공학박사 학위논문

자율 주행 차량의 심층강화학습 기반 긴급 차선 변경 경로 최적화

Deep Reinforcement Learning-based Path Optimization for Emergency Lane Change of Autonomous Vehicles

2021 년 8 월

서울대학교 대학원

기계항공공학부

송 준

DEEP REINFORCEMENT LEARNING-BASED PATH OPTIMIZATION FOR EMERGENCY LANE CHANGE OF AUTONOMOUS VEHICLES

DISSERTATION

SUBMITTED TO THE SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING AND THE COMMITTEE ON GRADUATE STUDIES OF SEOUL NATIONAL UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Jun Song

August 2021

자율 주행 차량의 심층강화학습 기반 긴급 차선 변경 경로 최적화

Deep Reinforcement Learning-based Path Optimization for Emergency Lane Change of Autonomous Vehicles

지도교수 강 연 준

이 논문을 공학박사 학위논문으로 제출함

2021 년 4 월

서울대학교 대학원

기계항공공학부

송　　준

송준의 공학박사 학위논문을 인준함

2021 년 6 월

위 원 장 : ____________

부위원장 : ____________

위　　원 : ____________

위　　원 : ____________

위　　원 : ____________

To my parents and my family

In memories of

Prof. Chong-Nam Chu

Abstract

Jun Song

School of Mechanical and Aerospace Engineering

The Graduate School

Seoul National University

The emergency lane change is a risk itself because it is made instantaneously in emergency such as a sudden stop of the vehicle in front in the driving lane. Therefore, the optimization of the lane change trajectory is an essential research area of autonomous vehicle. This research proposes a path optimization for emergency lane change of autonomous vehicles based on deep reinforcement learning. This algorithm is developed with a focus on fast and safe avoidance behavior and lane change in an emergency.

As the first step of algorithm development, a simulation environment was established. IPG CARMAKER was selected for reliable vehicle dynamics simulation and construction of driving scenarios for reinforcement learning. This program is a highly reliable and can analyze the behavior of a vehicle similar to that of a real vehicle. In this research, a simulation was performed using the Hyundai I30-PDe full car model. And as a simulator for DRL and vehicle control, Matlab Simulink which can encompass all of control, measurement, and artificial intelligence was selected. By connecting two simulators, the emergency lane change trajectory is optimized based on DRL.

The vehicle lane change trajectory is modeled as a 3rd order polynomial. The start and end point of the lane change is set and analyzed as a function of the lane change distance for the coefficient of the polynomial. In order to optimize the coefficients. A DRL architecture is constructed. 12 types of driving environment data are used for the observation space. And lane change distance which is a variable of polynomial is selected as the output of action space. Reward space is designed to maximize the learning ability. Dynamic & static reward and penalty are given at each time step of simulation, so that optimization proceeds in a direction in which the accumulated rewards could be maximized. Deep Deterministic Policy Gradient agent is used as an algorithm for optimization.

An algorithm is developed for driving a vehicle in a dynamic simulation program. First, an algorithm is developed that can determine when, at what velocity, and in which direction to change the lane of a vehicle in an emergency situation. By estimating the maximum tire-road friction coefficient in real-time, the minimum distance for the driving vehicle to stop is calculated to determine the risk of longitudinal collision with the vehicle in front. Also, using Gipps' safety distance formula, an algorithm is developed that detects the possibility of a collision with a vehicle coming from the lane to be changed, and determines whether to overtake the vehicle to pass forward or to go backward after as being overtaken. Based on this, the decision-making algorithm for the final lane change is developed by determine the collision risk and safety of the left and right lanes.

With the developed algorithm that outputs the emergency lane change trajectory through the configured reinforcement learning structure and the general driving trajectory such as the lane keeping algorithm and the adaptive cruise control algorithm according to the situation, an integrated algorithm that drives the ego vehicle through the adaptive model predictive controller is developed.

As the last step of the research, DRL was performed to optimize the developed emergency lane change path optimization algorithm. 60,000 trial-and-error learning is performed to develop the algorithm for each driving situation, and performance is evaluated through test driving.

Keyword: Deep Reinforcement Learning (DRL), Neural Network, Autonomous Vehicle, Artificial Intelligence, Emergency Lane Change, Evasive Steering, Collision Avoidance, Trajectory Planning, Vehicle Control

Student Number: 2012-23168

Contents

List of Figures

List of Tables

List of Symbols

π	=	Policy of DRL agent
a	=	An action of policy
s	=	A state of environment
r	=	Instant reward value
S	=	Set of states
A	=	Set of action
T	=	Transition probability function
R	=	Rewards function
γ	=	Discount factor of DRL agent
t	=	Simulation time step

μ	=	Actor network of DDPG agent
Q	=	Critic network of DDPG agent
α	=	Learning rate
l	=	Lane width
x_f	=	Longitudinal lane change distance
C_{ego}	=	Ego vehicle object
C_1	=	Vehicle object in front of the ego vehicle
C_2	=	Vehicle object in the rear left of the ego vehicle
C_3	=	Vehicle object in the rear right of the ego vehicle
$p_{r,i,x}$	=	Relative longitudinal distance between the ego vehicle and vehicle object, C_i

$v_{r,i,x}$	=	Relative longitudinal velocity between the ego vehicle and vehicle object, C_i in inertial frame.
ρ	=	Yaw angle in inertial frame
$\dot{\rho}$	=	Yaw rate in vehicle frame
v_{ego}	=	Ego vehicle velocity in vehicle frame
d_{brake}	=	Minimum braking distance
c_l	=	Lane change starting index
r_1	=	Reward according to ego vehicle longitudinal position
r_c	=	Trigger function of r_1
ω_1	=	Weight factor for r_1
i_c	=	Lane change direction index

sr_1	=	Static reward according to lane change complete
sr_c	=	Trigger function of sr_1
x	=	Center of mass longitudinal position of ego vehicle in inertial frame
y	=	Center of mass lateral position of ego vehicle in inertial frame
$V_{ex} = \dot{x}$	=	Center of mass longitudinal velocity of ego vehicle in inertial frame
p_1	=	Penalty according to ego vehicle later deviation
y_t	=	Reference lateral position of current state
ω_3	=	Weight factor for p_1
p_2	=	Penalty according to ego vehicle yaw rate
ω_4	=	Weight factor for p_2

sp_1	=	Static penalty according to distance between ego vehicle and surrounding vehicles
sp_c	=	Trigger function for sp_1
$d_{l,i}$	=	Relative distance between the ego vehicle and the vehicle object, C_i
ω_5	=	Weight factor for sp_1
sp_2	=	Static penalty according to ego vehicle velocity
sp_{c2}	=	Trigger function for sp_2
ω_6	=	Weight factor for sp_2
R_t	=	Summation of rewards
$\mathbf{x(t)}$	=	States of Kalman filter
$\mathbf{y(t)}$	=	Measurements of Kalman filter
F_x	=	Longitudinal tire force

F_y	=	Lateral tire force
ω_h	=	Wheel velocity
a_x	=	Longitudinal acceleration in vehicle frame
a_y	=	Lateral acceleration in vehicle frame
ϵ	=	Maximum friction coefficient
σ_x	=	Wheel slip ratio
K	=	Slip-slope
φ	=	System output of parameter identification form
$e(t)$	=	Identification error
m	=	Ego vehicle mass
μ	=	Friction coefficient of each tire
C_{LO}	=	Longitudinal collision detection index function
V_{tx}	=	Target vehicle longitudinal velocity

a_m	=	Maximum acceleration
C_{LA}	=	Lateral collision detection index function

Chapter 1 Introduction

1.1. Research Background

The quality of vehicle technology is greatly improved according to continuous research and development in the automotive industry. However, due to the many complex factors in surroundings, traffic accidents are still considered a major problem causing death. In 2015, the World Health Organization revealed that road traffic crashes resulted in more than 1.2 million deaths each year, making road traffic accidents the leading global cause of death. According to the United States (US) Department of Transportation's National Highway Traffic Safety Administration, 37,462 people were killed in crashes on US road in 2016, 5.6% more from the 35,485 in 2015. In addition, the latest statistics from China's Ministry of Transportation found that about 63,194 people died in crashes in 2019 (China Bureau of Statistics, 2019). This shows a slight reduction, but still very serious. Therefore, governments, institutes and manufacturers are paying great attention to vehicle safety. A reliable study found that 70% of accidents were the fault of the driver and most traffic accidents were avoidable [1]. One solution to decrease mortality is intelligent transportation systems and self-driving vehicles. In 2013, the US government proposed a pilotless automobile plan in an effort to reduce fatalities [2]. Automated vehicle technology can significantly improve traffic safety, reduce traffic congestion, and has attracted a lot of attention in recent years [3] [4]. Some automated vehicles such as Google Car, nuYonomy and Apple Car are already being tested on the highway. However, existing tests for automated vehicles met some safety problems in driving in a real traffic environment due to the complexity of the transportation system. In recent autonomous-driving car tests, a series of traffic accidents have occurred. and one of the important causes of these accidents is that the control algorithms built into the autonomous-driving car have failed in the face of dynamic changes in the real traffic environment.

To solve this problem, autonomous vehicles based on artificial intelligence (AI) are being widely studied in recent years [5] [6]. Collision avoidance is a key safety factor for autonomous vehicles. Lane change maneuvers are generally more desirable as they interfere with traffic flow to a minimum, compared to slowing down to a complete stop without hitting an obstacle [7]. There have been a few researches in lane change of vehicles such as

A lot of research has recently been performed on automated lane change of vehicles. Particularly, end-to-end, a new paradigm involving cognition and decision-making, has been proposed [8]. The existing method for autonomous vehicle is composed of the recognition of the environment according to sensors implemented on vehicles, decision making according to recognized environment and path planning to create future trajectory, and then follow a given path through control. However, end-to-end learning is based on machine learning, integrating recognition, decision making, and planning, and generating control inputs [8].

Deep reinforcement learning (DRL) is a combination of deep learning and reinforcement learning. Reinforcement learning learns how to maximize numerical rewards by relating situations and actions [9]. DRL provides a high level of optimization by combining the advantages of reinforcement learning and deep learning such as function approximation and expression learning properties[10]. In addition, DRL allows vehicles to learn from their actions instead of labeled data [11]. Labeled data is not required where vehicle agents take action and are rewarded.

In emergency that could cause an accident, vehicle control algorithms must immediately provide feasible maneuvers such as trajectories or paths. So real-time-based algorithms are needed. The standardization organization validates the performance of vehicles and algorithms through several test cases described previously and tests them on a vehicle test site [12]. One is the Double Lane Change Test (DLC) as defined in ISO-3888-2 [13]. DLC test also known as the “Moose test” aims at how well the target vehicle can avoid the obstacle that appears suddenly. The DLC test is described as follows. The target vehicle starts to drive at an start lane which has 12m length. And the test is finished when the vehicle exits ends of an exit lane which has the same size of start lane. The side lane which is between the start lane and the exit lane has an offset of 1 m. The longitudinal distance between the start position of the start lane and side lanes is 13.5m, and the length between the side and the end of the exit lane is 12.5m. This test requires throttle to be releases 2 meters after entering. And the rest of the tests are performed using only the steering without throttle or brake actuation. Testing is generally conducted with or without Electronic Stability Control (ESC) system. However, the DLC test also has limitations. This is because evasive steering in an actual driving environment cannot be the same as a standardized test environment. In the case of the sudden appearance of an obstacle, i.e. sudden stop of the vehicle in front in the actual driving environment, it is necessary to consider not only the distance to the obstacle, but also the vehicle in the lanes on both sides. Therefore, in order to implement a stable avoidance behavior, tests must be conducted by simulating the actual driving situation. For stable evasive steering, it is necessary to optimize the lane change trajectory based on driving environment.

In this study, trajectory optimization for emergency lane change based on DRL in actual driving conditions with a highly detailed vehicle dynamics is performed. In a general driving situation, when the vehicle in front of ego vehicle suddenly stops, the tire-road friction coefficient is estimated in real time to determine whether it can stop through braking. When it is not possible to stop, the ego vehicle decides whether to change lanes to the left or to the right, and decides whether to overtake or to be overtaken. After the decision making, the ego vehicle changes lanes along the optimized trajectory based on driving environment, i, e. both sides vehicles, vehicle in front, ensuring collision avoidance and vehicle controllability.

1.2. Previous Research

In order to solve issues related to lane change control of autonomous vehicles, many researchers have studied about trajectory planning. The methodology of the researches can be classified like follows: Geometric methods, empirical methods, and optimal nonlinear control algorithms. Finally, machine learning-based solutions are emerging in recent years. The geometric method uses curve fitting according to the input of a vehicle state and obstacles information and creates a path with a combination of straight line, arcs or splines [14]. Although these methods can compute optimum values fast, the dynamic viability of the generated trajectories is not guaranteed. Therefore, it must be evaluated later [15].

Some AI-based optimization can be found among the empirical approaches that use searching or random sampling. On the contrary to geometric approaches, these approaches need heavy calculation process. To reduce complexity, Ferguson et al. used sample time-based purification and Likhachev et al. used adaptive purification for the same purpose [16] [17]. However, because of a discrete solution of these method, the Hybrid A* algorithm is used to connect a continuous state to the discrete solution. The nonlinear optimization-based approaches define the problem as a nonlinear optimization problem (NLP) to ensure dynamic feasibility, where the technique is usually based on a geometric method to create a trajectory and establish a value function [18] [19]. Dynamic feasibility has trade off relation with the computation speed. So, These methods create a limited minimization problem [20].

Machine learning approaches can reduce these trade-offs problem. Supervised learning is one of the solution approaches. The nonlinear optimization problem solver can generate valid trajectories for various simulation conditions and use them as data set for a neural network training to generalize the problem and generate path in real time [21]. However, it is not always possible to generate enough data sets for training. RL-based technique uses trial and error method as a solution to the problem mentioned above. And the reinforcement learning agents are trained according to experimental data set obtained by a large number of trials [9] [22]. End to end method is usually used in reinforcement learning approaches that makes action space composed of steering and throttling commands according to the states which means environments. These studies need sensors to acquire environmental information, such as grid topology [23], lidar sensor [24], camera [11] or ground truth [25]. Some research group pay attention to decision-making, like the determination of action of the agent, such as lane-change, lane-keeping, cruise control, etc. Microscopic simulation is used in theses group [26]. Though hybrid solutions combination of decision making and end-to-end control exist, there are only a few researches dealing with how to define a trajectories through a geometrical approach through RL [27] [28].

In recent years, a few researches focus directly on the lane-change and evasive maneuvers using RL. Theses can be classified into four group: evasive steering path optimization, decision making for lane change, collision avoidance and real-time path planning. Feher et al. performed RL-based evasive path optimization [29]. They use standardized tests which is ISO-3888-2 to gain optimized trajectory. However, it is difficult to apply to the actual driving environment because they have optimized the trajectory only in the standardized test. An et al. used RL to decide lane change situation. It is hard to respond to emergency situations because they perform stable lane change of a pre-determined trajectory in a normal driving condition [30]. Duan et al. developed the hierarchical RL-based lane change decision making algorithm. However, it is also a normal driving situation algorithm. And they focused on time spent from starting point to target point [31]. Kim et al. developed RL-based vehicle control algorithm to prevent collision with traffic violations at intersections. But they didn't focus on trajectory optimization but collision avoidance. And they performed simulation on low-speed conditions [32]. Zhang et al. performed path optimization using RL of flying robot. They treated the robot as a geometric mass point. So, it can't be applied to real environment because optimization without dynamics.

It is shown that trend of recent research about RL-based autonomous vehicle driving are simulation to reach the target point in normal driving situations, for algorithm development under a vehicle simulator, and to develop controller of steering wheel. However, there are a few limitations: Few researches on algorithm development using event driven strategy, performing only standardized tests or aiming at avoidance itself, and using a simplified vehicle dynamics model.

1.3. Research Objective

In this study, four kinds of main objective are achieved. The objective of the first part is the construction of the simulation environment. In order to apply the results of this study to the actual road environment, it is important to implement a precise simulation environment. The simulation environment was implemented using IPG CARMAKER and Matlab Simulink. By using the Hyundai I30-PDE full car model and the MF 5.2 tire model, accurate and reliable vehicle dynamics environment was built. In addition, an emergency scenario which is the same environment as the actual emergency situation is implemented to simulator. The scenario consists of four steps: normal driving situation, emergency situation occurrence, decision making, and lane change. In the scenario, the tire-road friction coefficient is reflected in order to make relation between vehicle and road.

The second objective, the principal section of this study, is the development of the deep reinforcement learning (DRL) structure. In case of emergency, for rapid and stable lane change, this study optimizes the lane change trajectory through DRL. For the optimization, the structure of observations, action, and rewards must be constructed robustly. The agent of DRL performs actions in the direction of maximizing rewards based on observations. Based on appropriate observations and rewards, DRL structure is constructed to perform an optimized trajectory action. In this study, the agent is constructed using a neural network and trained using the deep deterministic policy gradient (DDPG) algorithm.

The third object is the development of an integrated control algorithm for vehicle control. The vehicle integrates three algorithms to drive suitable for driving conditions: decision making, path planning, and vehicle controller. In the decision-making algorithm, it is determined whether to change lanes by detecting a collision with the vehicle in front while normal driving. It also determines whether to overtake or be overtaken by detecting collisions with the left and right vehicles. Then, the direction of lane change to ensure stability is determined. The path planning algorithm is composed of DRL agents and provides the optimal lane change trajectory at the moment of decision making. Finally, the vehicle is controlled with and optimized path through adaptive model predictive controller (AMPC). By integrating the three algorithms, the vehicle drives in a developed scenario.

The final object is the training of DRL agent and the evaluation of DRL agent results. Based on the established simulation environment, vehicle model, and DRL agent, the agent is trained 60,000 times. The agent performs optimization in the direction of maximizing the reward through the trial & error method. In addition, the optimized agent is used to evaluate the optimized trajectories according to the driving situation, driving environment, and also evaluate the vehicle states.

This study proposes a new algorithm for lane change control of autonomous vehicles to avoid collision and ensure safety. The proposed method in this study can be improvement compared to previous researches.

- The simulation environment constructed in this study can provide accurate vehicle dynamics model and road environment the same as actual driving condition. The environment includes Hyundai I30-PDE full car model and MF tire model which is the most detailed vehicle model and road condition which is applying tire-road friction coefficient. Most papers used simplified vehicle dynamics and didn't consider tire-road friction coefficient. In this study, tire-road friction coefficient was considered in detail using slip-slope method.

- Deep reinforcement learning structure in this study can provide suitable architecture for autonomous vehicles. Because observation and reward don't include specific vehicle parameter such as unique characteristics of certain vehicle, DRL structure can be applied to various vehicles.

- The proposed algorithm enables autonomous vehicles to drive and evade collision safely. Most existing researches provided algorithm for normal driving condition lane change or algorithm focused on collision avoidance itself. But this algorithm cannot only be used in normal driving condition but also in emergency condition. And it can provide stable controllability after collision avoidance.

1.4. Dissertation Overview

The reminder of this dissertation is organized as follows. Specially, study outline from Chapter 2 to Chapter 7 is summarized in Figure 1.1.

- ***Chapter 2: Simulation Environments*** presents various states to be analyzed through this study. First, detailed vehicle dynamics was established. Hyundai I30-PDE full car model is a vehicle parameter model for vehicle dynamics simulator IPG CARMAKER. IPG CARMAKER provides the most detailed vehicle dynamics so that it can be the best simulation environment in this study. IPG CARMAKER provides 7 types of vehicle dynamics: vehicle body modelling, suspension kinematics, aerodynamics, steering system, powertrain, brake, tire. The vehicle body is a multi-body system which is characterized through different bodies. The motion of the multi body system is described with differential and with algebraic equations. The principle of d'Alembert is applied to get differential equations of motion for the generalized coordinates of the system. Kinematics of suspension which comprises spring, damper, buffer, and stabilizer describes the special movements of a wheel due to compression and steer action. Aerodynamics in this study is ignored. Steering system is modelled according to 2nd order differential equations. Brake system consider brake type and brake pressure. Powertrain system is not considered in this research. And tire is modelled using Magic Formula (MF) 5.2 ver.

Road environment is also established on IPG CARMAKER. The road condition contains friction coefficient likewise actual driving road so that tire-road friction coefficient can be estimated then minimum braking distance of ego vehicle can be estimated according to tire-road friction coefficient.

- ***Chapter 3: Fundamentals*** presents about deep neural network, reinforcement learning, and deep reinforcement learning.

  Because main topic of this study is DRL-base optimization, chapter 3 shows that the RL algorithm used in this study and theoretical base of DRL. And because DRL contains Deep neural network, Layers of neural network used in this study is shown and also the theoretical base of neural network.

- ***Chapter 4: DRL-enhanced lane change*** presents necessity of trajectory optimization and type of lane change trajectory. Lane change trajectory can be simplified into 3rd order polynomial shape. And then the coefficients of the polynomials will be the action of the DRL agent. For optimization training of DRL agent, DRL structure is built. Observation presents real-time states of environment such as ego vehicle velocity, relative distance between ego vehicle and target vehicle, etc. It is used as characteristic properties that is ground for agent actions. Reward is target properties that need to be maximized. DRL training is progressed to maximize reward according to observations. Neural network architecture is the structure of the DRL agent. For the detailed feature extraction of observation, neural network is implemented. And action is the output of the agent. The goal of this study is optimization of 3rd order polynomial shaped lane change trajectory, agent outputs the coefficients as action. And action values will be optimized through training in Chapter. 6.

- ***Chapter 5: Autonomous Driving Algorithm Integration*** presents hierarchical algorithm that is needed for driving in simulation environment. Longitudinal collision detection algorithm estimates the tire-road friction coefficient in real time. Using this, the minimum braking distance of the ego vehicle is estimated. If the distance to the vehicle in front is less than the minimum braking distance, a collision will occur, and thus a collision is determined based on this algorithm. Lateral collision detection algorithm detects the state of the vehicle following the left and right lanes. When the lane needs to be changed through the longitudinal collision detection algorithm, Gipps' safety distance is used to detect whether a collision with a vehicle on the rear side is encountered when the lane is changed in the current vehicle velocity. If there is no collision, make a lane change toward the larger safety distance side. In the warning of a collision, the vehicle speed is lowered with maximum braking, and the vehicle is overtaken by the vehicle in the rear side and then the lane change is performed. Lane change direction decision algorithm determines which direction to change lanes. When overtaking and overtaken are determined through the later collision detection algorithm, the lane is changed through comparison of left and right value scale.

- ***Chapter 6: Training & Results*** presents training results of deep reinforcement learning agent. Training was performed 60,000 times and it shows feasible results. And various properties according to operating conditions are evaluated.

- ***Chapter 7: Conclusion*** shows a summary of the study.

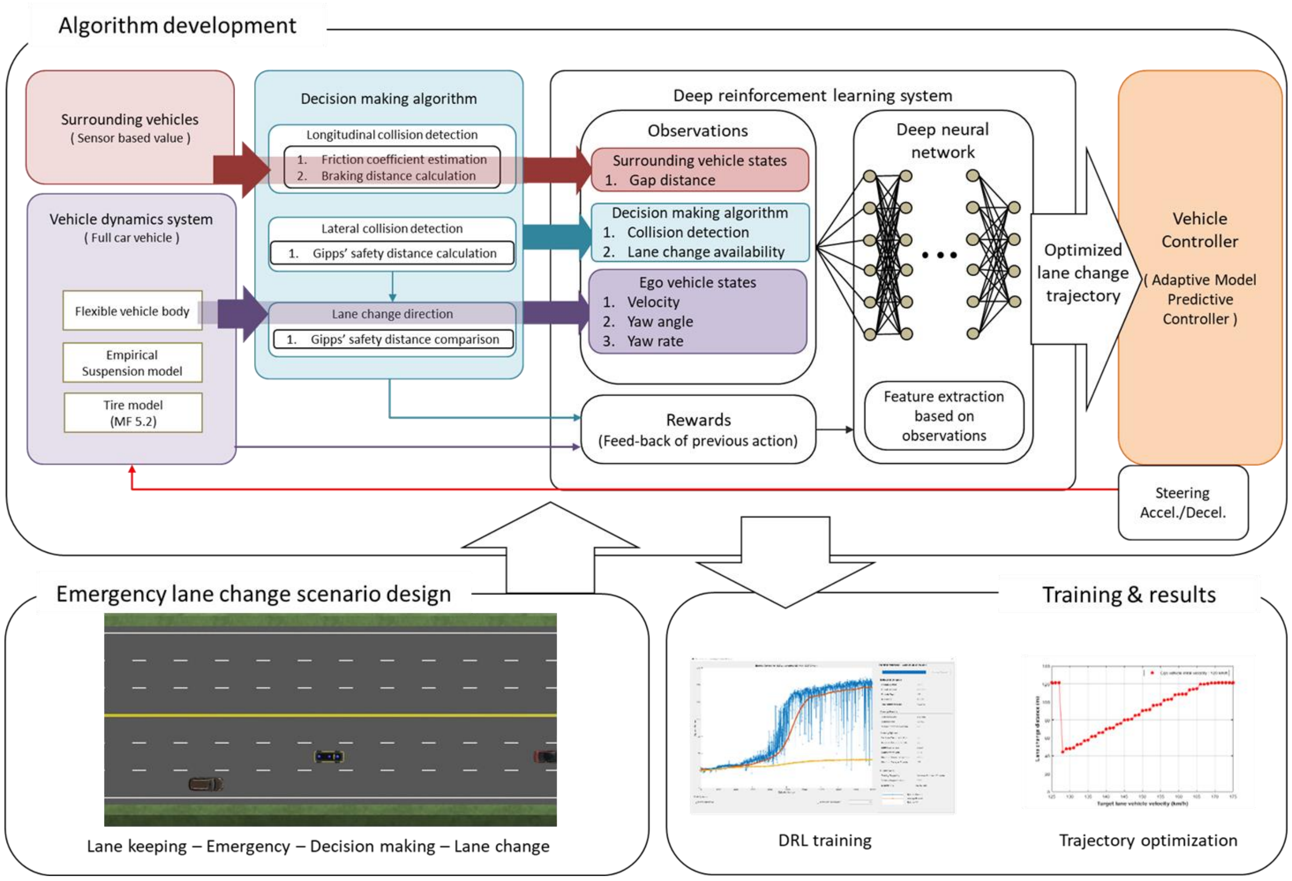

Algorithm development
Surrounding vehicles (Sensor based value)
Decision making algorithm
Longitudinal collision detection
1. Friction coefficient estimation
2. Braking distance calculation
Vehicle dynamics system (Full car vehicle)
Lateral collision detection
1. Gipps' safety distance calculation
Flexible vehicle body
Lane change direction
1. Gipps' safety distance comparison
Empirical Suspension model
Tire model (MF 5.2)
Deep reinforcement learning system
Observations
Surrounding vehicle states
1. Gap distance
Decision making algorithm
1. Collision detection
2. Lane change availability
Ego vehicle states
1. Velocity
2. Yaw angle
3. Yaw rate
Deep neural network
Feature extraction based on observations
Rewards (Feed-back of previous action)
Optimized lane change trajectory
Vehicle Controller
(Adaptive Model Predictive Controller)
Steering Accel./Decel.
Emergency lane change scenario design
Lane keeping – Emergency – Decision making – Lane change
Training & results
DRL training
Trajectory optimization

Figure 1.1 Outline for overall study

Chapter 2
Simulation Environment

2.1. Simulator

As the first process of this study, a simulator is selected. This study is conducted through simulation based on a virtual model, but the goal is to apply the algorithm to actual vehicles, so a simulation environment requires very detailed model similar to the actual environment. In addition, programs that can use appropriate algorithms for reinforcement learning training are needed. Based on these properties, IP CARMAKER and Matlab Simulink were selected.

First, IPG CARMAKER is a multi-body vehicle dynamics simulator, and CARMAKER models realistically and precisely simulate a wide variety of vehicle types along with their handling characteristics, the road and the surrounding environment, driver behavior and the traffic situation in the virtual world. It provides the widest variety of pre-defined models for vehicles, as well as a data set generator for quickly defining new models with a high degree of precision and realistic behavior to the vehicle dynamics. It implements multibody system efficiently – non-linear, expandable and real-time capable. And it provides realistic generation of roads or specific test tracks.

Table 2.1 The description of vehicle components

Body	Parts of the body
Vehicle’s body	All sprung masses beside engine, trimloads
Trimloads	Constant loads
Wheel suspension -front left -front right -rear left -rear right	All unsprung masses without the wheel, like link, wheel carrier, suspension leg, wishbone mount…
Wheel	All rotating masses, like tire, rim, bearing, brake disc…
External and internal forces/torques and constraints	
Suspension Force elements	
Aerodynamics	
Kinematics and Compliance	
Tire forces/torques	

In this study, Hyundai I30-Pde full car model was used as a pre-defined vehicle model. Vehicle dynamics model is implemented based on the parameters of the full car model.

The simulated vehicle is a multi-body system which is characterized through different bodies. They are generated and optimized with MESA VERDE [33]. Following Table 2.1 is the description of vehicle components

Figure 2.1 shows the vehicle dynamics model which are involved in the vehicle model calculation. The dynamics model involves highly detailed systems which contain empirical kinematics and compliance model. And Figure 2.2 demonstrates the chain of calculation of the vehicle model. Detailed vehicle dynamics equations are described in the reference manual of IPG CARMAKER.

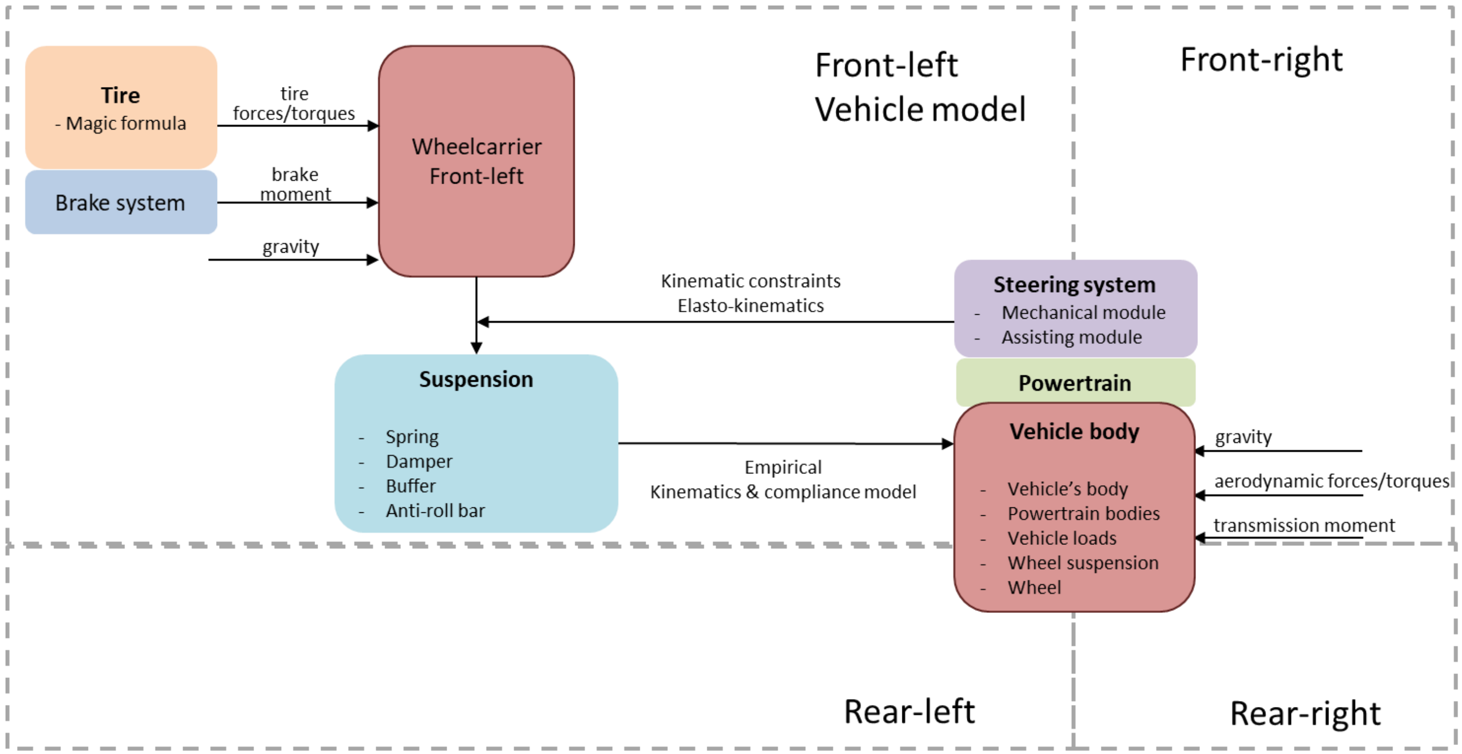


Figure 2.1. Highly detailed vehicle dynamics model

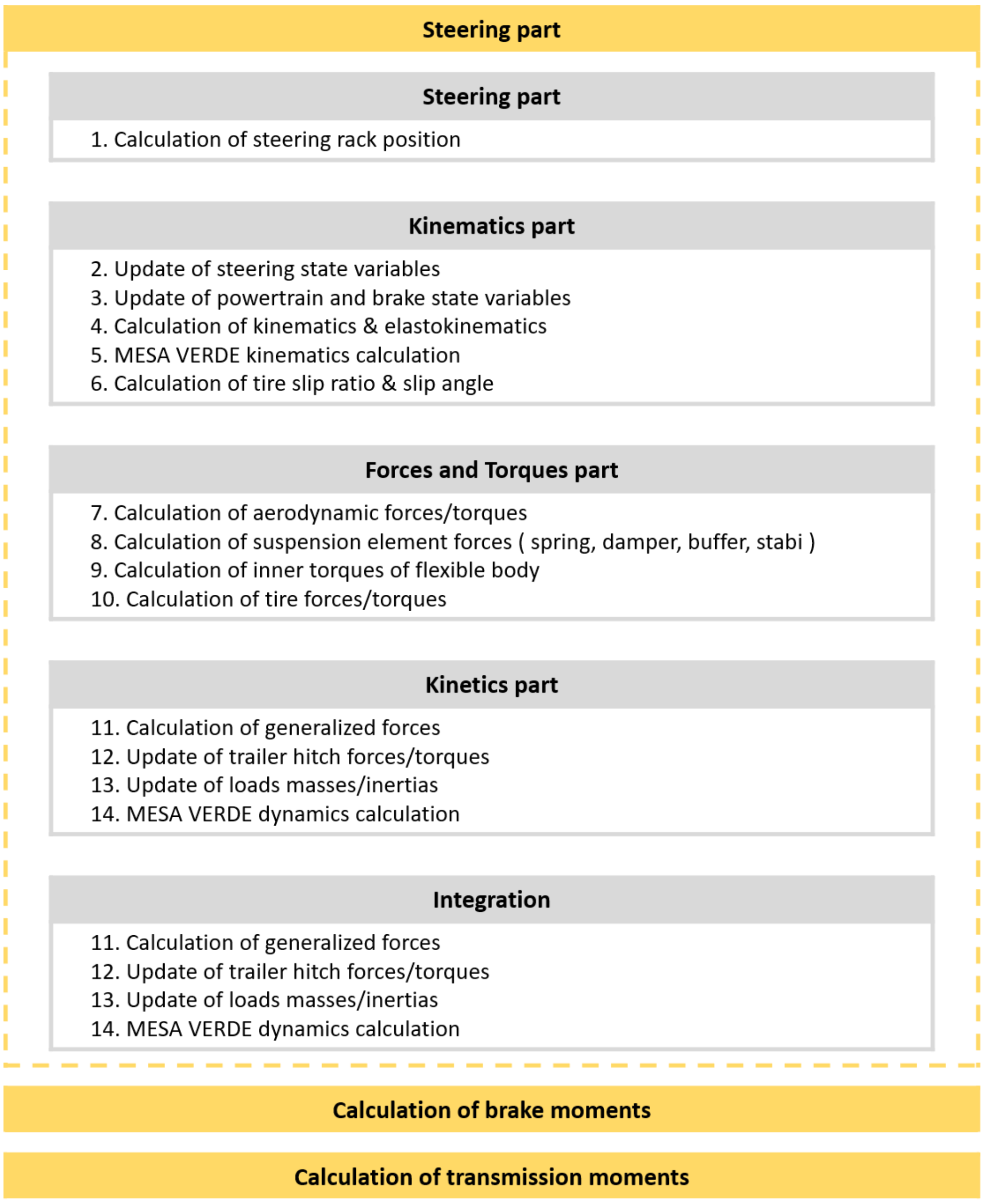

Steering part
Steering part
1. Calculation of steering rack position
Kinematics part
2. Update of steering state variables
3. Update of powertrain and brake state variables
4. Calculation of kinematics & elastokinematics
5. MESA VERDE kinematics calculation
6. Calculation of tire slip ratio & slip angle
Forces and Torques part
7. Calculation of aerodynamic forces/torques
8. Calculation of suspension element forces (spring, damper, buffer, stabi)
9. Calculation of inner torques of flexible body
10. Calculation of tire forces/torques
Kinetics part
11. Calculation of generalized forces
12. Update of trailer hitch forces/torques
13. Update of loads masses/inertias
14. MESA VERDE dynamics calculation
Integration
11. Calculation of generalized forces
12. Update of trailer hitch forces/torques
13. Update of loads masses/inertias
14. MESA VERDE dynamics calculation
Calculation of brake moments
Calculation of transmission moments

Figure 2.2. Chain of vehicle body motion calculation

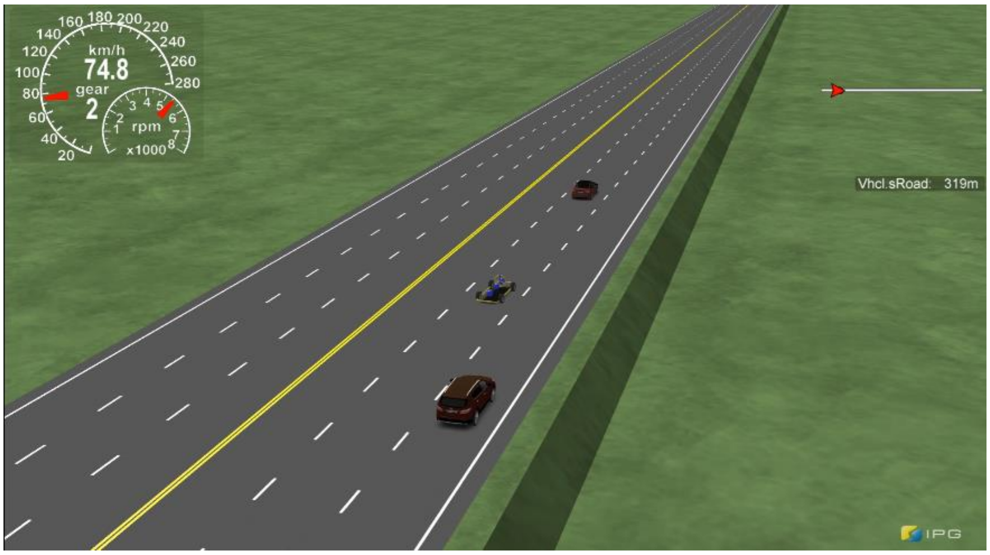


Figure 2.3. CARMAKER driving environment

Through detailed dynamics model as shown in Figure 2.2 above, this study can develop a reliable vehicle control algorithm applicable to real vehicles. In addition, like Figure 2.3, It is possible to make driving environment more similar to actual environment by applying the actual tire-road friction coefficient.

Second, Matlab Simulink is selected to train DRL agent and control vehicle which is defined in IPG CARMAKER. As shown in Figure 2.4, Simulink is a graphical programming environment for simulating and analyzing multidomain dynamical systems. It is widely used in automatic control and digital signal processing. In addition, it has excellent compatibility with other software such as CARMAKER. Most of all, it has powerful tools and functions for reinforcement learning. It provides functions and block for learning policies using RL algorithms including DQN, A2C, and DDPG. This policy can be used to implement algorithms for complex systems such as autonomous systems. Policies can be implemented using deep neural networks, polynomial, or lookup tables. In addition, agent training can be accelerated using parallel computing.

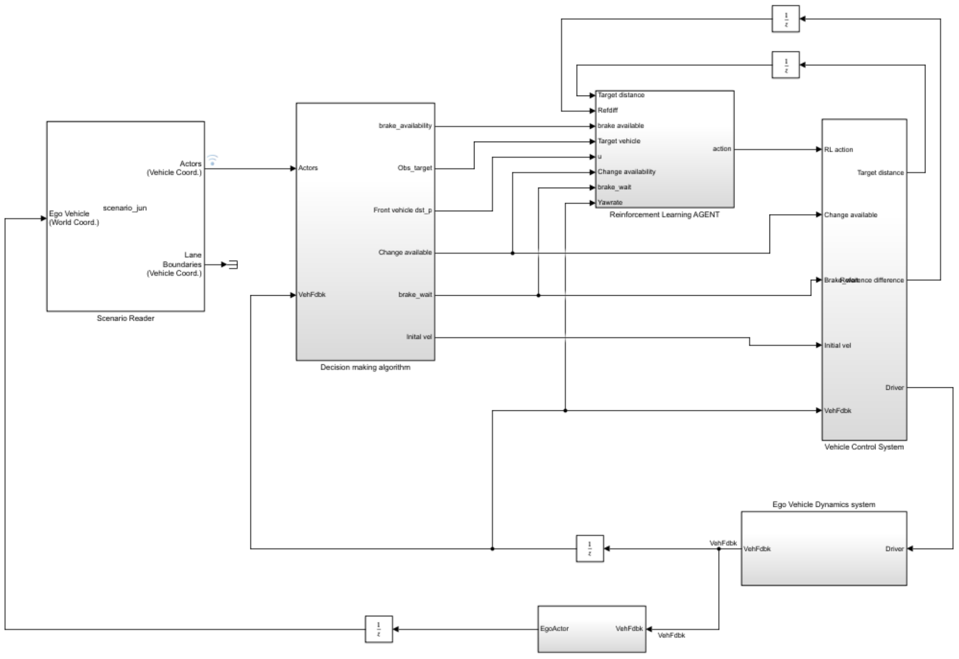


Figure 2.4. Matlab Simulink model

2.2. Scenario

In order to develop an algorithm applicable to the actual environment, it is important not only to select appropriate simulator which uses a detailed vehicle model, but also to make a scenario that imitate the actual driving conditions. This is because optimizing the trajectory through a scenario different from the real environment eventually leads to unexpected results when applying the algorithm to the actual driving condition. In this study, four stages of driving situations were set as s scenario for lane change in emergency situations. As shown in Figure 2.5, in the first situation, the ego vehicle maintains the distance to the vehicle in front, like adaptive cruise control, and drives in normal condition. The second situation represents a situation in which the vehicle in front suddenly stops or an obstacle appears. In this study, the vehicle in front does not stop through braking but stops momentarily in order to express a case in which a sudden obstacle appearance. In addition, since lane change in an unstoppable situation is the goal which should be optimized, lane change is not performed if the distance to the vehicle in front is stoppable. Collision detection is determined through the longitudinal collision detection algorithm, which will be discussed in Chapter 5. The third one is a situation in which, after an emergency, it is decided in which way or in which direction to change lanes. This is determined through the lateral collision detection algorithm and the lane change direction decision algorithm, which will be discussed in Chapter 5. The fourth situation is evasive steering. Evasive steering is performed based on the lane change trajectory modeled as a 3rd order polynomial. The coefficients of the lane change trajectory are optimized through reinforcement learning.

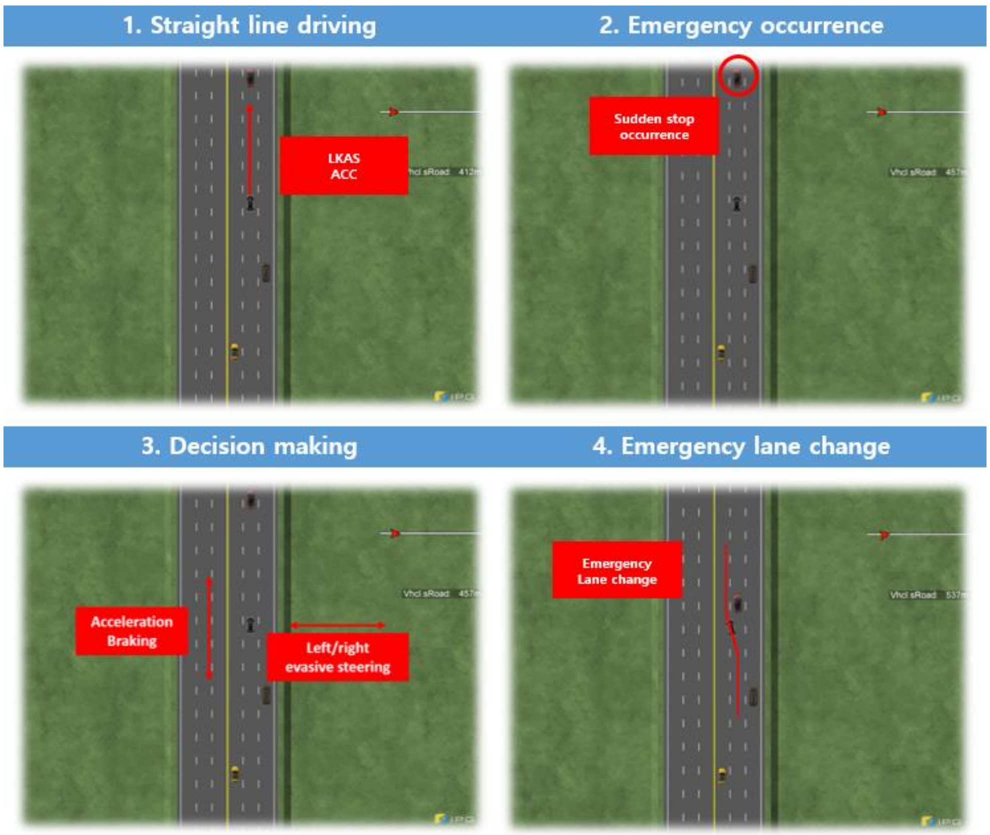


Figure 2.5. Driving scenario

Chapter 3
Methodology

This chapter details the methods used to optimize lane change trajectory. Reinforcement learning process is composed of training and evaluation like most machine learning approaches. This chapter presents the training phase of reinforcement learning of the neural networks.

3.1. Reinforcement learning

The reinforcement learning is composed of environments which means state-space, action-space and reward-space. And the agent of RL system takes solution from the action-space according to environments. And the agent takes action to make the accumulated reward which means summation of reward-space maximize based on state-space which is an input of the agent. During the training phase, agent learn a policy, π, to make the accumulated rewards maximized. The policy defines action, a, to be taken depending on the state, s. Then the state-space is changed to the new state, s', and the reward, r, is returned. RL problems are modeled as a Markov Decision Process (MDP), which is defined as a tuple modeled as a Markov Decision Process (MDP), which is defined as a tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of states, A is the set of actions, $T: S \times A \rightarrow S$ is the state transition probability function, $R: S \times A \times S \rightarrow \mathbb{R}$ is the reward function and $\gamma \in [0,1]$ is the discount factor. MDP meets Markov properties expressed as $\Pr(S_{t+1} = s' \mid S_0, S_1, \cdots, S_{t-1}, S_t) = \Pr(S_{t+1} = s' \mid S_t)$. This means that the probability distribution of future states depends only on the current state and behavior, not the history of the previous state. At every time step, t, the agent's goal is to maximize the future discounted returned, defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{3.1}$$

Where r_{t+k} is the reward given in step $t + k$ [9]. The RL-system needs to be built to optimize the trajectory mentioned in Chapter 1. The system is made in Simulink, and the components are presented in Chapter 5.

3.2. Deep reinforcement learning

In this study, the Deep Reinforcement Learning (DRL) method is selected to optimize lane change trajectory. Recently DRL has been greatly improved and utilized. In [34], a deep neural network is used for function estimation of value-based reinforcement learning. This method is applicable only to tasks with discrete action space. Because the action space of this study is continuous, deep deterministic policy gradient method (DDPG) is selected to solve a continuous control task [35]. Actor-critic network is used in DDPG algorithm. And deep neural networks represent policy $\mu(s|\theta^{\mu})$ and value $Q(s,a|\theta^{Q})$ in it. To solve the learning instability problem due to deep neural networks, replay buffer and the target networks are adopted. The algorithm can be more data-efficient by using replay buffer. Because training samples are distributed independently and equally. The target network makes the parameters change more slowly. The critic network is trained using the following Bellman equation,

$$Q^{\mu}(s_t,a_t) = \mathbb{E}_{r_t,s_{t+1}}{\sim}E[r(s_t,a_t) + \gamma Q^{\mu}(s_{t+1},\mu(s_{t+1})) \tag{3.2}$$

a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) is as Equation 3.3.

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \tag{3.3}$$

The following policy gradient is used for updating the actor.

$$\nabla_{\theta^{\mu} J} = \mathbb{E}_{s_t \sim \rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t|\theta^{\mu})}] \tag{3.4}$$

the sampled policy gradient is calculated as

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{i} \nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_i} \tag{3.5}$$

The actor is updated with policy gradients, the gradients is calculated from the Temporal error. And the critic-network is updated with gradients. The TD error can be calculated by the following Equation 3.6. And it can be used to update the weights of critic-networks.

$$L = \frac{1}{M}\sum_{i}(y_i - Q(s_i, a_i|\theta^Q)^2) \tag{3.6}$$

According to policy gradient, the policy network is also updated. DDPG uses a soft update to make the stability of algorithm greater than other algorithm. In other words, DDPG algorithm adopt a strategy that blend between the regular and target network weights slowly.

Since a DDPG algorithm is a proper DRL algorithm for optimization problem requiring continuous action spaces, this method is adopted to optimize path planning with continuous control action for the vehicle.

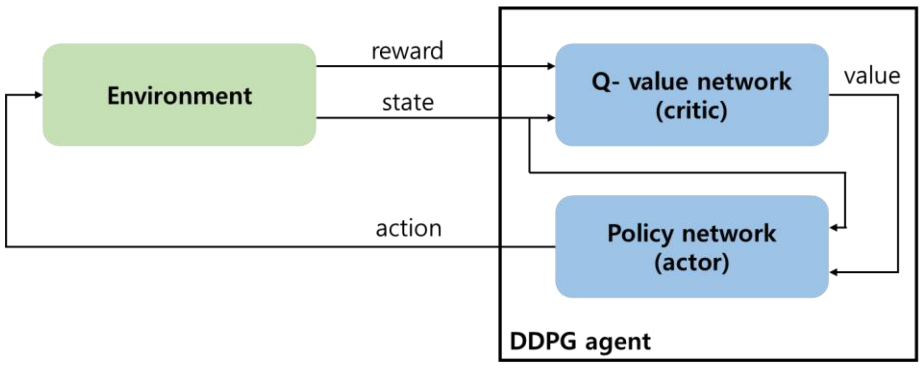


Figure 3.1. Deep Deterministic Policy Gradient (DDPG) structure

3.3. Neural network

The actor and the critic of the DDPG agent mentioned previous subset are consist of two neural network each other. The function of the actor-network is making an optimal parameters of lane change trajectories. And critic-network learns to evaluate the actor to improve the learning process.

The actor-network is composed of 10-element input layers and implemented in Matlab Simulink.

The first layer is feature input layer function,$featureInputLayer(N_f)$, and N_f is the number of observations, Feature input layer is an input layer that inputs feature data to a network and applies data normalization. When a data set of numeric scalars represents features, this layer can be used. Four fully connected hidden layers, $fullyConnectedLayer(L)$, where, L is the number of neurons, follow feature input layer. For hidden layers, batch normalization and rectified linear unit (ReLu) activation function is used. The ReLu layer performs a threshold operation that sets all values less than zero to zero for each element of the input. 9th element of actor network is a hyperbolic tangent activation function, $tanhLayer$. This layer applies the $tanh$ function on the layer inputs. And the last element layer is the scaling layer, $scalingLayer$. This function linearly scales and biases an input array, giving an output $Y = Scale.* U + Bias$. Because $tanhLayer$ gives bounded output that falls between -1 and 1. This layer is useful for scaling and shifting the outputs of nonlinear layers.

The second layer is critic-network. Critic network is composed of two parallel sub layers: state path, action path. State path consists of feature input layer, fully connected layer, ReLu layer, and addition layer. Action path is composed of one feature input layer and one fully connected layer. This path is added in the middle of the state path, to addition layer. Table 3.1 Summarizes the architecture of both networks.

Table 3.1. Parameters of neural networks

Actor network	
Learning rate (α)	0.0001
Mini batch size	64
Structure of fully connected layer	[100 100 100 1]
Critic network	
Learning rate (α)	0.001
Gradient threshold	1
L2 regularization	0.0001
Discount factor	0.99
Head 1 Structure of fully connected layer	[100 100 100 1]
Head 2 Structure of fully connected layer	[100]

Chapter 4
DRL-enhanced Lane change

In this Chapter, the deep reinforcement learning structure is designed for optimizing the lane change trajectory in an emergency situation based on a given driving environment and scenario. First, we describe the lane-changing trajectory as a 3rd order polynomial. And second, the observation, action, and reward space of deep reinforcement learning to obtain the coefficients of polynomial is designed, and the DDPG agent for optimal learning is constructed.

4.1. Necessity of Evasive Steering Trajectory Optimization

Lane change refers to the act of moving a vehicle to the left or right lanes. When vehicle is driving on a straight road, the lane change situation can be divided into two types scenario. The first is changing lanes in order to get to the target point quickly in normal driving situations. This is a sufficiently relaxed situation and enables the driver or an autonomous vehicle to stably change lanes. Research on this type of lane change has been sufficiently conducted since the past and is currently commercialized. The second is changing lanes in case of emergency situations. In particular, when an obstacle appears or the vehicle in front of the ego vehicle suddenly stops and a collision with the vehicle in front cannot be avoid, the ego vehicle performs evasive steering to avoid a collision. Changing to a different lane by evasive steering is the only way to ensure traffic flow and driver safety. However, in this case, unlike a normal driving situation, the driver cannot sufficiently recognize the surrounding information, which may cause another accident. On the other hand, since autonomous vehicles continuously observe the surrounding traffic environment in real time, it is possible to effectively avoid crash by evasive steering by judging risks in real time.

However, if the steering wheel is simply steered in the direction in which the vehicle intends to move, it is not possible to guarantee the stability of the vehicle and passenger's safety. In this study, lane change in an emergency situation is analyzed into two extreme cases. The first is when the autonomous vehicle steers too slow or the steering angle is insufficient. As shown in Figure 4.1, in this case, the vehicle collides with an obstacle or the vehicle in front of the ego vehicle before completing the lane change. Therefore, in order to prevent such an accident, faster evasive steering that means short lane change trajectory is required. Second case shown in Figure 4.2 is occurred when the steering angle is too much or the vehicle steers to fast. In this case, the vehicle can successfully avoid the crash with the obstacle ahead, but if the vehicle's behavior exceeds the limitation of vehicle dynamics, the vehicle loses controllability, resulting in other type of accidents such as spin, roll-over.

Therefore, for effective and stable emergency lane change, an optimized lane change trajectory between the above two aspects of lane change cases is required. By optimizing the lane change trajectory, it is possible to avoid collision with obstacles ahead by changing lanes quickly and other types of accidents can be prevented by securing vehicle controllability through safe lane change.

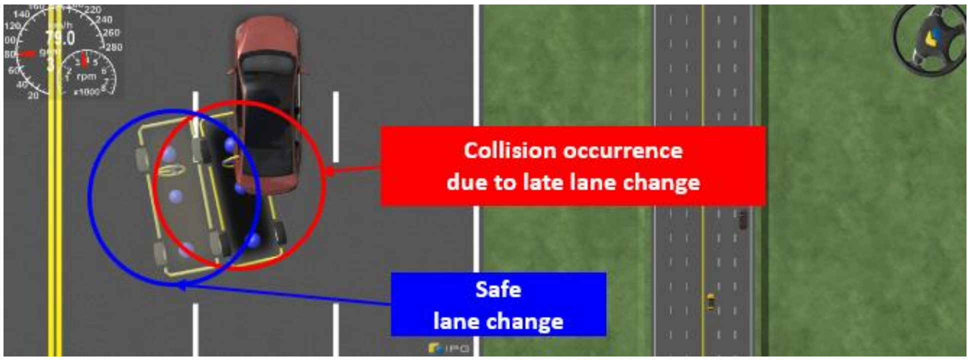


Figure 4.1. Slow lane-change inducing crash with the vehicle in front

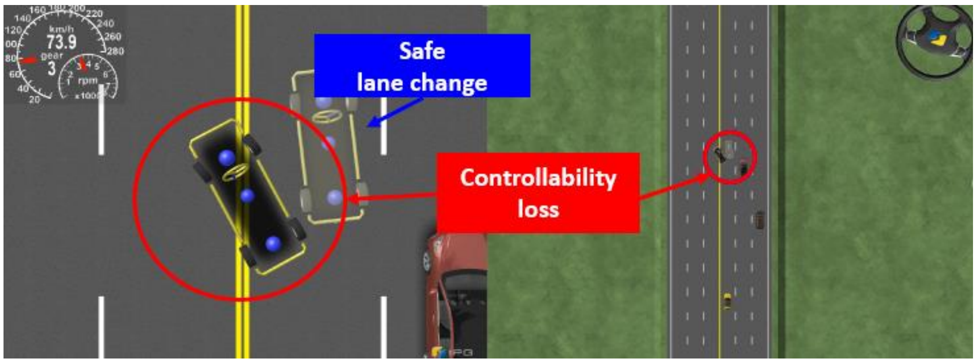


Figure 4.2. Rapid lane-change inducing loss of controllability

4.2. Trajectory Planning

In this study, to describe the lane change trajectory, a 3rd order polynomial lane change trajectory model is selected like following equation.

$$y = f(x) = c_a x^3 + c_b x^2 + c_c x + d_c \tag{4.1}$$

This model has advantages in that it can be expressed with one explicit equation compared to the conventional arc and straight-line method. And it can be calculated more simply than 5th order polynomial model so that optimization cost can be decreased.

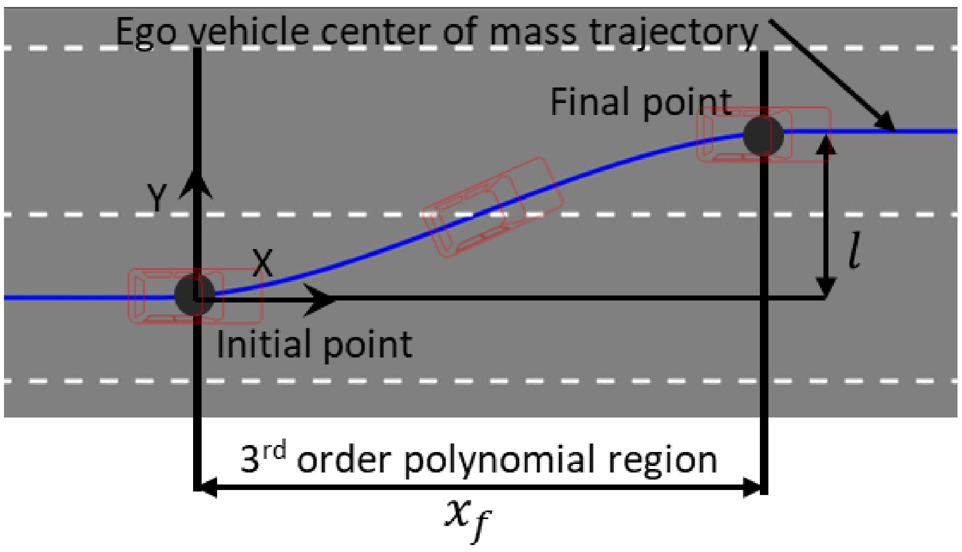

Ego vehicle center of mass trajectory
Final point
Y
X
l
Initial point
3rd order polynomial region
x_f

Figure 4.3. 3rd order polynomial trajectory approximation

As shown in Figure 4.3, assuming that the position of the ego vehicle mass center at the starting point of lane change is (0,0), the boundary conditions at the start and end point of lane change are given as follows.

$$\begin{gathered} f(0) = 0 \\ f'(0) = 0 \\ f\left(x_f\right) = l \\ f'\left(x_f\right) = 0 \end{gathered} \tag{4.2}$$

Modelling an equation based on the boundary conditions, the following equations can be obtained.

$$\begin{gathered} a_c = -\frac{8}{x_f^3} \\ b_c = \frac{12}{x_f^2} \\ c_c = d_c = 0 \end{gathered} \tag{4.3}$$

Finally, the lane change expressed as a 3rd order polynomial is determined according to x_f, which is the distance from the lane change start point to the end point as shown in Figure 4.3. Therefore, in this study, the agent is trained and evaluated to provides optimal x_f according to driving environment in real time through deep reinforcement learning.

4.3. DRL Structure

In this study, as mentioned before, the emergency lane change trajectory described as a 3rd order polynomial is optimized through deep reinforcement learning. It is important to design a robust reinforcement learning structure for stable and fast emergency lane change trajectory optimization.

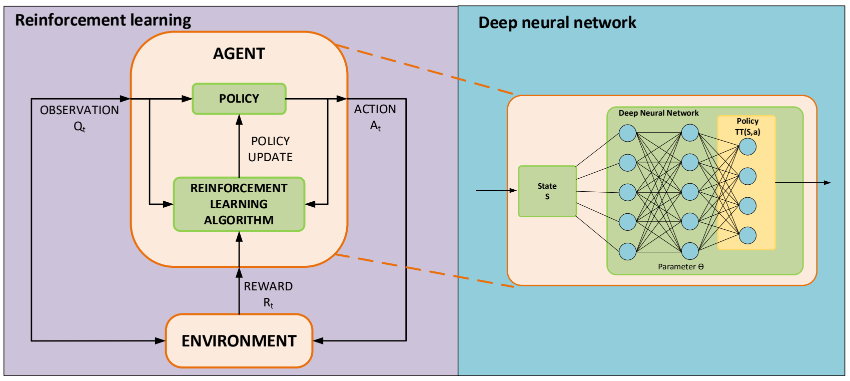


Figure 4.4. Deep reinforcement learning agent architecture

Figure 4.4 shows the deep reinforcement learning agent, environment, and architecture. In this chapter, such an artificial intelligence structure is designed.

4.3.1. Observation(states)

The agent receives no information from the environments because of the model-free training system. It only receives quantified observation variables closely related to the actions. To optimize lane change trajectory, the status variables are to specify the vehicles' relation between front, left side, right side and ego vehicle. Also be to specify the ego vehicles motion parameters.

As mentioned in the scenario in Chapter 2.2, the simulation is performed with a straight line of three lanes. The ego vehicle, C_{ego}, and object 1, C_1,drive on middle lane. And object 2, C_2, and object 3, C_3, are drive on left and right lane respectively. During normal driving situation, C_1 suddenly stops and the ego vehicle decides to change lanes through the relation with object C_1, C_2, C_3. According to these relations, the observation space consists of 12 continuous states.

$$[p_{r,1,x}, v_{r,1,x}, p_{r,2,x}, v_{r,2,x}, p_{r,3,x}, v_{r,3,x}, \psi, \dot{\psi}, v_{ego,x}, d_{brake}, x_f, c_l] \qquad (4.4)$$

$p_{r,1,x}$ and $v_{r,1,x}$ shows the relative longitudinal distance and velocity with C_1 in the scenario. $p_{r,2,x}, v_{r,2,x}, p_{r,3,x}$ and $v_{r,3,x}$ represent the relative longitudinal distance and velocity with C_2 and C_3 respectively. This observation space is selected as a factor to avoid collisions with vehicles during lane change and to decide lane change motion. $\rho, \dot{\rho}$, and $v_{ego,x}$ represent the yaw angle yaw rate, and longitudinal velocity of the ego vehicle respectively. These are selected as a factor affecting the stability of the vehicle during lane change. d_{brake} is the minimum braking distance to stop, which is selected as a factor related to collision avoidance with the C_1. x_f is the action feed back from the previous time stop, and c_l is the index that determines whether changes lane or not, which will be described later in Chapter 5, and is selected as a factor involved in the collision with C_2 and C_3. As mentioned in Chapter 3 above, the observations are normalized to the range [0, 1] based on the experience gained in previous reinforcement learning systems. This is because training takes more time when one variable has a large difference in size from the other variable in a high-dimensional observation.

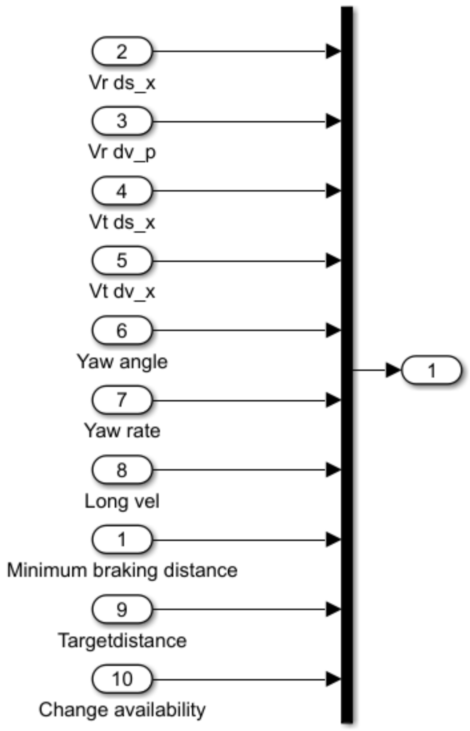


Figure 4.5. Observation-space of Simulink model

Table 4.1. Observation-space of Simulink model

Observation name	Object
Obj#1 relative longitudinal distance Obj#1 relative longitudinal velocity Obj#2&3 relative longitudinal distance Obj#2&3 relative longitudinal velocity	Relative motion parameters - Collision prevention - Decision making
Ego vehicle yaw angle	Ego motion parameters - Vehicle stability
Ego vehicle yaw rate	
Ego vehicle longitudinal velocity	
Minimum braking distance	Collision prevention
Lane change distance (x_f)	Action space feed back
Lane change availability	Decision making

4.3.2. Action

The agent has only one continuous action output, which significantly reduces the complexity of the training task. As mentioned in chapter 4.2, it is possible not by finding all points of lane change trajectories through reinforcement learning, but by simplifying trajectory points to an explicit function 3rd order polynomial. And for more effectiveness, Outputs of the action space, x_f, is limited to between 3 (m) to 100 (m). The action parameters determine the trajectory through which the ego vehicle is going to path. The manners of overtaking or being overtaken is decided through another decision-making algorithm which will be mentioned in chapter 5. The normalized action space [0, 1] can make the better training results than raw action space during the DRL training. So, the actor network uses normalization function and being connected scaling layer at the end of the layer to scale the normalized outputs into a x_f. Figure 4.6 shows the action system of Matlab Simulink block.

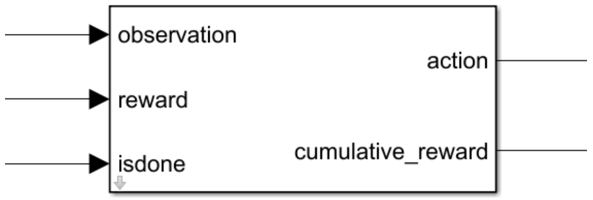


Figure 4.6. Action of Simulink model

Table 4.2. Action of Simulink model

Action	Limit
Lane change distance (x_f)	Min : 3 (m) Max : 100 (m)

4.3.3. Reward

The DRL learning is composed of a series of iterations. It is composed of steps in which the successive trial can be identified by accumulated reward. In an emergency lane change, the reward functions are to quantify how much good generated trajectory is, where the agent responds to a given state which means observation. The reward function has to be built to make the action be optimized.

The goal of the agent in DRL is not the reward value at each time step, but to maximize the accumulated reward during each episode. It is important to design a reward space to make the ego vehicle change the lane appropriately. The training of DRL is simulated for a lane change as an episodic task. And it means that the ego vehicle will be in a specific state at the end of the episode. Thus, a lane change is defined as moving into the next lane without collision. To meet this goal, a reward function is constructed as follows.

1. Ego vehicle longitudinal position (r_1)

$$
\begin{aligned}
&r_c(x) = \begin{cases} x & if\ i_c = 1 \\ 0 & otherwise \end{cases} \\
&r_1 = \omega_1 * r_c(\ \dot{x}, i_c)
\end{aligned}
\tag{4.5}
$$

This function is a reward function to encouraging the ego vehicle to move forward. r_c is a trigger function. ω_1 is a weight factor for reward. $\dot{x}$ is ego vehicle longitudinal velocity. and i_c is a lane change decision index value.

2. Lane-change complete index (sr_1)

$$
\begin{aligned}
&sr_c(l, y, \rho) = \begin{cases} 1 & if\ y \cong l\ and\ \dot{y} \cong 0 \\ 0 & otherwise \end{cases} \\
&sr_1 = \omega_2 * sr_c(l, y, \rho)
\end{aligned}
\tag{4.6}
$$

This function is a static reward function that encourages rapid successful lane change. sr_c is a trigger function. l is target lateral position. y is ego vehicle lateral position of current state. and ρ is ego vehicle yaw angle.

3. Ego vehicle lateral deviation (p_1)

$$p_1 = \omega_3 * |y_t - y| \tag{4.7}$$

This function is a penalty function that indicates the deviation between the current ego vehicle's lateral position and the reference lateral position. y_t is reference lateral position of current state, y is current lateral position, and ω_3 is weight factor.

4. Ego vehicle yaw rate (p_2)

$$p_2 = \omega_4 * |\dot{\rho}| \tag{4.8}$$

This function is also a penalty function that indicates the yaw stability. $\dot{\rho}$ is ego vehicle yaw rate, and ω_4 is weight-factor.

5. Distance between ego vehicle and surroundings (sp_1)

$$sp_c(x) = \begin{cases} 1 & if\ |d_{l,i}| < 4 \\ 0 & otherwise \end{cases}$$
$$sp_1 = \omega_5 * \sum sp_c(d_{l,i}) \tag{4.9}$$

This function is a static penalty function that indicates vehicle controllability.sp_c is trigger function, $d_{l,i}$ is distance between ego vehicle and surrounding vehicles especially c_1 and c_2. And ω_5 is weight factor

6. Ego vehicle velocity (sp_2)

$$sp_{c2}(x) = \begin{cases} 1 & if\ |d_{l,i}| < 4 \\ 0 & otherwise \end{cases}$$
$$sp_1 = \omega_5 * \sum sp_c(d_{l,i}) \tag{4.10}$$

This function is also a static penalty function that prevent excessive low speed of ego vehicle. sp_{c2} is a trigger function. $\dot{x}$ is ego vehicle longitudinal velocity. and ω_6 is weight factor

The sum of the rewards is expressed as follows.

$$R_t = r_1 + sr_1 - p_1 - p_2 - sp_1 - sp_2 \tag{4.11}$$

Through the training of deep reinforcement learning, the total value of the reward in one episode is output by accumulating the reward value of equation. At every time step. And training proceeds in the direction of maximizing this accumulated reward value.

The first reward value is a function that rewards the vehicle so that it can change lanes quickly. When the index value for the moment when lane change starts is triggered, compensation is rewarded as much as the ego vehicle longitudinal velocity. Second is the reward given when the lane change is completed. In order to prevent a reference trajectory from agent that the ego vehicle cannot pass along, a criterion for the success of lane change is established, and when it is completed, additional compensation is rewarded. Third to sixth are about penalties. Third one shows the difference between the lateral reference position and the current lateral position. Stable driving is impossible if the actual behavior of the vehicle differs greatly from the reference trajectory provided by agent. In other words, since this is an index indicating the controllability of the ego vehicle, optimization proceeds in the direction of minimizing it. Fourth represents the ego vehicle's current yaw rate. As the yaw rate increases, the ego vehicle gets more unstable and the ride comfort decreases. Therefore, optimization proceeds in the direction of minimizing the yaw rate. Fifth is the penalty of the distance between the ego vehicle and surrounding vehicles. If the distance becomes too close, the risk of collision increases, so it prevents the ego vehicle from getting close to surrounding vehicles. Sixth is the penalty for the vehicle's excessive low velocity. In order to prevent a vehicle from colliding, it is possible to slow down and change lanes in an overtaken method, but it is reasonable to change lanes as quickly as possible to get out of the danger situation, so a penalty is imposed if the vehicle is too slow.

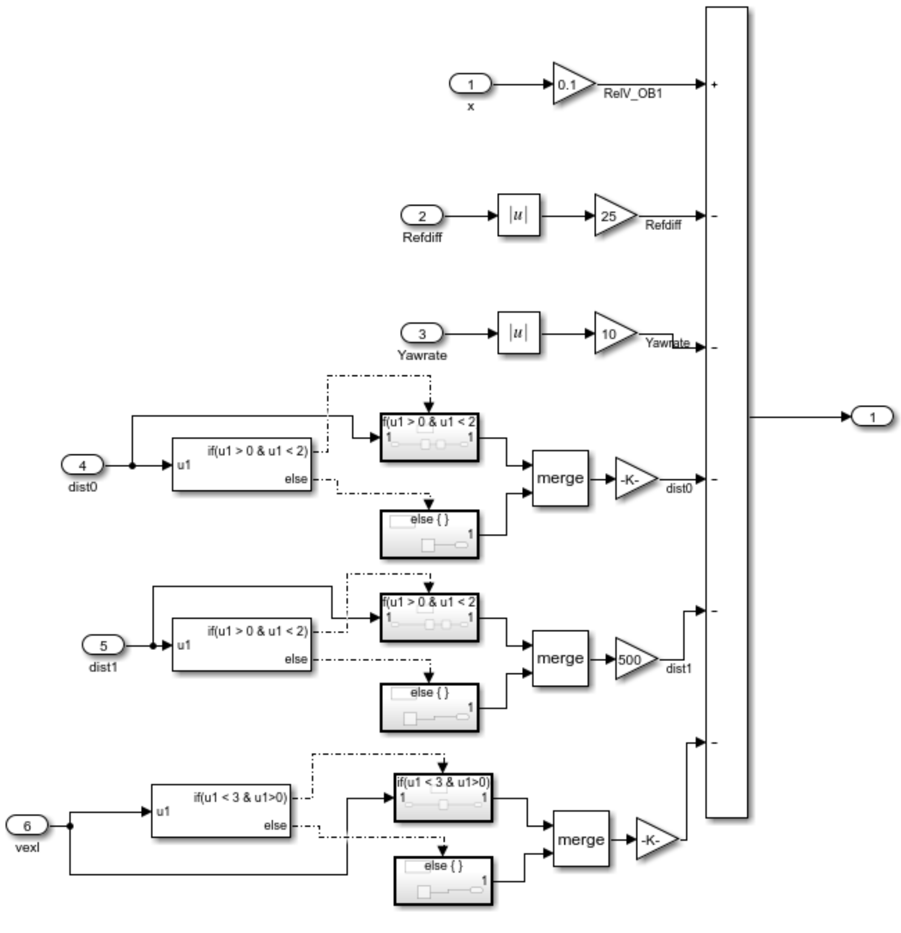
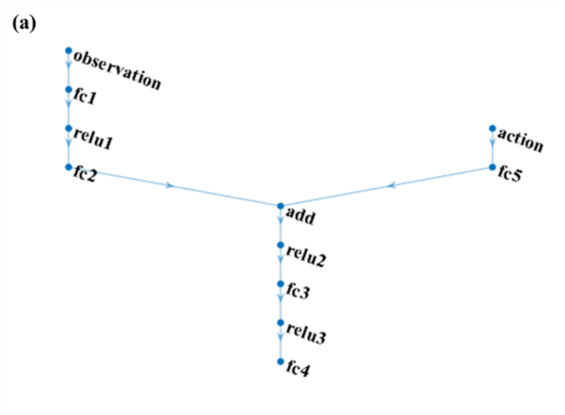


Figure 4.7. Reward-space of Simulink model

Table 4.3. Reward-space of Simulink model

Reward	**Object**
Ego vehicle longitudinal position	Encouraging progress
Static reward	
Lane-change complete	Encouraging rapid successful lane change
Penalty	
Position difference between reference trajectory and ego vehicle position	Vehicle controllability
Ego vehicle yaw rate	Vehicle stability
Static penalty	
Distance between ego vehicle and obj#1	Collision prevention
Distance between ego vehicle and obj#2	
Ego vehicle velocity	Prevention of excessive low speed



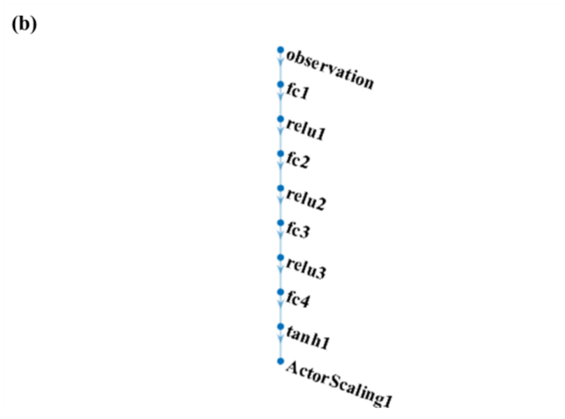


Figure 4.8. Neural network architecture: (a) Critic network, (b) Actor network

4.3.4. Neural Network Architecture

As mentioned briefly in previous Chapter 3.3, the agent constructed in this study contains the neural network architecture. The DDPG algorithm was used in this study. The DDPG algorithm consists of an actor network and a critic network as shown in Figure 4.8.(a). All of the network structure constructed in this study is shown in Figure 4.8 and Table 4.4 & 4.5.

Table 4.4. Critic-network structure

State path		
	Neurons	**Name**
Feature Input Layer	12	Observation
Fully Connected Layer	100	Fc1
Rectified Linear Unit		Relu1
Fully Connected Layer	100	Fc2
Addition Layer	2	Add
Rectified Linear Unit		Relu2
Fully Connected Layer	100	Fc3
Rectified Linear Unit		Relu3
Fully Connected Layer	1	Fc4
Action path		
Feature Input Layer	1	Action
Fully Connected Layer	100	Fc5

Table 4.5. Actor-network structure

	Neurons	**Name**
Feature Input Layer	12	Observation
Fully Connected Layer	100	Fc1
Rectified Linear Unit		Relu1
Fully Connected Layer	100	Fc2
Rectified Linear Unit		Relu2
Fully Connected Layer	100	Fc3
Rectified Linear Unit		Relu3
Fully Connected Layer	1	Fc4
Hyperbolic Tangent Layer		Tanh1
Scaling Layer		Actorscaling1

State of environment also known as observation, and action of the previous time step moment are combined as $s_a = (obs, x_{f,t-1})$. They are adopted as the input to the actor-network. Therefore, the number of neurons of actor-network input-layer which is feature input layer is 12. Meanwhile, the actor network's hidden layer utilizes four fully connected networks; each layer contains 100 neurons. The fully connected layer is followed by batch normalization (BN), before the ReLu layer is adopted as the activation function. At the same time, the layer of the network chooses tanh as the activation function to map the network output between the interval [-1, 1]. And the scaling layer the last layer of the networks makes the outputs of tanh layer to the x_f. After the observation and action are merged as $s_c = (s_a, x_f)$, then the critic network receives the merged value as input. The number of critic network input-layer neurons is 13. The critic network's hidden layer utilizes five fully connected layers and each layer contains 100 neurons. The fully connected layer is followed by batch normalization like actor network.

4.3.5. Deep Deterministic Policy Gradient (DDPG) agent

DDPG adopts the Actor-Critic framework, including the Actor and Critic network which mentioned previous steps. And Table 4.6, shows the hyperparameters used to train the DDPG agent. The online policy and the target policy network which adopt the deterministic policy is included in actor network to get a definite action from the environments. And the online Q network and the target Q network is included in the critic network. And the Bellman equation of the function Q is used to evaluate the action. The pseudo code of the DDPG algorithm is shown in Table 4.7. And the DDPG algorithm flow is shown in Figure 4.9. The input state $(d_1, \dots, d_{11}, x_{f,t-1})$ and output action x_f are represented accordingly. The action is the output of the policy because of a deterministic policy of DDPG algorithm. So, it requires relatively less data to maximize efficiency. However, this can cause the environment not to be fully explored. So, to make the algorithm fully analyze the environment, the OU stochastic process was adopted.

Table 4.6. Hyperparameters used to train the DDPG agents

Discount factor, γ	0.99
Target smooth factor	0.001
Mini-batch size	64
Target network update frequency	100
Replay memory size	1,000,000
Ornstein-Uhlenbeck parameters	
Mean attraction constant	0.15
Decay rate of the standard deviation	0
Noise model standard deviation	0.3
Minimum standard deviation	0

Table 4.7. Pseudo code of the DDPG algorithm.

```
Randomly initialize Critic online Q network parameters $\theta^Q$ and Actor's online policy network parameters $\theta^\mu$.
Initialize Critic target Q network parameters $\theta^{Q'} \leftarrow \theta^Q$ and Actor's target policy network parameters $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize experience replay memory (R).
For $episode = 1,$ M do
    Initialize the OU random process D for the exploration of action
    Input initial observation state $s_1$
    For $t = 1, T$ do
        Choose action $a_t$ according to current strategy $\mu(s_t)$ and noise $D_t : a_t = \mu(s_t) + D_t$
        Take the action $a_t$, the reward $r_t$, and the new state $s_{t+1}$.
        Store the process $(s_t, a_t, r_t, s_{t+1})$ in $R$.
        Sampling from R to get the process $(s_i, a_i, r_i, s_{i+1})$ of batch $N$
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{\mu'})//Q'$ is the state-action value calculated by the target Q network, and $\mu'$ is the current strategy obtained by the target policy network.
        Update Critic's online Q network by minimizing the loss function: $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)^2\right)$
        Update the Actor's online policy network with sampling gradient: $\nabla_{\theta\mu}\mu \Big| s_i \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta\mu}\mu(s|\theta^\mu)|_{s_i}$
        Update Critic's target Q network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
        Update Actor's target policy network: $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
    End for
End for
```

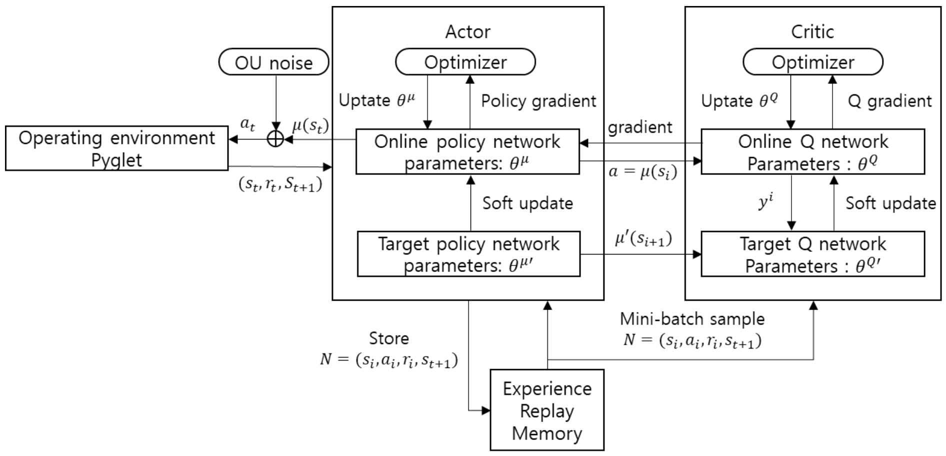

Actor
Optimizer
Uptate θ^{μ}
Policy gradient
OU noise
a_t
$\mu(s_t)$
Operating environment Pyglet
(s_t, r_t, S_{t+1})
Online policy network parameters: θ^{μ}
Soft update
Target policy network parameters: $\theta^{\mu'}$
gradient
$a = \mu(s_i)$
$\mu'(s_{i+1})$
Critic
Optimizer
Uptate θ^{Q}
Q gradient
Online Q network Parameters : θ^{Q}
y^i
Soft update
Target Q network Parameters : $\theta^{Q'}$
Store
$N = (s_i, a_i, r_i, s_{t+1})$
Mini-batch sample
$N = (s_i, a_i, r_i, s_{t+1})$
Experience Replay Memory

Figure 4.9. DDPG algorithm flow chart.

Chapter 5
Autonomous Driving Algorithm Integration

In this chapter, the algorithm for driving the vehicle in a given driving scenario is integrated. First, the algorithm that determines whether to start changing lanes while the ego vehicle is driving. The minimum braking distance to detect the risk of collision with the vehicle in front, and Gipps' safety distance is calculated to detect the risk of collision with the vehicle in front. Based on this, the lane change direction is set. Second, a path planning algorithm is designed using the DRL structure designed in Chapter 4. Third, the vehicle controller is configured and an integrated algorithm that can finally drive the vehicle is designed.

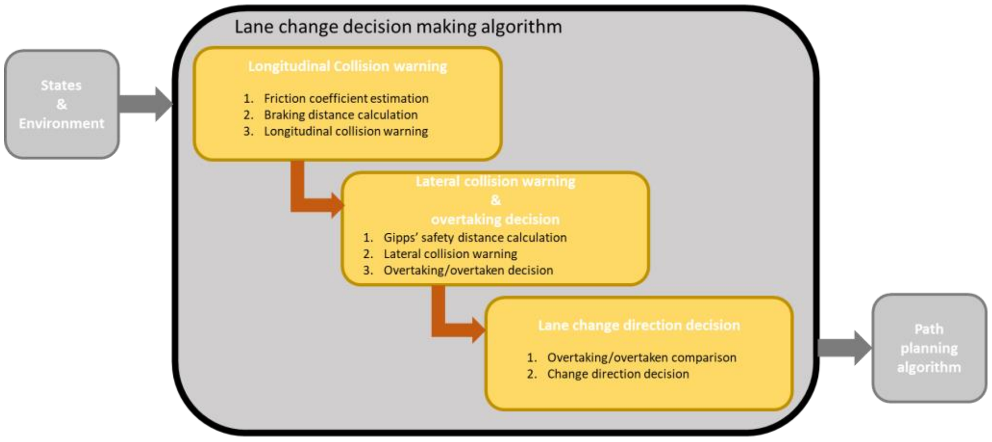

States & Environment
Lane change decision making algorithm
Longitudinal Collision warning
1. Friction coefficient estimation
2. Braking distance calculation
3. Longitudinal collision warning
Lateral collision warning & overtaking decision
1. Gipps' safety distance calculation
2. Lateral collision warning
3. Overtaking/overtaken decision
Lane change direction decision
1. Overtaking/overtaken comparison
2. Change direction decision
Path planning algorithm

Figure 5.1. Lane change decision making algorithm

5.1. Lane Change Decision Making

Prior to initiating a lane change, the ego vehicle must determine when and at what speed and direction to start to change the lane. First, in order to prevent a collision with the vehicle in front, it is necessary to detect the danger of a collision with a vehicle in front. In addition, in order to prevent a collision with a vehicle coming from the direction of the target lane, a collision with a vehicle from the side must be detected. And it is necessary to decide whether to overtake or being overtaken the target lane vehicle according to the distance between ego vehicle and target lane vehicle. Finally, a lane change direction is determined by comparing the distances of between ego vehicle and the vehicles on both lanes.

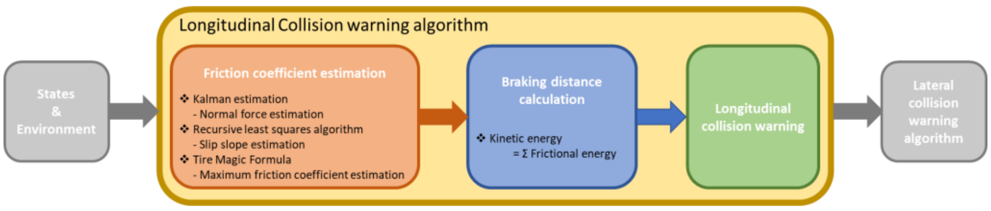

States & Environment
Longitudinal Collision warning algorithm
Friction coefficient estimation
❖ Kalman estimation
- Normal force estimation
❖ Recursive least squares algorithm
- Slip slope estimation
❖ Tire Magic Formula
- Maximum friction coefficient estimation
Braking distance calculation
❖ Kinetic energy = Σ Frictional energy
Longitudinal collision warning
Lateral collision warning algorithm

Figure 5.2. Longitudinal collision detection algorithm

5.1.1. Longitudinal Collision Detection

In order to detect a collision with the vehicle in front, the tire-road friction coefficient must first be estimated. Since the friction coefficient can be estimated based on the normal force and longitudinal force of the ego vehicle, the longitudinal traction force is first estimated. And the longitudinal traction force is estimated using Kalman estimation method by expressing the vehicle states in state-space form due to the difficulty of measuring the force directly, and the equation is as follows.

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathrm{A}(\mathrm{t})\mathbf{x}(\mathbf{t}) + \mathrm{B}(\mathrm{t}) + \omega_{\mathrm{h}}(\mathrm{t})\mathbf{y}(\mathbf{t}) = \mathrm{C}(\mathrm{t})\mathbf{x}(\mathbf{t}) + \mathrm{v}(\mathrm{t})$$

$$\mathbf{y}(\mathbf{t}) = \mathrm{C}(\mathrm{t})\mathbf{x}(\mathbf{t}) + \mathrm{v}(\mathrm{t})$$

$$\mathrm{x}(\mathrm{t}) = [\mathrm{F} \quad \Omega \quad \varphi]^{\mathrm{T}} \tag{5.1}$$

where, $\mathrm{F} = [\mathrm{F}_{\mathrm{x,fl}} \quad \mathrm{F}_{\mathrm{x,fr}} \quad \mathrm{F}_{\mathrm{x,rl}} \quad \mathrm{F}_{\mathrm{x,rr}} \quad \mathrm{F}_{\mathrm{y,f}} \quad \mathrm{F}_{\mathrm{y,r}}]$

$\Omega = [\omega_{\mathrm{h,fl}} \quad \omega_{\mathrm{h,fr}} \quad \omega_{\mathrm{h,rl}} \quad \omega_{\mathrm{h,rr}}]$

$$\mathrm{z}(\mathrm{t}) = [a_x \quad a_y \quad \omega_{\mathrm{h,fl}} \quad \omega_{\mathrm{h,fr}} \quad \omega_{\mathrm{h,rl}} \quad \omega_{\mathrm{h,rr}} \quad \varphi]^{\mathrm{T}}$$

where, $\mathrm{a}_{\mathrm{x}} = \frac{1}{\mathrm{m}}(\mathrm{F}_{\mathrm{XF}} + \mathrm{F}_{\mathrm{XR}} - \mathrm{F}_{\mathrm{a}})$ (5.1)

$$\mathrm{a}_{\mathrm{y}} = \frac{1}{\mathrm{m}}(\mathrm{F}_{\mathrm{YF}} + \mathrm{F}_{\mathrm{YR}})$$

Based on the estimated longitudinal force, the tire-road friction coefficient is estimated. The coefficient varies depending on the slip-ratio of the wheel during driving. In this study, it is assumed that the maximum friction coefficient is applied during full braking situation. Since the coefficient increases linearly as the slip ratio increases when the slip-ratio of the wheel is small, slip-slope estimation was used to estimate the maximum friction coefficient using this slope, and the equation is as follows.

$$\epsilon = \frac{F_x}{F_z} = K\sigma_x \tag{5.3}$$

ϵ means the friction coefficient of current driving state. Since this is proportional to the current slip ratio, σ_x, the slip slope, K, can be derived. Using the MF formula used in the ego vehicle dynamics, a database of the maximum friction coefficient according to the slip-slope is constructed as shown in Figure 5.3. And the maximum friction coefficient is estimated using recursive least-squares (RLS) identification. To use RLS identification, the slip-slope model of equation () is transformed into a parameter identification form as expressed as follows. And $y(t)$ is ρ the system output, $\varphi^T(t)$ is

σ_x the regression vector, $\theta(t)$ is K the estimated parameter, and $e(t)$ is identification error.

$$\mathrm{y(t)} = \varphi^T(t)\theta(t) + e(t) \tag{5.4}$$

In this study, it is assumed that the maximum coefficient of friction is applied during full braking as mentioned above, so the calculation of the minimum braking distance is performed based on the estimated maximum coefficient. During full braking, the aerodynamic force and rolling resistance of the wheel is neglected due to small scale. Accordingly, the following equation is derived.

$$\frac{1}{2}mv_{l,x}^2 = (\mu_{FL}F_{z,FL} + \mu_{FR}F_{z,FR} + \mu_{RL}F_{z,RL} + \mu_{RR}F_{z,RR})\mathrm{S} \tag{5.5}$$

Here, S denotes the minimum braking distance, μ is maximum friction coefficient, and F_z is the normal force of the vehicle at each wheel. Based on the finally calculated distance, the longitudinal collision detection algorithm operates as follows, where C_{LO} is the braking availability index function, and x_i is the relative longitudinal distance.

$$C_{LO} = \begin{cases} 1 & if\ x_l < S \\ 0 & otherwise \end{cases} \tag{5.6}$$

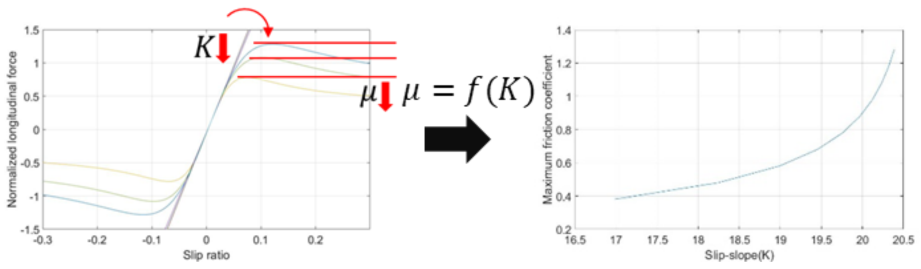
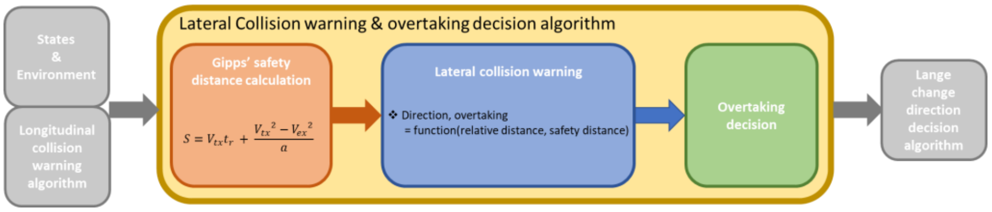


Figure 5.3. Slip-slope method using Recursive Least-squares identification



States & Environment
Longitudinal collision warning algorithm
Lateral Collision warning & overtaking decision algorithm
Gipps' safety distance calculation
$S = V_{tx}t_r + \frac{V_{tx}^2 - V_{ex}^2}{a}$
Lateral collision warning
❖ Direction, overtaking = function(relative distance, safety distance)
Overtaking decision
Lange change direction decision algorithm

Figure 5.4. Lateral collision detection algorithm

5.1.2. Lateral Collision Detection

In order to detect the risk of collision with a vehicle driving in the side lane, it is first necessary to determine the safety distance from the vehicle behind it. Gipps' safety distance formula is used to calculate the minimum distance at which the vehicle behind the vehicle cannot slow down and does not collide when changing lanes at the current speed. Following equation shows Gipps' safety distance, where S is safety distance, V_{tx} is target vehicle longitudinal velocity, V_{ex} is ego vehicle longitudinal velocity, t_r is reaction time, and a_m is maximum deceleration. It is assumed that the maximum deceleration of target vehicle is the same as the ego vehicle.

$$S = V_{tx}t_r + \frac{{V_{tx}}^2 - {V_{ex}}^2}{a_m} \tag{5.7}$$

Based on the safety distance, it is divided into two scenarios depending on whether the ego vehicle collides with the target lane vehicle. As shown in Figure 5.5, There are two cases when the longitudinal distance between the ego vehicle and the target lane vehicle is longer than the safety distance and the opposite. Scenario 1, it is possible to change lanes safely when changing lane as it is. But scenario 2 causes a collision.

Accordingly, in scenario 1, a strategy to change lanes in front of the target lane vehicle is selected, and in scenario 2, the ego vehicle changes the lane behind the target lane vehicle by decelerating the speed as much as possible through full braking. Based on these two strategies, the following equation is derived, where C_{LA} denotes collision detection index, and x_l denotes the relative longitudinal distance.

$$C_{LA} = \begin{cases} 1 & if\ d_x < S \\ 0 & otherwise \end{cases} \tag{5.8}$$

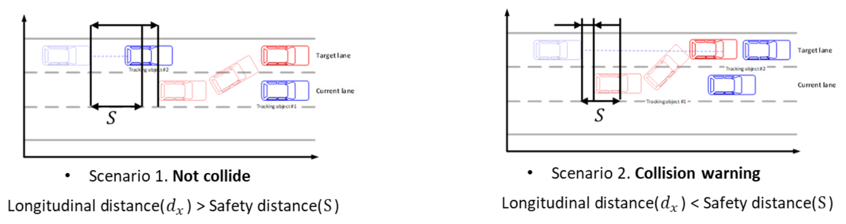

Target lane
Tracking object #2
Current lane
Tracking object #1
S
• Scenario 1. **Not collide**
Longitudinal distance(d_x) > Safety distance(S)
Target lane
Tracking object #2
Current lane
Tracking object #1
S
• Scenario 2. **Collision warning**
Longitudinal distance(d_x) < Safety distance(S)

Figure 5.5. Lateral collision detection according to Gipps' safety distance

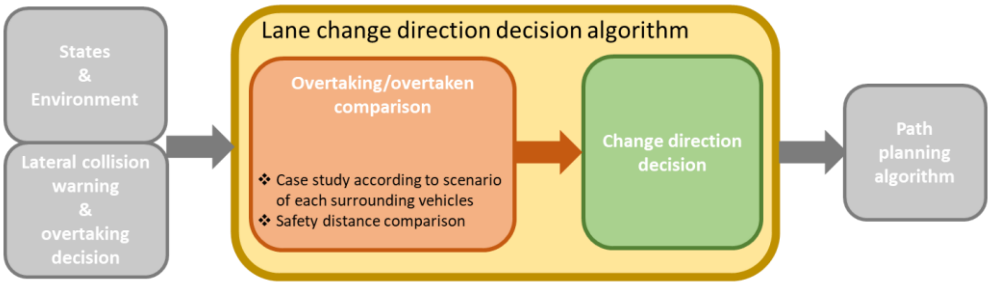
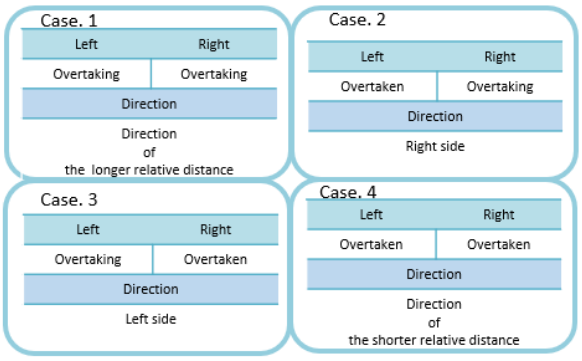


Figure 5.6. Lane change direction decision algorithm


Case. 1

Left	Right
Overtaking	Overtaking
Direction	
Direction of the longer relative distance	

Case. 2

Left	Right
Overtaken	Overtaking
Direction	
Right side	

Case. 3

Left	Right
Overtaking	Overtaken
Direction	
Left side	

Case. 4

Left	Right
Overtaken	Overtaken
Direction	
Direction of the shorter relative distance	

Figure 5.7. Case of lane change direction decision

5.1.3. Lane Change Direction Decision

Once a decision has been made whether to overtake or be overtaken for a particular lane, the ego vehicle must decide in which direction to change lanes.

In this study, the direction is determined through four divided lane change situations as shown in Figure 5.7. The first is the overtaking-overtaking case. In this case, since it is possible to change lanes without slowing down both sides, it makes sense to change lanes toward a longer safety distance lane. Second is the overtaken-overtaking case. As mentioned in the previous chpter, the priority of lane change is to get out of the danger situation quickly, so lane change is performed toward the right lane where overtake is possible. Likewise, in the case of overtake-overtaken, the lane is changed to the left lane. Lastly, in the overtaken-overtaken situation, the target lane vehicle must quickly pass the ego vehicle through deceleration. So the lane change is performed in a direction where the lane with a shorter safety distance.

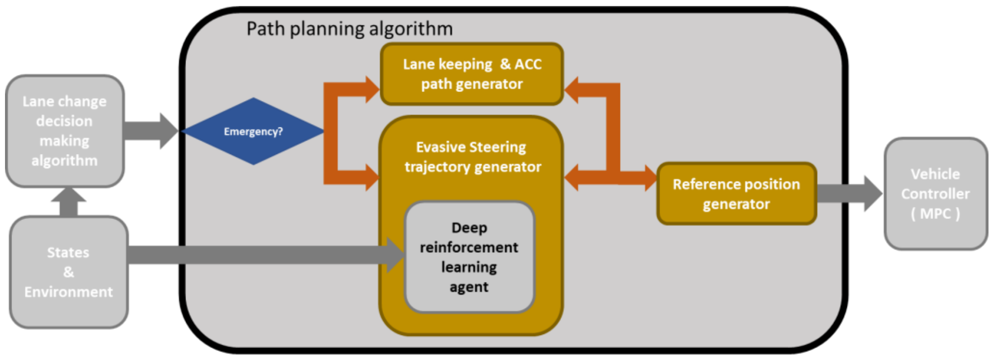

Path planning algorithm
Lane change decision making algorithm
States & Environment
Emergency?
Lane keeping & ACC path generator
Evasive Steering trajectory generator
Deep reinforcement learning agent
Reference position generator
Vehicle Controller (MPC)

Figure 5.8. Path planning algorithm

5.2. Path Planning

Based on the lane change decision making algorithm developed in Chapter 5.1.1, the chapter develops an algorithm that creates the path that the ego vehicle should be going to pass through. As in the simulation scenario, the ego vehicle drives at a constant distance from the vehicle in front in normal driving situation. Therefore, when it is not an emergency situation, it performs lane keeping and active cruise control and generates a path for this algorithm. In the meantime, when an emergency occurs, a trajectory for emergency lane change is immediately created according to driving environment and designated as a reference position so that the vehicle can prevent accident. The emergency lane change trajectory generator represented in the Figure 5.8

is a block composed of DRL agents constructed in Chapter. 4 and will be trained in Chapter 6 to generate an optimized trajectory.

5.3. Vehicle Controller

The trajectory created through the path planning algorithm is implemented as vehicle motion through the vehicle controller. In this chapter, the vehicle is controlled using an adaptive model predictive controller (Adaptive MPC)

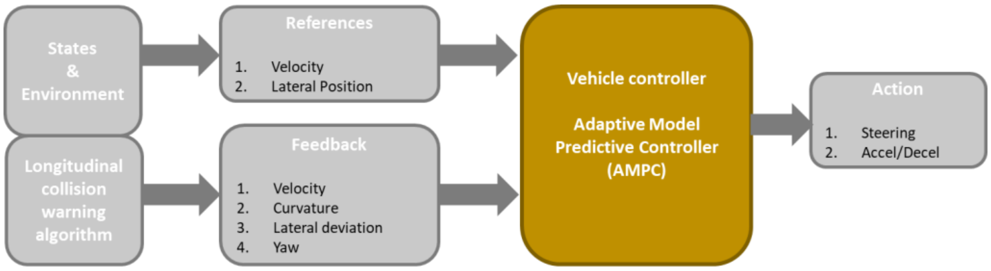


Figure 5.9. Vehicle controller using Adaptive model predictive controller (AMPC)

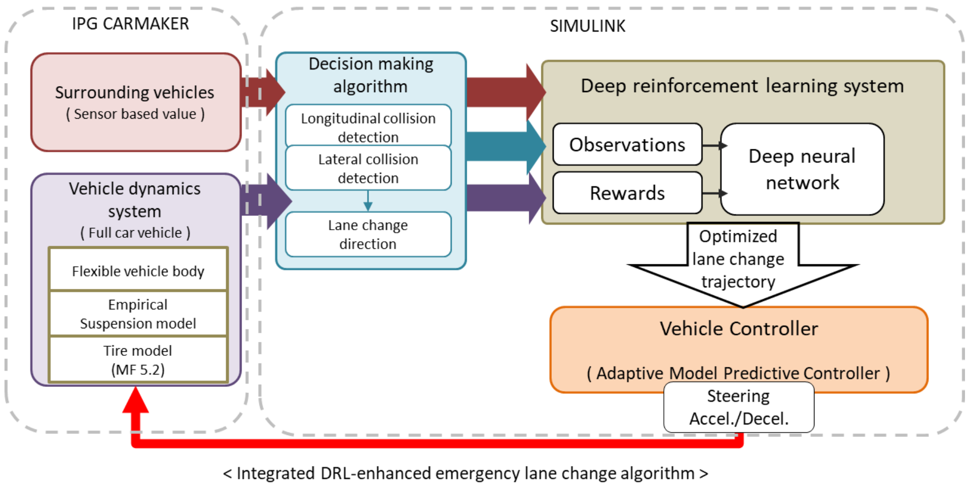

IPG CARMAKER
SIMULINK
Surrounding vehicles
(Sensor based value)
Vehicle dynamics system
(Full car vehicle)
Flexible vehicle body
Empirical Suspension model
Tire model (MF 5.2)
Decision making algorithm
Longitudinal collision detection
Lateral collision detection
Lane change direction
Deep reinforcement learning system
Observations
Rewards
Deep neural network
Optimized lane change trajectory
Vehicle Controller
(Adaptive Model Predictive Controller)
Steering
Accel./Decel.
< Integrated DRL-enhanced emergency lane change algorithm >

Figure 5.10. Integrated DRL-enhanced lane change algorithm

5.4. Algorithm Integration

In this chapter, the algorithm developed through Chapter 4 to Chapter 5 is integrated to optimize emergency lane change trajectory using deep reinforcement learning. First, IPG CARMAKER provides a vehicle dynamics simulation environment. It provides accurate states and environment information by performing reliable vehicle dynamics simulation with the Hyundai I30-Pde full car model. In Matlab Simulink environment, lane change decision making algorithm, path planning algorithm and vehicle controller are integrated. The lane change direction, overtake/overtaken, starting point of lane change are determined based on the carmaker's driving environment. These are applied as an input to the path planning algorithm, creating a reference point for lane keeping, active cruise control, and lane change. And the vehicle controller determines vehicle throttling and steering based on reference point.

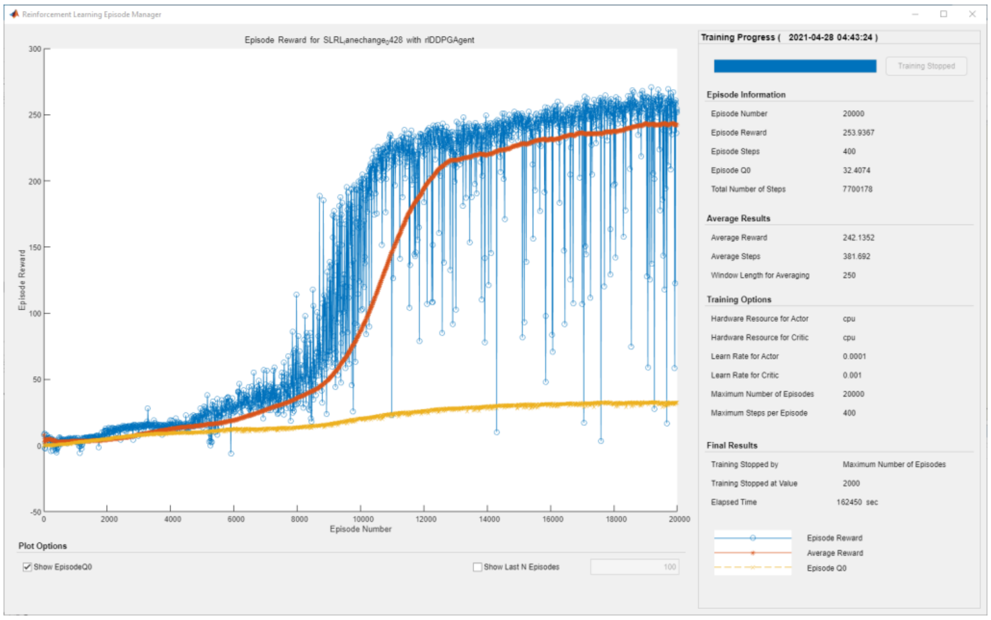
Chapter 6
Training & Results

In this chapter, applying the previously developed integrated control algorithm model for the emergency lane change, deep reinforcement learning was trained. The training result is an actor network that generates appropriate value needed to create the optimized trajectory from the state-space and a critic neural network that determines a Q value from the actor’s actions and state-space. In the optimized system, the Q value can get closer to the accumulated reward. So, it is significant that it can give feasibility of the action. As mentioned in chapter 4, the training of the DDPG agent applies noise to the action-space output to retain exploration. Therefore, the optimized agent also generates the results that include noise. In the evaluation step, the optimization quality of the agent is determined with the noise. If the accumulated reward converges to a pre-set value which means successive attempts or designated times of episode are over, the training ends. And the estimated Q value becomes closer to the accumulated reward.

In this study, the number of training episode is 60,000 with a powerfully configured computer (i9-9900K). it lasted 50 hours and 25 minutes with 8 parallel computing workers.

From the human driver point of view, a driver experiences emergency, such as a collision situation, a few times ever. And when the driver meets an emergency, the driver controls the vehicle abruptly, the vehicle comes close to the limit of stability. It is very unexpected situation, and the control depends on the driver's ability. However, the algorithm developed in this study is capable of change lanes safely and quickly for each traffic situation through training in dangerous situations 60,000 times.

Figure 6.1 is a graph that trains reinforcement learning by linking Matlab Simulink and IPG CARMAKER. The blue data on the graph means accumulated reward for each episode, and the orange graph means average reward. As a result of training 60,000 times, the average reward continuously increased and converged to 242.1352.



Training Progress (2021-04-28 04:43:24)

Episode Information

Episode Number	20000
Episode Reward	253.9367
Episode Steps	400
Episode Q0	32.4074
Total Number of Steps	7700178

Average Results

Average Reward	242.1352
Average Steps	381.692
Window Length for Averaging	250

Training Options

Hardware Resource for Actor	cpu
Hardware Resource for Critic	cpu
Learn Rate for Actor	0.0001
Learn Rate for Critic	0.001
Maximum Number of Episodes	20000
Maximum Steps per Episode	400

Final Results

Training Stopped by	Maximum Number of Episodes
Training Stopped at Value	2000
Elapsed Time	162450 sec

Plot Options

[✓] Show EpisodeQ0

☐ Show Last N Episodes 100

Figure 6.1. Deep Reinforcement Learning training graph

Figure 6.2 represents the lane change distance of the ego vehicle in each driving condition, and this is the optimized value through reinforcement learning. The x axis represents the trailing vehicle velocity of the target lane to change lanes. All three graphs (a), (b), and (c) have a constant x_f value when the velocity of the target vehicle is not significantly different from the velocity of the ego vehicle. However, when it reaches a certain velocity, it can be seen that the length of x_f decreases rapidly. This section is the moment when the ego vehicle changes from overtaking the target area to overtaken by the target area. In the overtaking section, the lane change distance is long enough from the point where braking is impossible because it is necessary to quickly change lanes in front of the target vehicle. However, since it is possible to change lanes from the moment the CG of the target vehicle passes the CG of the ego vehicle at the moment of the overtaken section, the ego vehicle allows the target vehicle to pass as quickly as possible through full braking. During that section, the ego vehicle travels in a straight line, so the distance at which lane change is possible is reduced. After x_f decreases rapidly, the value of x_f increases almost linearly as the velocity of the target vehicle gradually increases. If the value continuously increases and then increases to a certain velocity or mor, it has the same value as the lane change distance in the initial overtaking area. This is because the CG of the target vehicle is already ahead of the CG of the ego vehicle when the distance between the ego vehicle and the front vehicle is reduced below the braking distance to prevent collision with the vehicle in front, although it is an overtaken area.

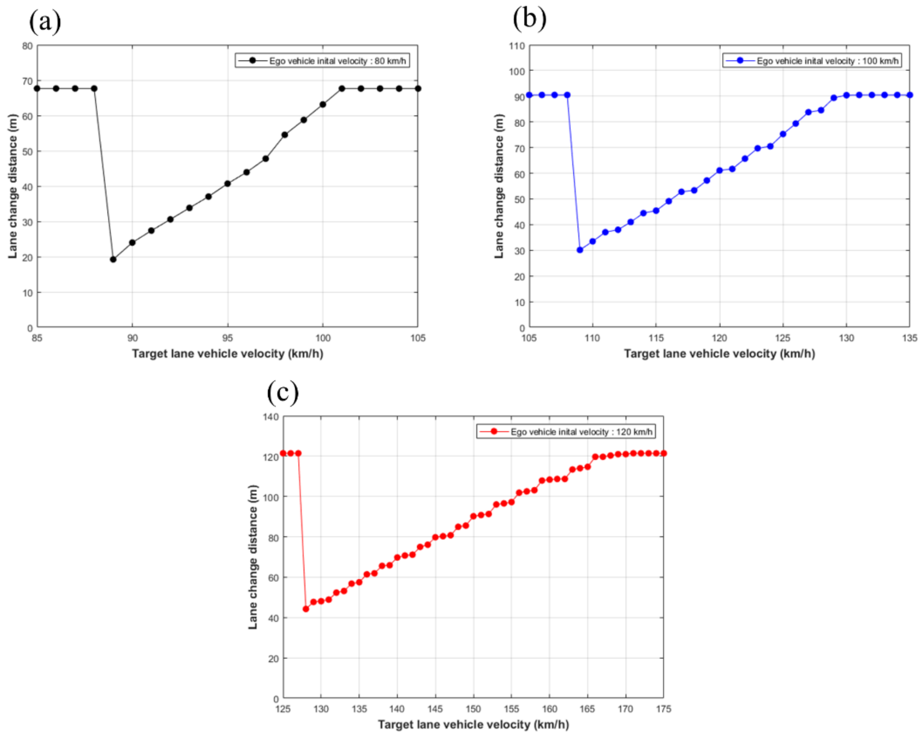


Figure 6.2. Lane change distance according to target vehicle velocity: (a) Ego vehicle velocity: 80km/h, (b) Ego vehicle velocity: 100km/h, (c) Ego vehicle velocity: 120km/h

Figure 6.3 represents the minimum distance between the ego vehicle and the front vehicle for each driving condition. As can be seen from Figure 6.3 (a), (b), and (c), it can be seen that regardless of the velocity condition, the minimum distance converges to 0.5m. This is related to the reward of reinforcement learning. Looking ate the reward in Chapter 4, it can be seen that the minimum distance does not decrease to less than 0.5m in order to maximize the reward because a large penalty is imposed if it is reduced to a certain distance from the vehicle in front.

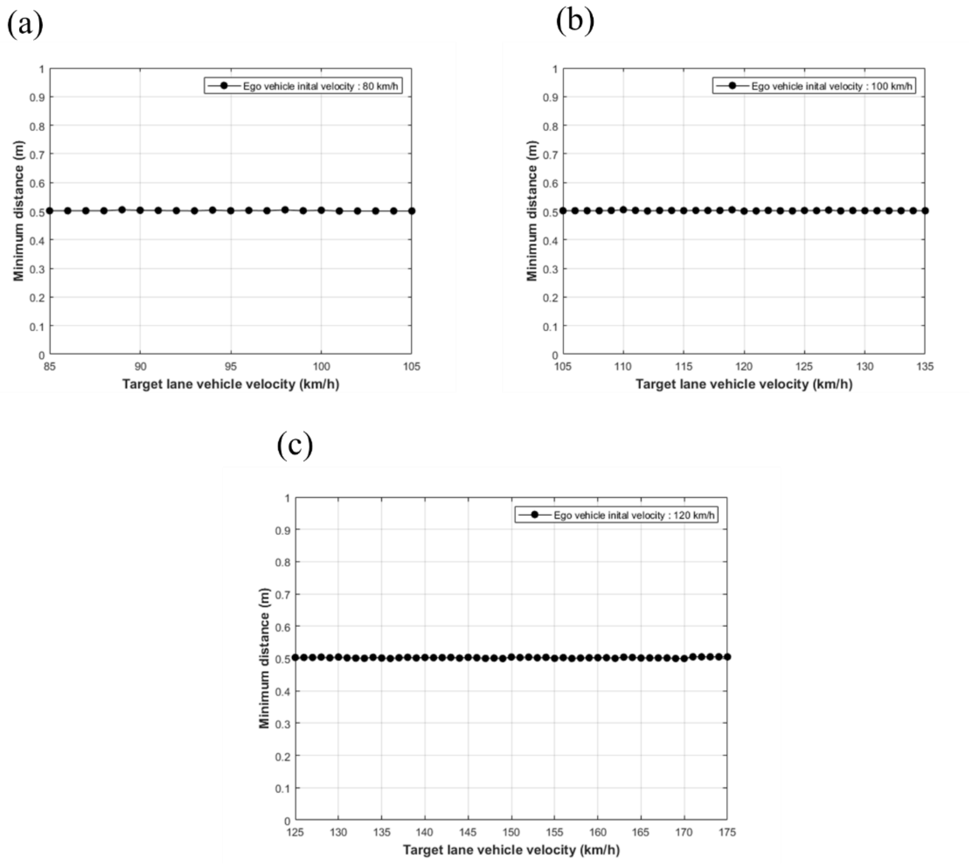


Figure 6.3. Minimum distance between ego vehicle and front vehicle according to target vehicle velocity: (a) Ego vehicle velocity: 80km/h, (b) Ego vehicle velocity: 100km/h, (c) Ego vehicle velocity: 120km/h

As shown in the Figure 6.2 and 6.3, it can be seen that the lane change trajectory through reinforcement learning is optimized. In order to change lanes quickly and safely, lane changes must be made in a direction that minimizes the minimum distance to the vehicle in front. Since the yaw rate penalty and lateral deviation in Chapter 4 increase as x_f gets shorter, optimization if made in the direction in which x_f is maximized in each condition. However, it can be seen that x_f does not decrease to less than 0.5m because it is necessary to prevent a collision with the vehicle in front due to the lengthening of x_f beyond a certain level. Figure 6.4 shows that the maximum yaw rate in each condition is inversely proportional to the distance of x_f. The maximum yaw rate in the overtaken area increases because it has to adapt to a short x_f. In Figure 6.4, it can be seen that the maximum yaw rate at the moment of switching to the overtaken area is the largest in each driving condition. In addition, it can be seen that the maximum yaw rate is larger in the scenario where the velocity of the ego vehicle is low. This is because, in the same scenario, the smaller the longitudinal velocity of the ego vehicle, the smaller the length of x_f, and thus the maximum slope of the 3rd order polynomials.

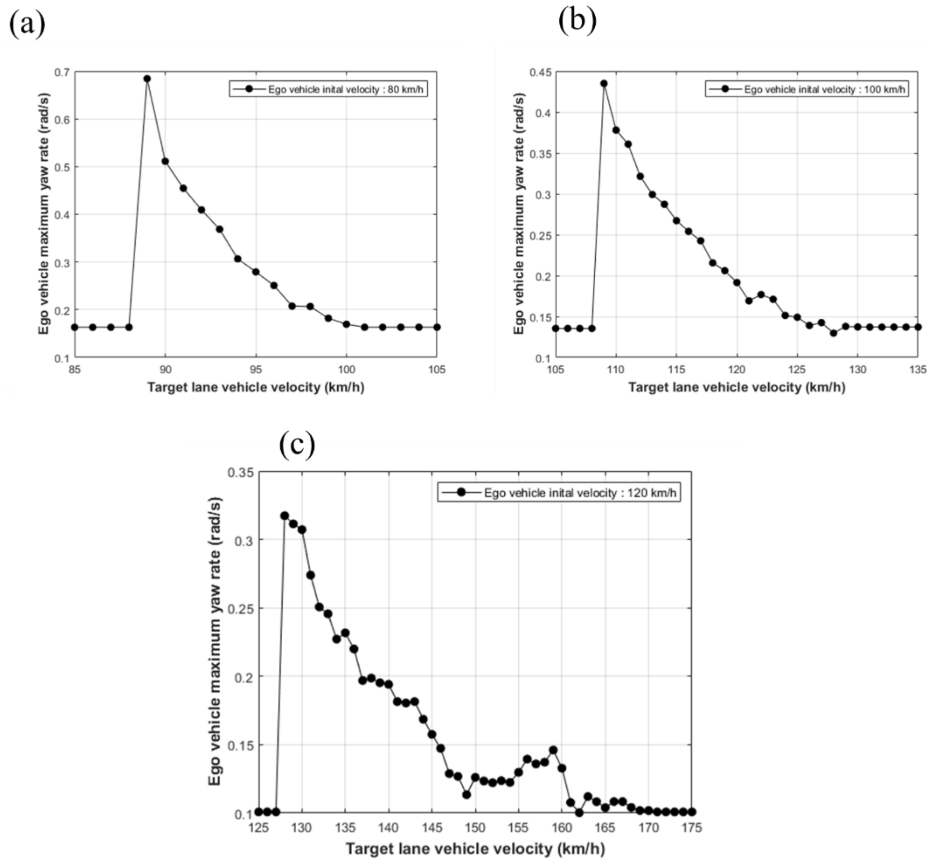


Figure 6.4. Maximum yaw rate of ego vehicle according to target vehicle velocity: (a) Ego vehicle velocity: 80km/h, (b) Ego vehicle velocity: 100km/h, (c) Ego vehicle velocity: 120km/h

Figure 6.5 is a graph of the maximum lateral forces in each driving conditions. The maximum lateral force also shows a similar pattern to the maximum yaw rate. It can be seen that the maximum lateral force at the moment of transition to the overtaken area is the largest, and this is due to the shortest x_f. Like the yaw rate, it can be seen that the maximum lateral force is the largest in the ego vehicle velocity condition of 80 km/h. The lateral force increases as the velocity increases, but as mentioned in Figure 6.4 above, the maximum value of the yaw rate at 80 km/h is the largest and the lateral force is proportional to the square of the angular velocity, so it can be confirmed that the largest lateral force appears in this section.

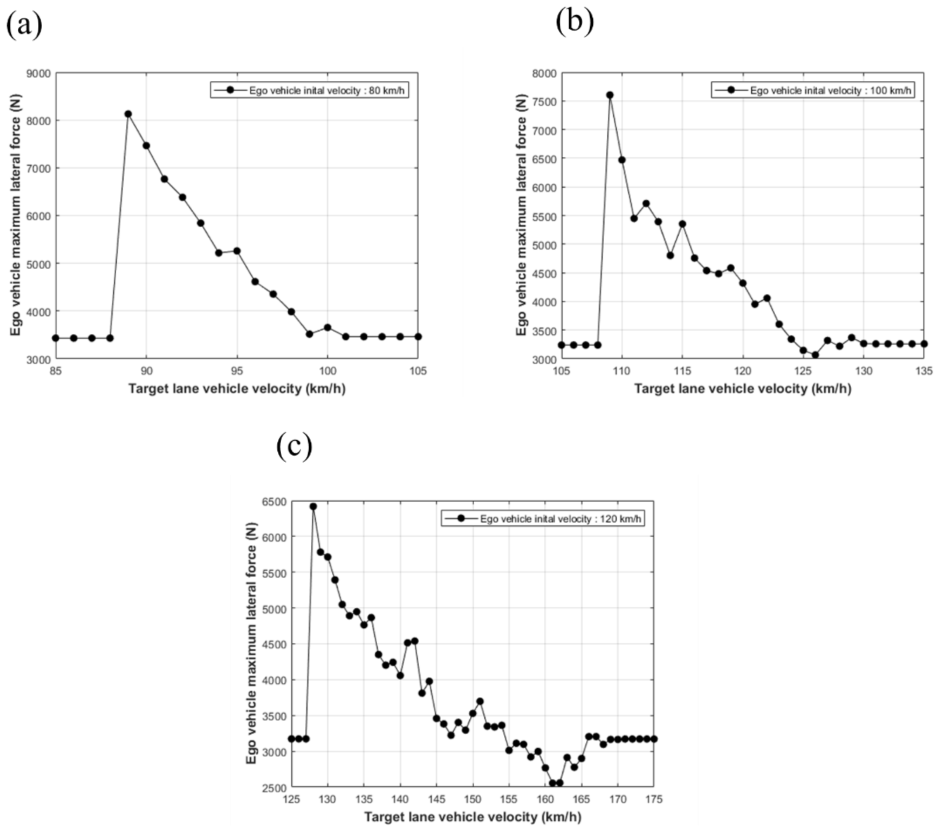


Figure 6.5. Maximum lateral force of ego vehicle according to target vehicle velocity: (a) Ego vehicle velocity: 80km/h, (b) Ego vehicle velocity: 100km/h, (c) Ego vehicle velocity: 120km/h

When changing lanes to the left or right, the direction of lane change is determined through the decision-making algorithm in Chapter 5. Figure 6.6 is a graph of the lane change direction according to the velocity of the trailing vehicles coming from the left and right lanes. The white part is an area where the left and right lanes have the same velocity, so that lane change is selective. The Light gray area indicates the area that changes to the left lane, and the dark gray indicates the area that changes to the right lane. The method of determining the lane change direction according to the speed of each lane vehicle is described in detail in Chapter 5.1.3.

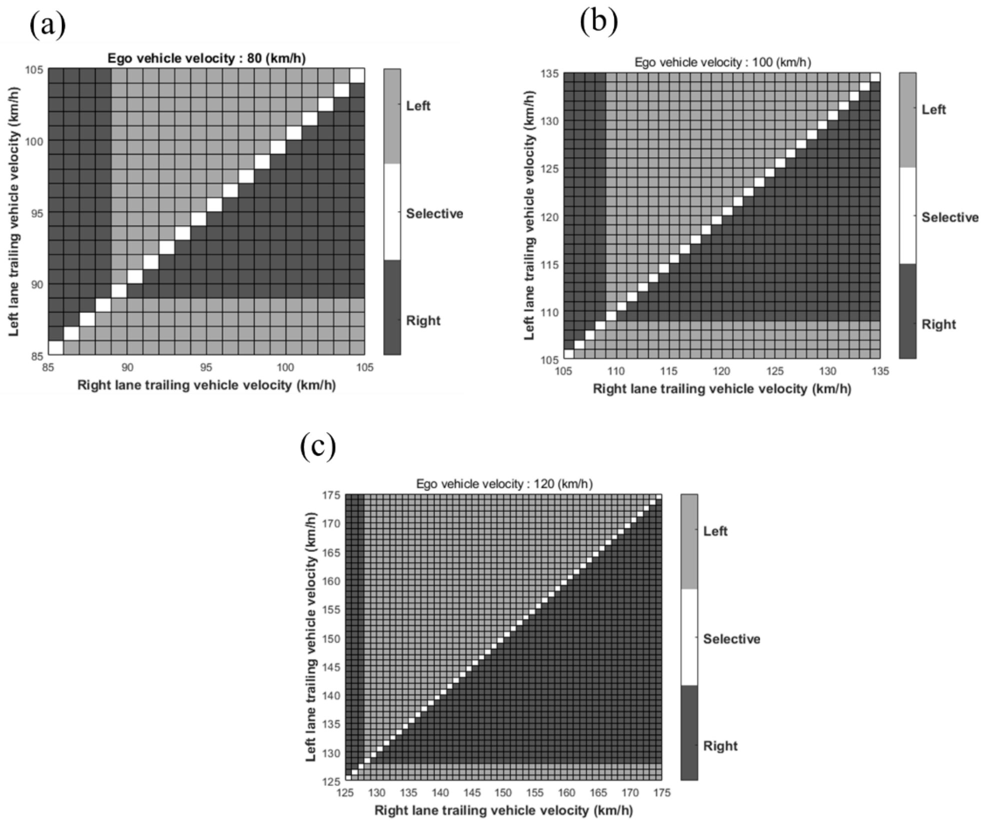


Figure 6.2. Lane change direction of ego vehicle according to target vehicle velocity: (a) Ego vehicle velocity: 80km/h, (b) Ego vehicle velocity: 100km/h, (c) Ego vehicle velocity: 120km/h

Chapter 7
Conclusion

In this study, an emergency lane change control algorithm for autonomous vehicles based on reinforcement learning was proposed.

Firstly, a simulator is selected. Because the simulation performed in this study requires very detailed model similar to the actual environment, IPG CARMAKER was selected. Hyundai I30-PDe full car model was used as a pre-defined vehicle model. Vehicle dynamics model is implemented based on the parameters of the full car model. And for deep reinforcement learning process, Matlab Simulink was chosen. The simulator shows high performance in the field of optimization because it is highly interoperable with IPG CARMAKER, and optimization and system measurement through reinforcement learning are possible.

Secondly, the scenario for simulation was constructed. Four stages of driving situations were set as a scenario for lane change in emergency situations. The first situation, the ego vehicle maintains the distance to the vehicle in front, like adaptive cruise control, and drives in normal condition. The second situation represents a situation in which the vehicle in front suddenly stops or an obstacle appears. The third one is a situation in which, after an emergency, it is decided in which way or in which direction to change lanes. This is determined through the lateral collision detection algorithm and the lane change direction decision algorithm. The fourth situation is evasive steering. Evasive steering is performed based on the lane change trajectory modeled as a 3rd order polynomial.

Third, to describe the lane change trajectory, a 3rd order polynomial lane change trajectory model is selected like following equation. This model has advantages in that it can be expressed with one explicit equation compared to the conventional arc and straight-line method. And it can be calculated more simply than 5th order polynomial model so that optimization cost can be decreased. The 3rd order polynomial can be optimized through get 4 coefficients. However, based on the initial conditions and the end conditions, the 3rd order polynomial is simplified by optimization for the lane change distance, x_f.

Fourth, a reinforcement learning structure was designed to optimize lane change trajectories through deep reinforcement learning. The reinforcement learning structure consists of observation space, actor, reward, and agent, respectively. And the selected parameters are followed

- Observation: Obj#1 relative longitudinal distance, Obj#1 relative longitudinal velocity, Obj#2&3 relative longitudinal distance, Obj#2&3 relative longitudinal velocity, ego vehicle yaw angle, ego vehicle yaw rate, ego vehicle longitudinal velocity, minimum braking distance, lane change distance, lane change availability.

- Action: 3rd order polynomial coefficients

- Reward: Ego vehicle longitudinal position, lane-change complete index, ego vehicle lateral deviation, ego vehicle yaw rate, distance between ego vehicle and surroundings, ego vehicle velocity

- Agent: Deep Deterministic Policy Gradient

Fifth, integrated autonomous driving algorithm to control ego vehicle was developed. Hierarchical algorithm that is needed for driving in simulation environment. Longitudinal collision detection algorithm estimates the tire-road friction coefficient in real time. Using this, the minimum braking distance of the ego vehicle is estimated. If the distance to the vehicle in front is less than the minimum braking distance, a collision will occur, and thus a collision is determined based on this algorithm. Lateral collision detection algorithm detects the state of the vehicle following the left and right lanes. When the lane needs to be changed through the longitudinal collision detection algorithm, Gipps' safety distance is used to detect whether a collision with a vehicle on the rear side is encountered when the lane is changed in the current vehicle velocity. If there is no collision, make a lane change toward the larger safety distance side. In the warning of a collision, the vehicle speed is lowered with maximum braking, and the vehicle is overtaken by the vehicle in the rear side and then the lane change is performed. Lane change direction decision algorithm determines which direction to change lanes. When overtaking and overtaken are determined through the later collision detection algorithm, the lane is changed through comparison of left and right value scale.

And the last, deep reinforcement learning was trained on developed algorithm and simulation environment. The algorithm is capable of change lanes safely and quickly for each traffic situation through training in dangerous situations 60,000 times. During training, the average reward continuously increased and converged to 242.1352.

The training was performed successfully, the 3rd order polynomial coefficients are fully optimized. And the evaluations according to optimized value are followed

- x_f value when the velocity of the target vehicle is not significantly different from the velocity of the ego vehicle. However, when it reaches a certain velocity, it can be seen that the length of x_f decreases rapidly.
- The minimum distance between ego vehicle and the front vehicle converges to 0.5m regardless of driving conditions.
- Maximum yaw rate at the moment of transition to the overtaken area is the largest

- Maximum lateral force at the moment of transition to the overtaken area is the largest

This study contributes to two aspects. Since trajectory optimization through reinforcement learning generally does not include vehicle dynamics, it is difficult to apply when the vehicle model changes. However, since vehicle control in this study is performed through a separate controller, a general purpose algorithm that can be applied even if the vehicle model is changed was developed. Secondly, by simulating a scenario similar to an actual emergency situation, even if it is applied to an actual vehicle, it can be quickly adapted. Consequently, it led the autonomous vehicle to drive safely. Theses results are expected to be easily and directly applicable to automobile industry.

References

1. Rolison, J.J., et al., *What are the factors that contribute to road accidents? An assessment of law enforcement views, ordinary drivers' opinions, and road accident records.* Accident Analysis & Prevention, 2018. **115**: p. 11-24.

2. Fagnant, D.J. and K. Kockelman, *Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations.* Transportation Research Part A: Policy and Practice, 2015. **77**: p. 167-181.

3. Bevly, D., et al., *Lane change and merge maneuvers for connected and automated vehicles: A survey.* IEEE Transactions on Intelligent Vehicles, 2016. **1**(1): p. 105-120.

4. Wang, M., et al., *Game theoretic approach for predictive lane-changing and car-following control.* Transportation Research Part C: Emerging Technologies, 2015. **58**: p. 73-92.

5. Rigas, E.S., S.D. Ramchurn, and N. Bassiliades, *Managing electric vehicles in the smart grid using artificial intelligence: A survey.*

IEEE Transactions on Intelligent Transportation Systems, 2014. **16**(4): p. 1619-1635.

6. Garcia-Pulido, J., et al., *Recognition of a landing platform for unmanned aerial vehicles by using computer vision-based techniques.* Expert Systems with Applications, 2017. **76**: p. 152-165.

7. Peng, T., et al., *A new safe lane-change trajectory model and collision avoidance control method for automatic driving vehicles.* Expert Systems with Applications, 2020. **141**: p. 112953.

8. Schwarting, W., J. Alonso-Mora, and D. Rus, *Planning and decision-making for autonomous vehicles.* Annual Review of Control, Robotics, and Autonomous Systems, 2018.

9. Sutton, R.S. and A.G. Barto, *Reinforcement learning: An introduction*. 2018: MIT press.

10. Arulkumaran, K., et al., *A brief survey of deep reinforcement learning.* arXiv preprint arXiv:1708.05866, 2017.

11. Wolf, P., et al. *Learning how to drive in a real world simulation with deep q-networks*. in *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017. IEEE.

12. Szalay, Z., et al., *Development of a test track for driverless cars: vehicle design, track configuration, and liability considerations.* Periodica Polytechnica Transportation Engineering, 2018. **46**(1): p. 29-35.

13. -2:, I., *Passenger cars–test track for a severe lane-change manoeuvre–Part 2: obstacle avoidance.* 2011.

14. Vorobieva, H., et al. *Geometric continuous-curvature path planning for automatic parallel parking*. in *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*. 2013. IEEE.

15. Li, X., et al. *A practical trajectory planning framework for autonomous ground vehicles driving in urban environments*. in *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015. IEEE.

16. Ferguson, D., T.M. Howard, and M. Likhachev. *Motion planning in urban environments: Part ii*. in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008. IEEE.

17. Likhachev, M., et al. *Anytime Dynamic A*: An Anytime, Replanning Algorithm*. in *ICAPS*. 2005.

18. Bian, C., et al., *Active collision algorithm for autonomous electric vehicles at intersections.* IET Intelligent Transport Systems, 2018. **13**(1): p. 90-97.

19. Hegedüs, F., et al., *Model based trajectory planning for highly automated road vehicles.* IFAC-PapersOnLine, 2017. **50**(1): p. 6958-6964.

20. Lin, Y., J. McPhee, and N.L. Azad. *Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning*. in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019. IEEE.

21. Hegedüs, F., et al., *Motion planning for highly automated road vehicles with a hybrid approach using nonlinear optimization and*

artificial neural networks. Strojniski Vestnik-Journal of Mechanical Engineering, 2019. **65**(3): p. 148-160.

22. Ly, A.O. and M.A. Akhloufi, *Learning to drive by imitation: An overview of deep behavior cloning methods.* IEEE Transactions on Intelligent Vehicles, 2020.

23. Folkers, A., M. Rick, and C. Büskens. *Controlling an autonomous vehicle with deep reinforcement learning*. in *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019. IEEE.

24. Lee, J., T. Kim, and H.J. Kim. *Autonomous lane keeping based on approximate Q-learning*. in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2017. IEEE.

25. Xia, W., H. Li, and B. Li. *A control strategy of autonomous vehicles based on deep reinforcement learning*. in *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*. 2016. IEEE.

26. Ye, Y., X. Zhang, and J. Sun, *Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity*

simulation environment. Transportation Research Part C: Emerging Technologies, 2019. **107**: p. 155-170.

27. Ronecker, M.P. and Y. Zhu. *Deep Q-Network based decision making for autonomous driving*. in *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*. 2019. IEEE.

28. Fehér, Á., et al. *Proving ground test of a DDPG-based vehicle trajectory planner*. in *2020 European Control Conference (ECC)*. 2020. IEEE.

29. Fehér, Á., S. Aradi, and T. Bécsi, *Hierarchical Evasive Path Planning Using Reinforcement Learning and Model Predictive Control.* IEEE Access, 2020. **8**: p. 187470-187482.

30. An, H. and J.-i. Jung, *Decision-making system for lane change using deep reinforcement learning in connected and automated driving.* Electronics, 2019. **8**(5): p. 543.

31. Duan, J., et al., *Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data.* IET Intelligent Transport Systems, 2020. **14**(5): p. 297-305.

32. Kim, M., et al., *Unexpected collision avoidance driving strategy using deep reinforcement learning*. IEEE Access, 2020. **8**: p. 17243-17252.

33. Wittenburg, J., U. Wolz, and A. Schmidt, *MESA VERDE—A general-purpose program package for symbolical dynamics simulations of multibody systems*, in *Multibody Systems Handbook*. 1990, Springer. p. 341-360.

34. Mnih, V., et al., *Human-level control through deep reinforcement learning*. nature, 2015. **518**(7540): p. 529-533.

35. Lillicrap, T.P., et al. *Continuous control with deep reinforcement learning*. 2015. arXiv:1509.02971.

국문 초록

서울대학교

공과대학원

기계항공공학부

송 준

긴급 차선 변경은 주행 차선에서 선행차량 급정거와 같은 응급상황 발생시에 순간적으로 이루어지는 것이므로 그 자체에 위험성을 안고 있다. 지나치게 느리게 조향을 하는 경우, 주행 차량은 앞에 있는 장애물과의 충돌을 피할 수 없다. 이와 반대로 지나치게 빠르게 조향을 하는 경우, 차량과 지면 사이의 작용력은 타이어 마찰 한계를 넘게 된다. 이는 차량의 조종 안정성을 떨어트려 스핀이나 전복 등 다른 양상의 사고를 야기한다. 따라서

차선 변경 경로의 최적화는 자율 주행 차량의 응급 상황 대처에 필수적인 요소이다.

본 논문에서는 심층강화학습을 기반으로 자율 주행 차량의 긴급 차선 변경 경로를 최적화한다. 이 알고리즘은 선행차량의 급정거나 장애물 출현과 같은 응급상황 발생 시, 빠르고 안전한 회피 거동 및 차선 변경에 초점을 맞추어 개발되었다.

알고리즘 개발의 첫 번째 단계로서 시뮬레이션 환경이 구축되었다. 신뢰성 있는 차량 동역학 시뮬레이션과 강화학습을 위한 주행 시나리오 구축을 위하여 IPG CARMAKER 가 선정되었다. 이 프로그램은 실제 산업 현장에서 사용되는 높은 신뢰성을 가진 프로그램으로 실제 차량과 유사한 차량의 거동을 분석할 수 있다. 본 연구에서는 현대자동차의 I30-PDe 모델을 사용하여 시뮬레이션을 수행하였다. 또한 강화학습과 차량제어를 위한 프로그램으로 제어, 계측, 인공지능을 모두 아우를 수 있는 Matlab Simulink 를 선정하였다. 본 연구에서는 IPG CARMAKER 와 Matlab Simulink 를 연동하여 심층 강화 학습을 바탕으로 긴급 차선 변경 궤적을 최적화하였다.

차량의 차선 변경 궤적은 3 차 다항식의 형상으로 모델링 되었다. 차선 변경 시작 지점과 종료 지점을 설정하여 다항식의 계수를 차선 변경 거리에 대한 함수로 해석하였다. 심층 강화 학습을 기반으로 계수들을 최적화하기 위하여, 강화 학습 아키텍처를 구성하였다. 관측 공간은 12 가지의 주행 환경 데이터를 이용하였고, 강화 학습의 출력으로는 3 차 함수의 변수인 차선 변경 거리를 선정하였다. 그리고 강화 학습의 학습 능력을 극대화할 수 있는 보상 공간을 설계하였다. 동적 보상, 정적 보상, 동적 벌칙, 정적 벌칙을 시뮬레이션의 매 단계마다 부여함으로써 보상 총 합이 최대화될 수 있는 방향으로 학습이 진행되었다. 최적화를 위한 알고리즘으로는 Deep Deterministic Policy Gradient agent 가 사용되었다.

강화학습 아키텍처와 함께 동역학 시뮬레이션 프로그램에서의 차량 구동을 위한 알고리즘을 개발하였다. 먼저 응급상황시에 차량의 차선을 언제, 어떤 속도로, 어떤 방향으로 변경할 지 결정하는 의사결정 알고리즘을 개발하였다. 타이어와 도로 사이의 최대 마찰계수를 실시간으로 추정하여 주행 차량이 정지하기 위한

최소 거리를 산출함으로써 선행 차량과의 충돌 위험을 판단하였다. 또한 Gipps 의 안전거리 공식을 사용하여 변경하고자 하는 차선에서 오는 차량과의 충돌 가능성을 감지하여 그 차량을 추월해서 앞으로 지나갈지, 추월을 당해서 뒤로 갈 것인지를 결정하는 알고리즘을 개발하였다. 이를 바탕으로 좌측 차선과 우측 차선의 충돌 위험성 및 안정성을 판단하여 최종적인 차선 변경을 위한 의사결정 알고리즘을 개발하였다.

구성된 강화 학습 구조를 통한 긴급 차선 변경 궤적과 차선 유지 장치, 적응형 순항 제어와 같은 일반 주행시의 궤적을 상황에 맞추어 출력하는 알고리즘을 개발하고 적응형 모델 예측 제어기를 통해 주행 차량을 구동하는 통합 알고리즘을 개발하였다.

본 연구의 마지막 단계로서, 개발된 긴급 차선 변경 경로 생성 알고리즘의 최적화를 위하여 심층 강화 학습이 수행되었다. 총 60,000 회의 시행 착오 방식의 학습을 통해 각 주행 상황 별 최적의 차선 변경 제어 알고리즘을 개발하였고, 각 주행상황 별 최적의 차선 변경 궤적을 제시하였다.

주요어: 심층 강화 학습, 신경망, 자율주행자동차, 인공지능, 긴급 차선 변경, 회피 조향, 충돌 회피, 경로 계획, 차량 제어

학번: 2012-23157

Ph. D Dissertation Information

Submitted to the School of Mechanical and Aerospace Engineering and the Committee on Graduate Studies of Seoul National University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Title: Deep Reinforcement Learning-based Path Optimization for Emergency Lane Change of Autonomous Vehicles

Keywords: Deep Reinforcement Learning (DRL), Neural Network, Autonomous Vehicle, Artificial Intelligence, Emergency Lane Change, Evasive Steering, Collision Avoidance, Path Planning, Vehicle Control

Author: Jun Song

Advisor: Professor Yeon June Kang

Laboratory: Precision Engineering & Manufacturing Lab., Seoul National University

Contact: fkqnfkqn3@snu.ac.kr, http://prema.snu.ac.kr

Version: 1.0 – the final

Date: August 8, 2021