pace of the training process could save a significant amount of time by canceling the training process and searching for the reason/reasons behind of the lousy training results, fixing them and start the training process over again. Figure 3.15 shows one example of the training processes of the RECA system with 2000 episodes.

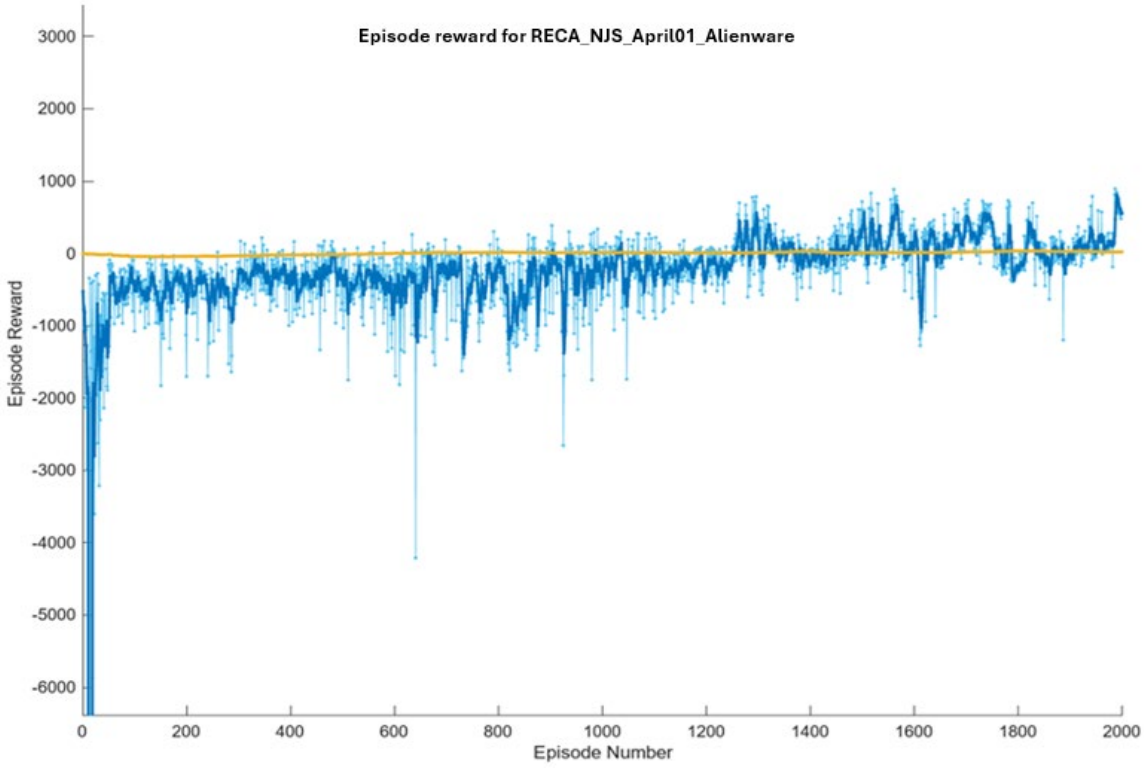

Figure 3.15 Example of training processes of the RECA system with 2000 episodes

The training process is broken into three stages; the plot shows these stages; from episode zero up to ~ 300, we can notice that the rewards are low, and that is expected in the early stages because the RL agent is still learning (uncertain) about the best action to take in each state because of the lack of experience. In this part of training, the agent explores random actions for the states and builds memory (experience) by recording the rewards relevant to each action for each state so it can use the most rewarding actions in the future. From episode ~300 to episode ~1500, we can see that the rewards are improving because the RL agent starts exploiting the experience he built from the first (exploration) stage but will keep the exploration beside the exploitation because the rewards are not as intended. After episode ~1500, it's clear that the rewards are in the

highest area for the whole process as the agent is very experienced at this time and able to pick the proper action for each state (observation) from the environment. In other words, the share of exploitation of the accumulated experience will dominate the share of the exploration in the third stage. At each step, the agent observes the state, takes an action, and receives rewards or penalties (low scores) from the environment. Figure 3.16 below shows the Simulink model of the training of the RECA system.

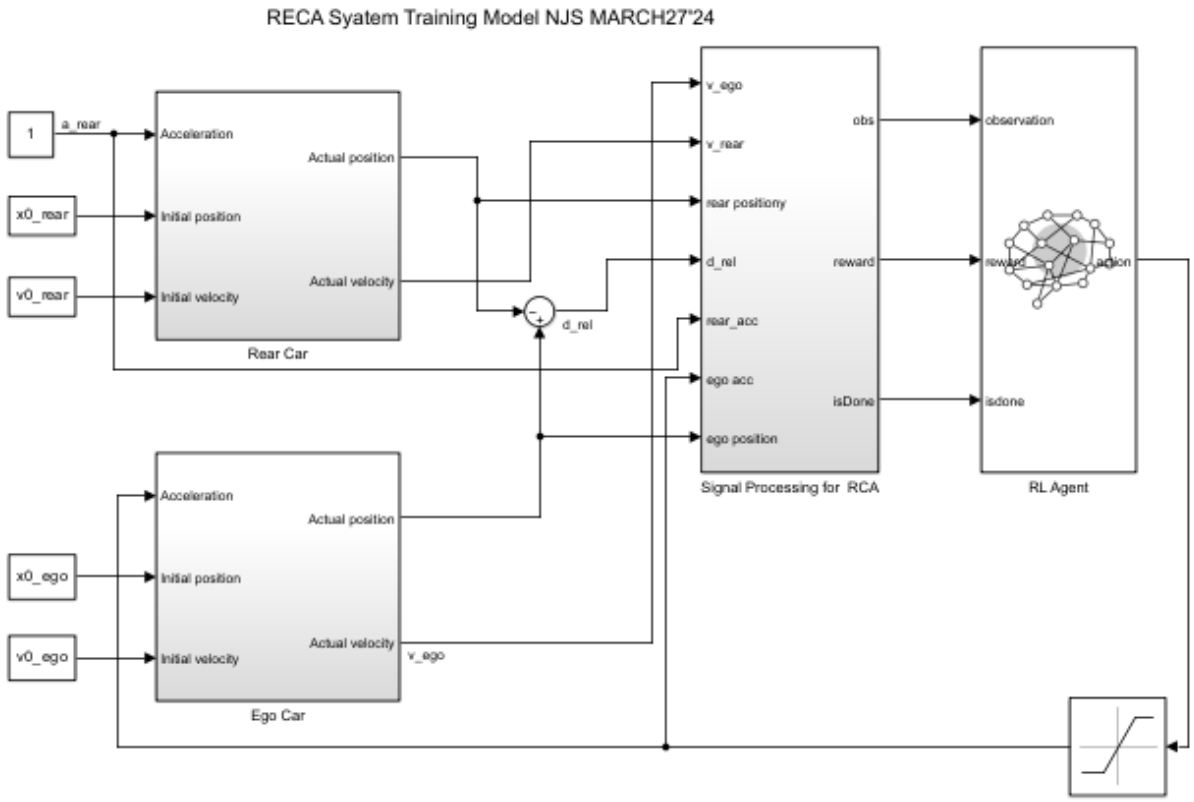

Figure 3.16 Simulink model of the training of the RECA system

3.10 Extracting and Simulating the Trained RL Agent of the RECA System

To verify the trained RL agent black box performance, using MATLAB, we extracted the trained RL agent and then created a Simulink model to test it. We tested it with different scenarios, such as initial positions and speeds for the rear and the ego cars.

A single-lane track was created for the test using Simulink. Figure 3.17 is for the Simulink model of the trained RL agent of the RECA system simulator.

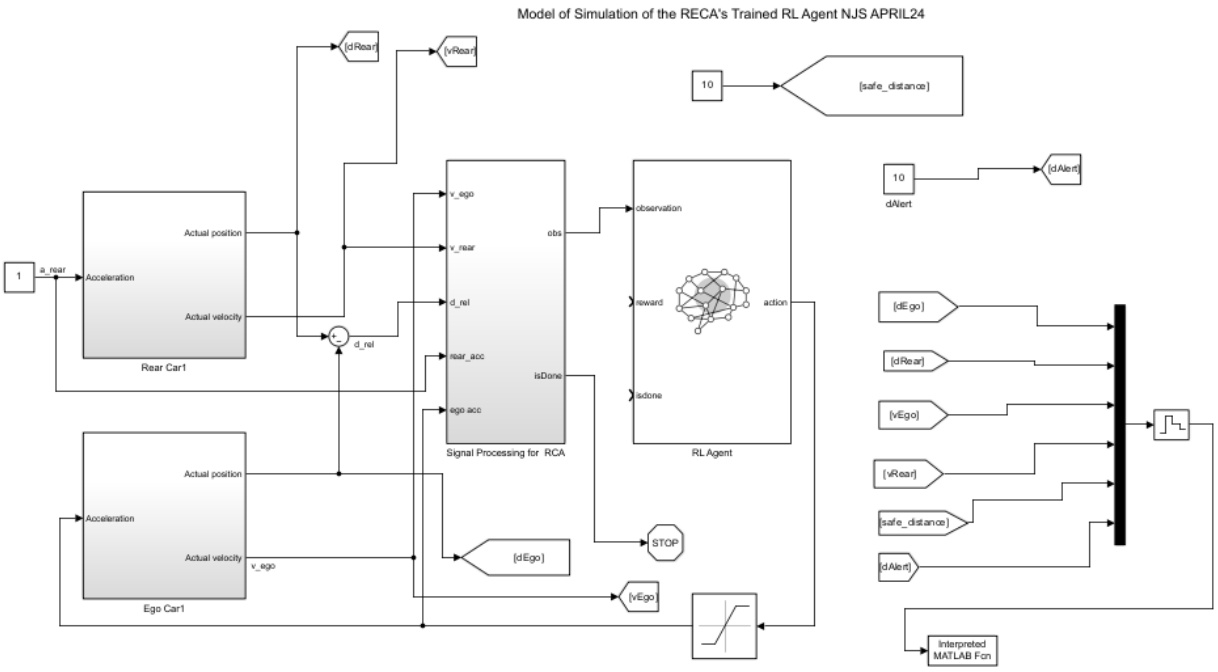

Figure 3.17 Simulink model of the simulator of the RL agent of the RECA system

Figure 3.18 shows the four inputs (observations) and the acceleration (output) of the trained RL agent of the RECA system used in the simulation.

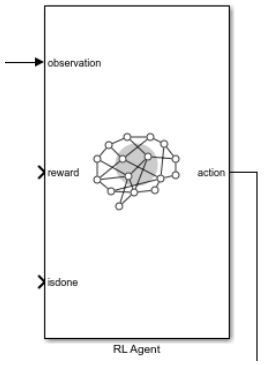

Figure 3.18 Inputs (observations)/output of the trained RL agent of the RECA system

3.11 The Results of Simulating the Trained RL Agent of the RECA System

Figures 3.19, 3.20, and 3.21 show the results of the simulation of the trained RL agent of the RECA system. The first two plots depict the rear and ego car's speeds, and the third shows relative speeds.

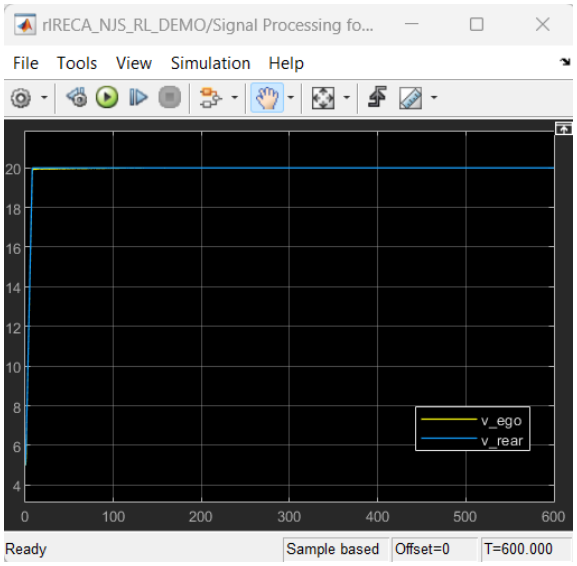

Figure 3.19 Speeds of Ego and Rear cars of the trained RL agent of the RECA system

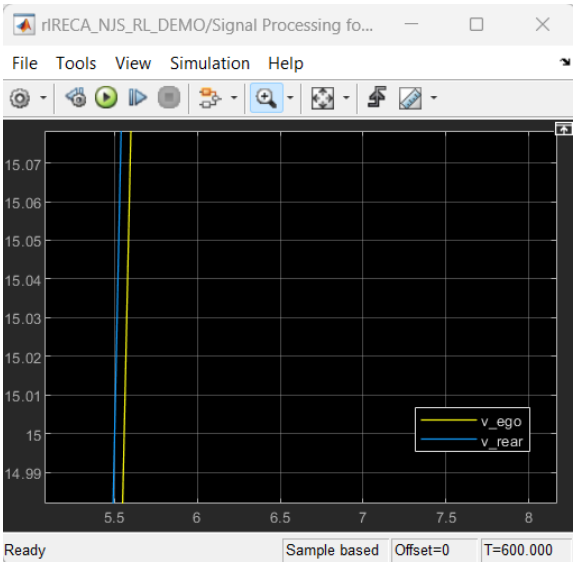

Figure 3.20 Speeds of Ego and Rear cars of the trained RL agent (magnified shot)

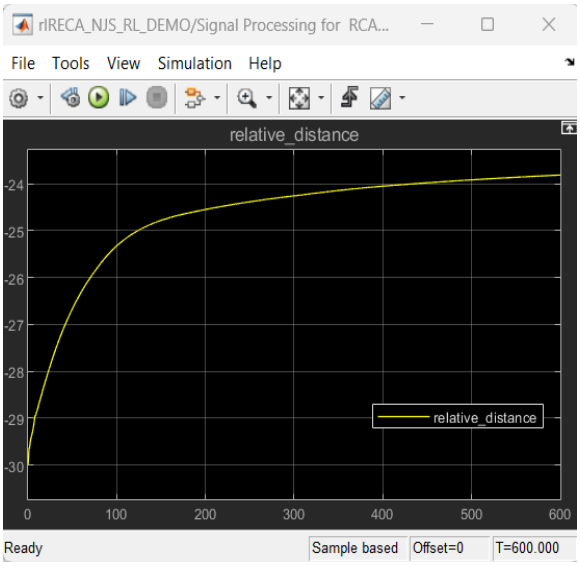

Figure 3.21 Relative distance of the trained RL agent of the RECA system

From the above relative distance plot, see Figure 3.21, we notice that the relative distance of this model decreased starting from the beginning of the simulation when the separation distance was 30 meters and then maintained (stabilized) above the safe distance. From Figure 3.20, the zoomed-in shot of the speeds of the two cars, we can see that the ego car was trying to catch up with the speed of the rear car at the early stages of the simulation, where the relative speed was not equal to zero, While at the later stages of the simulation, we can see that the ego car was able to match its speed with the rear car speed, see Figure 3.19 in the previous page above.

Below is a snippet captured from the RECA trained RL agent simulation. The animation track was set to 500 meters only because it would be tough to see the two cars and their speeds if we made it longer, while the simulation time was up to 1000 seconds to gain better insight into the ego car performance for a relatively long time, see Figure 3.22.

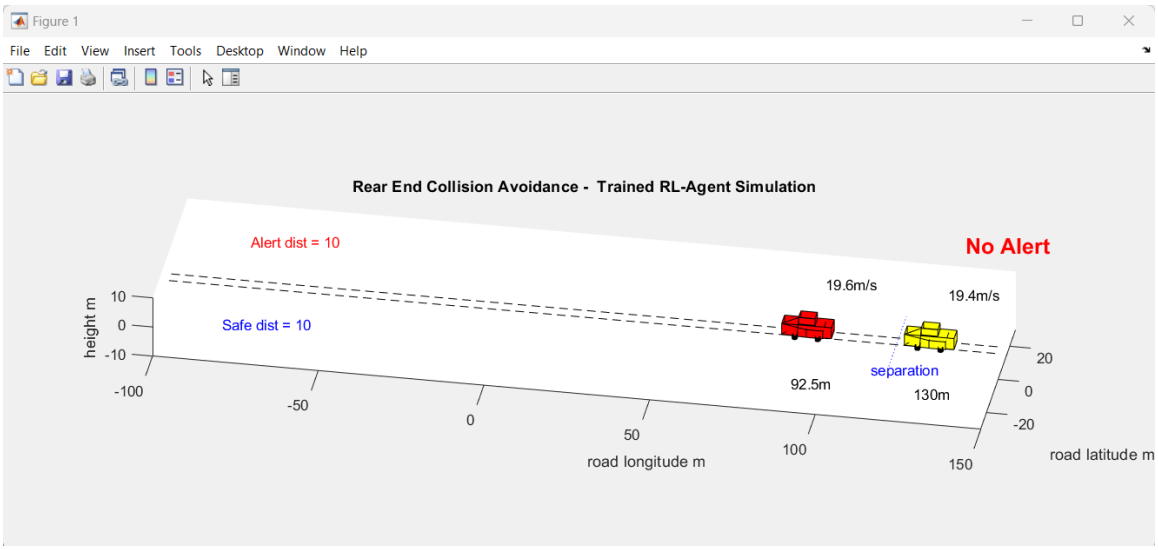

Figure 3.22 A snippet captured from the RECA trained RL agent simulation

3.12 The Results of Simulating the RECA System Without the RL Agent

Then, we simulated the RECA without the RL agent, so in this scenario, the RECA was inactive as the output of the trained RL agent disconnected from the model. This means the ego car is not receiving the acceleration signal from the RL agent (independent of the RL agent); the simulation model is shown in Figure 3.23. The results are depicted in Figures 3.24 and 3.25. We can notice from the two cars' speed plot that the ego car didn't change its initial speed while the rear car's speed was increasing, and from the relative speed plot, we can see that the relative speed dropped to zero, which means colliding between the two cars has been taken place. These results show the significance of the RL agent in operating the RECA system to maintain at least a minimum safe distance between the two cars.

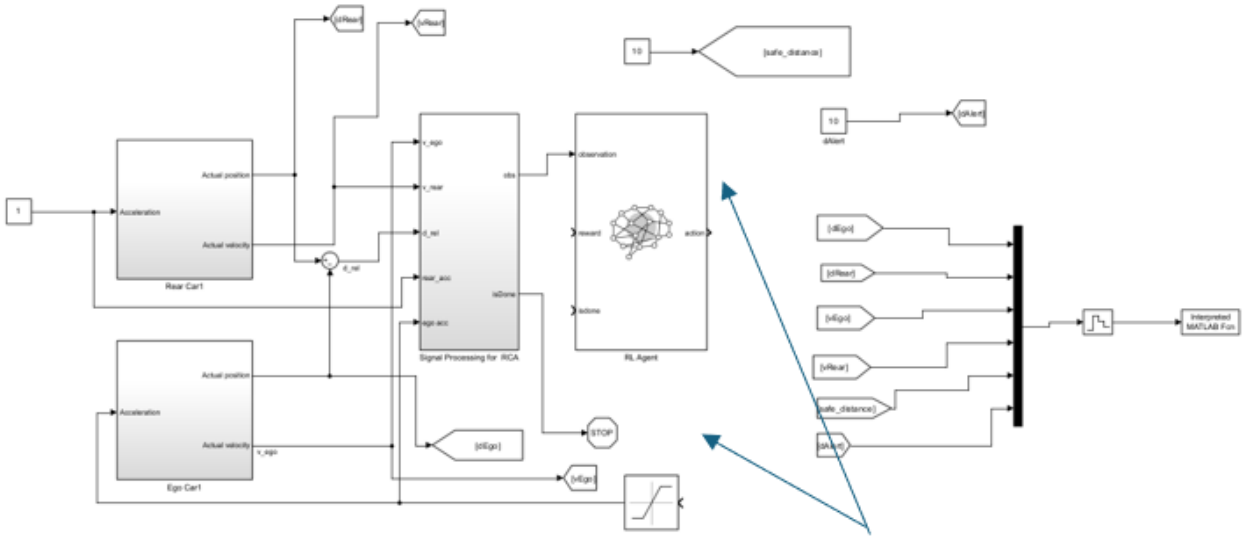

Figure 3.23 RECA system when the RL agent is inactive

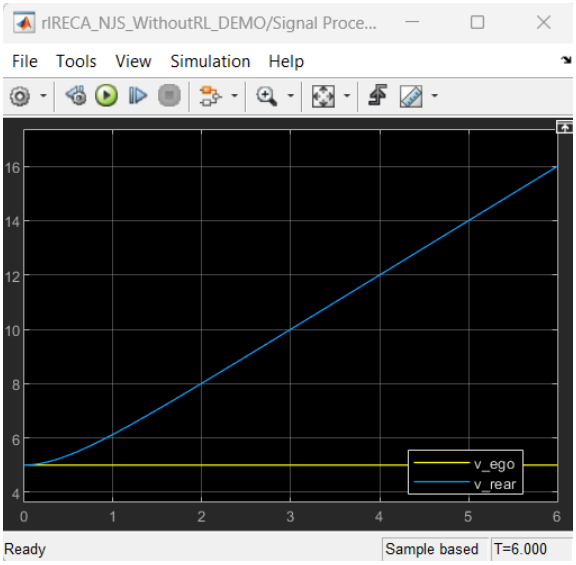

Figure 3.24 Speeds of Ego and Rear cars when the RL agent is inactive

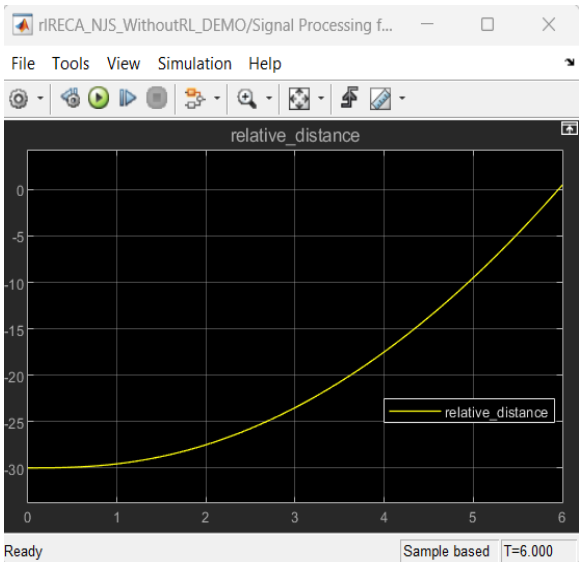

Figure 3.25 Relative distance when the RL agent is inactive (without RL agent)

Below is a snippet captured from the RECA simulation without the RL agent. In Figure 3.26, the rear car collides with the ego car.

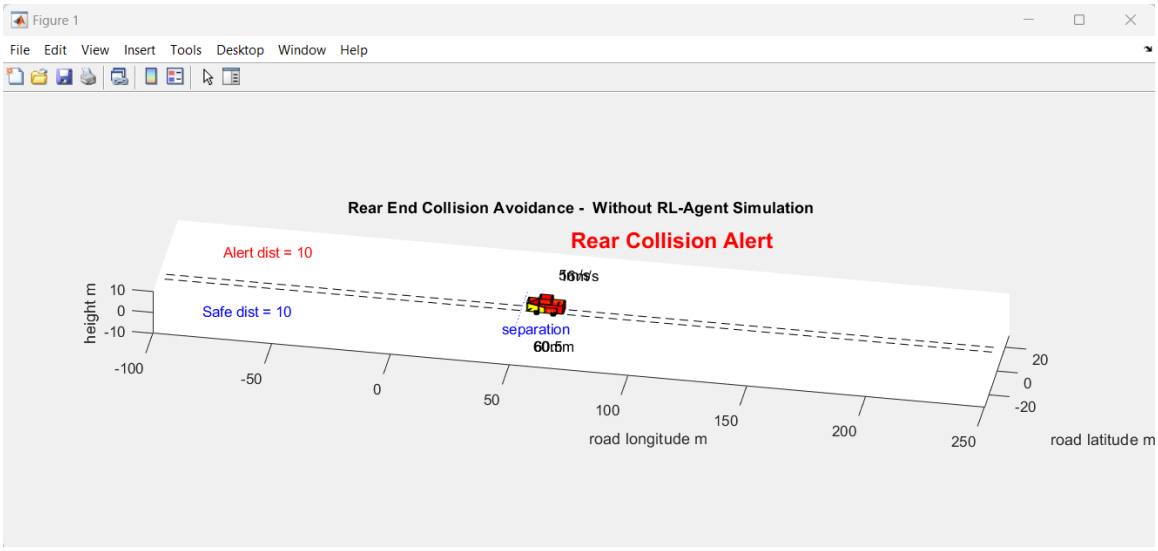

Figure 3.26 Snippet of the collision between the two cars when the RL agent is inactive

CHAPTER FOUR

EXTRACTION OF INFORMATION FROM AI DESIGN DECISIONS

Because AI systems become more complex and penetrate critical domains such as health, finance, and automotive, we need to deepen our understanding of how these sophisticated AI systems make decisions. Responding to that urgent need, in the mid-2010s, Explainable AI (XAI) started attracting attention [66]. In 2016, the Defense Advanced Research Projects Agency (DARPA) launched a program to explain AI decision-making processes without compromising their performance [67]. This program responded to the growing need for understandable AI decisions in the defense and other sectors.

The term "Explainable AI" and its current conceptual framework started gaining more formal recognition and momentum with the advent of more sophisticated machine learning models, which lacked transparency; this ambiguity about the inner working processes of the AI algorithms contributed to the perception of AI as a black box. This field continues to expand as various industries and regulatory bodies demand greater clarity on how AI models make decisions, significantly when these decisions impact human lives [68].

This thesis seeks to formulate the AI algorithm's model, illuminating the AI system strategies. By highlighting these AI algorithms' strategies to reach their decision-making, we can better understand the inner working process of these systems, which will help with complete trust, optimizing, and clarifying users' doubts.

4.1 Extracting Information from the Trained RL Agent Black Box of the RECA

The methodology for extracting information from the trained RL agent (black box) is structured in several key phases. It aims to dissect and understand the proposed RECA system's Reinforcement Learning (RL) algorithm.

4.1.1 The Inputs and Output of the Trained RL Agent Black Box

We developed a MATLAB code to extract the inputs (states) and output (accelerating force) from the Rear-end Collision Avoidance System trained reinforcement learning (RL) agent and save it to a states_actions.mat file, as shown in Figure 4.1.

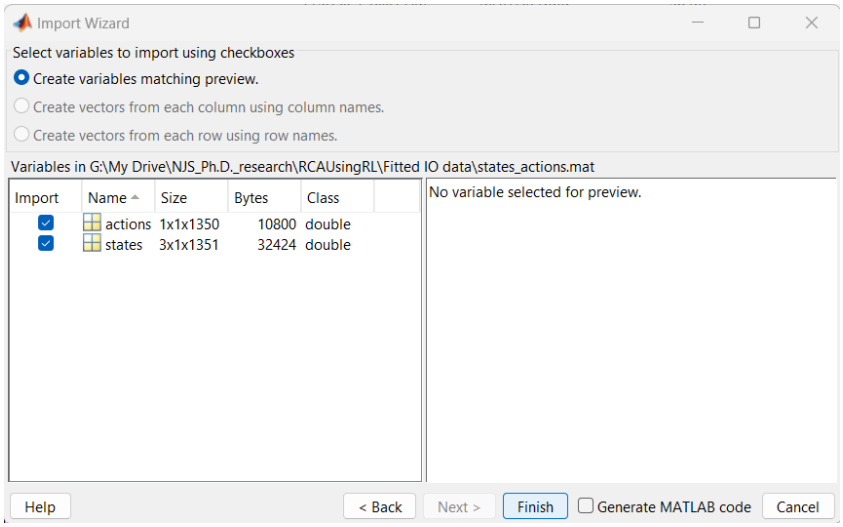

Figure 4.1 Saving input and output data of RL agent in a .mat file

Visualizing the states (input) and the action (output) of the trained RL agent gives us a preliminary insight into the agent's decision-making process. Figures 4.2 and 4.3 show the plots of the inputs and output of the RL-trained agent, respectively.

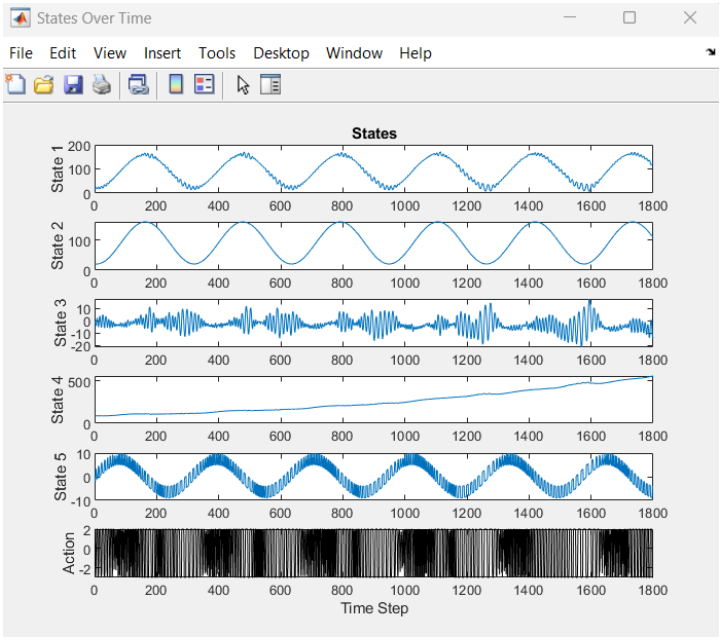

Figure 4.2 Trained RL agent inputs and output plots

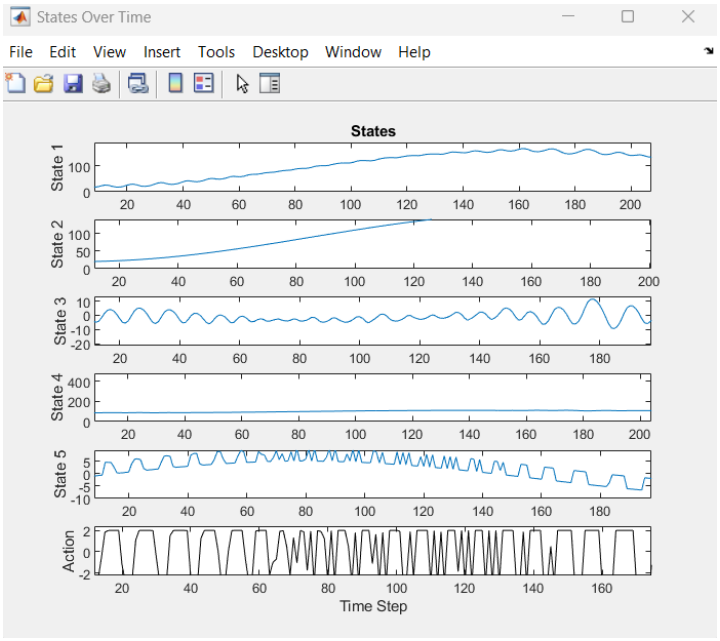

Figure 4.3: Trained RL agent inputs and output plots (zoomed-in)

4.1.2 AI Black Box and Info Extractor

Figure 4.4 shows a block diagram of the optimizing scheme for tuning an Info Extractor (IE) to emulate an AI black box (AIBB). The optimizing algorithm would adjust the parameters and functions of the IE to minimize a measure of the extraction errors.

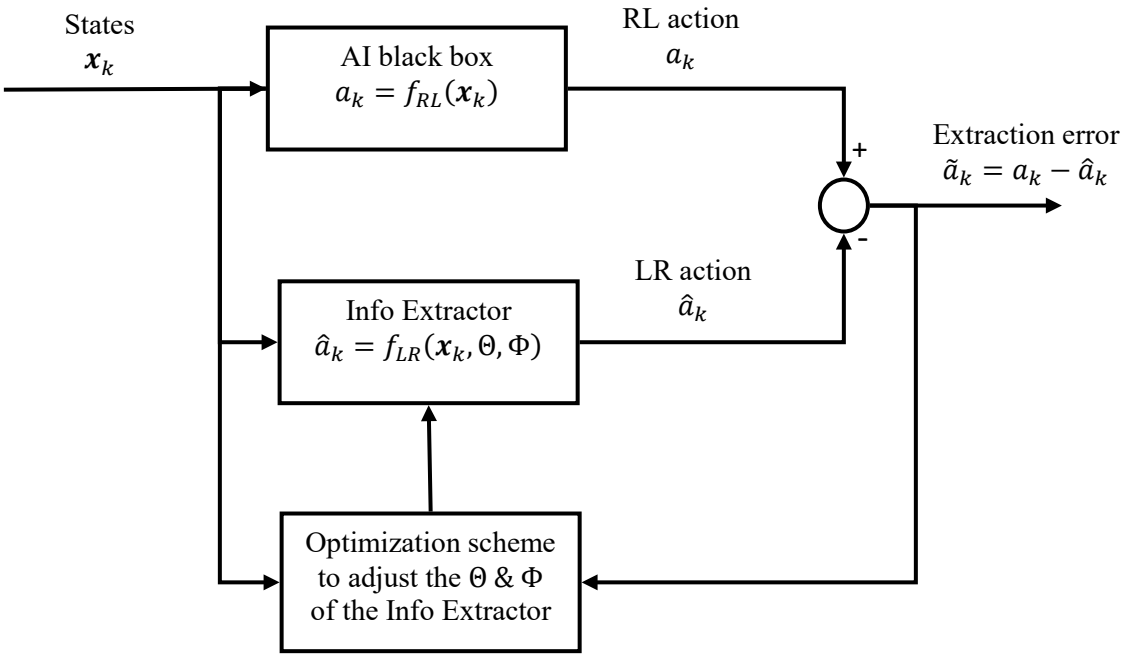

Figure 4.4 Tuning an Info Extractor to emulate an AI block box (AIBB)

For the focus of this thesis, we consider an AI decision to be made up of a reinforced learning (RL) scheme, which would be expressed in equation (4.1).

$$a_k = f_{RL}(\boldsymbol{x}_k) \quad RL\ control\ action\ AIBB \tag{4.1}$$

where $k \in \mathbb{N}$, $\boldsymbol{x}_k \in \mathbb{R}^n$ & $a_k \in \mathbb{R}$, are the sampling time index, sampled observation of state variables, and control action, respectively. f_{RL} is the functional mapping from $\boldsymbol{x}_k \; to \; a_k$, representing the RL decision processes the observation and produces the action.

Next, we consider an Infor Extractor to consist of a linear regression (LR) model, as shown in equation (4.2).

$$\hat{a}_k = f_{LR}(\boldsymbol{x}_k, \Theta, \Phi) \quad LR\ model\ IE \tag{4.2}$$

where Θ & Φ are the parameters and functions of the LR model. The Θ & Φ are to be determined to minimize a performance measure of errors. This thesis considers the Mean Squared Errors (MSE) given by equation (4.3).

$$\begin{aligned} \min_{\Theta,\Phi} MSE &= \min_{\Theta,\Phi} J(\boldsymbol{x}_k, \Theta, \Phi) \\ \min_{\Theta,\Phi} MSE &= \min_{\Theta,\Phi} \left(\frac{1}{N} \sum_{k=1}^{N} (a_k - \hat{a}_k)^2 \right) \\ &= \min_{\Theta,\Phi} \left(\frac{1}{N} \sum_{k=1}^{N} (f_{RL}(\boldsymbol{x}_k) - f_{LR}(\boldsymbol{x}_k, \Theta, \Phi))^2 \right) \end{aligned} \tag{4.3}$$

4.2 Linear Regression Model & Basis Functions

Basis functions are essential elementary math functions that can be combined to construct an original signal. For a simple linear regression model, the signal $\hat{a}$ can be written as a linear combination of elementary components $\phi_1, \phi_2, \cdots, \phi_n$ as in equation (4.4).

$$\hat{a}(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\phi} = \sum_{j=1}^{n} \theta_j \phi_j = \theta_1 \phi_1 + \theta_2 \phi_2 + \cdots + \theta_n \phi_n \tag{4.4}$$

where $\boldsymbol{\theta} = [\theta_1 \ \ \theta_2 \ \cdots \ \ \theta_n]^T$ are the coefficients and, $\boldsymbol{\phi} = [\phi_1 \ \ \phi_2 \ \cdots \phi_{\ n}]^T$ elementary components. For a more general linear regression model, see equation (4.5).

$$\hat{a}(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \Phi(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j \phi_j(\boldsymbol{x}) = \theta_1 \phi_1(\boldsymbol{x}) + \theta_2 \phi_2(\boldsymbol{x}) + \cdots + \theta_n \phi_n(\boldsymbol{x}) \tag{4.5}$$

where $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}) \ \ \phi_2(\boldsymbol{x}) \ \ \cdots \phi_n(\boldsymbol{x})]$ is the more general basis function.

4.2.1 Linear Regression Model with MSE Coefficients

Assuming the states–actions relationship can be approximated by a time-invariant 1st-order as in equation (4.6):

$$u \approx k_1 x_1 + k_2 x_2 + k_3 x_3 + \ k_4 x_4 + \ k_5 x_5 = \ \ K\boldsymbol{x} \tag{4.6}$$

u is the output (accelerating force), $K = [k_1 \ k_2 \ \ k_3 \ \ k_4 \ \ k_5]$ are the coefficients and $\boldsymbol{x} = [x_1 \, x_2 \, x_3 \, x_4 \, x_5]^T$ are the states (elementary components) which are the RL agent inputs. A system of linear equations can be represented in matrix form as shown below in equation (4.7):

$$\mathbf{u} \approx MK \tag{4.7}$$

$$M = \underbrace{\begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} & x_{4,1} & x_{5,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1,N} & x_{2,N} & x_{3,N} & x_{4,N} & x_{5,N} \end{bmatrix}}_{Nx5}, \ K = \underbrace{\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{bmatrix}}_{5x1}, \ \mathbf{u} = \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_N \end{bmatrix}}_{Nx1}$$

The mean-squared error MSE in (4.3), can be expressed in terms of $\boldsymbol{a}$ and $\mathbf{u}$ solution K_{MSE} is derived from minimizing the MSE.

$$MSE = \frac{1}{N}[\boldsymbol{a} - \mathbf{u}]^T[\boldsymbol{a} - \mathbf{u}] = \frac{1}{N}[\,\boldsymbol{a} - MK]^T[\boldsymbol{a} - MK] \tag{4.8}$$

where $\boldsymbol{a} = \underbrace{\begin{bmatrix} a_I \\ \vdots \\ \vdots \\ a_N \end{bmatrix}}_{Nx1} (RLA\ output)$ and $\mathbf{u} = \underbrace{\begin{bmatrix} u_I \\ \vdots \\ \vdots \\ u_N \end{bmatrix}}_{Nx1} (LRM\ output)$. Knowing the data $M\ (observable\ states)$ and $\boldsymbol{a}$, the Mean Square Error (MSE) estimation of K yields

$$K_{MSE} = (M^T M)^{-1} M^T \boldsymbol{a} \tag{4.9}$$

So the output of the LRM output (4.6) becomes

$$u = K_{MSE}\boldsymbol{x} = k_{1MSE}x_1 + k_{2MSE}x_2 + k_{3MSE}x_3 + k_{4MSE}x_4 + k_{5MSE}x_5 \tag{4.10}$$

4.3 Fit Linear Regression Model for the RECA System

A MATLAB function called fitlm, which stands for Fit Linear Model, is used to get a linear regression model fit to the input data of the RECA system. The linear regression model (LRM) for my RECA system was approximated, as shown in equation (4.11).

Action = 0.13658 + (-0.31919)*State_1 + (0.31911)*State_2 + (0)*State_3 + (0.0026726)*State_4 + (0.072491)*State_5 (4.11)

where Action is the output of the RL agent, and it's the accelerating force for the front (ego) car, 0.13658 is the bias, State_1 is the V_{Ego}, State_2 is the V_{Rear}, State_3 is the V_{Rel} , State_4 is the D_{Rel} , State_5 is the A_{Rel}.

We can notice that the coefficient of the relative speed (V_{Rel}) is equal to zero because it is redundant and V_{Rel} is implicitly included in the expression, as the LRM already encompasses both V_{Ego} and V_{Rear} and $V_{Rel} = V_{Ego} - V_{Rear}$.

4.4 Simulating the Extracted Linear Regression Model (LRM)

Using Simulink, we verified the LRM by simulating it. We used the same simulation model to simulate the trained RL agent. Still, we replaced the RL block with a function (green block) that uses the extracted model to calculate the action, which is the accelerating force of the ego car, as shown in Figure 4.5.

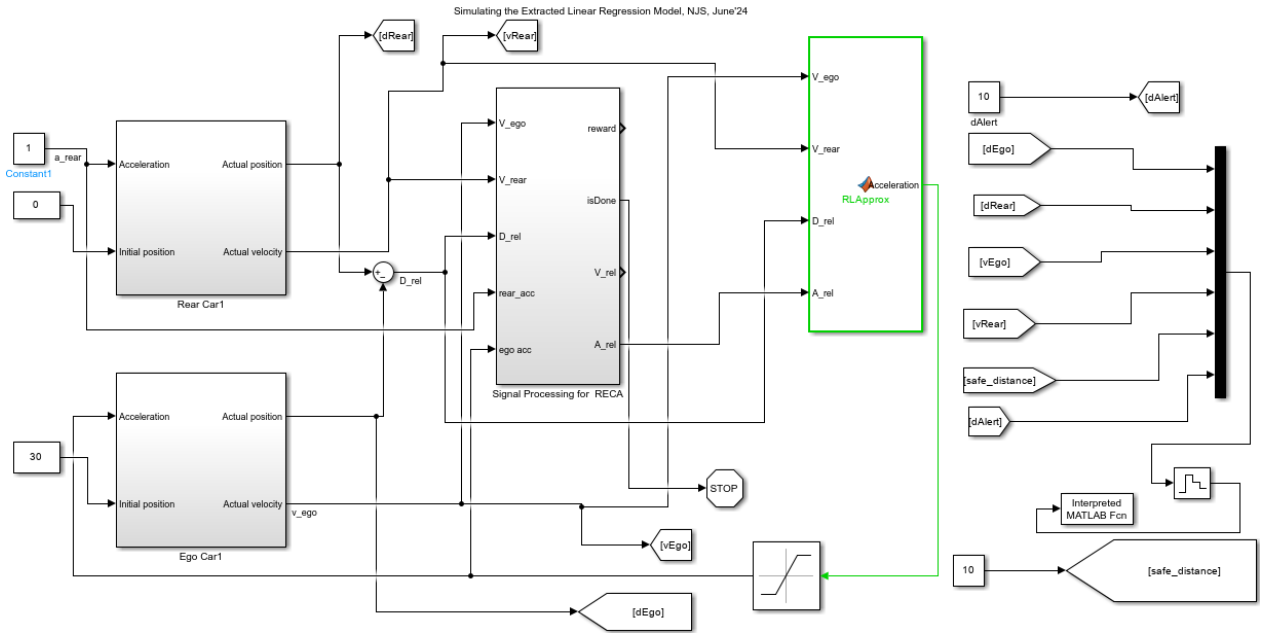

Figure 4.5 Simulink model of the simulation of the LRM of the RECA system

Figure 4.6 shows the four inputs, $V_{Ego}, V_{Rear}, D_{Rel}$, and A_{Rel} and the acceleration (output) of the RECA-extracted linear regression model (LRM). Unlike when we trained the RL agent, we had five inputs, $V_{Ego}, V_{Rear}, D_{Rel}$, A_{Rel} and V_{Rel}, but because the coefficient of the V_{Rel} was equal to zero in the extracted LRM, V_{Rel} was excluded from the LRM's inputs when we simulated it. The Simulink function is shown in Figure 4.7.

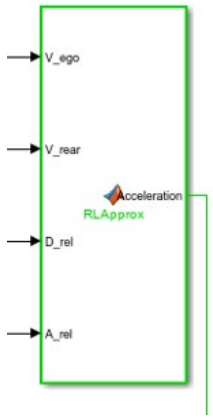

Figure 4.6 Inputs/output of the Simulink function of the RECA LRM

```
function Acceleration = RLApprox(V_ego,V_rear,D_rel,A_rel)

    Acceleration = 0.13658 + (-0.31919)*V_ego + (0.31911)*V_rear + ...
                   (0.0026726)*D_rel + (0.072491)*A_rel

end
```

Figure 4.7 The function calculates the output of the LRM of the RECA

4.5 Simulation Results of the LRM of the RECA System

Figures 4.8 and 4.9 show the RECA system's LRM simulation results. The first plot shows the relative speed, and the second shows the rear and ego car's speeds. From the relative distance plot, we notice that the relative distance of this model decreased starting from the beginning of the simulation when the separation distance was 30 meters and then maintained (stabilized) above the safe distance. From the speeds of the two cars plot, we can see that the ego was trying to catch up with the speed of the rear car starting

at the beginning of the simulation, where the initial speed was ~11 mph until the rear car speed saturated at ~45 mph, where at that speed, both cars' speeds maintained at the same level.

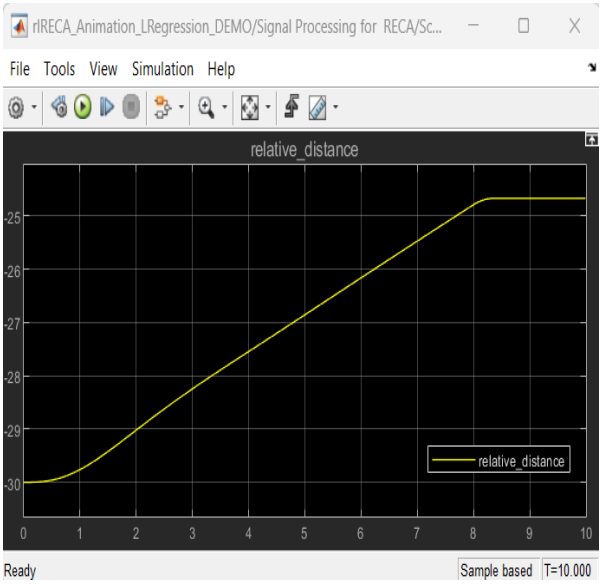

Figure 4.8 Relative distance of the LRM of the RECA system

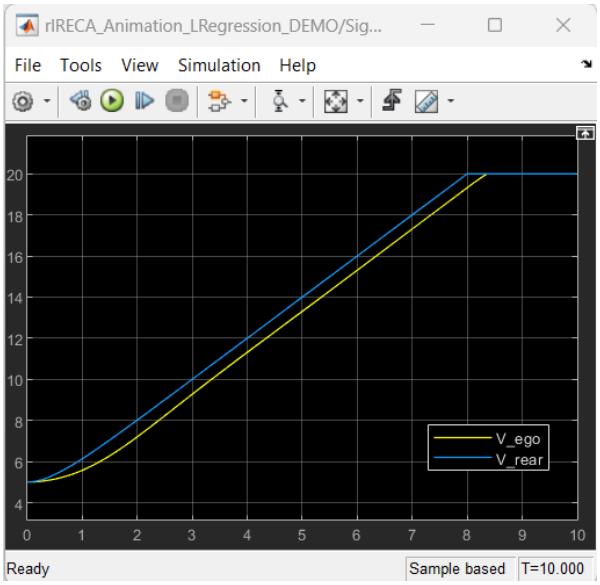

Figure 4.9 Speeds of ego and rear cars of the LRM of the RECA system

Figure 4.10 shows a snippet of the animation of the LRM simulation. The track was set to 150 meters only because seeing the two cars and their speeds would be tough if we made it longer. The simulation time was set to 10 to highlight the first stage when the ego is trying to catch up with the rear car speed from the initial rear speed of ~11 mph to the saturation speed of the rear car, 45 mph, which both cars maintain the same speed. From the snippet (between 0-50 m) of the animation of the LRM simulation, we can see that the rear car (red car) speed is 8.6 m/s, which is ~19.2 m/h. In comparison, the ego car (yellow car) speed is around 7.8 m/s, which is ~17.4 m/h. That shows that the ego car's speed is less than the speed of the rear car at that stage, but from Figure 4.9, we notice that the ego car could keep the same speed as the rear car after the second 8 of the simulation.

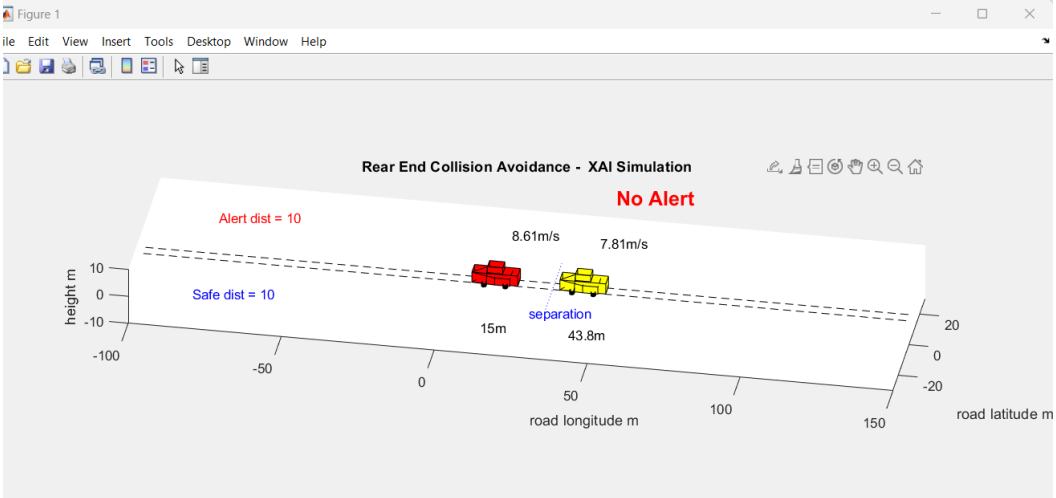

Figure 4.10 Snippet of the animation of the LRM

CHAPTER FIVE

EXPERIMENTAL VERIFICATION OF THE EXTRACTED INFORMATION FOR THE RECA SYSTEM

5.1 Introduction

After verifying the functionality of both the trained agent and the extracted information using the XAI method (function approximation) by using Simulink of MATLAB, the next step was verifying the extracted mathematical model (LRM) using hardware. Although dealing with hardware to validate a concept approved using software introduced challenges in many aspects, it was an irreplaceable step to verify the results we received thoroughly.

5.2 Hardware Used in the Experiments

Two scaled, four-wheeled robotic cars, also known as differential drive robots, were used to simulate the rear and ego cars, as shown in Figure 5.1.

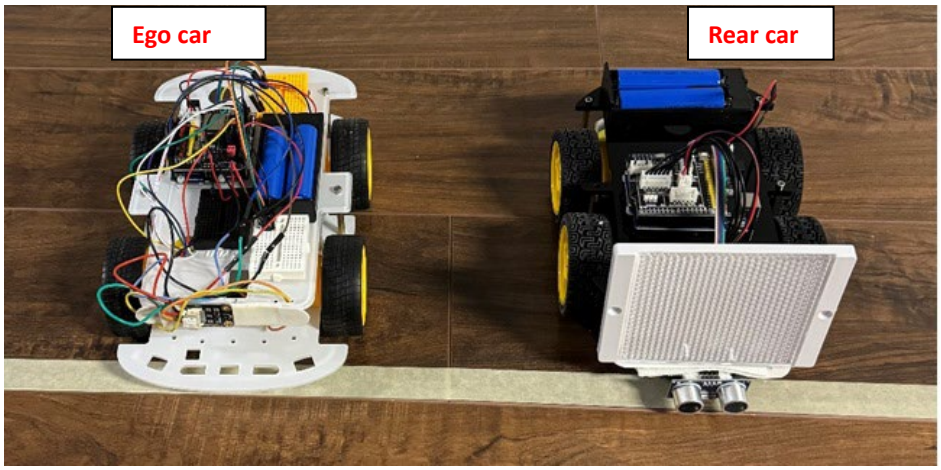

Figure 5.1 Differential drive scaled robotic cars used in the experiments

In these types of skid-steering robotic cars, direction control is typically achieved by independently varying the speeds of each side of the wheels. This differential in speed causes the robot to turn by skidding or rotating around a central point between the wheels. We had to run separate tests on the DC motors that came in car kits to ensure that these motors respond evenly to the power in terms of the number of revolutions per Minute, RPM. For this purpose, we used a compact photo tachometer and counter, type Reed Instruments R7050.

After several tests, we found speed variation among the motors; it seemed like the motors had different speed characteristics. We noticed, for example, that each motor has a slightly higher or lower RPM than the other. This problem resulted in an uneven straight-line motion and difficulty maintaining a straight path. By observing the speed differences for each wheel for the same PWM, I tried to solve this problem by calibrating each side and adjusting their speed by multiplying the PWM by some factor before sending them to the motors. That remedy wasn't so practical to solve the problem, especially since we faced another issue regarding these motors, and that was the motor shafts were wriggly, which caused the wheels to loosen after some experiments, and that worsened the case even more, so we had to search for a reliable solution to solve this problem. The solution we found effectively forced the robotic cars to keep moving straight, using metal rails originally made as holders. It is worth mentioning that the experiments were implemented in my house's basement.

5.2.1 Rear Car

For the rear car, we used an Arduino Uno R3-based scaled robotic car made by ELEGOO - Smart car v3. A 100 mm x 100 mm plastic reflector for photoelectric sensors

mounted on the upper chassis of the car for better reflecting of the light emitted by the Time-of-Flight (ToF) Ranging Sensor that will be used to measure the separation distance between the two cars, that sensor was fixed on the back of the ego car. An HC-SR04 ultrasonic sensor was used as a raging sensor; the sensor was used to implement an automatic emergency brake whenever the distance between the Ego and rear car is equal to or less than 15 cm. By using picture hanging strips, a logi 720P webcam fixed on the top of the reflector to capture the ego car during the demo. Figure 5.2 Rear car with the plastic reflector and ultrasonic sensor.

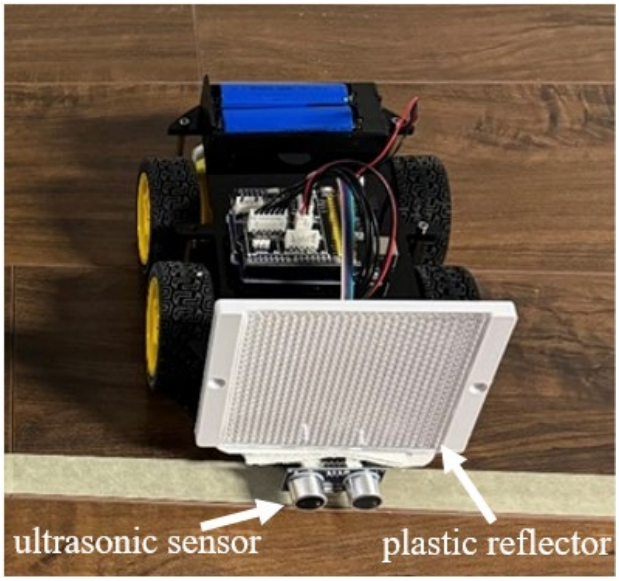

Figure 5.2 Rear car with the plastic reflector and ultrasonic sensor

5.2.2 Ego Car

Another scaled four-wheeled robotic car made by Osoyoo - V2.1 was used as an ego car and hosted the algorithm that implemented the LRM extracted from the trained RL agent. A VL53L1X Time-of-Flight (ToF) Long-Distance Ranging Sensor was fixed

on the upper chassis on the back of the ego car to measure the relative distance (D_{rel}) in real time between the two cars. Figure 5.3 shows the ego car and the ranging sensor. An LM393 Speed Measuring Module was fixed to the bottom side of the lower chassis and used to calculate the speed of the ego car, and the encoder disc was fixed to the motor shaft.

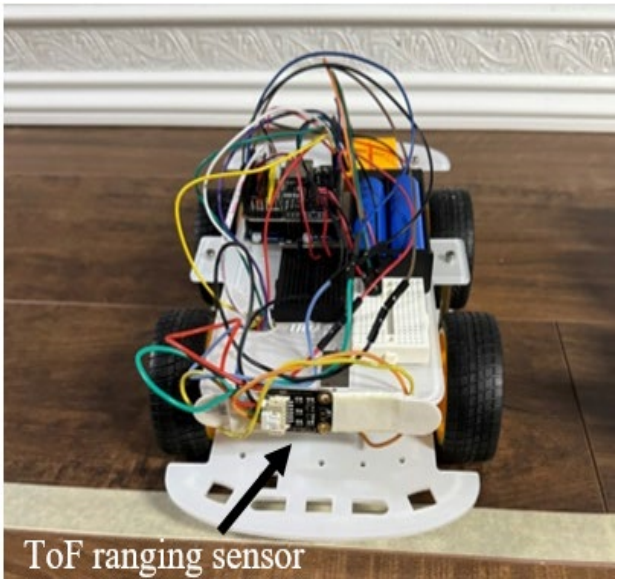

Figure 5.3 Ego car with the ToF ranging sensor

The intensive computation power required by the microcontroller in the ego car is mainly because of these two sensors whose data needs to be acquired promptly and the intensive calculations required to calculate different parameters used in LRM. Furthermore, all these calculations depend on the sensors' output; therefore, we needed a fast processor to complete all the abovementioned tasks and send control signals to the actuators (motors) in real-time. Considering that the microcontroller that came with the robotic car "Arduino Uno R3" was so slow for my application, we had to search for an

alternative. Arduino Uno R4, released in June 2023, was the proper one for this experiment because of the advanced features that the microcontroller came with, making the ego car very responsive and fast. The fact that the Arduino Uno R4 is hardware (voltage level) and software compatible with the Arduino Uno R3 facilitated the swapping process. Therefore, we didn't face troubles with the shield that came with the robotic car as the shield pinout fitted in the new board, and the voltage levels were kept as is without a change of 0-5V. The only part that needed to be replaced was the USB cable, which was replaced with a USB-Type C cable to fit the Arduino Uno R4 microcontroller.

5.3 Experiments Track

Five 80-inch-long hang tracks, totaling 400 inches "33 feet", were purchased and used in this experiment to ensure that both cars were moving straight. These rails worked well as they have grooves with a width suitable for the wheels of the scaled robotic cars. The unleveled floor of the basement was challenging in the experiments as it used to cause one or both cars to drift away occasionally from the track, especially at the ends of some of the track prices. Therefore, the main track was supported by other wood and metal materials I already had in my house, plus we used the picture hanging strips to tape the ends of the "white" metal track to the ground. During the experiments, an HD 1080p Logi type "C922 Pro Stream Webcam" was used to capture both cars with the whole track; the track and webcam are shown in Figure 5.4.

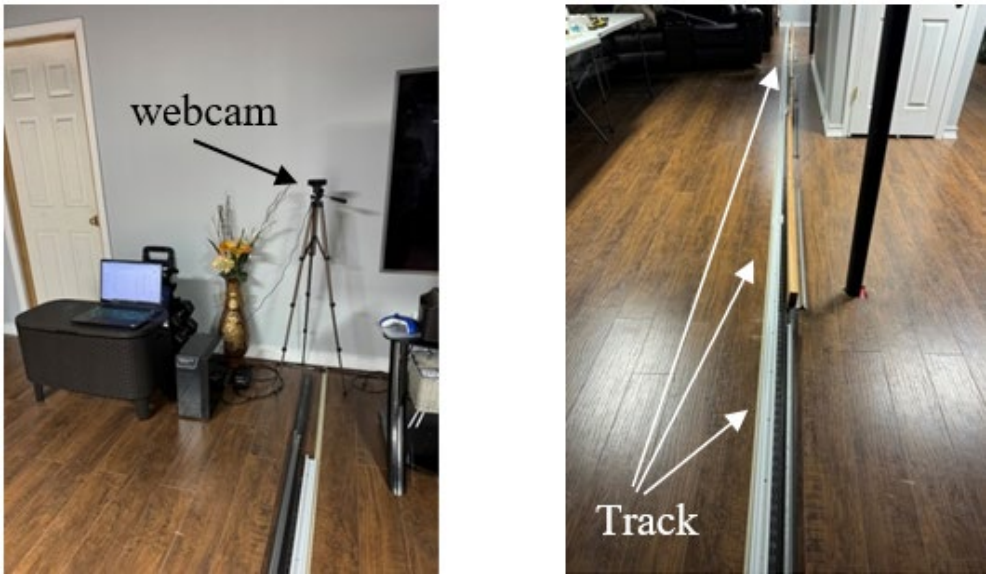

Figure 5.4 Track and webcam used in the experiments

5.4 Software and Algorithm Used in the Experiment

Both robotic cars used in this experiment, the Osoyoo V2.1 (ego car) and the Elegoo Uno R3 Smart Robot Car Kit V3 (rear car), came with some tutorials. Coding-wise, we didn't start from scratch as we used these tutorials and some ready-to-use functions, such as forward and stop functions. These tutorials were important as some sensors, like the ultrasonic HC-SR04 sensor, could not connect directly to the microcontroller but through the shield on top of the microcontroller. Therefore, we had to follow that configuration with special sockets made in that shield for each sensor that came with the car.

5.4.1 Rear Car Algorithm

Various acceleration and initial speeds were used for the rear car, while the separation distance was implemented manually by putting the car at varying distances from the ego car. Different accelerations (sine wave), initial speeds, and separation

distances were implemented to test the ego car's response in various scenarios and verify the reliability of the ego car's control algorithm (LRM) extracted from the trained RL agent. An automatic emergency brake was added to the code to stop the rear car whenever it gets dangerously close to the ego car.

5.4.1.1 Rear Car Algorithm Flowchart The flow chart below shows the rear car's algorithm; see Figure 5.5.

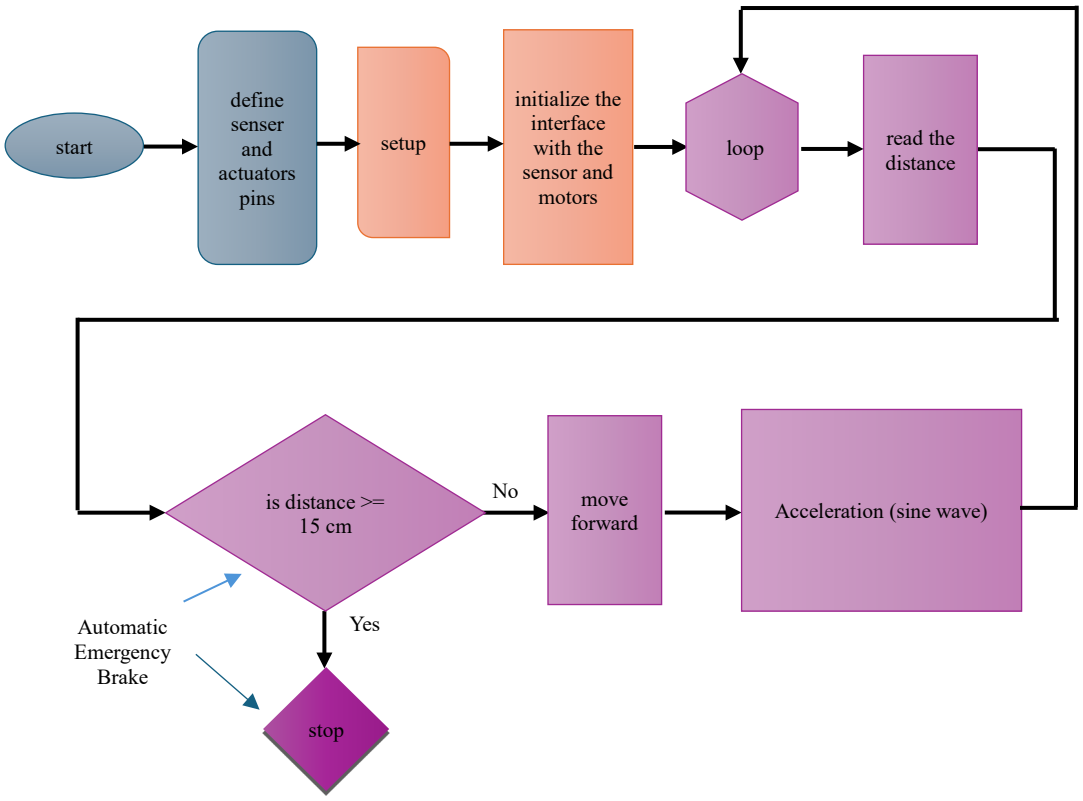

Figure 5.5 Rear car algorithm flowchart

5.4.2 Ego Car Algorithm (Using the Extracted LRM)

The extracted LRM from the trained RL agent was implemented in the ego (front) car algorithm. In order to use the Arduino Uno R4 board, I had to migrate to the latest

released version of the Arduino Integrated Development Environment IDE, IDE 2.3.2. Then, I downloaded all the libraries that come with the new IDE.

5.4.2.1 Measuring, Calibrating, and Filtering Distance After assembling the car and ensuring it acted as expected, like moving forward, backward, left, and right, we had to test the other components and ensure they worked as intended. We ran some tests on the ranging sensors that I had already to pick the most appropriate one for my application. We tested the HC-SR04 4-meter Ultrasonic Ranging Sensor, GARMIN LiDAR-Lite V3 40-meter Laser-based Optical Ranging Sensor, GP2Y0A21YK0F Sharp IR 10-80cm Analog Distance Sensor, VL53L1X Time-of-Flight (ToF) 4-meter Long-Distance Ranging Sensor, Plus we purchased and tested another sensor, TF-Luna LiDAR Range Finder Sensor 0.2m-8m Single-Point Ranging Module UART/I2C. Considering the required features needed in the raging sensor that fit these experiments, for example, covered distance, data rate, and the size of the ranging sensor, we ended up using the VL53L1X Time-of-Flight (ToF) ranging sensor. Even though we noticed that the distance reported by this sensor was nonlinear, we had to linearize the data. we used the curve fitting toolbox in MATLAB to fit data; below is the fitted distance equation (5.1).

$$D_{Rel} = 0.0007 \times rawDistance^2 + 0.8285 \times rawDistance + 15.1827 \quad (5.1)$$

To address the instability (fluctuations) of the ranging sensor readings because of the noise, we first tried to average the readings, starting with averaging small numbers of readings and observing the effect on their stability. We ended up with 40 readings to stabilize the reading. Still, we noticed that it was expensive timewise, especially calculating the control signal depending on the distance.

As the averaging technique didn't work in my case, we had to search for an alternative. We implemented the Low-Pass Filter (LPF) in the code by defining a function to smooth out noisy distance readings from the ToF sensor, as shown in equation (5.2).

$$filteredValue(t) = \alpha \times currentReading(t) + (1 - \alpha) \times \ldots$$
$$filteredValue(t - 1) \quad (5.2)$$

where $filteredValue(t)$, represents the filtered value at time t. $currentReading(t)$, represents the current sensor reading at time t. $filteredValue(t - 1)$, represents the filtered value from the previous time step t−1. α, the filter coefficient determines how much weight is given to the current reading versus the filtered value from the previous time step. Its value, in my case, is 0.1.

A weighted average of the current reading and the previous filtered value is calculated by the function that applies the LPF. Alpha is the weight given to the current reading and the $1 - \alpha$ is the weight of the previous filtered value. Through adjusting $\alpha\ to$ 0.1, I could control how much smoothing was applied to the sensor readings.

5.4.2.2 <u>Calculating Rear Car Speed</u> The speed of the rear car is calculated based on the distance change measured by the ToF sensor. The ToF sensor continuously measures the distance between the Ego and rear cars. The change in distance (ΔD_{Rel}) between consecutive measurements is calculated as in equation (5.3).

$$\Delta D_{Rel} = current\ distance - previous\ distance \quad (5.3)$$

where $current\ distance$ distance is the distance measured by the ToF sensor at the current time. $previous\ distance$, is the distance measured by the ToF sensor at the previous time. The time interval (Δt_{rear}) between consecutive distance measurements is calculated using the current time shown in equation (5.4).

$$\Delta t_{rear} = current\ time - previous\ time \tag{5.4}$$

where $current\ time$, is the current time when the distance measurement is taken. $previous\ time$, is the time of the previous distance measurement. Once the distance changes (ΔD_{rel}) and the time interval (Δt_{rear}) are determined, the speed of the rear car (V_{Rear}) is calculated as in equation (5.5).

$$V_{Rear} = \frac{\Delta D_{rel}}{\Delta t_{rear}} \tag{5.5}$$

5.4.2.3 <u>Calculating Rear Car Acceleration</u> The change in velocity (ΔV_{Rear}) of the rear car is calculated based on its current and previous velocities, as shown in equation (5.6).

$$\Delta V_{Rear} = current\ velocity - previous\ velocity \tag{5.6}$$

The time interval (Δt_{rear}) between consecutive velocity measurements is calculated as in (5.4) above. Once the velocity change (ΔV_{Rear}) and the time interval

(Δt_{rear}) are determined, the acceleration of the rear car (A_{Rear}) is calculated, as shown in (5.7).

$$A_{Rear} = \frac{\Delta V_{Rear}}{\Delta t_{rear}} \tag{5.7}$$

5.4.2.4 <u>Measuring Ego Car Speed</u> We used the LM393 Speed Measuring Sensor to measure the speed of the Ego car. This sensor consists of an infrared (IR) LED and a photodetector (phototransistor) facing each other with a small gap between them. When the wheel rotates, it interrupts the light beam, causing the photodetector to output a digital signal. Knowing the wheel circumference, which is 6.6 cm, we were able to calculate the speed of the Ego car, as shown in equation (5.8).

$$V_{Ego} = \frac{distance\ traveled\ by\ the\ wheel}{time\ interval} \tag{5.8}$$

5.4.2.5 <u>Calculating the Acceleration Equation of Ego Car</u> The equation of the action (output) of the trained RL agent in terms of the states (inputs) was extracted using linear regression. The acceleration of the ego car is calculated based on various factors, including relative distance, relative acceleration, Ego, and rear car speeds. This equation (5.9) represents a linear combination of multiple factors affecting the acceleration of the ego car, each multiplied by respective coefficients.

$$E_{Acc} = 0.13658 - 0.31919 \times V_{Ego} + 0.31911 \times V_{Rear} + 0.0026726 \times D_{Rel} + 0.072491 \times A_{Rel} \tag{5.9}$$

where the first term "0.13658" is the bias, V_{Ego}, is the speed of the ego car, V_{Rear}, is the speed of the rear car, D_{Rel}, is the relative distance and A_{Rel}, is the relative acceleration. Then, we implemented a numerical integration step to calculate the ego car's speed based on its acceleration over a period of time.

5.4.2.6 Ego Car Algorithm Flowchart The flowchart of the ego car's algorithm is shown below in Figure 5.6.

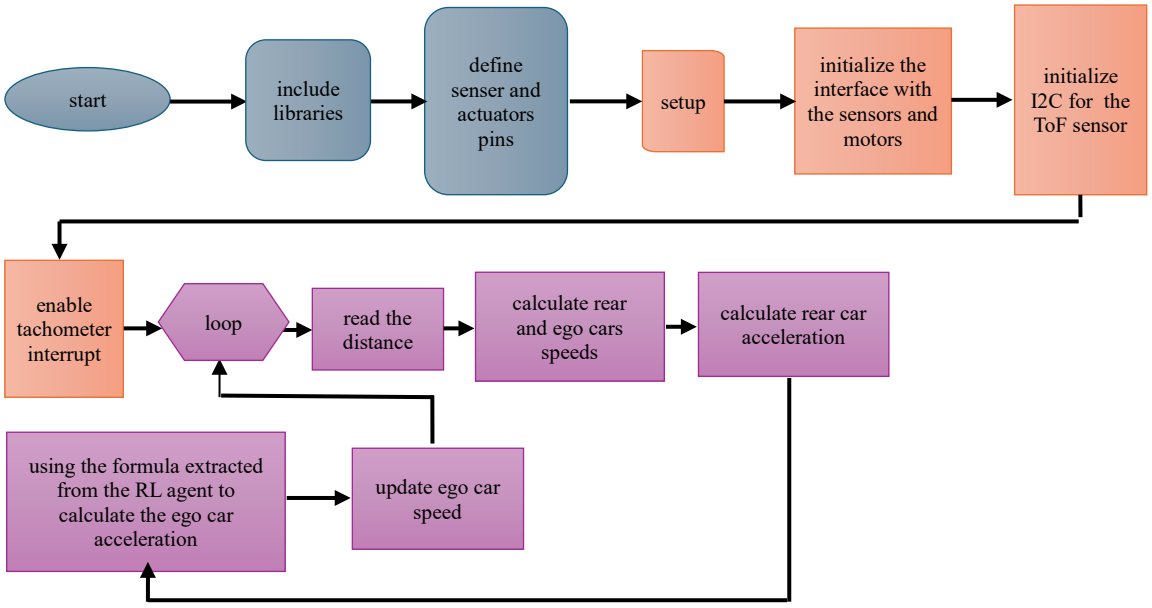

Figure 5.6 Ego car algorithm flowchart

5.5 Experimental Results

The next step is to visualize and display the experimental results. Visualizing the results from three different located cameras is essential. Over many days in my basement, we went through many testing scenarios.

5.5.1 Top View Camera Capture Both Cars and the Track

Figure 5.7 shows an HD 1080p Logi type "C922" Pro stream webcam that provides a top view of the whole track and both robotic cars. The scheme of videoing the track along with the two cars (top view) shows the webcam's position along with the track and the two cars, as shown in 5.8.

Figure 5.7 Webcam on a tripod to record footage of the track with both cars (top view)

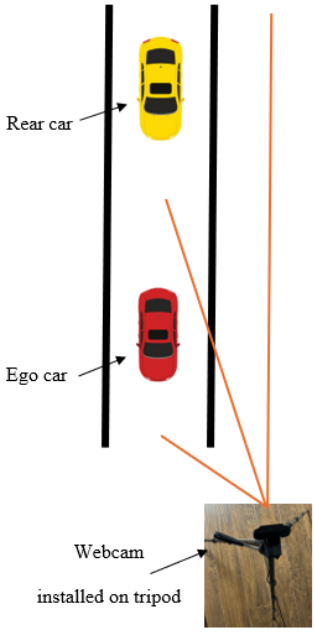

Figure 5.8 Scheme of videoing the track along with the two cars (top-view)

This top-view camera is used to video both cars in action and the track. This is important because it emphasizes the ego car's reaction to the rear car's actions, specifically when the rear car cruises, accelerates, or decelerates; see Figure 5.9 below, which shows some pictures from a video recorded by this camera.

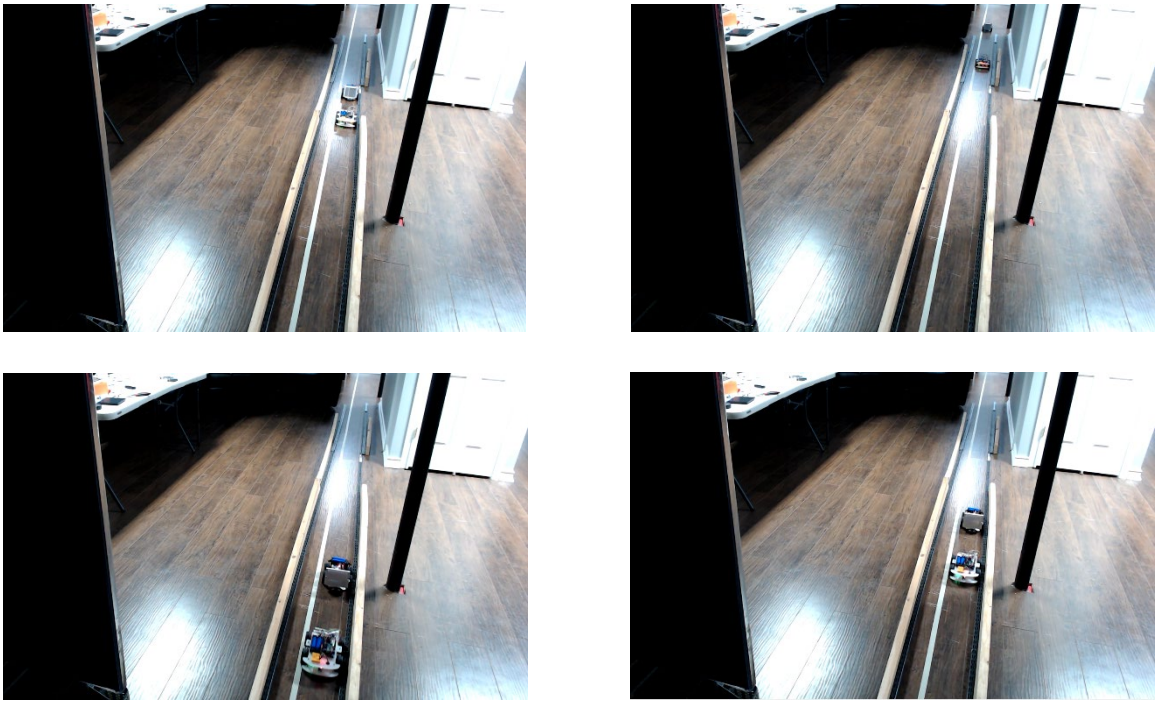

Figure 5.9 Some pictures from a video recorded by the top-view camera

5.5.2 Camera Mounted on the Back of the Ego Car to Video the Rear Car

A Logi 720P webcam mounted on the ego (lead) car captures the rear car, as shown in Figure 5.10. This scenario monitors the rear car from inside the ego car, simulating what passengers see and feel if they were inside the ego car. The scheme of the ego car's webcam videoing the rear car (top view) is shown in Figure 5.11. Some shots captured by this camera are shown below in Figure 5.12.

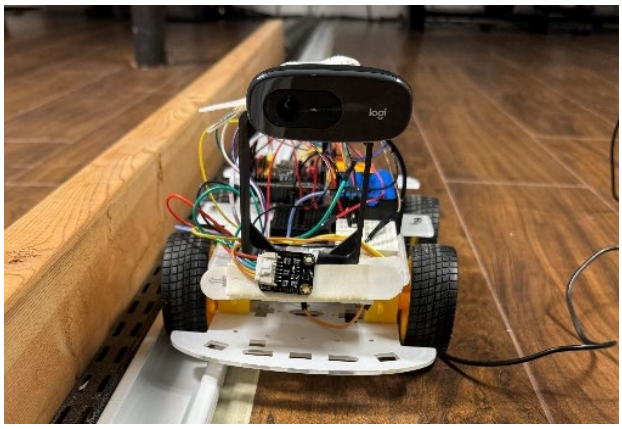

Figure 5.10 A Webcam is installed on the back of the ego car to video the rear car

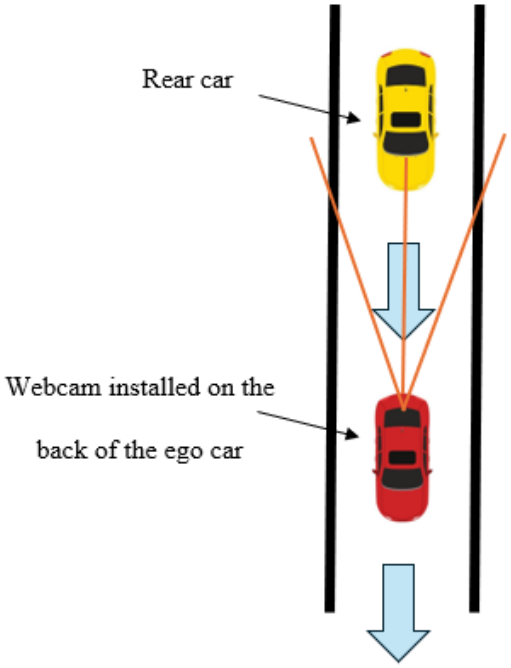

Figure 5.11 Scheme of ego car's webcam videoing the rear car (top-view)

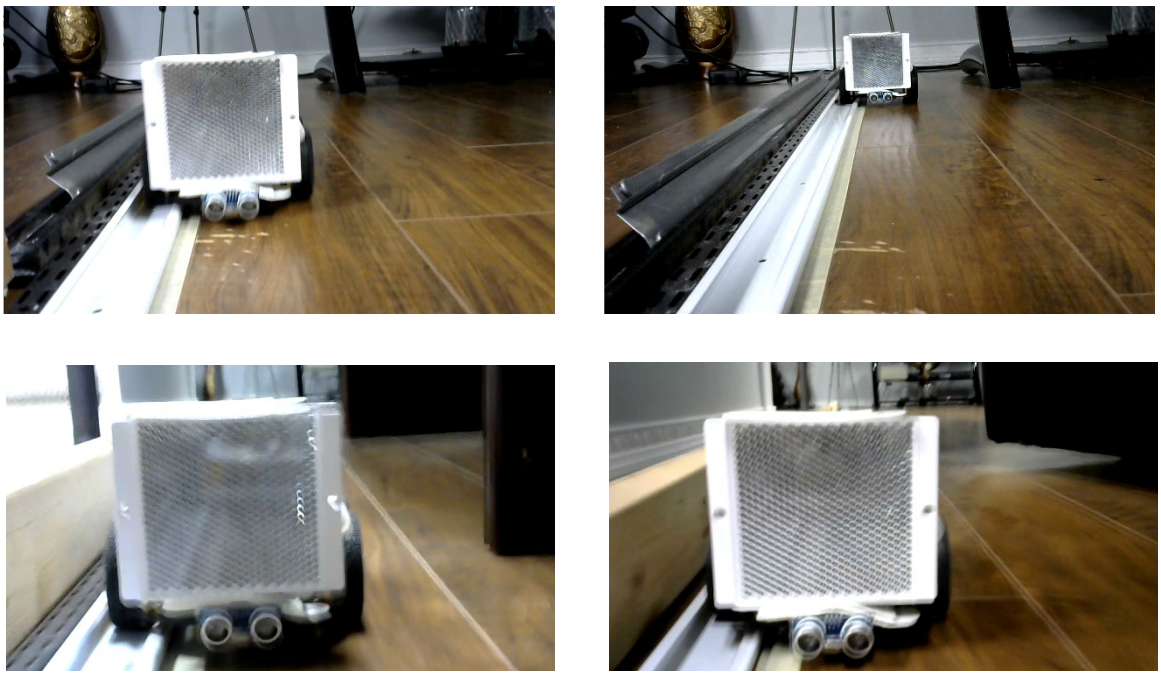

Figure 5.12 Some shots captured by the camera of the ego car

5.5.3 Camera Mounted on the Front of the Rear Car to Video the Ego Car

We removed the Logi 720P webcam from the Ego car and mounted it on the rear car to video the Ego car in motion, as shown in Figure 5.13. Figure 5.14 depicts the scheme of videoing the Ego car with the webcam installed on the rear car.

Figure 5.13 Webcam mounted to the rear car captures the front car

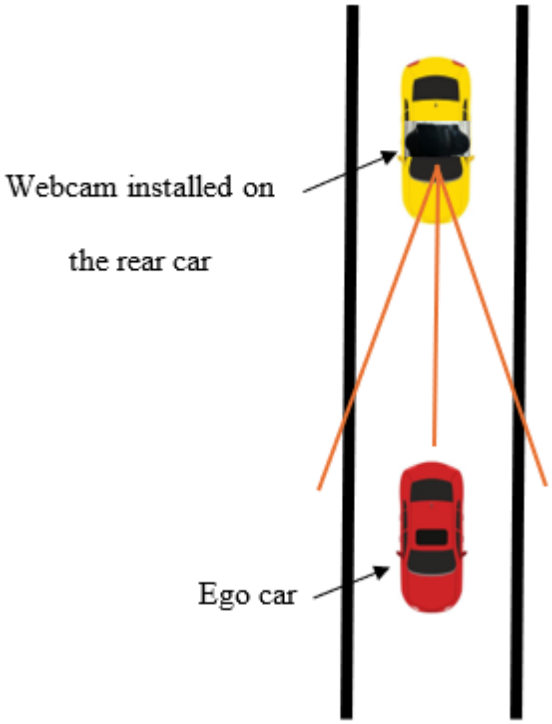

Figure 5.14 Scheme of rear car's webcam videoing the ego car (top-view)

This scenario monitors the ego car from inside the rear car, simulating what passengers see and feel if they are inside the rear car. Some shots from this scenario are shown below in Figure 5.15.

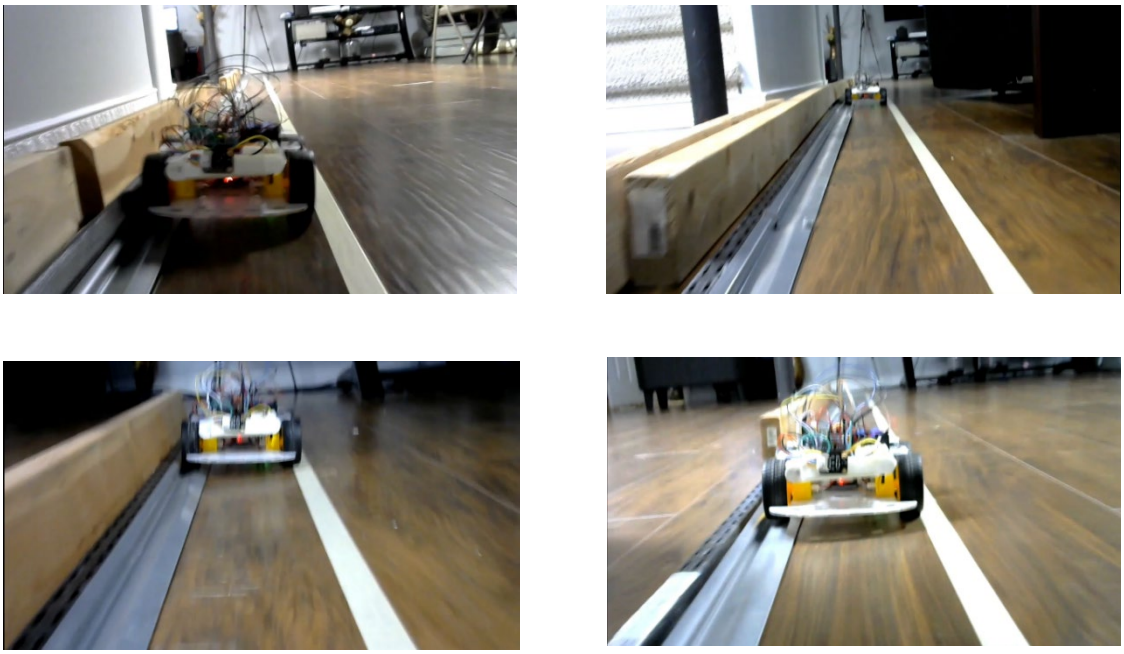

Figure 5.15 Some shots captured by the rear car camera

CHAPTER SIX

BASIS FUNCTIONS AND CONTROL ANALYSIS

Basis functions represent the essential functions of the approximated model that would formulate any data set in a mathematical formula. Combining basis functions with proper coefficients can precisely approximate any function within the domain. On the other hand, missing one or more functions may cause inaccurate representations. This defect can lead to erroneous modeling because those missing basis functions might be relevant to the functions' essential features. Consequently, the range of the basis functions could become unable to capture the data set's complexity, affecting the approximation's quality.

Basis function analysis (BFA) is very important in this thesis as it reveals the significance of each function of the linear regression model (LRM) we extracted from the trained agent of the RECA system. This analysis determines which function is a basis function (fundamental) and which is redundant or not essential (ineffective). Besides those benefits, BFA explains why the states (observations) assigned to the Reinforcement Learning (RL) agent as inputs were enough for the Reinforcement Learning Algorithm (LRA) in the agent training process to successfully achieve the task of maintaining a safe distance between the two cars involved in the study, which was the primary purpose of the Rear-End Collision Avoidance (RECA) system.

6.1 Rear-End Collision Avoidance Control Theory Analysis

Reinforcement Learning is an AI technique that solves control problems. However, the RLA is a black box, as was explained in this dissertation. Unraveling the

control theory used by the trained RL agent of the RECA is essential, just like basis function analysis (BFA) was necessary to deepen our comprehension of the inner workings of this complicated RLA and to unravel the reasons behind the good results that we got in terms of the RECA system, which in turn would help to increase our understanding and trusting to AI in general and to RL technique specifically.

Therefore, the next unreplaceable steps in my thesis, after modifying and developing a RECA system using RL, were extracting the trained RL agent (TRLA), verifying the TRLA by simulating it, extracting raw data from the TRLA, formulating an LRM from the data, verifying the reliability of the LRM by simulating it, a second step of verification to check the accuracy of the formulated LRM of the RECA system by using scaled robotic cars, and implementing basis functions analysis; therefore, the next irreplaceable step was to implementing control theory analysis (CTA) on the RECA system.

6.2 Rear-End Collision Avoidance Dynamics

The schematic in Figure 6.1 below shows the Rear-End Collision Avoidance System (RECA) dynamics. It shows the two cars involved in our system, the ego (front) and rear cars, with their positions, speeds, relative distance, and relative speed.

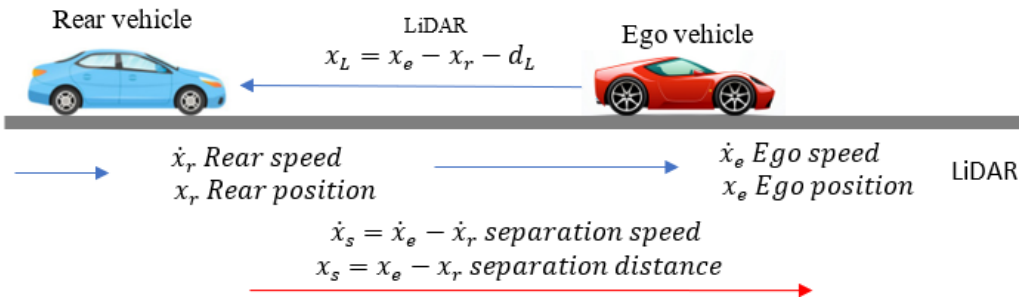

Figure 6.1 The Dynamics of the RECA System

6.2.1 Ego Car Dynamics

Equations (6.1) and (6.2) show the transfer function (TF) of the ego car's speed and position, respectively.

$$\dot{x}_e(s) \approx \frac{b_e}{s+a_e} u_e + \frac{d_e}{s+a_e} w_e \tag{6.1}$$

$$x_e(s) = \frac{1}{s} \dot{x}_e(s) \tag{6.2}$$

x_e and the $\dot{x}_e$ are the position and the speed of the ego car, respectively, u_e and w_e are the control and disturbance inputs, b_e is the input control gain and d_e is the disturbance gain. Equation (6.3) shows the state space equation (SSE) of the dynamics of the ego car.

$$\begin{bmatrix} \ddot{x}_e(t) \\ \dot{x}_e(t) \end{bmatrix} \approx \begin{bmatrix} -a_e & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_e(t) \\ x_e(t) \end{bmatrix} + \begin{bmatrix} b_e \\ 0 \end{bmatrix} u_e(t) + \begin{bmatrix} d_e \\ 0 \end{bmatrix} w_e(t) \tag{6.3}$$

6.2.2 Rear Car Dynamics

Equations (6.4) and (6.5) show that the rear car speed and position are $\dot{x}_r(t)$ & $x_r(t)$, respectively. Since their motion depends on the rear car driver, we treated and represented them as Brownian motion.

$$\dot{x}_r(s) = \frac{1}{s} w_r(s) \tag{6.4}$$

$$x_r(s) = \frac{1}{s} \dot{x}_r(s) \tag{6.5}$$

x_r and the $\dot{x}_r$ are the position and the speed of the rear car, respectively, and w_r, is unknown shaping input disturbance noise (acceleration or deceleration of the rear car, which is unpredictable). Equation (6.6) shows the state space equation (SSE) of the dynamics of the rear car.

$$\begin{bmatrix} \ddot{x}_r(t) \\ \dot{x}_r(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_r(t) \\ x_r(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w_r(t) \tag{6.6}$$

6.3 Relative Distance and Speed Between the Ego and Rear Vehicles

The separation (relative) distance between the two cars can be measured using a ranging sensor such as a Light Detection and Ranging (LiDAR) or any ToF sensor like the one we used in my application (Ranging Sensor). The relative distance can be represented as shown in equation (6.7).

$$x_s = x_e - x_r \tag{6.7}$$

x_s is the separation distance between the two cars, x_e is the ego car position, and x_r is the rear car position.

The LiDAR measures bumper-to-bumper distance as shown in equation (6.8).

$$x_L = x_e - x_r - d_L \tag{6.8}$$

d_L is car length, assuming both cars are the same length, and the LiDAR is mounted in the center of the ego car.

Compensating equation (6.7) into equation (6.8) gives the relationship shown in equation (6.9).

$$x_L = x_s - d_L \tag{6.9}$$

Equation (6.10) approximates the rate of change of the distance (speed) reported by the LiDAR distance.

$$\dot{x}_L \triangleq \frac{dx_L}{dt} \tag{6.10}$$

The backward difference approximation is shown in equation (6.11).

$$\hat{\dot{x}}_L \approx \frac{x_L(k) - x_L(k-1)}{\Delta T} = \frac{x_{L,k} - x_{L,k-1}}{\Delta T} \tag{6.11}$$

So since $\dot{x}_L = \dot{x}_e - \dot{x}_r$, the rear vehicle speed can be estimated (computed) in equation (6.12).

$$\hat{\dot{x}}_r = \dot{x}_e - \hat{\dot{x}}_L \tag{6.12}$$

6.4 Dynamics of Separation in the Time Domain

From (6.7), we know $x_s = x_e - x_r$; We can then define the separation motion variables, as shown in (6.13) and (6.14).

$$x_s = x_e - x_r$$
$$\dot{x}_s = \dot{x}_e - \dot{x}_r \quad (6.13)$$
$$\ddot{x}_s = \ddot{x}_e - \ddot{x}_r \quad (6.14)$$

$\dot{x}_s$ the relative speed, $\dot{x}_e$ is the ego car speed, and $\dot{x}_r$ is the rear car speed.

We can also show that the dynamics for separation are represented by the equation of motion shown in equation (6.15).

$$\begin{bmatrix} \ddot{x}_s(t) \\ \dot{x}_s(t) \end{bmatrix} = \begin{bmatrix} \ddot{x}_e(t) \\ \dot{x}_e(t) \end{bmatrix} - \begin{bmatrix} \ddot{x}_r(t) \\ \dot{x}_r(t) \end{bmatrix} \quad (6.15)$$

Equation (6.16) represents the dynamics of the two vehicles' separation (center to center) affected by ego driver input u_e, the rear car speed $\dot{x}_r$, and unknown disturbances, $d_e w_e(t) - w_r(t)$.

$$\begin{bmatrix} \ddot{x}_s(t) \\ \dot{x}_s(t) \end{bmatrix} = \underbrace{\begin{bmatrix} -a_e & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_s(t) \\ x_s(t) \end{bmatrix} + \begin{bmatrix} b_e \\ 0 \end{bmatrix} u_e(t)}_{controllable\ term} + \underbrace{\begin{bmatrix} -a_e \\ 0 \end{bmatrix} \dot{x}_r(t)}_{\substack{uncontrollable \\ observable \\ term}} \ldots$$
$$+ \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix} \left(d_e w_e(t) - w_r(t) \right)}_{\substack{uncontrollable \\ unobservable \\ term}} \quad (6.16)$$

Equation (6.17) represents the generic form of the state space model (SSM) that shows how the state vector evolves based on the linear relationships defined by the matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$. The generic form of the SSM is usually used to describe linear time-invariant (LTI) systems, in which the system's future state is determined by its current state and any control inputs applied to it.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{6.17}$$

$\dot{\boldsymbol{x}}$ denotes the rate of change (derivative) of the state vector $\boldsymbol{x}$, indicating how the state changes over time. $\boldsymbol{A}$ represents the state matrix, which describes the system's dynamics and relates $\boldsymbol{x}$ to the $\dot{\boldsymbol{x}}$. $\boldsymbol{B}$ is the input matrix, which describes how the control input $\boldsymbol{u}$ affects the state $\boldsymbol{x}$. While $\boldsymbol{x}$ denotes the state vector, representing the system's state at any given time. The last variable is $\boldsymbol{u}$ which represents the input vector (external control signals applied to the system) [69-70].

We extended generic SSM by incorporating disturbances and uncertainties, as shown in equation (6.18). This new formula is more reliable for real-world applications as it considers external influences such as noise and counts for uncertainties.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{D_0}\boldsymbol{w_0} + \boldsymbol{D_u}\boldsymbol{w_u} \tag{6.18}$$

$\boldsymbol{x} = \begin{bmatrix} \dot{x}_s(t) \\ x_s(t) \end{bmatrix}$ represents the states (speed and displacement) of the separation

$\boldsymbol{u} = u_e(t)$ represents the accelerator (pedal) input to the ego car

$\boldsymbol{w_O} = \dot{x}_r(t)$ represents the observable computable speed of the rear car

$\boldsymbol{w_u} = d_e w_e(t) - w_r(t)$ represents the unobservable disturbances.

From equation 6.16, we have:

$$\boldsymbol{A} = \begin{bmatrix} -\boldsymbol{a_e} & \boldsymbol{0} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \boldsymbol{b_e} \\ \boldsymbol{0} \end{bmatrix}, \ \boldsymbol{D_O} = \begin{bmatrix} -\boldsymbol{a_e} \\ \boldsymbol{0} \end{bmatrix}, \boldsymbol{and}\ \boldsymbol{D_u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The block diagram of the extended State Space Model is shown in Figure 6.2.

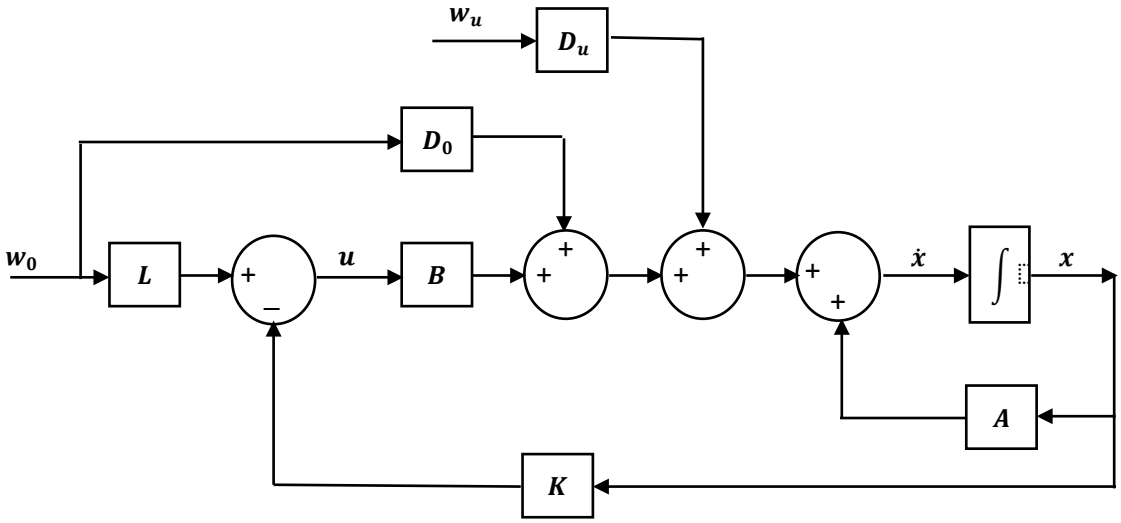

Figure 6.2 The block diagram of the extended State Space Model

6.5 Control Analysis

The control input for the state-space model consists of system state feedback ($-\boldsymbol{Kx}$) and a feedforward term of observable variables ($\boldsymbol{Lw_0}$).

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{D_0 w_0} + \boldsymbol{D_u w_u} \tag{6.19}$$

$$\boldsymbol{u} = -\boldsymbol{Kx} + \boldsymbol{Lw_0} \tag{6.20}$$

6.5.1 First Set of Basis Functions for the LRM

The state feedback control is given in equation (6.20).

$$\begin{aligned} u(t) &= -\boldsymbol{Kx} + \boldsymbol{Lw_0} \\ &= -\begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} \dot{x}_s(t) \\ x_s(t) \end{bmatrix} + [l_1]\dot{x}_r(t) \\ &= -k_1\dot{x}_s(t) - k_2 x_s(t) + l_1\dot{x}_r(t) \\ &= -k_1(\dot{x}_e(t) - \dot{x}_r(t)) - k_2(x_L(t) + d_L) + l_1\dot{x}_r(t) \\ &= \underbrace{(-k_1)\dot{x}_e(t)}_{Ego\ Speed} + \underbrace{(k_1 + l_1)\dot{x}_r(t)}_{Rear\ Speed} + \underbrace{(-k_2)x_L(t)}_{\substack{Relative\ Distance \\ (LiDAR)}} + \underbrace{(-k_2)d_L}_{\substack{constant \\ (bias)}} \end{aligned} \tag{6.21}$$

Equation (6.21) provides the first set of the basis functions for the RECA system. We can see three essential functions besides the bias: ego speed, rear speed, and relative distance. We noticed that the relative acceleration, one of the original states of the RL agent at the training stage, is not among the basis functions, indicating that it's not essential for the system; the same thing happened to relative speed, which the extracted LRM already omitted. Our control analysis verified the previous results we got from the LRM regarding the relative speed and added relative acceleration as noncritical in the RECA system.

6.5.2 Second Set of Basis Functions for the LRM

Further analysis led to the second set of basis functions, as in equation (6.22).

$$\begin{aligned} u(t) &= (-k_1)\dot{x}_e(t) + (k_1 + l_1)(\dot{x}_e - \hat{\dot{x}}_L) + (-k_2)x_L(t) + (-k_2)d_L \\ &= \underbrace{(l_1)\dot{x}_e(t)}_{Ego\ Speed} + \underbrace{(-k_1 - l_1)(\hat{\dot{x}}_L)}_{Relative\ Speed} + \underbrace{(-k_2)x_L(t)}_{\substack{Relative\ Distance \\ (LiDAR)}} + \underbrace{(-k_2)d_L}_{\substack{Constant \\ (bias)}} \end{aligned} \tag{6.22}$$

Equation (6.22) provides the second set of the basis function for the RECA system. In the second set, there are three essential functions besides the bias: ego speed, relative speed, and relative distance. We noticed that the relative acceleration was not within the basis function for the second time, so we concluded that the relative acceleration was not critical to the RECA system. we noticed in the second set that the relative speed is within the basis functions, but the rear car speed is not. From both sets, we concluded that we need only two of these three functions, ego car speed, rear car speed, and relative speed because having any two of them means we implicitly have the third one, as depicted in equation (6.13), $\dot{x}_s = \dot{x}_e - \dot{x}_r$.

6.6 Simulating the LRM With the Basis Functions Only

Verifying the basis functions analysis is crucial to confirm the credibility of the analysis and the results. We used the first set of the basis functions, shown in equation (6.21) above. The first set of LRM basis functions encompasses rear car speed, ego car speed, and the relative distance beside the bias. It's worth remembering that the second of the basis functions, shown in equation (6.22), encompasses, besides the bias, the ego car speed, relative speed, and relative distance. Because the LRM excluded the relative speed by setting the coefficient of the relative speed to zero, we cannot simulate it because, in this case, we have only one speed, which is the ego car speed, but as explained before, we need at least two of the three speeds. Figure 6.3 shows the Simulink model of the first set of the basis functions of the LRM.

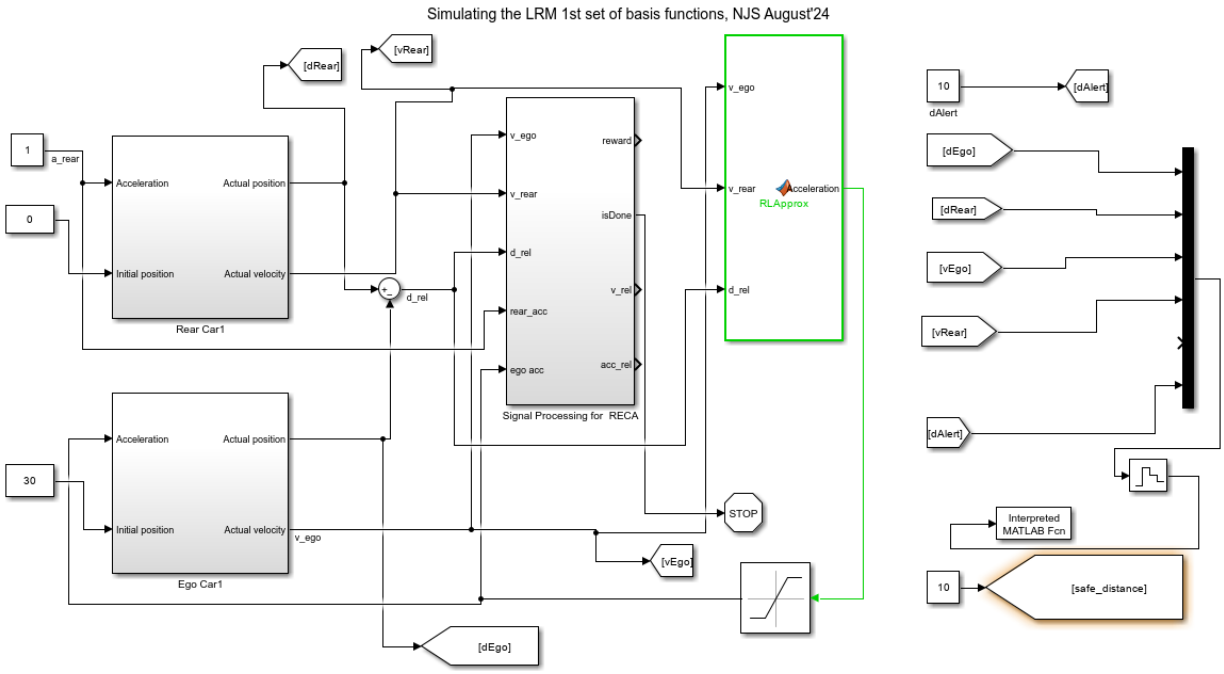

Figure 6.3 Simulink model of the first set of the basis functions of the LRM

6.7 Results of Simulating the LRM With Basis Functions

Figures 6.4 and 6.5 show the simulation results of the LRM with the basis functions. The first plot shows the ego and rear car speeds, and the second shows the relative distance.

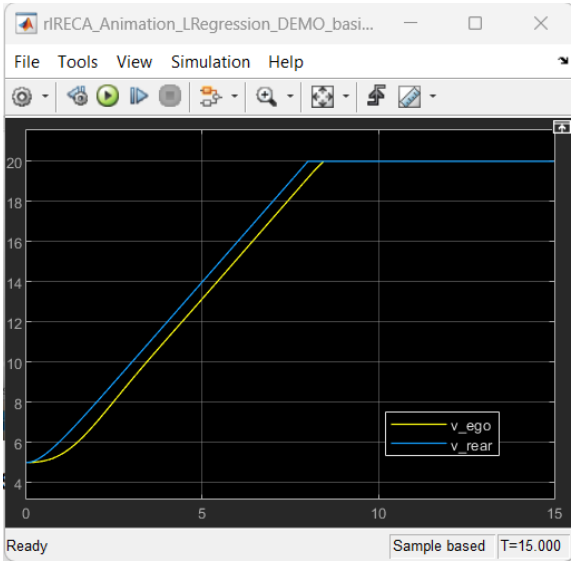

Figure 6.4 Ego and rear car speeds of the 1st set of the LRM basis functions

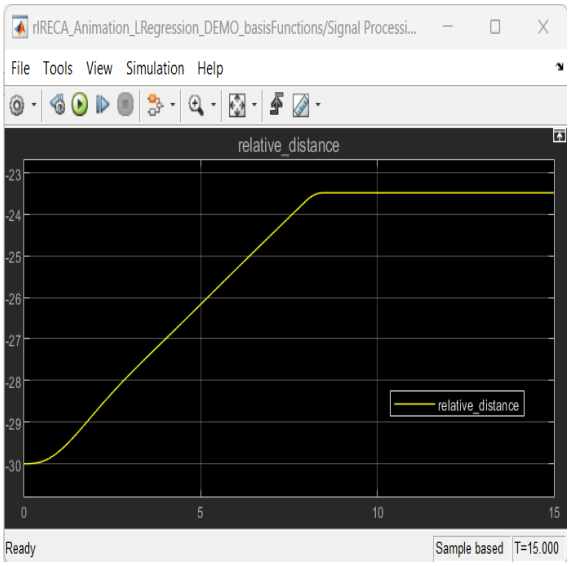

Figure 6.5 Relative distance of the 1st set of the LRM basis functions

From the above relative distance plot, we notice that the relative distance of this model decreased starting from the beginning of the simulation when the separation distance was 30 meters. Then, it was stabilized and maintained above the safe distance. From the speeds of the two cars plot, we can see that the ego was trying to catch up with the speed of the rear car starting at the beginning of the simulation, where the initial speed was ~11 mph until the rear car speed saturated at ~45 mph, where at that speed, both cars' speeds maintained at the same level.

6.8 Analysis of Closed-Loop Stability

To further understand the necessity of the basis functions, we analyze the stability of the closed-loop state-space model as follows. Substituting equation (6.20) into equation (6.19), we obtained:

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{B}(-\boldsymbol{Kx} + \boldsymbol{Lw_0}) + \boldsymbol{D_0 w_0} + \boldsymbol{D_u w_u}$$

$$= (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} + (\boldsymbol{D_0} + \boldsymbol{BL})\boldsymbol{w_0} + \boldsymbol{D_u w_u}$$

$$\boldsymbol{A} - \boldsymbol{BK} = \begin{bmatrix} -\boldsymbol{a_e} & \boldsymbol{0} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{b_e} \\ \boldsymbol{0} \end{bmatrix} [\boldsymbol{k_1} \quad \boldsymbol{k_2}] = \begin{bmatrix} -\boldsymbol{a_e} - \boldsymbol{b_e k_1} & -\boldsymbol{b_e k_2} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix}$$

The stability of the closed-loop matrix $\boldsymbol{A} - \boldsymbol{BK}$ depends on $\boldsymbol{k_1}$ and $\boldsymbol{k_2}$. It is necessary to make sure that $-\boldsymbol{a_e} - \boldsymbol{b_e k_1} < 0$ and $-\boldsymbol{b_e k_2} < 0$, so that $\boldsymbol{A} - \boldsymbol{BK} = \begin{bmatrix} -\boldsymbol{a_e} - \boldsymbol{b_e k_1} & -\boldsymbol{b_e k_2} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix}$ is a matrix with stable eigenvalues.

If $k_2 = 0$, then A-BK becomes $\boldsymbol{A} - \boldsymbol{BK} = \begin{bmatrix} -\boldsymbol{a_e} - \boldsymbol{b_e k_1} & \boldsymbol{0} \\ \boldsymbol{1} & \boldsymbol{0} \end{bmatrix}$.

It will have a zero eigenvalue and cause the system to not converge in tracking. We can conclude from equations 6.21 and 6.22 that the relative distance (ranging sensor readings)

must be used as a basis function (compensable) in the LRM. This argument shows that we must use:

$$\underbrace{(-k_2)x_L(t)}_{\substack{Relative\ Distance \\ (LiDar)}} + \underbrace{(-k_2)d_L}_{\substack{Constant \\ (bias)}}$$

If $k_1 = 0$, then A-BK becomes $\boldsymbol{A - BK} = \begin{bmatrix} -\boldsymbol{a_e} & -\boldsymbol{b_e k_2} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}$.

In this case, the system would lose stability because it would lose the ability to control the relative speed between the ego and rear cars. So, in this case, if the rear car starts accelerating, the ego car won't be able to increase the speed as it's speed constrained with a constant value, $-\boldsymbol{a_e}$.

6.9 Simulating the RECA System Excluding D_{Rel} from the Basis Functions

Then, we simulated the RECA system with excluding the D_{Rel} from the basis functions. The Simulink model is shown in Figure 6.6. The results are depicted in Figures 6.7 and 6.8. We can notice from Figure 6.7 that both cars' speeds initially started at ~11 mph (5 mps) while the rear car's speed was increasing; the ego car's speed wasn't fast enough to catch the ego car's speed. Figure 6.8 shows that the relative speed dropped to zero, which means the two vehicles collided. These results show the significance of $\boldsymbol{k_2}$ which is the LiDAR to the complete set of the basis functions to maintain a minimum safe distance between the two cars.

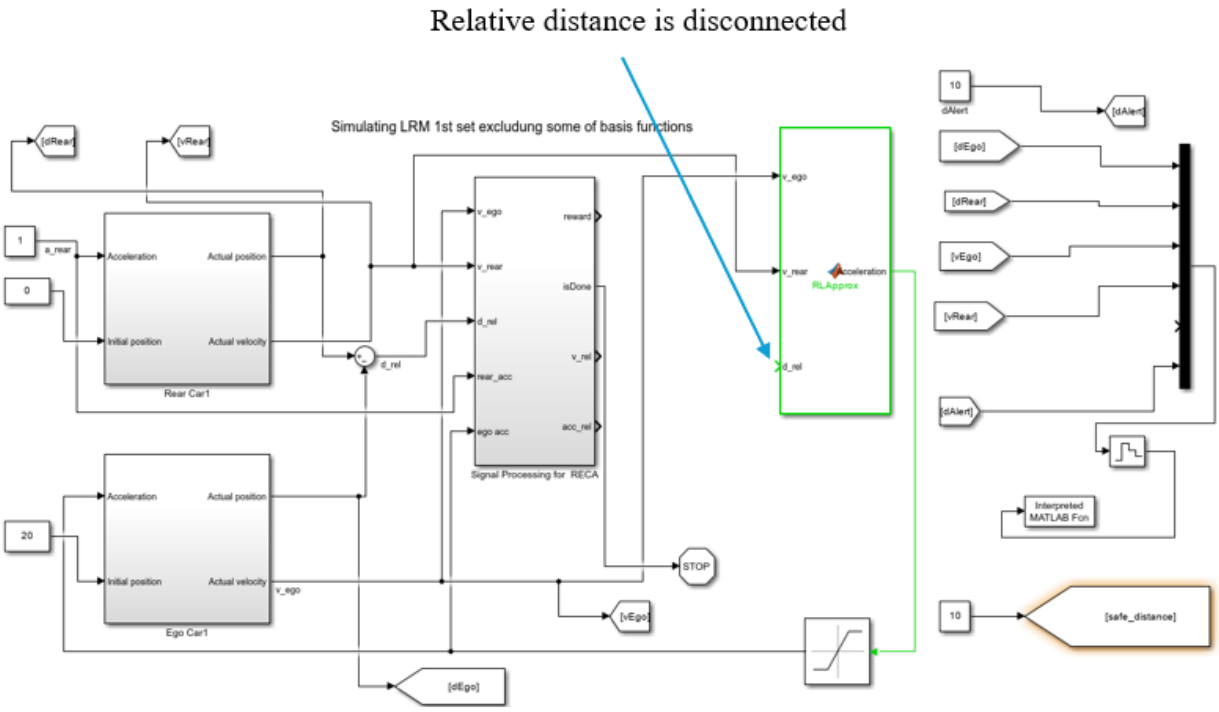

Figure 6.6 Simulink model of the RECA system excluding D_{Rel} from basis functions

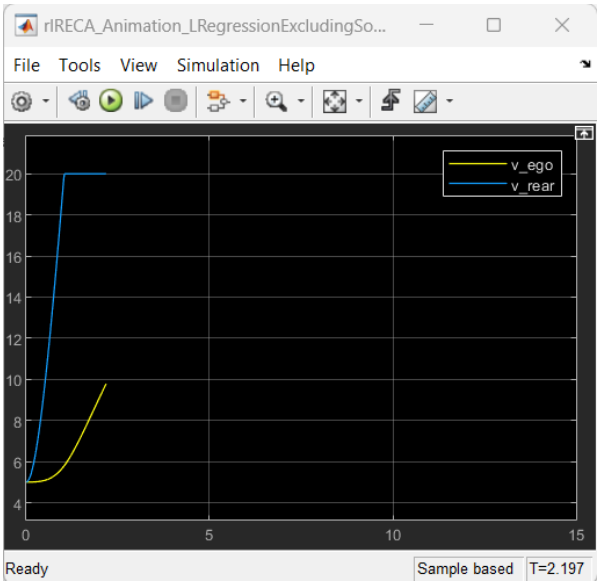

Figure 6.7 Speeds of Ego and Rear cars when excluding D_{Rel} from basis functions

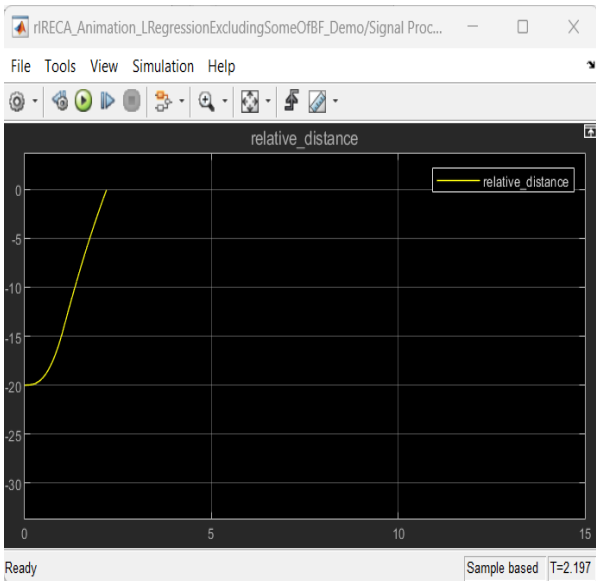

Figure 6.8 Relative distance when excluding D_{Rel} from basis functions

Figure 6.9 shows a snippet captured from the RECA simulation when D_{Rel} excluded from the basis functions of the linea regression model (LRM). The D_{Rel} was excluded by setting $k_2 = 0$,

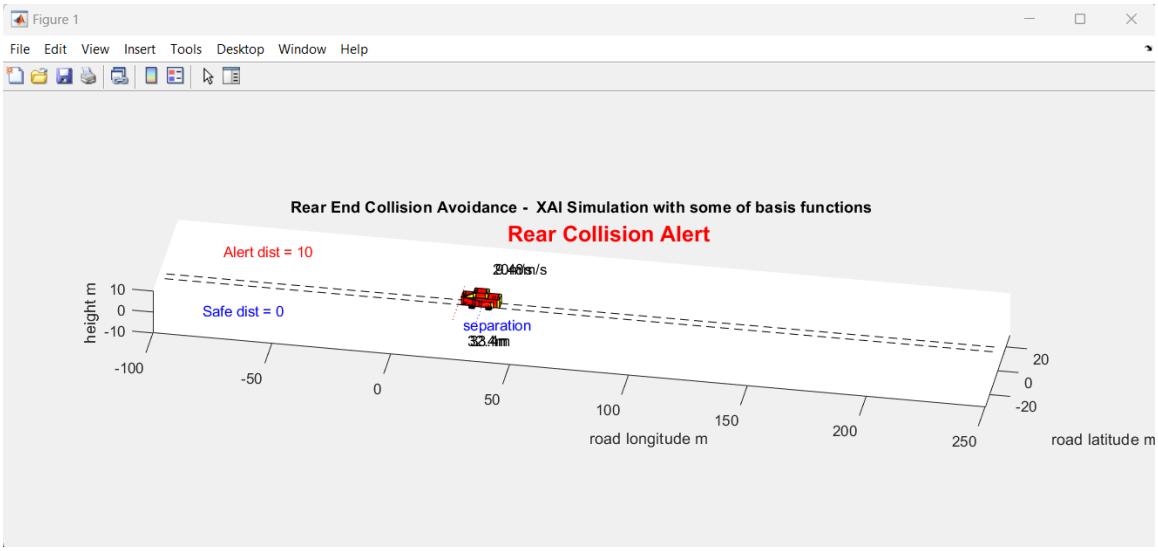

Figure 6.9 Snippet from the animation when excluding D_{Rel} from basis functions

CHAPTER SEVEN

CONTRIBUTION, RECOMMENDATION, CONCLUSION AND FUTURE WORK

7.1 Contribution

With the increase of human dependence on the products and technologies that use AI, where using AI-based products is considered a daily routine for most people nowadays; however, the ways that the neural networks in these algorithms are taking to come up with their decisions are still very complex and ambiguous. This thesis contributes to demystifying this complexity by addressing the problem of a black box of AI algorithms and making AI-based systems interpretable by humans. Unraveling the cover of these algorithms will profoundly contribute to increasing trust in many fields, such as healthcare, finance, legal, and automotive, and make AI-based products more user-friendly.

This thesis introduces several innovative contributions and original work, including:

1. Formulate the RECA as a novel scheme for the Advanced Driving Assistance System (Chapter 3.5)
2. Formulate the Reinforcement Learning method to train an RL agent to operate as a Rear-End Collision Avoidance (RECA) system (Chapter 3.6).
3. Use MATLAB Simulink and RL toolbox to train the RL agent (black box) (Chapter 3.9).
4. Use MATLAB to extract the RECA-trained RL agent (Chapter 3.10).
5. Simulate the trained RECA-RL agent to verify it (Chapter 3.10).

6. Extract information from a trained RECA RL agent using an explainable artificial intelligence (XAI) technique (function approximation) (Chapter 4.1).
7. Fit the extracted information to a Linear Regression Model (LRM) (Chapter 4.3).
8. Verify the LRM of the RECA system using Simulink (Chapter 4.4).
9. Verify the LRM of the RECA system on a scaled robotic (ego car) (Chapter 5.4.2).
10. Use the state-space control principle to reveal the LRM basis function set (Chapter 6.5).
11. Verify the basis functions of the LRM using Simulink (Chapter 6.6).
12. Analyze the stability of XAI with the state-space control model (Chapter 6.8).

7.2 Recommendation

This thesis could be significant for the theory and application sides in many fields; for example, in the theory and research part, we recommend the researchers refer to this thesis as a starting point that it may open the door for them to explore other XAI techniques and using different control techniques to understand the results. On the application side, they can use other AI techniques, such as deep learning and machine learning; plus, they may explore different fields, such as health and finance. In academia, the instructors in the engineering departments can cite some of the application parts of this thesis and assign it as a project to graduate-level students. These types of projects are significant because they prepare the students to deal with real-world challenges in their career lives. In industries such as automotive, this thesis is expected to help scientists and engineers who work with these complicated algorithms respond to stakeholders' and end users' demands for an acceptable explanation of how AI algorithms produce their results.

This thesis offers thorough work that could be used as guidance for scientists and engineers willing to start working in the field of explainable artificial intelligence (XAI) with real-world applications.

7.3 Conclusion

This dissertation explores and develops an explainable artificial intelligence (XAI) model for a rear-end collision avoidance (RECA) system implemented by a reinforcement learning trained agent. The RECA RL agent was formulated using Simulink, and the training was done using MATLAB, Reinforcement Learning toolbox, and Simulink. The trained RL agent was verified using Simulink. MATLAB code was developed to extract the states (inputs) and the action (output) from the trained RL agent. A linear regression model (LRM) was extracted and optimized from the trained RL agent's inputs. The extracted LRM was simulated and verified using Simulink. Using two scaled robotic cars, LRM was tested and verified. Basis function and control analysis were implemented to determine the essential (basis) states (functions) and to explain why the trained RL agent was able to achieve the task of maintaining a safe distance between the two cars of the RECA system. finally, the basis function formula was simulated and verified using Simulink.

Chapter two of the dissertation provides a brief overview of the problem of the black box of AI networks. The chapter also proposes using function approximation as an explainable artificial intelligence (XAI) to model AI networks. Also, this chapter proposes implementing control and basis function analysis for the RECA system and verifying the results using software and hardware.

Chapter three of the dissertation provides a detailed overview of using the RL technique as a case study for this thesis. This chapter explains RL, its essential components, and the DDPG algorithm used in this thesis. Then, it discusses the available systems of the RECA and suggests a novel idea to address this problem. It also overviews the methodology used to train the RECA agent using MATLAB, Simulink, and the Reinforcement Learning toolbox, as well as modeling and simulating the RECA with RL, and finally displays and discusses the results.

Chapter four of the dissertation describes how information is extracted from the RECA system's trained RL agent black box. It explains how the trained agent's raw inputs/output data are extracted and plotted using MATLAB. Then, it delves into how to use linear regression to model the trained RL agent. The last topics discussed in this chapter were simulating the LRM using Simulink and explaining the results.

Chapter five of the dissertation provides a detailed overview of the experimental part to verify the LRM extracted from the trained RL agent. we used two scaled robotic cars on a straight track for the experiments. This chapter details the two scaled cars with their microcontroller features, all the sensors both vehicles are equipped with, and the track used in these experiments. The flowcharts of both car's algorithms and all the math used to calculate the variables are also included in this chapter.

Chapter six of the dissertation explains how control theory substantiates the linear regression (LR) function extracted from the trained reinforce learning (RL) scheme. A state space control technique was used to formulate a state-space model for the RECA system. We then presented a state feedback and reference feedforward control input, illustrating a set of necessary variables. We showed that the input can then be

transformed into a combination of the basis functions corresponding to the extracted linear regression (LR) function. This explains a way to choose LR basis functions. Having the necessary basis variables also allows the RL algorithm to converge to a solution in the first place. Additionally, the control analysis explains how the stability of the RECA can be affected if some variables are missing.

In summary, AI algorithms can be complex and perceived as black boxes. This dissertation demonstrates a technique for extracting information from an AI black box and explaining the input/output relationship using linear regression of basis functions. The work is analyzed using mathematical formulation and supported by simulation and hardware experiments applied to a rear-end collision avoidance scheme.

7.4 Future Work

In the theory part, we will use other control theories, such as Fuzzy Logic, to analyze our system. Despite that, the emphasis of our dissertation, " Extraction of Information from AI Design Decisions – Theoretical and Practical Case Study," is on extracting information from AI design decisions regardless of the application. However, we can work on the practical part to make it more applicable for the real world by involving a third car in front of the ego can and allowing for lane change when available. In this case, we will have a Forward and Rear-Ends Collision Avoidance, as shown in Figure 7.1.

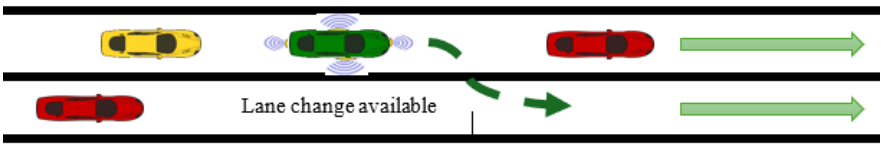

Figure 7.1 Forward and Rear-Ends Collision Avoidance

REFERENCES

[1] Tom Mitchell, "Three future trends in AI: perception natural language processing and the proliferation of mobile phone technology"[J]", Shanghai Informatization, vol. 09, pp. 23-24, 2019.

[2] Dwivedi, Yogesh K., et al. "Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy." International journal of information management 57 (2021): 101994.

[3] Borenstein, Jason, Joseph Herkert, and Keith Miller. "Self-driving cars: Ethical responsibilities of design engineers." IEEE Technology and Society Magazine 36.2.

[4] https://www.mathworks.com/products/reinforcement-learning.html (2017): 67-75.

[5] https://www.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-for-adaptive-cruise-control.html.

[6] Adaptive Cruise Control System Using Model Predictive Control - MATLAB & Simulink (mathworks.com).

[7] https://www.mathworks.com/help/stats/fitlm.html.

[8] https://raven.aero/courses/artificial-intelligence-for-aviation-applications/#:~:text=Such%20tasks%20may%20include%20visual,in%20many%20industries%2C%20including%20aviation.

[9] https://www.skillup-air.eu/2021/12/artificial-intelligence-in-aviation/.

[10] Das, Arun, and Paul Rad. "Opportunities and challenges in explainable artificial intelligence (xai): A survey." arXiv preprint arXiv:2006.11371 (2020).

[11] Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. Introduction to linear regression analysis. John Wiley & Sons, 2021.

[12] Benefits and Use Cases of AI in the Automotive Industry (appinventiv.com).

[13] Akamatsu, Motoyuki, Paul Green, and Klaus Bengler. "Automotive technology and human factors research: Past, present, and future." International journal of vehicular technology 2013.1 (2013): 526180.

[14] https://www.idtechex.com/en/research-article/a-history-of-adas-emergence-to-essential/25592.

[15] Wikipedia, Anti-lock braking system, homepage, https://en.wikipedia.org/wiki/Anti-lock_braking_system.

[16] https://carlinesmotors.ie/the-evolution-of-car-safety/.

[17] AUTOGLASS, The Evolution of ADAS: A Look Back and Ahead, homepage,https://www.autoglass.ie/the-evolution-of-adas-a-look-back-and-ahead/#:~:text=The%20First%20ADAS&text=It%20wasn't%20until%20the,significant%20milestone%20in%20vehicle%20safety.

[18] CARADAS, ADAS Sensors Guide: The Different Sensors ADAS Systems, homepage, Use https://caradas.com/adas-sensors-guide/.

[19] S. Jain et al., "Blockchain and Autonomous Vehicles: Recent Advances and Future Directions," in IEEE Access, vol. 9, pp. 130264-130328, 2021, doi: 10.1109/ACCESS.2021.3113649.

[20] A. Validi, T. Ludwig and C. Olaverri-Monreal, "Analyzing the effects of V2V and ADAS-ACC penetration rates on the level of road safety in intersections: Evaluating simulation platforms SUMO and scene suite," 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Vienna, Austria, 2017, pp. 38-43, doi: 10.1109/ICVES.2017.7991898.

[21] Liang Li, G. Lu, Yunpeng Wang and Daxin Tian, "A rear-end collision avoidance system of connected vehicles," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp. 63-68, doi: 10.1109/ITSC.2014.6957667.

[22] T. Kasuga and S. Yakubo, "Design of a Highly Safe Model Vehicle for Rear-End Collision Avoidance Considering Multiple Faults of Sensors," 2009 International Conference on Computational Intelligence, Modelling and Simulation, Brno, Czech Republic, 2009, pp. 63-68, doi: 10.1109/CSSim.2009.20.

[23] A. Hosny, M. Yousef, W. Gamil, M. Adel, H. Mostafa and M. S. Darweesh, "Demonstration of Forward Collision Avoidance Algorithm Based on V2V Communication," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2019, pp. 1-4, doi: 10.1109/MOCAST.2019.8741580.

[24] NISSAN MOTOR CORPORATION, Rear Cross Traffic Alert, homepage, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/RCTA/.

[25] BOSCH, Rear Cross Traffic Warning (Warring of crossing vehicles when backing out of a parking space), homepage, https://www.bosch-mobility.com/en/solutions/parking/rear-cross-traffic-warning/.

[26] https://patents.google.com/patent/US6831572B2/en.

[27] A. Kimura and M. T. Nadege, "RoI Editing Tool Using AI for Radiotherapy Simulation," 2020 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), Boston, MA, USA, 2020, pp. 1-3, doi: 10.1109/NSS/MIC42677.2020.9508040.

[28] J. L. C. Sanz and Y. Zhu, "Toward Scalable Artificial Intelligence in Finance," 2021 IEEE International Conference on Services Computing (SCC), Chicago, IL, USA, 2021, pp. 460-469, doi: 10.1109/SCC53864.2021.00067.

[29] Y. A. Rani, A. Balaram, M. R. Sirisha, S. A. Nabi, P. Renuka and A. Kiran, "AI Enhanced Customer Service Chatbot," 2024 International Conference on Science Technology Engineering and Management (ICSTEM), Coimbatore, India, 2024, pp. 1-5, doi: 10.1109/ICSTEM61137.2024.10561155.

[30] S. K. Vishnumolakala, S. C. C, N. P. Subheesh, P. Kumar and R. Kumar, "AI-Based Research Companion (ARC): An Innovative Tool for Fostering Research Activities in Undergraduate Engineering Education," 2024 IEEE Global Engineering Education Conference (EDUCON), Kos Island, Greece, 2024, pp. 1-5, doi: 10.1109/EDUCON60312.2024.10578646.

[31] J. Henriksson, M. Borg and C. Englund, "Automotive Safety and Machine Learning: Initial Results from a Study on How to Adapt the ISO 26262 Safety Standard," 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), Gothenburg, Sweden, 2018, pp. 47-49.

[32] R. Nautiyal, R. S. Jha, S. Kathuria, Y. Chanti, N. Rathor and M. Gupta, "Intersection of Artificial Intelligence (AI) in Entertainment Sector," 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2023, pp. 1273-1278, doi: 10.1109/ICOSEC58147.2023.10275976.

[33] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders and K. -R. Müller, "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications," in Proceedings of the IEEE, vol. 109, no. 3, pp. 247-278, March 2021, doi: 10.1109/JPROC.2021.3060483.

[34] S. Chakraborty et al., "Interpretability of deep learning models: A survey of results," 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 2017, pp. 1-6, doi: 10.1109/UIC-ATC.2017.8397411.

[35] Preece, A., Harborne, D., Braines, D., Tomsett, R., & Chakraborty, S. (2018). Stakeholders in Explainable AI. ArXiv. /abs/1810.00184.

[36] A.S. Albahri, Ali M. Duhaim, Mohammed A. Fadhel, Alhamzah Alnoor, Noor S. Baqer, Laith Alzubaidi, O.S. Albahri, A.H. Alamoodi, Jinshuai Bai, Asma Salhi, Jose Santamaría, Chun Ouyang, Ashish Gupta, Yuantong Gu, Muhammet Deveci, A systematic review of trustworthy and explainable artificial intelligence in healthcare: Assessment of quality, bias risk, and data fusion, Information Fusion, Volume 96, 2023, Pages 156-191, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2023.03.008. (https://www.sciencedirect.com/science/article/pii/S1566253523000891).

[37] Devineni, Siva Karthik. "From chaos to clarity: revolutionizing industries with AI for enhanced trust, efficiency, and innovation." Journal of Scientific and Engineering Research 11.4 (2024): 116-128.

[38] Mensah, George Benneh. "Artificial intelligence and ethics: a comprehensive review of bias mitigation, transparency, and accountability in AI Systems." Preprint, November 10 (2023).

[39] Cheatham, Benjamin, Kia Javanmardian, and Hamid Samandari. "Confronting the risks of artificial intelligence." McKinsey Quarterly 2.38 (2019): 1-9.

[40] Leslie, David. "Understanding artificial intelligence ethics and safety." arXiv preprint arXiv:1906.05684 (2019).

[41] Cheng, Lu, Kush R. Varshney, and Huan Liu. "Socially responsible ai algorithms: Issues, purposes, and challenges." Journal of Artificial Intelligence Research 71 (2021): 1137-1181.

[42] Reed, Chris. "How should we regulate artificial intelligence?." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 376.2128 (2018): 20170360.

[43] Zhang, Zidong, Dongxia Zhang, and Robert C. Qiu. "Deep reinforcement learning for power system applications: An overview." CSEE Journal of Power and Energy Systems 6.1 (2019): 213-225.

[44] Oh, Junhyuk, et al. "Discovering reinforcement learning algorithms." Advances in Neural Information Processing Systems 33 (2020): 1060-1070.

[45] Wang, Jane X., Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. "Learning to reinforcement learn." arXiv preprint arXiv:1611.05763 (2016).

[46] MathWorks, Deep Deterministic Policy Gradient (DDPG) Agents homepage, https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html.

[47] Hassija, Vikas, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir

Hussain. "Interpreting black-box models: a review on explainable artificial intelligence." Cognitive Computation 16, no. 1 (2024): 45-74.

[48] David Abels, Abels, and Annes, PC Personal injury lawyers homepage, https://www.daveabels.com/rear-end-collisions-frequent-type-collision/#:~:text=In%20fact%2C%20roughly%201.7%20million,are%20injured%20in%20the%20crashes.

[49] Sebi, Nashwan, Narendra Kintali, and Ka C. Cheok. "Rear-end Vehicle Collision Avoidance using Reinforcement Learning." Proceedings of 39th International Confer 98 (2024): 36-45.

[50] Lagoudakis, Michail G., and Ronald Parr. "Reinforcement learning as classification: Leveraging modern classifiers." Proceedings of the 20th International Conference on Machine Learning (ICML-03). 2003.

[51] J. Jia and W. Wang, "Review of reinforcement learning research," 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Zhanjiang, China, 2020, pp. 186-191, doi: 10.1109/YAC51587.2020.9337653.

[52] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

[53] Busoniu, Lucian, Robert Babuska, and Bart De Schutter. "A comprehensive survey of multiagent reinforcement learning." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38.2 (2008): 156-172.

[54] https://www.interviewkickstart.com/blogs/learn/reinforcement-learning-teaching-machines-optimal-decisions#:~:text=Reinforcement%20Machine%20Learning%20is%20the,behavior%20in%20different%20specific%20situations.

[55] Pecka, Martin, and Tomas Svoboda. "Safe exploration techniques for reinforcement learning–an overview." Modelling and Simulation for Autonomous Systems: First International Workshop, MESAS 2014, Rome, Italy, May 5-6, 2014, Revised Selected Papers 1. Springer International Publishing, 2014.

[56] M. Everett, Y. F. Chen and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 3052-3059, doi: 10.1109/IROS.2018.8593871.

[57] Younes, Younes & Barczyk, Martin. (2022). Adaptive Nonlinear Model Predictive Horizon Using Deep Reinforcement Learning for Optimal Trajectory Planning. Drones. 6. 323. 10.3390/drones6110323.

[58] Deterministic vs. Stochastic Policies in Reinforcement Learning | Baeldung on Computer Science.

[59] https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html.

[60] chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.wnins.com/print/AvoidingRearEndCollisions.pdf.

[61] D. H. Good, K. Krutilla, S. Chien, L. Li, and Y. Chen, "Analysis of potential co-benefits for bicyclist crash imminent braking systems," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 2017, pp. 1-6, doi: 10.1109/ITSC.2017.8317877.

[62] Vahidi, Ardalan, and Azim Eskandarian. "Research advances in intelligent collision avoidance and adaptive cruise control." IEEE transactions on intelligent transportation systems 4.3 (2003): 143-153.

[63] Nekovee, Maziar, and Jing Bie. "Rear-end collision: Causes and avoidance techniques." Wireless Vehicular Networks for Car Collision Avoidance. New York, NY: Springer New York, 2013. 99-119.

[64] https://www.mitre.org/news-insights/publication/real-world-effectiveness-model-year-2015-2020-advanced-driver-assistance.

[65] chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://newsroom.aaa.com/wp-content/uploads/2024/02/UPDATED-Reverse-AEB-Report-FINAL-2.24-2.pdf.

[66] Malinverno L, Barros V, Ghisoni F, Visonà G, Kern R, Nickel PJ, Ventura BE, Šimić I, Stryeck S, Manni F, Ferri C, Jean-Quartier C, Genga L, Schweikert G, Lovrić M, Rosen-Zvi M. A historical perspective of biomedical explainable AI research.

[67] Defense Advanced Research Project Agency, Explainable Artificial Intelligence (XAI) (Archived) homepage, https://www.darpa.mil/program/explainable-artificial-intelligence.

[68] Jangoan, S., Krishnamoorthy, G., Muthusubramanian, M., & Sharma, K. K. (2024). Demystifying Explainable AI: Understanding, Transparency, and Trust. International Journal For Multidisciplinary Research, 6(2), 1-13.

[69] State Space Model: Know Representation, Basic Concept & How to Find It? (testbook.com), homepage, https://testbook.com/electrical-engineering/state-space-model.

[70] matrices - How to obtain a possible state space representation of this 2nd order transfer function? - Mathematics Stack Exchange.