Title	Architecture definition and verification of an automated driving system with a perception system : using V2X communication method to find safety zone
Sub Title	
Author	金, 帥(Jin, Shuai) 西村, 秀和(Nishimura, Hidekazu)
Publisher	慶應義塾大学大学院システムデザイン・マネジメント研究科
Publication year	2020
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2020年度システムエンジニアリング学 第312号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002020-0002

Master's Dissertation 2020

Architecture Definition and Verification of an Automated Driving System with a Perception System: Using V2X Communication Method to Find Safety Zone

Shuai Jin

（Student ID Number：81834510）

Supervisor　Hidekazu Nishimura

September 2020

Graduate School of System Design and Management,

Keio University

Major in System Design and Management

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81834510	Name	JIN SHUAI

Title

Architecture Definition and Verification of an Automated Driving System with a Perception System: Using V2X Communication Method to Find Safety Zone

Abstract

In recent years, although the occurrence of traffic accidents has been greatly reduced in some developed countries, it has now entered a bottleneck period and needs new ways to improve the traffic safety. The automated driving system (ADS) is considered as an effective measure to further reduce traffic accidents. The current mainstream ADS mainly relies on sensors to perceive the physical environment around the ego vehicle. However, perception based only on sensors is not safe enough as there may exist the risk of sensor failure. With the development of vehicle to everything (V2X) communication technology, if V2X can be applied to the perception system of ADS, it will broaden the method of collecting information so as to make ADS information sources become more abundant. Based on this background, this research is intended to propose an ADS with a perception system that can conduct V2X communication to understand the environment.

In this paper, the automation definition of automated driving by different countries is introduced. It also analyzes the existing major ADS structure. Besides, the architecture of proposed ADS is defined by SysML (Systems Modeling Language). It describes the context of automated driving, illustrates the use case of ADS, introduces the internal systems of ADS, and details the activities of the perception system. Furthermore, the concept of safety zone is introduced to ensure the driving safety of the vehicles. This research takes the intersection as a known example scenario to describe how the V2X communication system collects information and how the perception system calculates the safety zone.

Based on the aforementioned model, the ADS is simulated on the SUMO open-source driving simulator, programmed by Python language. In the process of simulation, the parameters of vehicle communication delay, communication distance and braking distance are considered. The simulation result shows that the minimum safety zone value is positive and in the range of 0 ~ 1.5 m, which can reach the conclusion that the proposed ADS system is feasible and is able to guarantee the vehicle safety.

Key Word(5 words)

Automated Driving System, Perception System, V2X Communication, Model-Based Systems Engineering, Safety Zone, System Architecture

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Research Background

1.1.1 Road Safety is Serious

According to the road safety report published by the United Nations, in recent decades, the number of road deaths in most parts of the world is increasing. If this trend continues, this number is expected to reach 2.4 million a year by 2030. In addition, traffic accidents cause around 20 to 50 million non-fatal injuries every year, and these avoidable injuries overload the strained health care system in many developing countries[1-3]. From this perspective, traffic safety needs to be paid more attention to, and effective methods should be developed to reduce the number of traffic accidents.

On the other hand, due to the early emphasis on traffic safety in developed countries, many measures have been adopted to improve people's awareness of safe driving. The number of traffic accidents in some countries, such as Japan, the United States and Canada, has decreased year by year. However, the overall trend tends to be flat in recent three years, which indicates that the impact from policy or social aspects may have been very limited. Nevertheless, the Japanese government has set a goal of halving the number of death caused by traffic accidents in the coming 10 years. In order to achieve this goal at an early date, the government considers that the development and popularization of the Automated Driving System (ADS) is an important tool to reduce traffic accidents and the number of death in the next stage[4-5].

For another thing, urban traffic includes different types of road scenarios such as intersections, curves, overpasses, etc. The traffic intersection is the area where all kinds of vehicles gather and change directions in the city. It is an essential node of

the road traffic network, and also the frequent happening place of urban road traffic accidents. According to the investigation report conducted by National Highway Traffic Safety Administration (NHTSA), there are more than 7 million traffic accidents every year in the United States, about 36% of these cases occurred at intersections. Jingwei Qian's research shows that 36% of traffic accidents in China occur at intersections in 2017[6-7]. As for the proportion of accidents at intersections in the world, the value of most countries is in the range of 30% to 60%[8]. The high traffic accident rate and low traffic efficiency make the traffic intersection become the bottleneck of the traffic system. In the experimental part of this research, the traffic intersection is also selected for simulation. Because of the complexity of traffic in some cases and the limited ability of human observation, the ADS may have a strong ability to collect information, so as to ensure traffic safety. The above-mentioned background can reflect the message that the current road safety is serious and the research of ADS has the promoting significance for the mitigation of the traffic safety.

1.1.2 Motivation - ADS with a Better Perception System

As for ADS, perception, decision-making and command execution are the three important processes. The perception system is intended to collect the general information of the surrounding physical environment, including traffic signals, front and backward objects parameters, ego vehicle parameters, and drive's status, etc. The ADS perceives the environment through sensors. The sensors help the command system with ADS to carry out preliminary crisis intervention. It is the subsystem of great importance to ADS[9]. However, this perception method based on sensors has certain limitations, because the sensors have their own ideal working conditions. In the case of poor conditions, sensors may fail. For example, in very dark weather, color cameras may not work normally, although there can exist infrared cameras in ADS, but the information source is relatively limited at

this time. The information obtained by ADS can not completely reflect the physical environment.

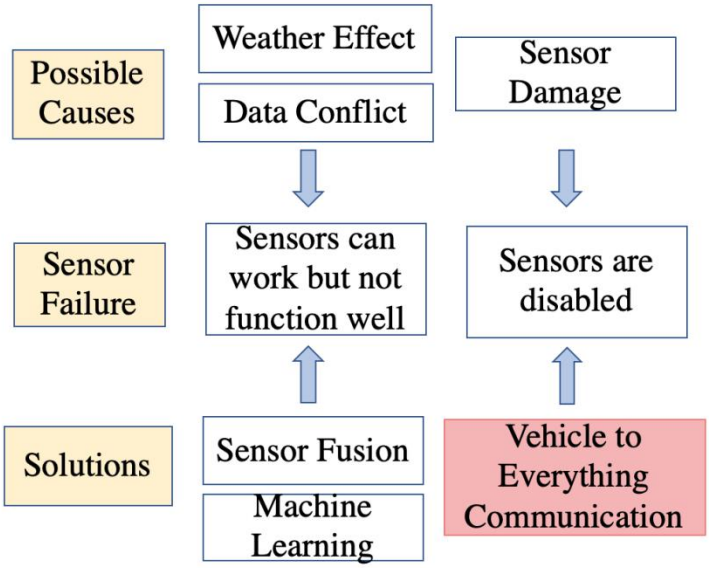


Figure 1.1 Analysis of Sensor Failure

Figure 1.1 analyzes the sensor failure, which can be divided into two cases. The first case is that the sensor can work, but can not achieve the expected working status. The possible causes are the influence of the weather, the quality of the sensor and the data conflict, etc[10-11]. The solutions to this problem are introducing more advanced sensor fusion algorithms and adopting mechanical learning method to improve the accuracy of the sensors. Another situation is that the sensor is fully adapted due to some limitations and the possible solution is to use the Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) communication technology to get information from physical environment[12]. Based on the above contents, it can be understood that only relying on sensors to collect information may not very reliable, and other ways of collecting information need to be broadened.

1.1.3 Motivation - V2X Communication

Vehicle wireless communication is a new generation of information communication technology that connects the ego vehicle with everything (V2X), in which V represents the vehicle and X represents any object that interacts with the ego vehicle. As shown in the Figure 1.2, V2X is a mesh network. Nodes (vehicles, infrastructure, pedestrian and telecom network) in this big network can transmit and capture signals. With V2X, ego vehicle can obtain the unknown parameters of the surrounding environment like the states of the other vehicles and objects nearby, which may include the basic information such as speed, position, driving direction, etc[13-14]. Then the designated security algorithm may be used to processing the acquired information, classifying the information according to the priority, and give early warning to other "nodes" in dangerous situation.

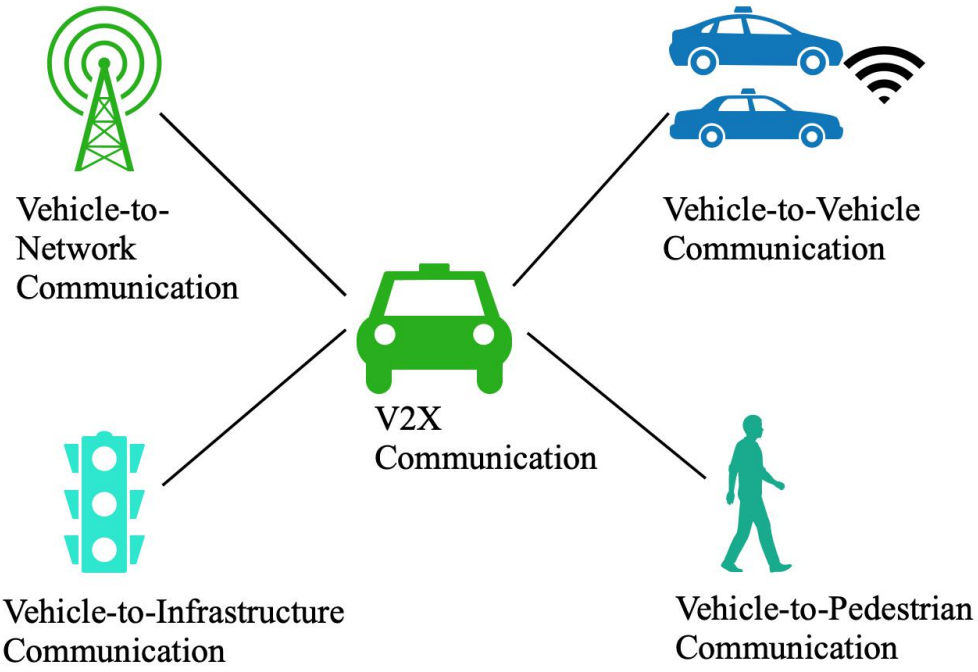


Figure 1.2 Introduction of V2X Communication[15]

V2X can detect the surrounding objects differently from the traditional sensors detection method, which can help ego vehicle and even other vehicles avoid danger in extreme cases. Therefore, it may help to improve the accuracy by collecting more formation.

Improving traffic efficiency is possible with the help of V2X. It is of great significance to alleviate urban traffic congestion, energy conservation and emission reduction. Speed guidance and vehicle platooning is the typical application for V2X. The speed guidance means the ego vehicle can collect the timing information of traffic infrastructure through the communication unit, and broadcast the current status of the infrastructure as well as the approaching time to the surrounding vehicles[16]. After receiving this information, the ego vehicle calculates the recommended driving speed based on the current position and gives command to the other vehicles to conduct safe driving operation. Vehicle platooning is human-driven vehicles or automated driving vehicles that maintain real-time information interaction with the first vehicle in the platooning through V2X communication. It can achieve stable follow-up of multiple vehicles within a certain vehicle distance, speed, and has other functions including lane keeping and tracking, cooperative lane change, etc[17]. Both speed guidance and vehicle platooning are implemented in the simulation part of this research.

1.2 Research Problem

As aforementioned in chapter 1.1.1 and 1.1.2, with the further research on automated driving, the limitations of perception based on sensors are highlighted. ADS is not effective in dealing with sensor failure. Sensor failure may cause ADS having no data for operation and lead to potential risks for vehicles. For another

thing, road safety is serious and the happening rate of vehicle accidents at the intersections is high. The current ADS needs to be improved.

1.3 Research Purpose

In order to address the problem that the ADS may not have enough data for normal operation, the V2X communication method is introduced to the perception system to collect the necessary information. Applying V2X to the perception system may improve the reliability of ADS by taking its advantages. Meanwhile, safety zone is defined to protect the vehicle safety at the intersection. In summary, this research is committed to propose an ADS architecture with a perception system using new detection method and verify its feasibility.

1.4 Research Approach

Reasonable system modeling is important for conducting ADS research. There are many ways to model large-scale projects. in the past, large-scale projects usually adopted document-based system engineering method. However, document-based system engineering method has limitations that the completeness, consistency, and relationship between requirements, design, engineering analysis, and test information are difficult to assess because the information is spread across multiple documents. It is also difficult to understand a specific part of the system and perform the necessary traceability as well.

Model-based system engineering (MBSE) is regarded as a more effective method compared with document-based method, which can manage complexity, improve design quality and communication between developers. Systems Modeling Language (SysML) is a graphical modeling language for MBSE[18]. It can be used for specification, analysis, design, verification and evaluation of various systems and system of systems (SoS). Figure 1.3 describes different types of

SysML diagrams, which can simulate the system from various perspectives. For example, the block definition diagram can describe the system composition and introduce the system context. The requirement diagram can be used to describe the physical and functional requirements of the system. Activity diagram can introduce the operation situation of the system in depth, etc. One of the benefits of SysML is it allows the traceability between requirements and design model. Therefore, it is suitable for building ADS architecture.

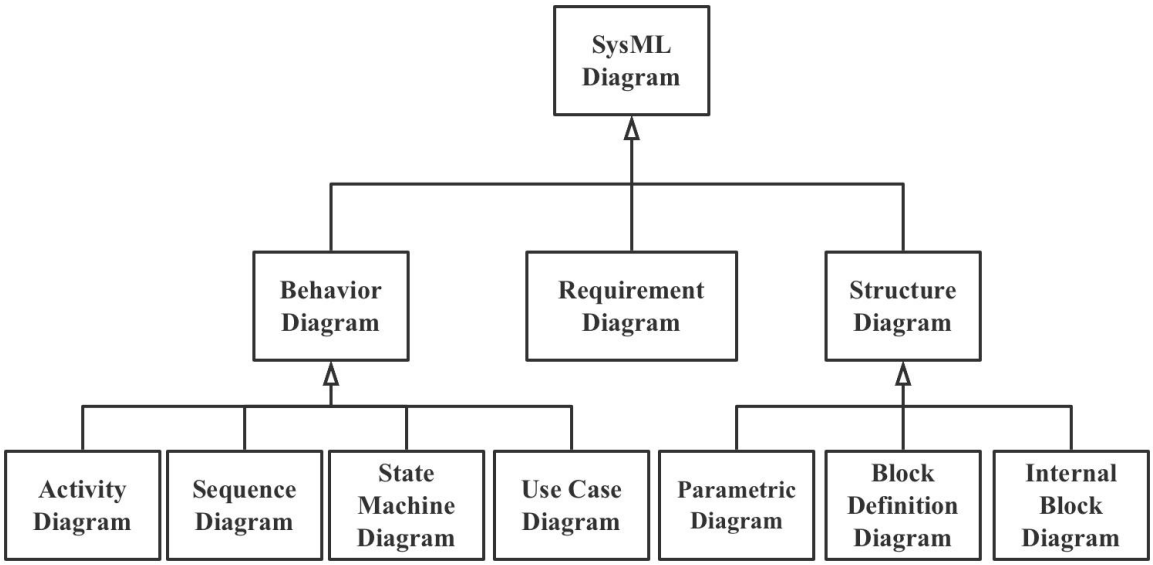


Figure 1.3 SysML Diagram Taxonomy

This research uses SysML to model the ADS based on the system engineering methodology, describing the operation activities of the perception system using the V2X communication method in detail. In addition, the ADS architecture is programmed using the SUMO simulator, and the simulation is performed at the selected scenario.

1.5 Dissertation Structure

This dissertation includes 6 main chapters and followed by the acknowledgment.

The first chapter elaborates the need of automated driving and the basic formation of V2X communication. In the second part of this chapter, the problem that perception system within ADS relies too much on sensors is specified. Specific research methods and propose are described.

The second chapter introduces the concept and definition of ADS, and takes Apollo and Autoware ADS as examples to describe the system architecture and the workflow of the perception system. The third section explains the safety of the ADS system from the perspective of internal functions and expected functions of the system.

The third chapter models ADS based on the systems modeling language, elaborating on the composition of ADS and modeling the behavior and activities of the perception system.

The fourth chapter describes the basic parameters and settings of the simulation while the fifth chapter illustrates the experimental results, analysis and discussion. Besides, it also mentions the future research direction based on the review of the deficiency of the simulation. The sixth chapter is an overall summary of this research. Finally, there is an acknowledge chapter for this paper.

2. ADS Architectures in the Literature

2.1 The Definition of Automation Level for Automated Driving

In 2013, the National Highway Traffic Safety Administration (NHTSA) first announced the grading standard of automated driving vehicles. It divided the vehicle automation into 4 levels: driver assistance, partial automation, conditional automation and complete automation. The American Automotive Engineers Association (SAE) issued the SAE J306 standard on the basis of NHTSA's classification in 2014. As shown in Table 2.1, there are some similarities between the two standards, which are based on whether the vehicle can control some key driving functions to distinguish automated driving and non-automated driving. In addition to describing the detailed functions corresponding to each level, SAE emphasizes the dynamic driving grading index, and defines the drivers' role in driving process separately, so as to refine the level of automation. The U.S. government uses the level of automation standard developed by SAE as a federal guideline for automated driving[19].

As shown in the table, the automation of automated driving is divided into 0 ~ 5 levels according to the grading standard developed by SAE, which respectively correspond to the human driving, assisted driving, partial assisted driving, conditional automated driving, highly automated driving, and full automated driving. The Level 1 system can assist the driver to complete some simple driving tasks. In order to improve the driving safety, vehicles in this level is equipped with LDW (lane departure warning), FCW (forward collision warning), BSD (blind spot detection function) and TSR (traffic sign recognition) alarm system. For example, The LDW can assist the driver in managing the lateral motion of the vehicle by controlling the electronic steering system. In this level, the vehicle control, environmental monitoring and system response is conducted by the driver.

Table 2.1 Level of Automation by NHTSA and SAE Standard[19]

Automation Level		Name	Definition	Monitor	Operator
NHTSA	SAE				
L0	L0	Human driving	Only driven by human	Human driver	Human driver
L1	L1	Assisted driving	The vehicle provide the assistance of steer-wheel or speed while the hum ab driver control the rest	Human driver	Human driver
L2	L2	Partial assisted driving	The vehicle provide a certain assistance of steer-wheel and speed while the human driver control the rest	Human driver	Human driver
L3	L3	Conditional automated driving	The vehicle control the majority of the driving operation while human driver still need to pay attention to it	Vehicle	Human driver
L4	L4	Highly automated driving	The vehicle control all the driving operation while human need no attention but only for specific situation	Vehicle	Vehicle
	L5	Full automated driving	The vehicle control all the driving operation while human need no attention	Vehicle	Vehicle

Level 2 can control the vehicle both horizontally and vertically. In this level, the driver needs to monitor the surrounding environment, correct the mistakes and take over the authority in time when the system makes a wrong judgment.

Level 3 is conditional automation, which can replace the driver to complete some driving tasks and have some environmental monitoring functions. Still, the driver needs to regain the driving control right in time when the system sends out a request.

Level 4 system is a highly automated driving system, the system itself is responsible for vehicle control, environmental monitoring and system response, but there also exists driver control mode.

The level 5 system is fully automated. The driver does not need to be involved in driving activities at all. ADS will take over driving completely.

German Federal Highway Research Institute (BASt) divides the automation of automated driving into five stages: driver control stage, assassinated driving stage, partial automated driving stage, highly automated driving stage and fully automated driving stage, corresponding to 0 ~ 5 levels of SAE standard respectively. At the same time, the rule also describes the legal consequences that may occur when the automobile is at different level of automation, so as to facilitate vehicle automation laws and regulations[20].

China Automotive Engineering Association also defines the process of automated driving into five levels. Different from the above standards, China Automotive Engineering Association particularly classifies the network connection of vehicles, emphasizing the application of network technology in the field of automated driving[21].

2.2 Analysis of Existing ADS Architecture

2.2.1 Baidu Apollo ADS

Apollo was released by Baidu company in April 2017 in order to provide an open, complete and secure ADS platform for stakeholders in the automobile industry. Currently, This system is able to cover some of the tough automated driving tasks and have corresponding solutions. Baidu named the project "Apollo", borrowing the meaning of the Apollo moon landing program[22].

Figure 2.1 shows the system architecture of Apollo 3.0. It contains five subsystems including Turkey solution, open vehicle certificate platform, hardware subsystem, software subsystem and cloud service subsystem. The followings listed below are the introduction to the subsystems：

- Open vehicle certificate platform is a basic subsystem. It will identify whether the vehicle has the Apollo ADS qualification before starting the system.
- Turkey solution classifies the current types of vehicles. As different types of vehicles have different features, the parameter settings in the ADS may vary as well. This subsystem will help to improve the safety of automated driving.
- The hardware subsystem is mainly composed of sensors, including the computing unit, GPS, camera, radar, HMI hardware, etc. Mapping, localization, perception, controlling and HMI functions are integrated in the software subsystem. The data gathered by sensors will be transferred to the software subsystem for further processing and then be sent to software cloud service to perform decision-making operation and command execution[23]. The vehicle will act accordingly based on the command.

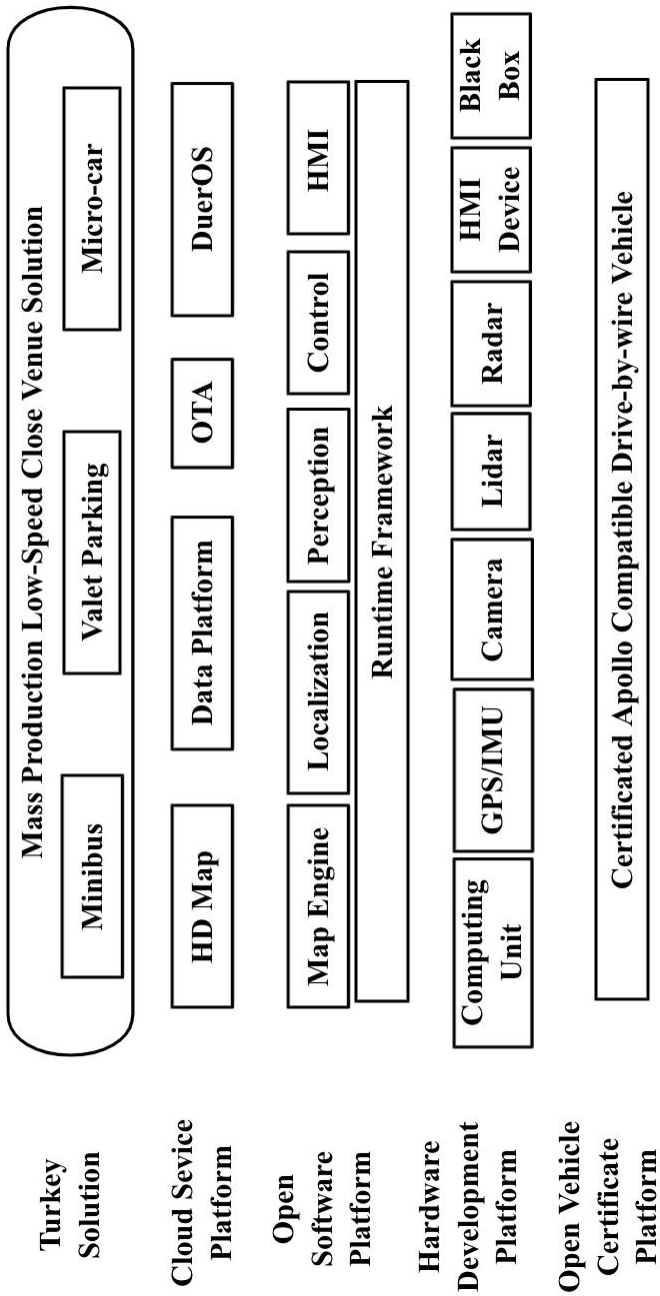


Figure 2.1 Apollo 3.0 System Architecture[23]

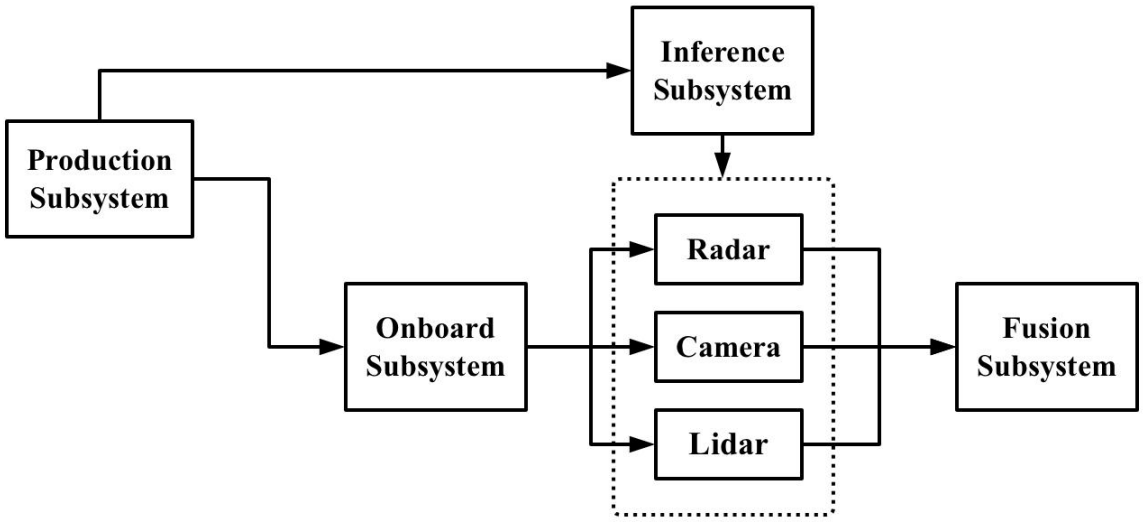


Figure 2.2 Activity Analysis of Apollo ADS Perception System

Figure 2.2 shows the workflow of Apollo perception process.

- The entry of perception activity is in the production subsystem, and it is triggered by external launch operation from the command system to start the onboard and interface substitute.
- The onboard subsystem is used to guide different sensors, including lidar, radar, and camera. The process of guiding each sensor is similar, which can be divided into prepossessing, object recognition, the region of interest (ROI) filtering and tracking.
- The inference subsystem has the function of deep learning. After the deep learning model is trained, it needs to be deployed. Inference actually accelerates the model in the process of deployment, and mainly implements three deep learning models like Caffe, TensorRT and paddle deployment. After that, loaded deployment and online calculation will be carried out.
- Camera subsystem mainly realizes lane recognition, traffic light detection, obstacle identification and tracking functions.

- Radar subsystem mainly realizes obstacle recognition and tracking. The millimeter-wave radar can report obstacle information, and here it is responsible for tracking obstacles.
- Lidar subsystem mainly realizes obstacle recognition and tracking (segmentation, classification, recognition, etc.) of lidar point cloud.
- Fusion subsystem is responsible for fusing the sensing results of the above sensors.

From the ADS architecture of Apollo, it can be seen that the perception activity is supported by deep learning to improve the accuracy of generated data. After the first data processing, it will be transmitted to the fusion subsystem for data fusion, thus enhancing the reliability of the final data and ensuring driving safety to a certain extent. However, in this system architecture, there is a high dependence on sensors, which can not guarantee the normal environment perception of ADS in various situations.

2.2.2 Autoware ADS

Autoware ADS was first officially released in August 2015 by the research group of Nagoya University under the guidance of Shinpei Kato. In late December 2015, Kato Nobuhira founded Tier IV to maintain Autoware and apply it to the real vehicle. Over time, autoware has become a recognized open source project. As shown in the Figure 2.3, Autoware has three subsystems named sensing, computing and actuation. It supports localization, mapping, object detection and tracking, traffic light recognition, mission and motion planning, lane detection and selection, vehicle control and sensor fusion. The sensing subsystem includes cameras, lidar, radar, GPS. Autoware is suitable for running in urban cities. Additionally, highways, freeways, mountainous regions, and geofenced areas can also be supported[24].

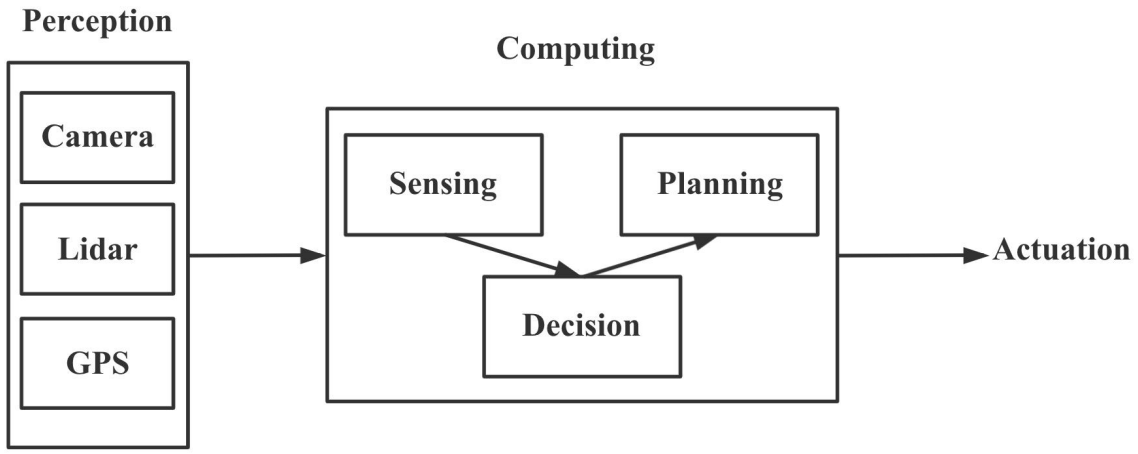


Figure 2.3 Autoware ADS Architecture

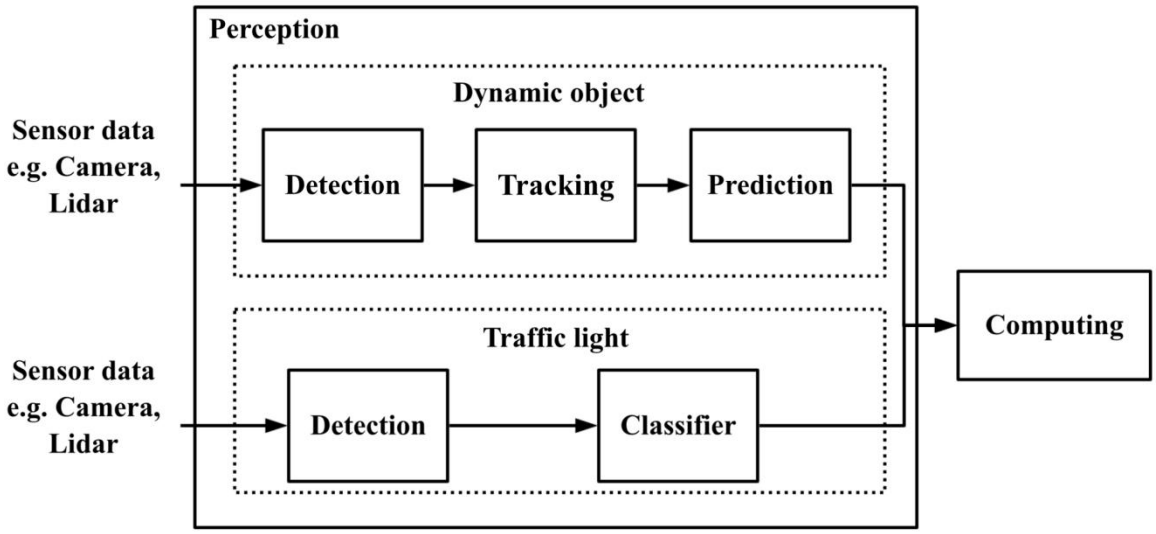


Figure 2.4 Activity of Autoware ADS Perception System

Figure 2.4 describes the perception activities of Autoware ADS, the data collected by sensors will also be pre-processed. The surrounding objects will be sensed and the objects are numbered. The state of objects such as acceleration, predicted path, position and other conditions will be analyzed and sent to the computing system. At the same time, it will also examine the current traffic light status and output the color of the traffic light.

It can be understood that autoware ADS also attaches great importance to the accuracy of sensor data, but similarly, it can not deal with the special situation of sensor failure. Based on the analysis of the above two mainstream ADS perception system. It can be found that the current perception system mainly relies on sensors to receive data, and there exists no other detection method. Therefore, the application of V2X method to perceive the physical environment is of great significance to the current ADS research.

2.3 The Safety Standards for ADS

With the deepening of the research on automated driving, ADS continues to be added with new functions. The development and integration of these functions greatly increase the complexity of the system. In order to judge the safety of the design process and ensure that the functional quality of ADS can meet the requirements of safe driving, a systematic analysis of all possible risks should be carried out, and corresponding countermeasures should be designed to reduce the risks to an acceptable level[25]. For this reason, ISO 26262 recommends to use hazard analysis and risk assessment (HARA) methods to identify the possible hazards specified by different system components, and strictly control every step of the system life cycle based on the method of system engineering, so as to ensure that the ADS can meet the requirements of safety objectives[26-27].

On the other hand, There are a large number of sensors and complex algorithms with advanced functions in ADS. Even if there is no problem in the system hardware, if the decision made by the processing algorithm is wrong based on the data sent from sensors, these may also cause ADS to make commands that violate the safety requirements[28]. Different from ISO 26262 functional safety, the safety of intended functionality (SOTIF) is aimed at the harm caused by the misjudgment coming from the system function restrictions, but all components of

the system are under normal operation. Figure 2.5 is the analysis of SOTIF and ISO 26262 in V model. SOTIF safety and ISO26262 functional safety are also accompanied by HARA, risk assessment, functional improvement, verification and validation[29]. As for SOTIF, it allows new functional improvement to the ADS first and then conducts V&V strategy. After that, the evaluation of the known use cases will be done for verification. Meanwhile, the system will be validated in unknown use cases to check its performance. This research focuses on adding new V2X communication function to the system and testing it in the known environment (traffic intersection), the safety zone is adopted to verify the expected functional safety of the proposed ADS.

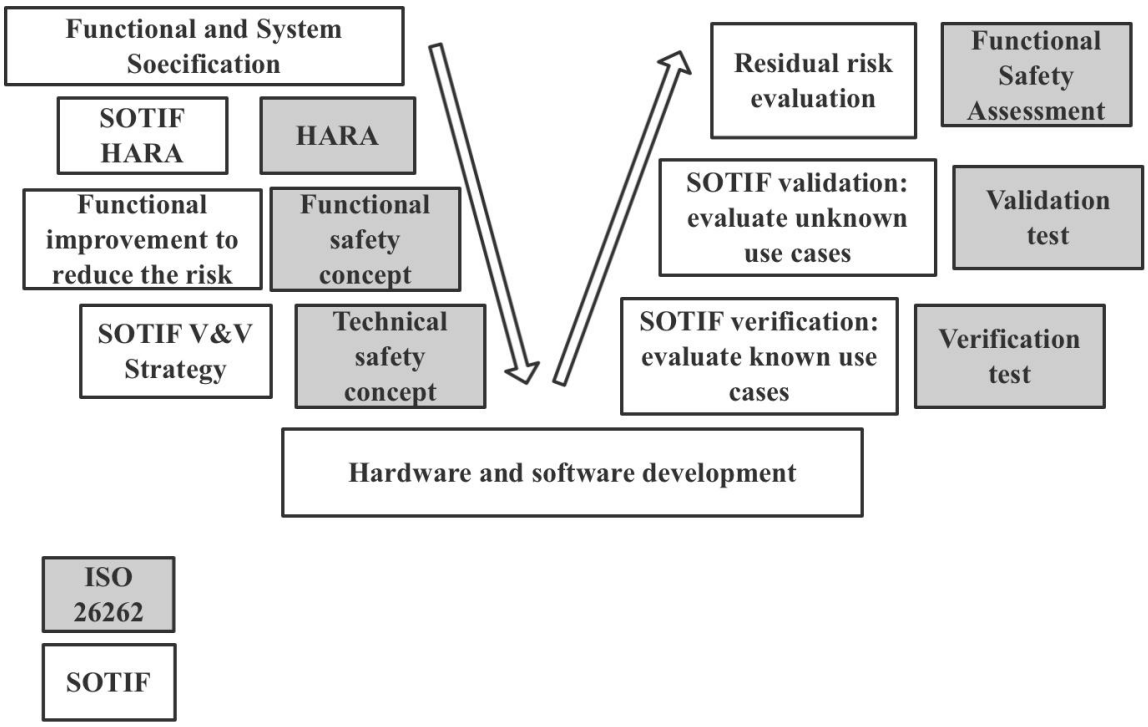


Figure 2.5 Analysis of SOTIF and ISO 26262 in V Model

3. ADS Model to Define the Architecture

Based on the study of the ADS in the second chapter, this chapter describes the automated driving context, derives the basic functions of ADS, and analyzes the contents of the external physical environment. The block definition diagrams are used to describe the components of the systems, and the activity diagrams are used to explain the behavior of the subsystems. In addition, the scenario of vehicle platooning passing through the intersection using ADS with V2X communication method is modeled as well.

3.1 ADS Operational Use Case

The automated driving level defined for the ADS in this research is level three, which means that the driver will still have the interactive relationship with ADS. In this process, the driver can directly or indirectly control the vehicle through ADS. The driver can choose to drive manually or monitor ADS to operate the vehicle. As shown in Figure 3.1, ADS is able to allow interaction with the driver.

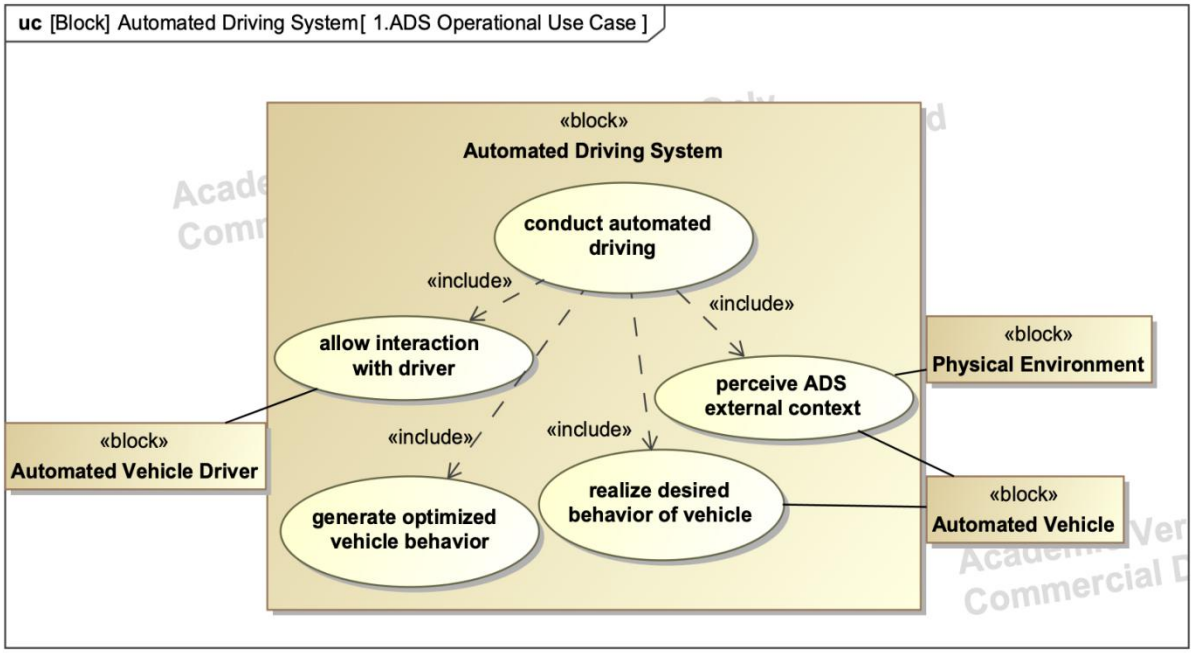


Figure 3.1 ADS Operational Use Case

Besides, ADS also has the functionalities of perceiving the external context, integrating and optimizing the vehicle behavior, and controlling the vehicle to realize the desired behavior.

3.2 ADS Operational Context

Figure 3.2 shows ADS operational context, referring to the classification made by Nishimura's research[30]. It includes physical environment, automated vehicle, driver and automated driving system. Physical environment includes traffic facilities, environmental conditions and traffic participants. Here ADS is separated from the automated vehicle, because for ADS, the ego vehicle exists outside the system boundary, but it is within the big picture of the context.

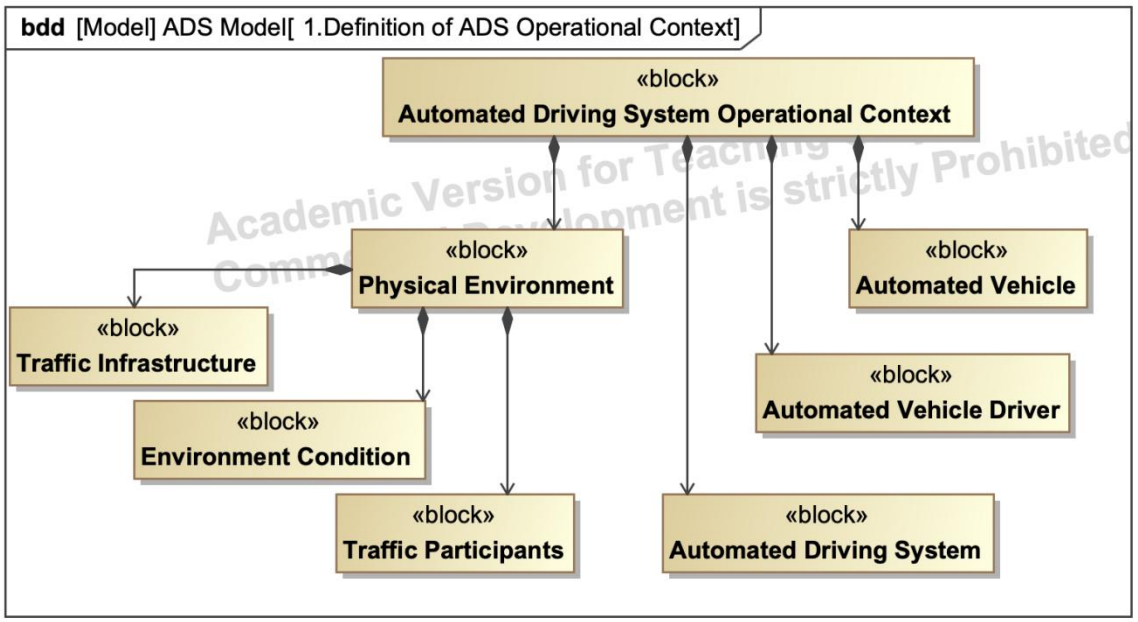


Figure 3.2 ADS Operational Context

3.3 Definition of Physical Environment

Figure 3.3 is a detailed decomposition of the physical environment mentioned in Figure 3.2, which refers to the research of Yun[31]. Environment condition contains the physical characteristics corresponding to the natural environment. For example, visibility, natural lighting and weather conditions. The weather includes wind, clouds, thunder, air temperature and precision. The various combination of these integrals can form the complex and changeable environments, including bad and good environmental situations. As for ADS, it needs to be able to work normally under a variety of situations.

On the other hand, the transportation infrastructure includes roads and intelligent transportation system besides the road furniture. It is related to V2X described in section 1.1.3, including traffic sign, traffic signal, pavement markings and V2I communication system. The traffic sign, traffic signal will guide the vehicle to follow the basic traffic regulations, which can be detected by sensors within the perception system. V2I communication system needs to communicate with ADS and transmit corresponding information to both the ego and other vehicles in order to help them understand the current environment situation and achieve safe driving.

In addition, the block definition diagram also decomposes traffic participants according to whether roads are used. None road users and road users of potential obstacles constitute the traffic participants. If the road is used, then it is classified in detail by whether it is human. Therefore, road users include the human road user and surround vehicles.

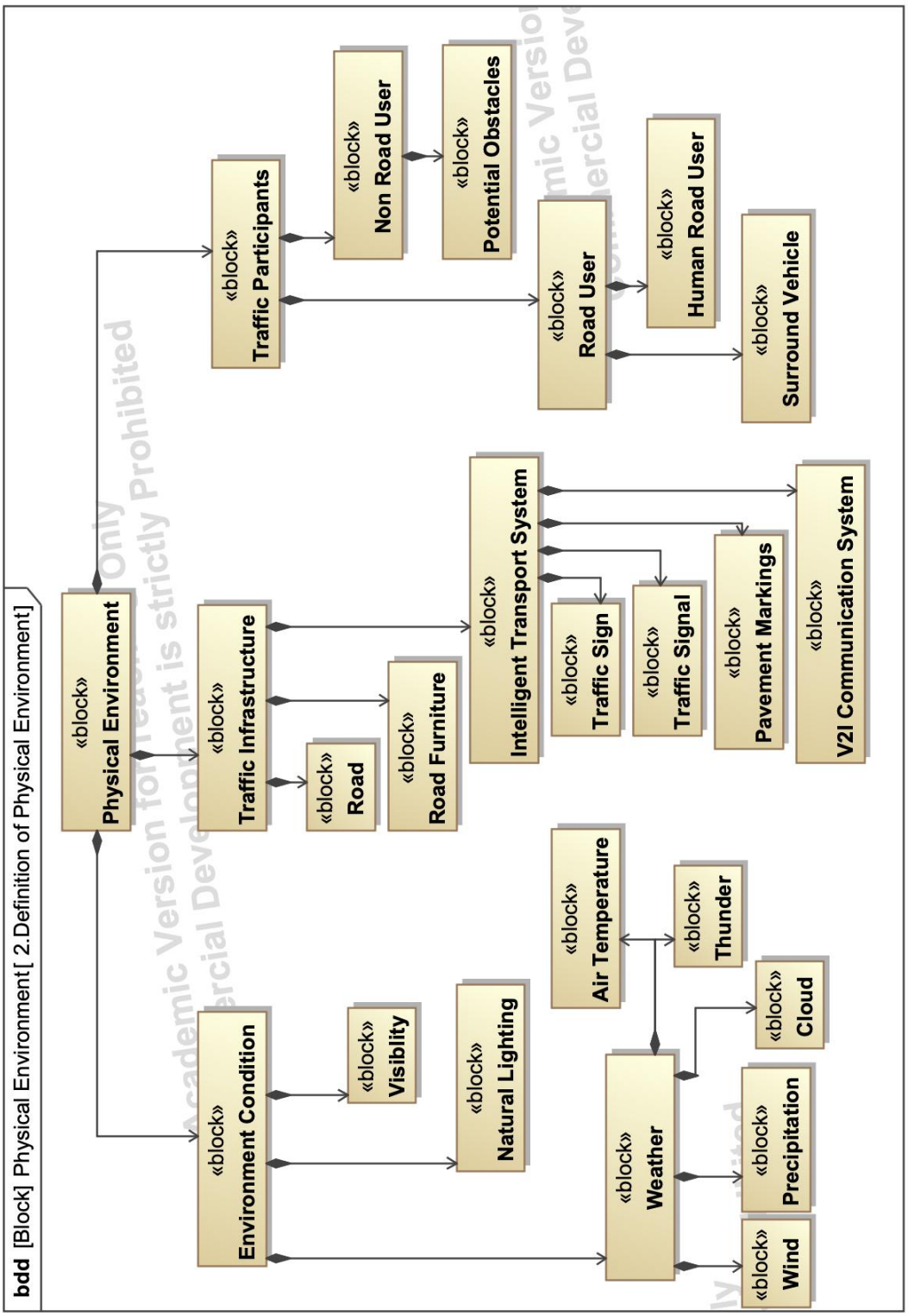


Figure 3.3 Definition of Physical Environment

3.4 Definition of ADS Components

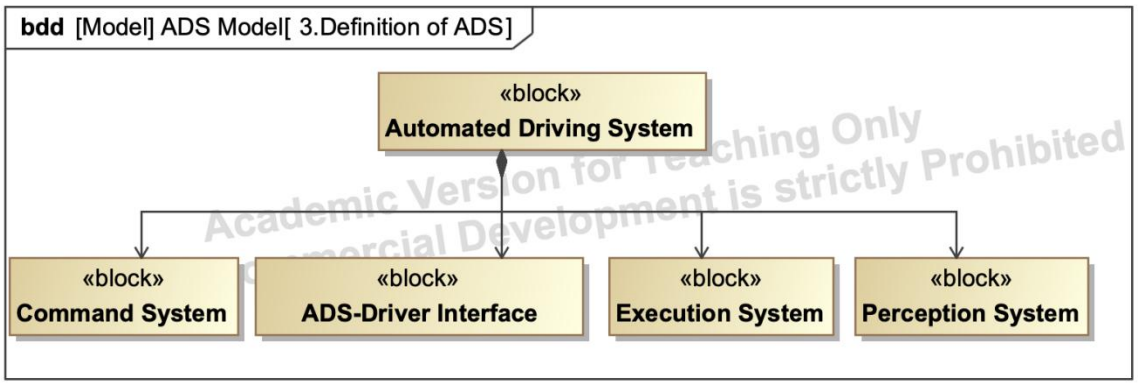


Figure 3.4 Definition of ADS

In light of the analysis of existing ADS in Chapter 2.2, Figure 3.4 describes the composition of ADS by block definition diagram. ADS consists of the command system, ADS-driver interface, execution system and perception system. ADS driver interface is a platform for interaction between the system and the driver, which can transmit information to the driver and obtain instructions from the driver as well. Perception system is responsible for obtaining ADS context from outside and conducting a preliminary analysis. Meanwhile, the perception system can obtain the driver's status to avoid fatigue driving. The execution system will execute the instructions issued by the command system to control the vehicle.

Figure 3.3 is the definition of the perception system and also the review of ADS System of Systems (SoS). The sensor unit, computing unit and V2X communication unit constitute the perception system. With reference to the system architecture of Apollo and Autoware ADS, the computing unit makes a preliminary refinement of the information obtained by the sensor in order to filter out the correct data. On the other hand, it also reduces the operating burden for the command system. The V2X communication unit includes a DSRC system for short-range communication and an LTE-V system suitable for medium and long-distance communication with low latency. The sensor unit comprises the GPS system, radar system, camera system and lidar system.

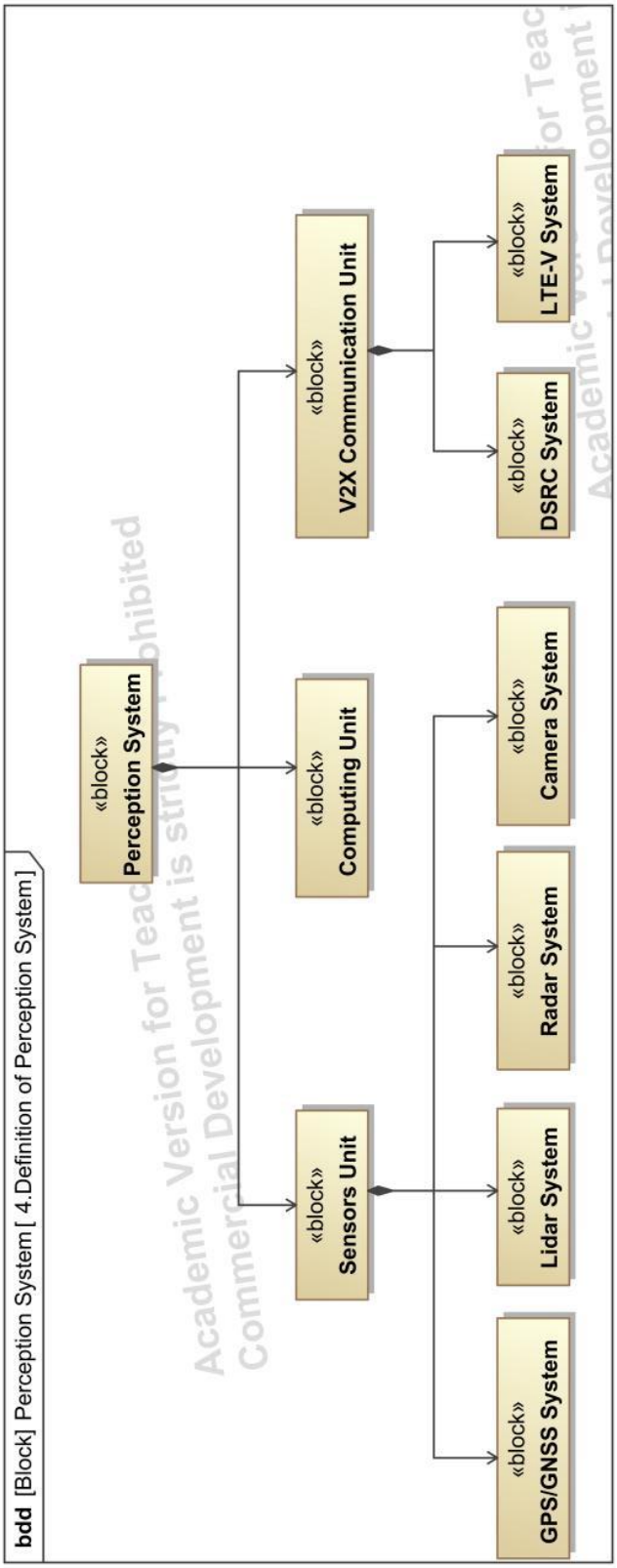


Figure 3.5 Definition of the Perception System

3.5 Activity Analysis of ADS Operation

According to the definition of ADS described in Figure 3.2 ~ 3.5, this section will model the operation activities of the perception system. The activity diagram of SysML is used to show the activities of ADS, the external environment and automated vehicle.

First, the driver will begin the entire activity by turning on the ADS. Subsequently, the driver will choose the current driving mode, that is, manual operation or using ADS to drive. Since the ADS in this study is at level 3, when using ADS for driving, the driver will monitor the driving activities to deal with the unexpected situations that may be encountered during driving process to ensure driving safety.

ADS will start working immediately after then. ADS will process the driver's command or operation, and will also analyze the driver's state to prevent fatigue driving. During the driving process, the information of the physical environment will be collected by ADS. The detailed procedures of the sensors collecting information are described in Figure 3.7. ADS will obtain traffic infrastructure state, environment condition and traffic participants state. At the same time, V2X communication will also be carried out, and communication information will be output to the physical environment. After ADS completes generating the operation command, it will communicate the vehicle parameters and driving conditions to the driver through the ADS-driver interface to make sure that driver see the whole picture of the automated driving, and then control the vehicle to perform the corresponding operations.

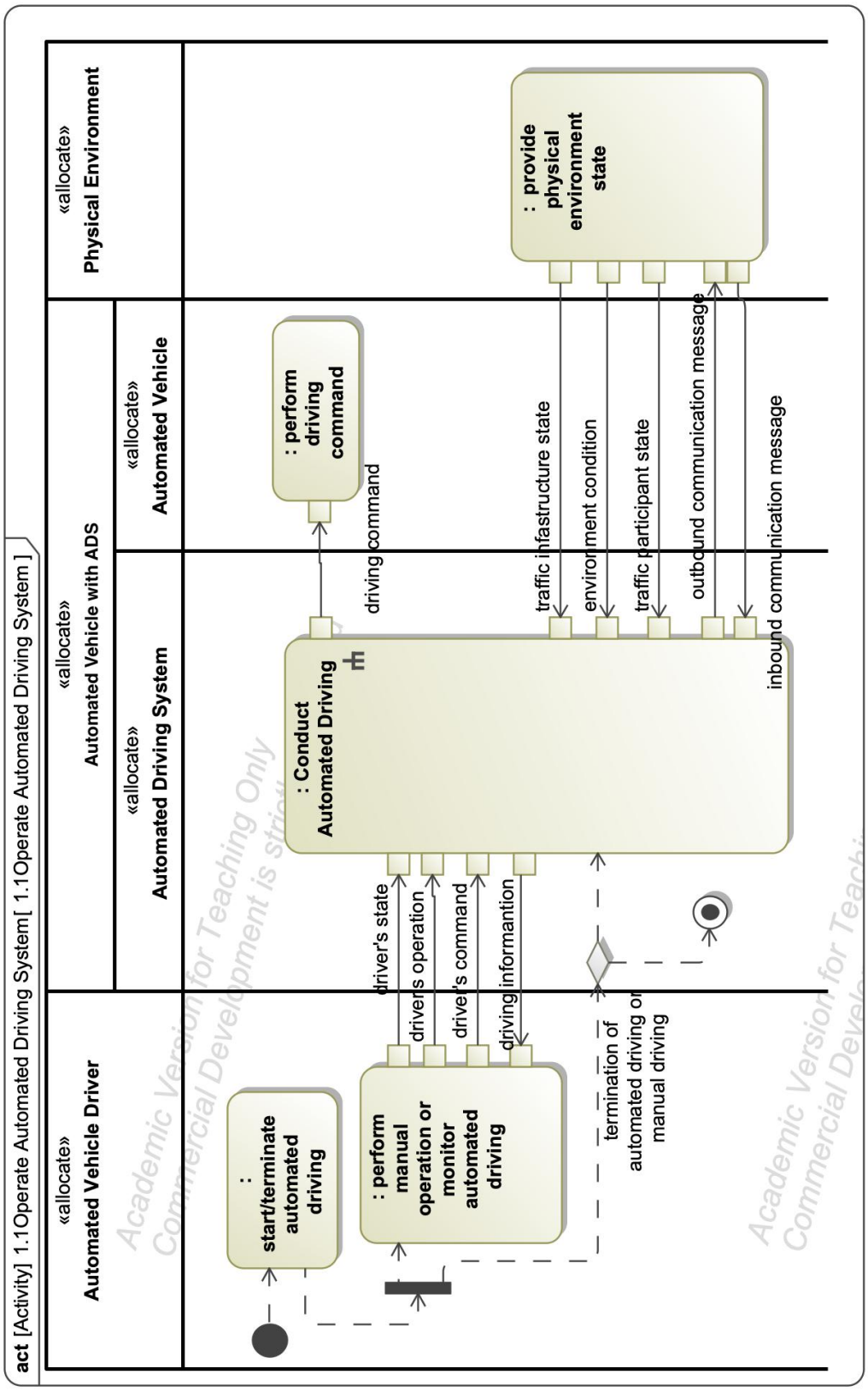


Figure 3.6 Activity Diagram of ADS Operation - 1

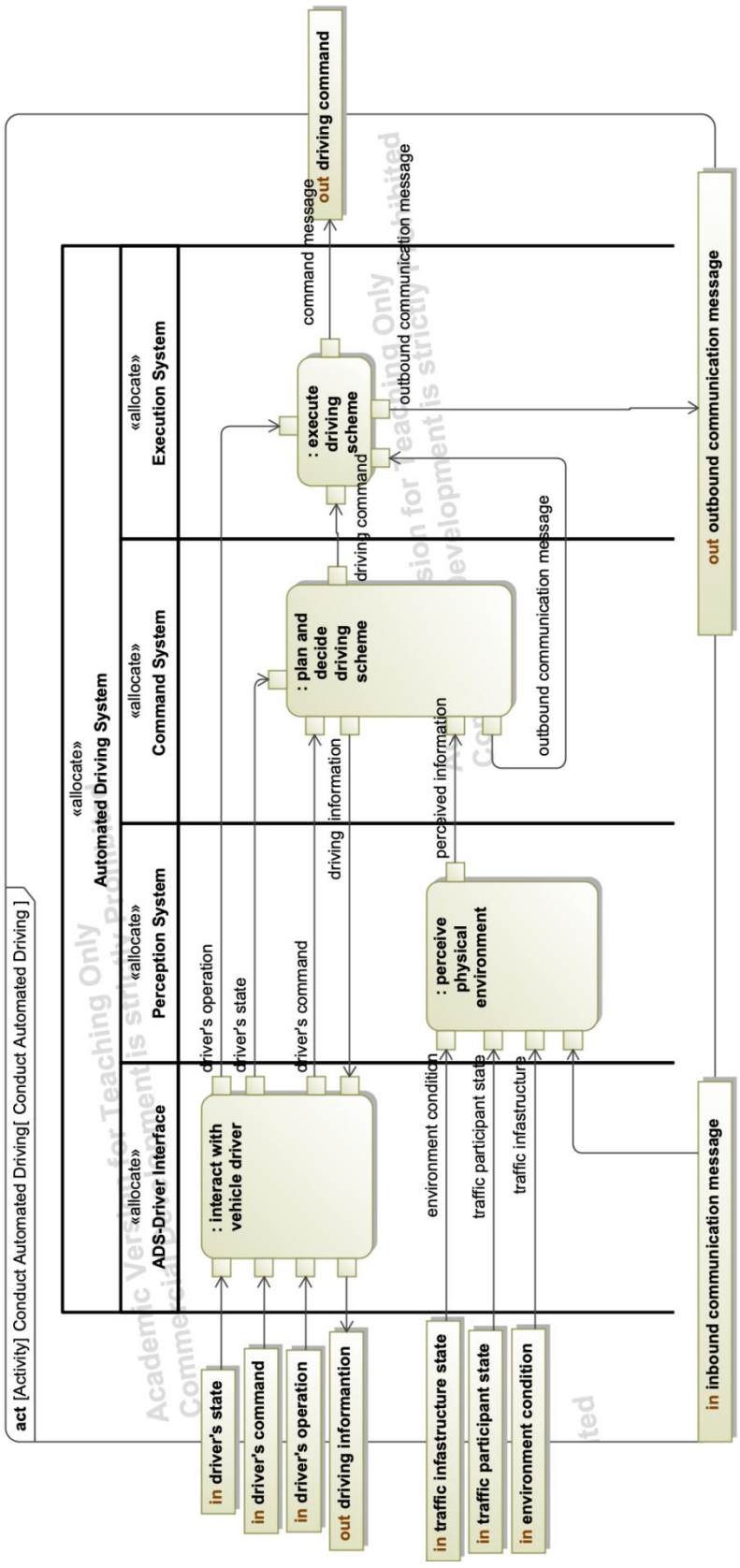


Figure 3.7 Activity Diagram of ADS Operation - 2

Figure 3.7 is based on Figure 3.6 to further describe the internal activities of perception system. It respectively decomposing the activities of four components of ADS. The ADS-driver interface will interact with the driver to collect the driver's status, operations and commands. It transmits these status and command information to the command system. If the obtained information is the driver's operation, it will directly transmit the command information to the execution system to operate the vehicle. The perception system will detect traffic infrastructure state, traffic participants state, environment condition, and conduct V2X communication to obtain inbound information. After preliminary processing internally, the information is sent to the command system. The command system will make the driving command after getting all needed data, and then send the driving command and the message to be broadcast to the physical environment to the execution system for execution. The execution system will pass the outbound message to the environment.

The internal activities of the perception system are modeled in Figure 3.8 and 3.9. It can be seen from Figure 3.9 that the environment condition, traffic infrastructure state and traffic participates state will be detected and monitored by the sensor unit, and then the information will be sent to the computing unit. Similarly, as shown in Figure 3.9, the V2X communication unit will perform communication operations to accept LTE-V signals. Then it will analyze the received inbound communication message to get the data of time, location, speed and ID. Computing unit will analyze the information transmitted by the other two units. After that, the safety zone information will be calculated and passed to the command system for making further decision, which will help to determine the current safety level of ego vehicle or other objects.

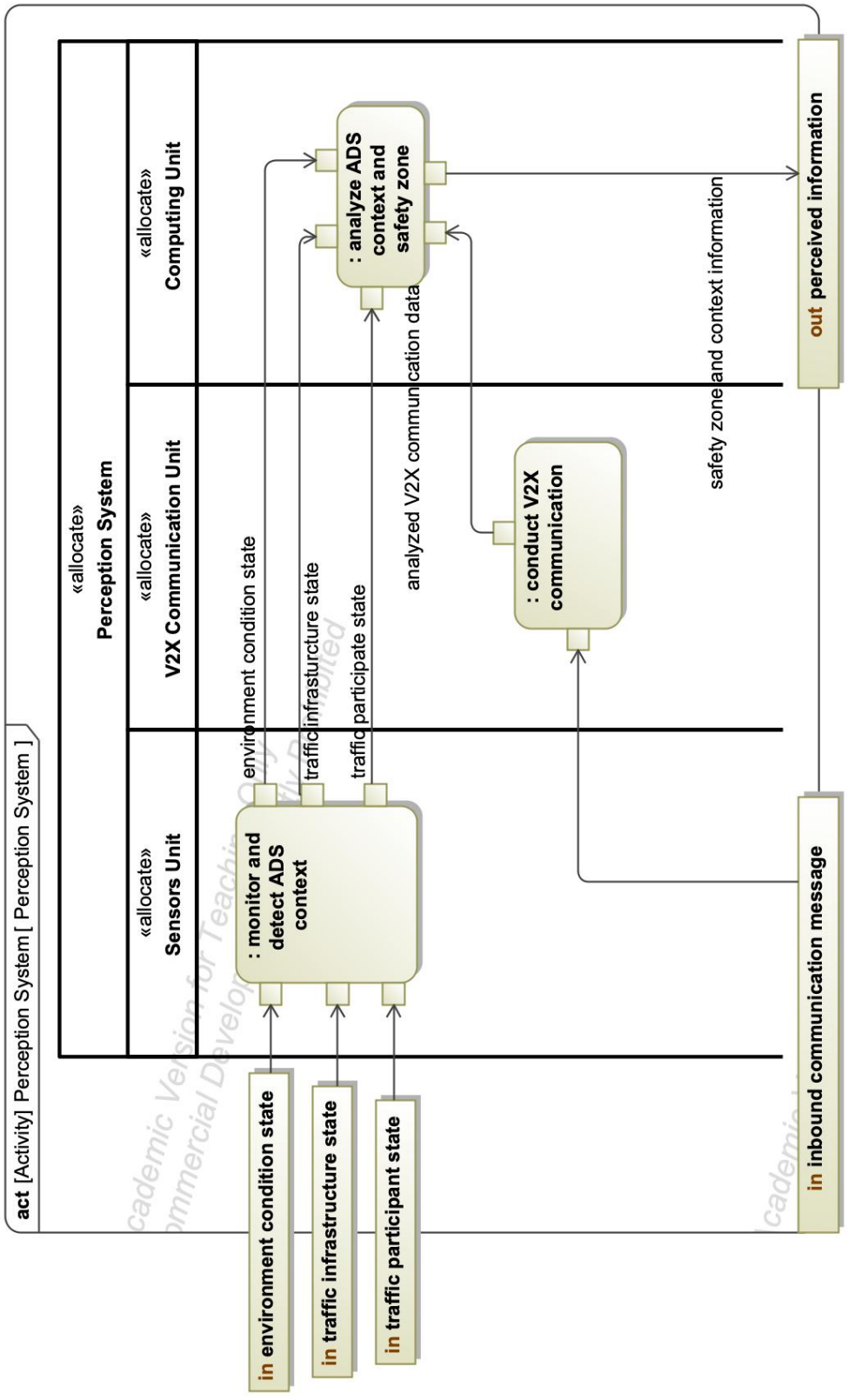


Figure 3.8 Activity Diagram of Perception System

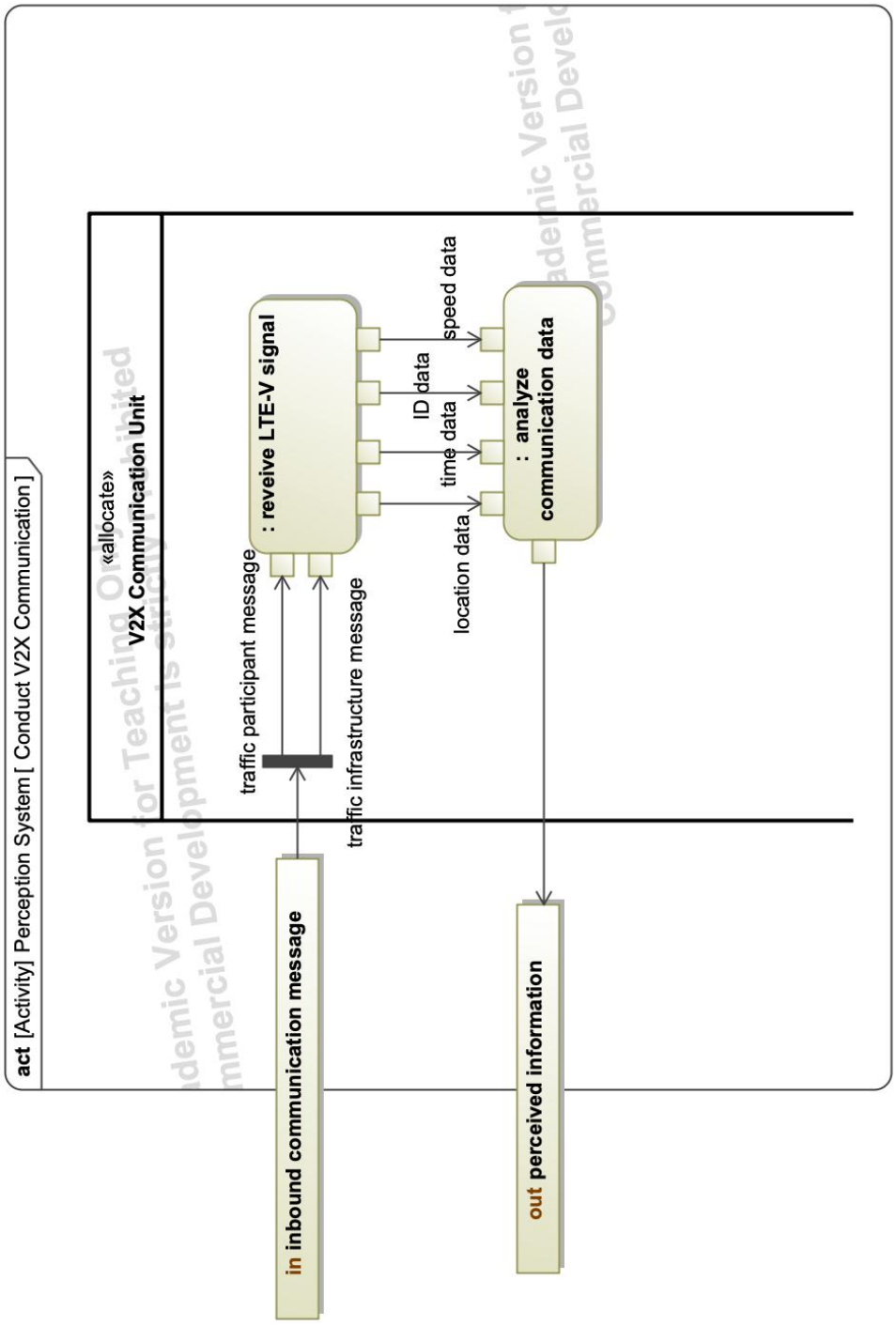


Figure 3.9 Activity Diagram of V2X Communication Unit

3.6 Activity Analysis of Conducting V2X Communication at the Intersection

According to the background described in Chapter 1.1, the happening rate at traffic intersections in various countries is very high. Based on this situation, it is also needed to verify the functionality of the ADS through the verification of traffic intersection scenario. Figure 3.10 depicts the situation where the vehicle uses ADS V2X function to cross the intersection and interact with the physical environment (including other vehicles and traffic lights). Specific vehicle parameters and experimental conditions will be explained in detail in Chapter 4. In this experimental scenario, ego vehicle is the first car in the platooning. As mentioned in Chapter 1, platooning is an important form of V2V communication. The ADS will try to ensure the overall safety of platooning.

The ADS perception system will first communicate with other vehicles to obtain data, and then decompose the data into the vehicle's ID, speed, vehicle length and position. After that, it will pass the information to the computing unit for analysis. When the ego vehicle reaches the communication distance, V2X communication unit will communicate with the infrastructure to obtain the position, time and traffic light signal information of the traffic lights, so as to analyze the safety distance. When the time is insufficient, ADS will calculate the ID and deceleration of the potential head vehicle to ensure the safety of the potential head vehicle and its followers. This part of the data will be transmitted through the execution system. When the potential head vehicle reaches the braking distance, the deceleration operation will begin. The ego vehicle itself will pass through the intersection under the control of ADS.

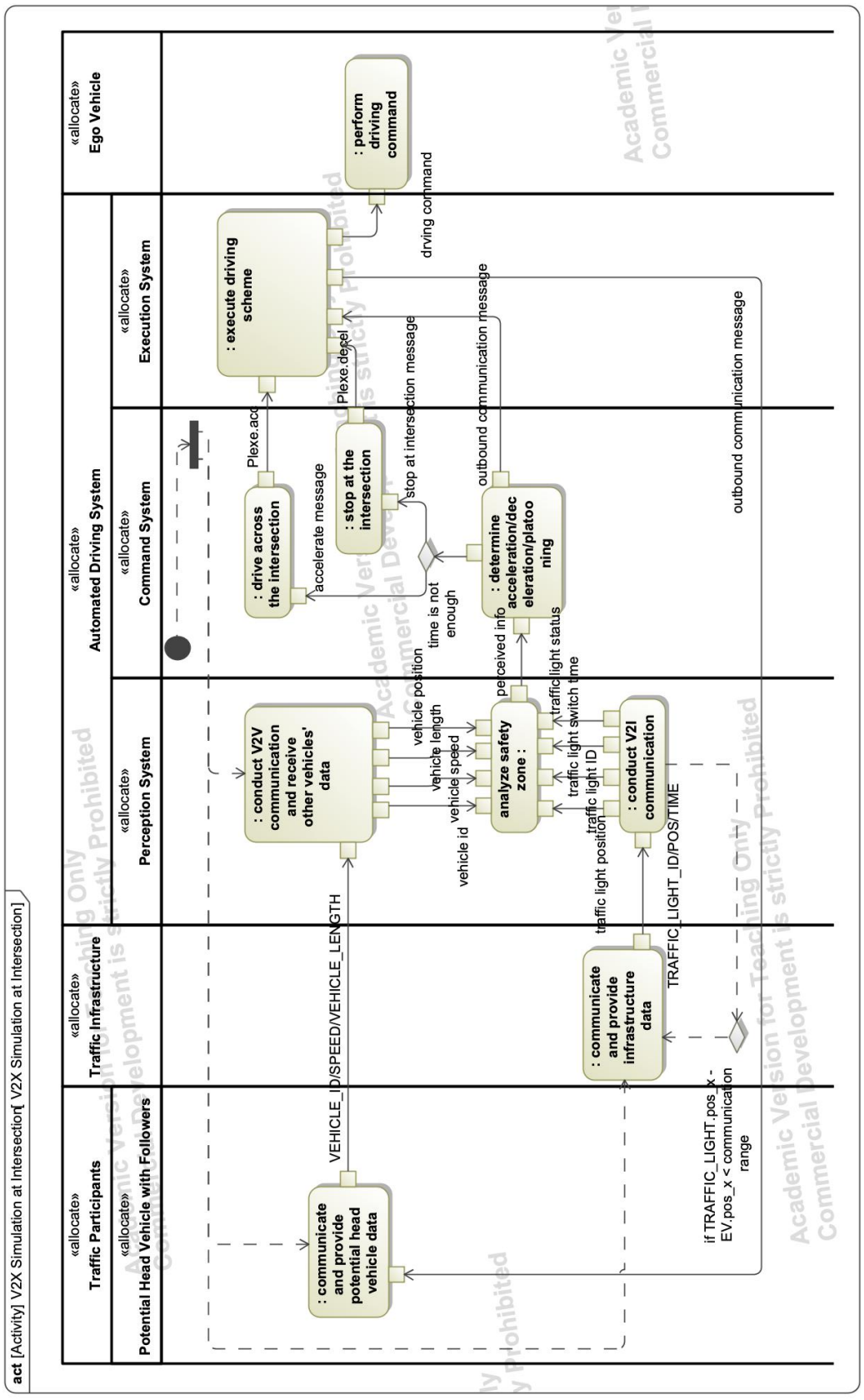


Figure 3.10 Activity Diagram of Conducting V2X Communication at the Intersection

3.7 Sequence Analysis of Conducting V2X Communication at the Intersection

Based on the model situation described in Figure 3.10, as described on Figure 3.11, V2X communication at the traffic intersection can be divided into 4 different substages. The first substage is "before passing the intersection". The second substage is "checking communication range", and the remaining substages are "preparing brake" and "passing the intersection". Through this classification, the state and specific operation of ego vehicle are clearly decomposed.

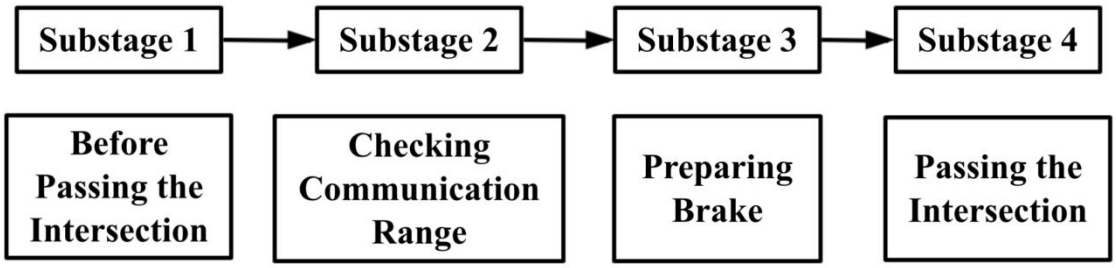


Figure 3.11 Substage Decomposition of Conducting V2X Communication at the Intersection

As shown in the sequence diagram of Figure 3.12 and Figure 3.13, all activities are carried out among traffic light, ego vehicle and potential head vehicle. In the "before passing the intersection" substage, there is V2X communication mainly among the ego vehicle and the potential head vehicle. The communication among them is parallel to each other and is carried out simultaneously. In order to focus on the main communication process, the communication between ego vehicle and other vehicles is omitted here. During the "checking communication range" substage, in addition to interacting with the potential head vehicle, the ego vehicle also communicates with traffic light after reaching the communication range to determine and locate the ID information of the potential head vehicle. Thus in the third substage it can be able to accurately communicate the braking command with

designated vehicles. In the final "passing the intersection" substage, the ego vehicle obtained the current traffic light status through V2I communication. When it is available, the potential head vehicle will receive a command to guide the potential head vehicle platooning to drive through the intersection.

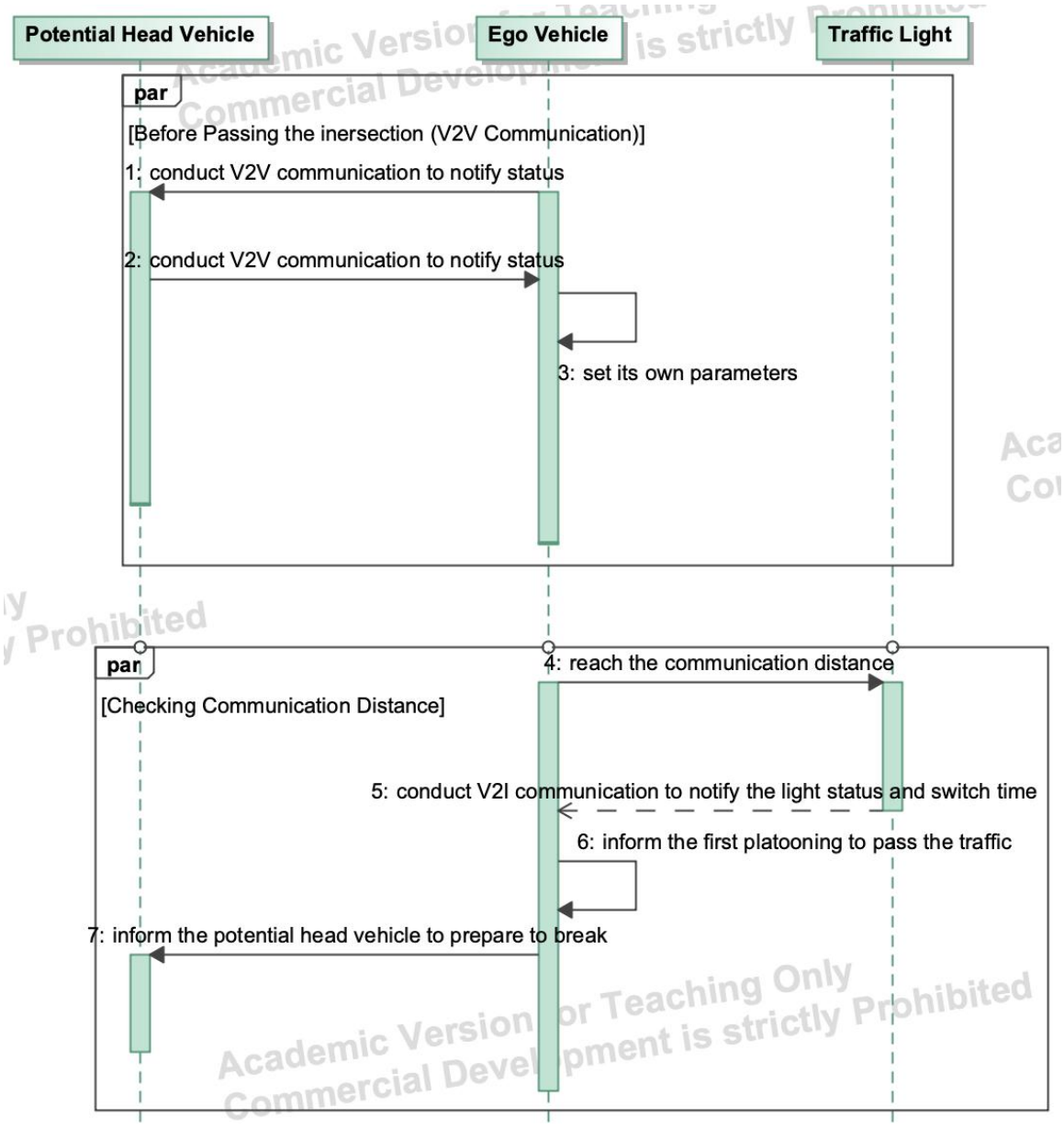


Figure 3.12 Sequence Diagram of Conducting V2X Communication at the Intersection - 1

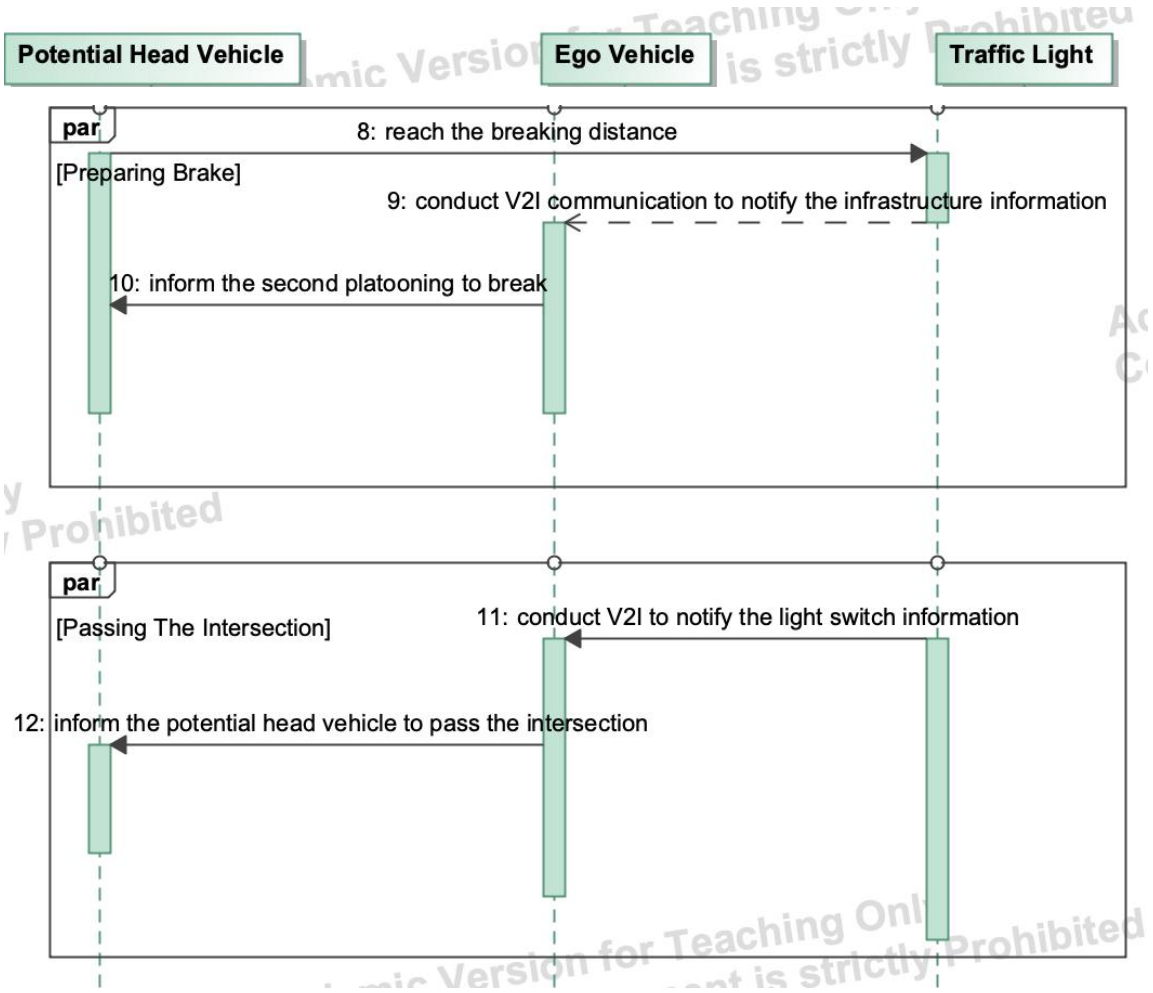


Figure 3.13 Sequence Diagram of Conducting V2X Communication at the Intersection - 2

3.8 Parameter Analysis of Conducting V2X Communication at the Intersection

This chapter is a summary of the parameters of the V2X simulation implemented at the intersection based on the aforementioned chapters 3.6 to 3.5. Figure 3.13 lists the structure and constraints. Simulation at the intersection includes automated driving system, safety index, ego vehicle, physical environment. Physical environment includes traffic light and potential head vehicle with followers. Safety index is the ultimate safety zone, which is used to measure the safety distance standard. The value is defined as 0.3 ~ 0.5 vehicle lengths, the unit is meters. There are 5 formulas in the picture:

(1) The formula for conduct V2V communication and receive other vehicles' data:

- The Parameters are:

PV_pos.x (Potential head vehicle position)

PV_ID (Potential head vehicle ID)

PV_V (Speed of potential head vehicle)

EV_pos.x (Ego vehicle position)

EV_ID (Ego vehicle ID)

EV_V (Speed of ego vehicle)

- The constraint is:

V2V=(PV_pos.x, PV_ID, PV_V, EV_pos.x, EV_ID, EV_V)

This constraint mainly expresses the content of V2V communication between vehicles, covering the ID, speed and location information. As there is a specific introduction of information exchange between vehicles by programming language, it is not a actual formula but the expression of the relationship here. The formula is explained in Chapter 4.

(2) The formula for conduct V2I communication

- The parameters are:

TL__Time (The duration of each light of traffic light)

TL_PHASE (Current color of traffic lights)

TL_pos.x (Location of traffic lights)

TL_ID (Traffic light ID)

PV_pos.x (Potential head vehicle position)

PV_ID (Potential head vehicle ID)

PV_V (Speed of potential head vehicle)

EV_pos.x (Ego vehicle position)

EV_ID (Ego vehicle ID)

EV_V (Speed of ego vehicle)

- The constraint is:

V2I=(PV_pos.x, PV_ID, PV_V, EV_pos.x, EV_ID, EV_V, TL_Time,

TL_PHASE, TL, pos.x_TL_ID)

This constraint mainly describes the content of V2I communication between traffic light and vehicles, covering the ID, speed, location, time and phase information. As there is a specific way of expressing information exchange between vehicles in the programming language, it is not a specific formula but the expression of the relationship here. The formula is explained in Chapter 4.

(3) The formula for drive across the Intersection

- The parameters are:

PV_ID (Potential head vehicle ID)

PV_V (Speed of potential head vehicle)

TL__Time (The duration of each light of traffic light)

TL_PHASE (Current color of traffic lights)

- The constraint is:

DAI=(PV_pos.x, PV_ID, PV_V, TL_Time, ADP)

This constraint mainly illustrates the process of accelerating the potential head

vehicle after waiting for the green light.

(4) The formula for determine acceleration/deceleration/platooning

- The parameters are:

PV_pos.x (Potential head vehicle position)

PV_ID (Potential head vehicle ID)

PV_V (Speed of potential head vehicle)

EV_pos.x (Ego vehicle position)

EV_ID (Ego vehicle ID)

EV_V (Speed of ego vehicle)

- The constraint is:

ADP=(PV_pos.x, PV_ID, PV_V, TL_Time, TL_PHASE, TL_pos.x)

This constraint shows the decision of ADS command system to accelerate or decelerate or maintain platooning. Only the relationship is expressed here, the detailed acceleration formula is explained in Chapter 4.

(5) The formula for stop at the intersection

- The parameters are:

Sz (Safety Zone Index)

EV_pos.x (Ego vehicle position)

EV_V (Speed of ego vehicle)

TL__Time (The duration of each light of traffic light)

TL_pos.x (Location of traffic lights)

TL_ID (Traffic light ID)

- The constraint is:

SAI=(Sz, EV_pos.x, EV_V, EV_V, TL_ID)

This constraint illustrates the decision-making process of ADS. The whole process is related to the above parameters. ADS will determine whether the Potential head vehicle (PHV) can pass through the intersection.

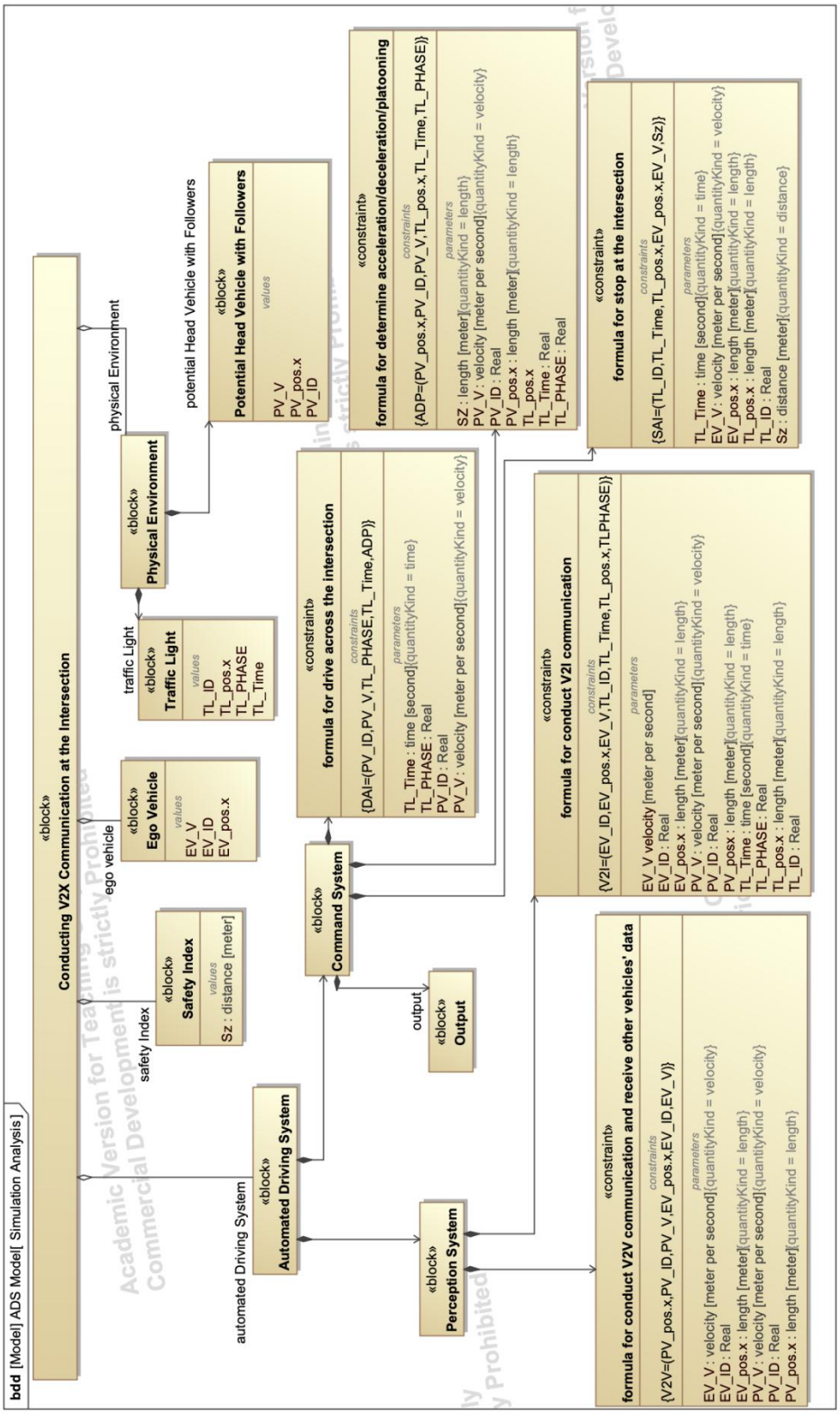


Figure 3.14 Structure and Constraints of Conducting V2X at the Intersection

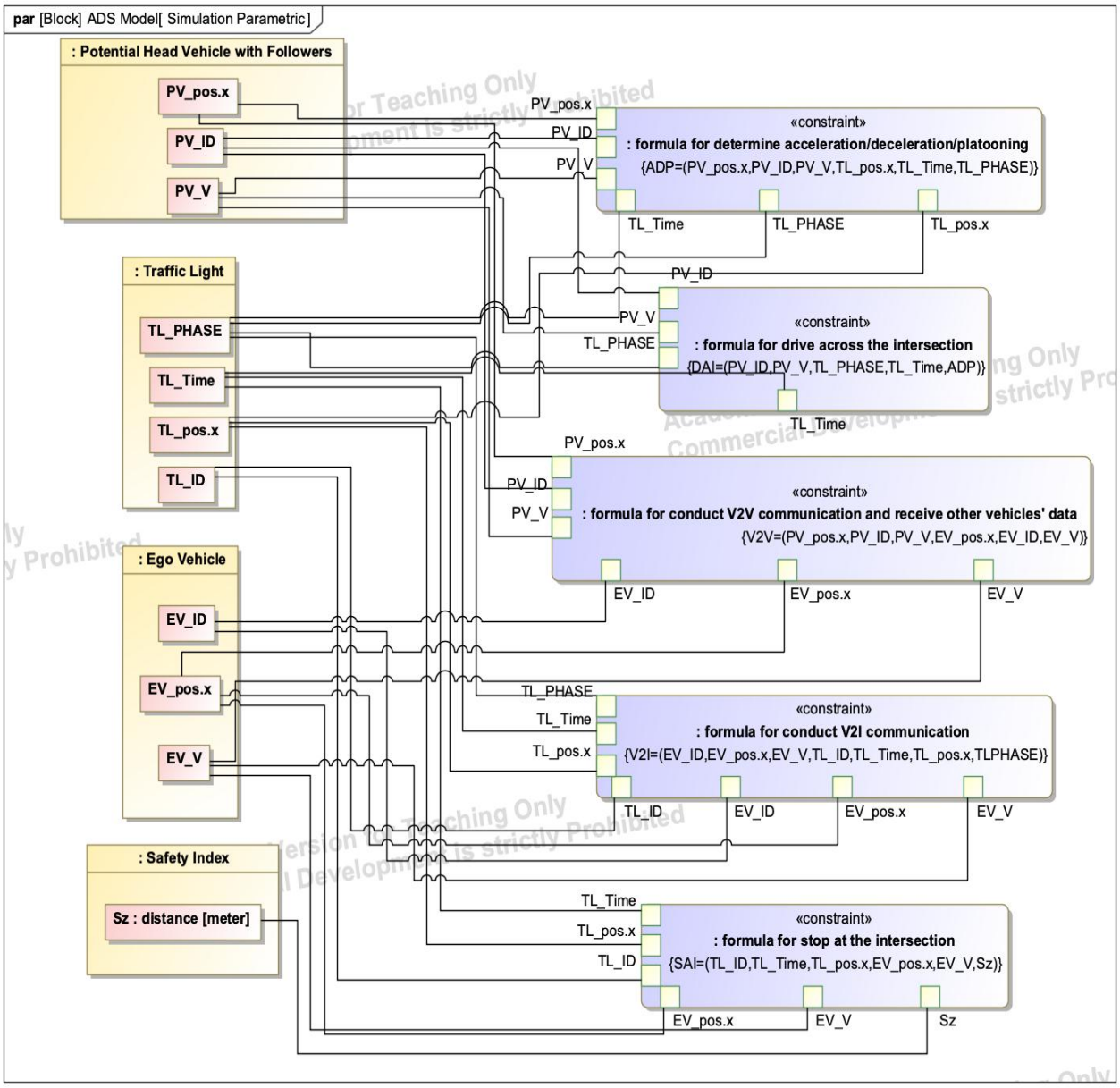


Figure 3.15 Parameters for Conducting V2X Communication at the Intersection

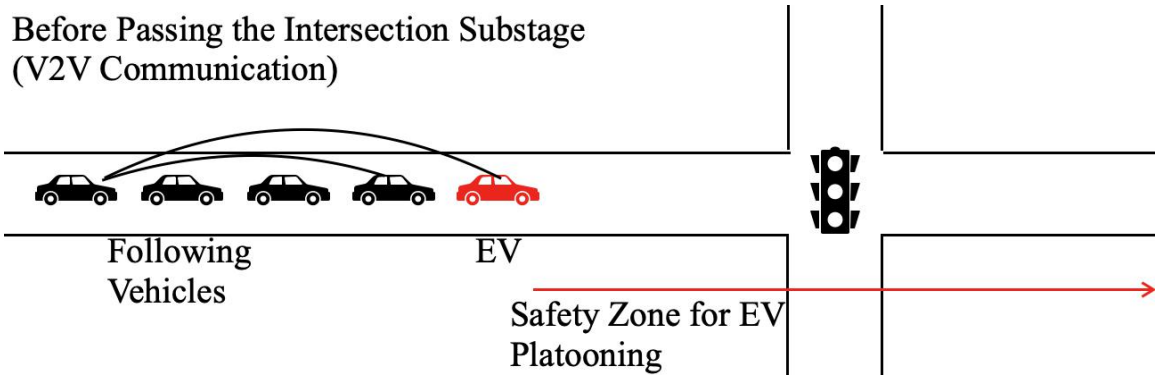
4. Simulation Model

This chapter is based on the basic situation shown in Figure 3.10 ~ 3.14 to simulate the implementation of ADS at the intersection. According to the background of section 2.3, the concept of safety zone is introduced to check whether ADS can achieve the expected performance. The simulation is based on SUMO simulator and Python 2.

4.1 Simulation Scenario

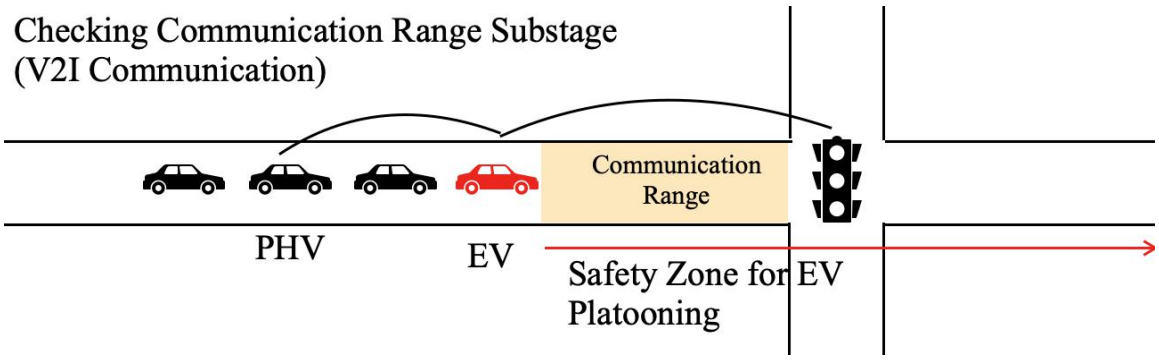
First, the purpose of the simulation is to verify the ADS model established in chapter 3. ADS is responsible for ensuring the safety of vehicles. The desired scenario for the experiment needs to include the application of V2X communication. According to the background of the first chapter regarding the frequent occurrence of traffic intersection accidents, ADS needs to prove the feasibility of the system itself by testing in such a scenario. On the other hand, ADS can not only help its own to ensure driving safety through communication, but also guide other vehicles to avoid risks through its command generation. Based on the above factors, the details of the scenario settings are as follows: As shown in the Figure 4.1 ~ 4.2, 8 vehicles will pass through the intersection. The ego vehicle (EV) is at the front of the vehicle. The EV is equipped with ADS to control the itself. Other vehicles are equipped with the communication system as well as the auto-braking system, but does not include the ADS system. The traffic light at the intersection is also equipped with a communication system that can send out information such as the status, time and distance. Figure (a) describes that in the initial stage of the simulation, the EV will conduct V2V communication with subsequent vehicles, interact with each other to exchange ID information, distance and other information, and set the driving mode to the vehicle platooning. In this mode, vehicles in the platooning will maintain the same driving form and

follow the instructions output by the EV. ADS will protect the overall safety of the platooning. When the EV reaches the communication distance in Figure (b), the EV will communicate with the traffic light, then the traffic lights will send parameters to the EV. ADS will conduct calculations in real time to determine which vehicles in the platooning cannot pass through the intersection. Among them, the vehicles that cannot pass the traffic lights will be classified as the second platooning by the ADS system. Its first vehicle is named Potential Head Vehicle (PHV). ADS will communicate with the PHV to inform the PHV and vehicles behind it to decelerate in advance. PHV will perform the deceleration operation after reaching the braking distance as Figure (c). In Figure (d), PHV platooning will accelerate and pass the intersection. During all processes, ADS will calculate the safety distance in real time to ensure the safety of all platooning.

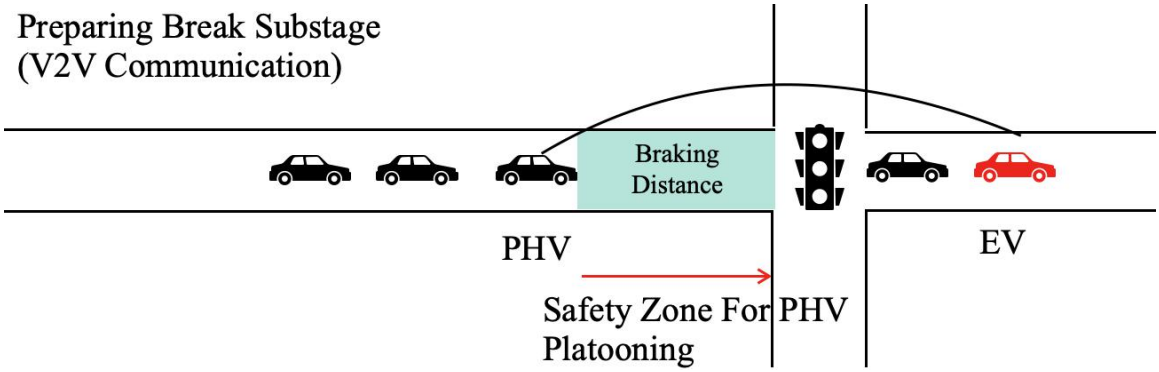


(a) Before Passing the Intersection Substage

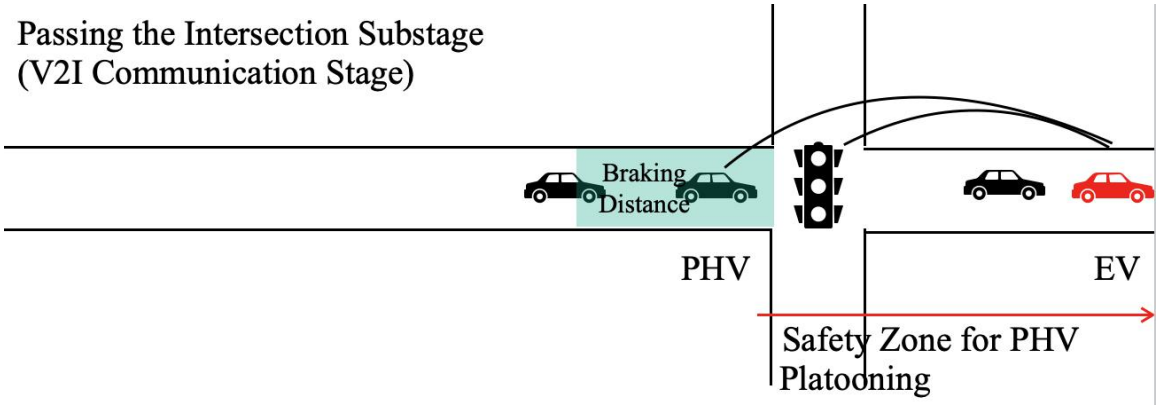
Figure 4.1 Substages Analysis of Conducting V2X Communication at the Intersection - 1



(b) Checking Communication Range Substage



(c) Preparing Breaking Substage



(d) Passing the Intersection Substage

Figure 4.2 Substages Analysis of Conducting V2X Communication at the Intersection - 2

4.2 Safety Zone in the Simulation

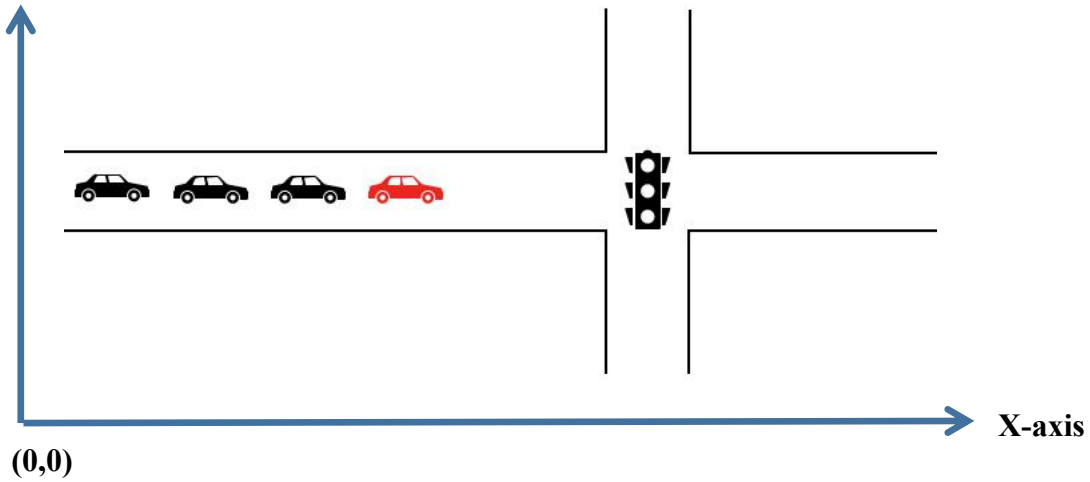


Figure 4.3 Definition of Map Axis

Based on the introduction of automated driving safety zone in Chapter 2, the safety zone of this experiment is defined. Since there is only a single lane in this experiment, so **the safety distance is used to represent the safety zone**. As shown in the Figure 4.3, in the simulation, the lower left corner of the map is marked as the origin of coordinates. The horizontal axis is marked as the X-axis. The position of each object is the absolute distance from the origin of the coordinates on the X-axis. At the beginning of the experiment, the time is recorded as 0 second.

The classification and formula of safety zone based on scenarios are as follows:

- As shown in the Figure 4.4, if the light is green and the vehicle platoon can pass the intersection, then the safety zone is:

  Safety Zone = Safety Distance = the length of the road - the head vehicle's position in x axis

$$S_{sz} = S_{sd} = S_{lr} - S_{hv} \quad (1)$$

- If the vehicle platooning can not pass the intersection and the light is still green:

  ① As shown in Figure 4.5 (b), when it didn't reach the braking distance, the safety zone is:

  Safety Zone = Safety Distance = the length of the road - the head vehicle's position in x axis

$$S_{sz} = S_{sd} = S_{lr} - S_{hv} \tag{2}$$

  ② As shown in Figure 4.5 (c), when it reaches the braking distance, the safety zone is:

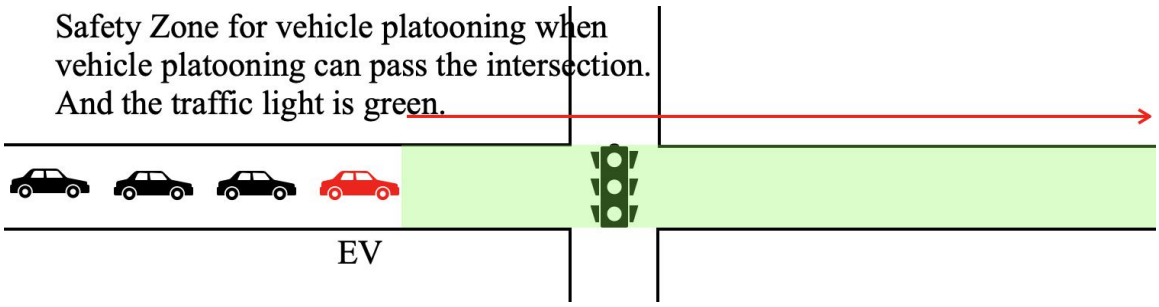
  Safety Zone = Safety Distance = the stopping line's position in x axis - the head vehicle's position in x axis

$$S_{sz} = S_{sd} = S_{sl} - S_{hv} \tag{3}$$

- As shown in Figure 4.5 (d), if the vehicle platooning can not pass the intersection and the light is yellow or red:

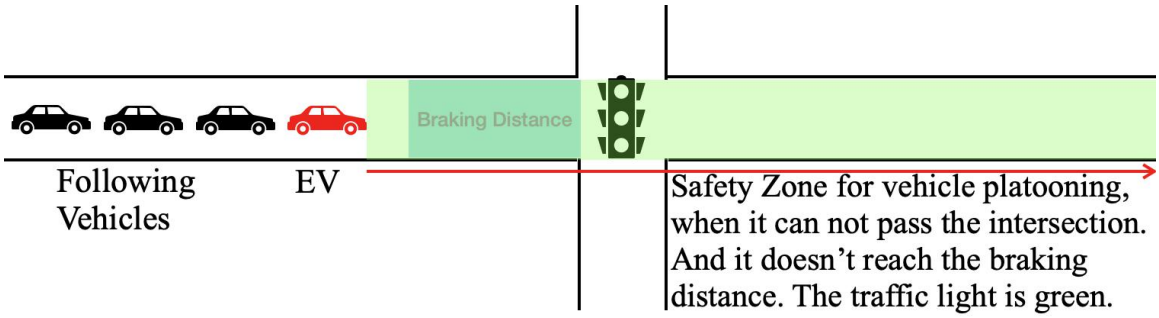
  Safety Zone = Safety Distance = the stopping line's position in x axis - the head vehicle's position in x axis

$$S_{sz} = S_{sd} = S_{sl} - S_{hv} \tag{4}$$

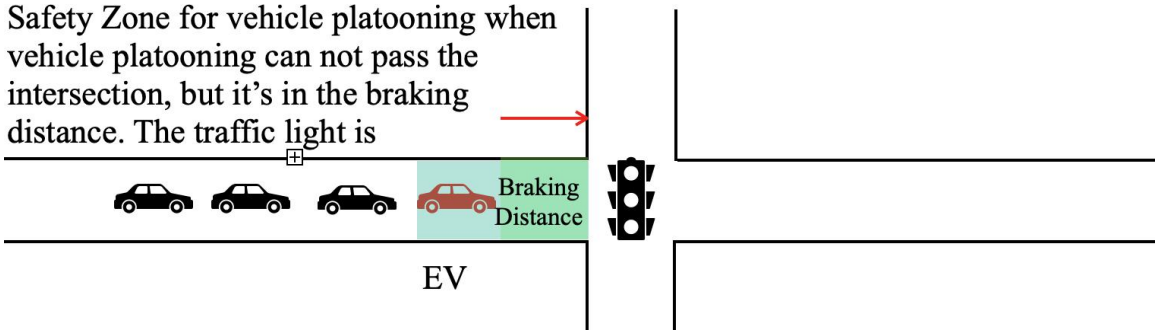


(a)

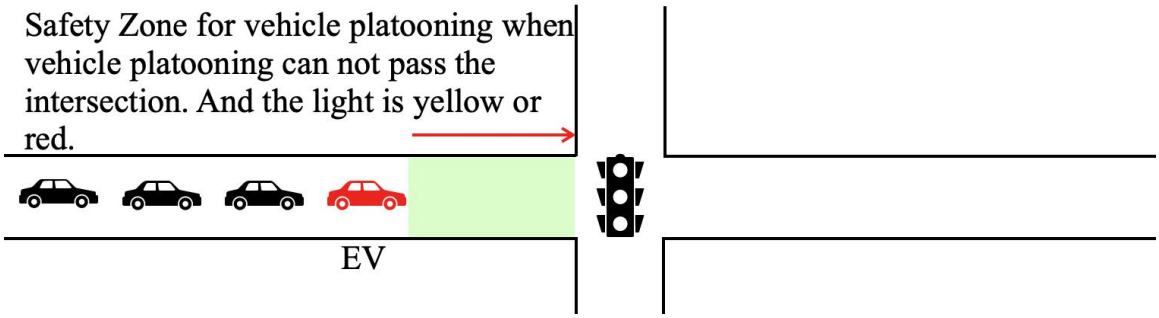
Figure 4.4 Definition of Safety Zone in Simulation - 1



(b)



(c)



(d)

Figure 4.5 Definition of Safety Zone in the Simulation - 2

4.3 Simulation Environment

4.3.1 Simulator Introduction

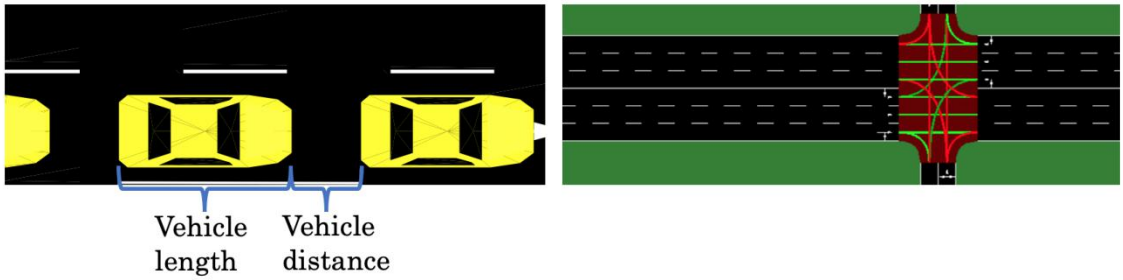


Figure 4.6 Vehicle and Map Model Established for Simulation

As this experiment needs to simulate the communication situation, SUMO simulator is adopted as the experimental platform. SUMO is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks. It allows for inter-modal simulation with a broad set of tools for scenario creation. In this experiment, the situation of communication delay is also simulated as 1 ms. The vehicle model uses the API interface of the sumo vehicle library. Figure 4.6 shows the map model established in this simulator. The intersection is in the middle of the road. The followings are the basic data for vehicle and map model.

Table 4.1 List of Vehicle and Map data

Vehicle Model	Value	Map Model	Value
Vehicle length	4 m	Road length	1000 m
Vehicle speed	30 ~ 40 km/h	Road width	3 m
Wheels friction	0.7	Green light period	10 s
Engine efficiency	0.9	Red light period	7 s
Engine action time	0.1 s	Yellow light period	3 s
Brakes action time	0.2 s		

4.3.2 Programming Environment

The code for simulation is programmed based on Python 2. The editor used is Spyder 2.0, and the virtual environment of the program is based on various python function libraries under Anaconda. The hardware environment is described as follows:

- OS: Windows 10.
- Processor: 2 GHz.
- Memory: 4 GB RAM.
- Graphics: 512 MB.
- DirectX: Version 11.
- Network: None.
- Storage: 60 GB system space available.

4.4 Simulation Parameters

This section will mainly describe the specific parameter settings.

Since the experiment is simulated in the intersection scenario, the vehicle speed is the normal urban speed, and the value of V_0 is 30 km/h ~ 40 km/h. The vehicle length S_l is 4 meters, and the distance between vehicles is related to the speed, as shown in Formula 5:

$$S_l = 1.5 \cdot V_0 + 2 \tag{5}$$

The braking distance is related to the initial speed as well. Considering the safety of the vehicle during driving, the braking distance is 3 times the current speed. At the same time, because the vehicle engine requires a certain reaction time to start braking, 0.2 seconds is added as the time compensation. The communication

distance is the braking distance plus the distance that the vehicle can travel in one second. The two formulas are defined as follows:

$$S_{bd} = \frac{V_0}{3.6}(3 + 0.2) \tag{6}$$

$$S_{cr} = S_{bd} + V_0 \tag{7}$$

As shown in Figure 4.1 (b), (c), the EV also needs to locate the PHV for further communication. The process of positioning is to calculate the vehicle ID. The calculation formula based on programming language is as follows:

$$T_a = \text{traci.trafficlight.getPhase(traffic light)}$$

$$T_l = T_a - \text{traci.simulation.getTime()}$$

$$N_{id} = \frac{V_0 \cdot T_l - S_{cr}}{S_l + S_d} \tag{8}$$

The term of “traci.trafficlight.getPhase(traffic light)” is used to know the next switch time of the traffic light while the “traci.simulation.getTime()” is to understand the current simulation time.

On the other hand, because the vehicle model in the SUMO simulator is designed to imitate a real vehicle, the parameters of the vehicle's engine efficiency, wheels friction, and braking reaction time are added in the original model. Therefore, when the car decelerates, the deceleration of the car does not start from the the theoretical value, nor does it change gradually. The EV needs to passed deceleration information to the PHV. This deceleration information needs to take the above situation into account, so it is necessary to compensate for the deceleration[33-34]. Here, K is defined as a compensation factor, as shown in the deceleration formula.

$$d = \frac{V_0^2}{2(S_{tl} - S_{phv})} \cdot K \tag{9}$$

$$K_n = \frac{\overline{a_{tn}}}{\overline{a_{an}}} \tag{10}$$

$$K = \frac{\sum_0^n K_n}{n} \tag{11}$$

In order to calculate the appropriate K, this simulation will be divided into two parts, the first is to collect the average actual deceleration value $\overline{d_{an}}$ at each different speed, and then calculate the average theoretical deceleration value $\overline{d_{tn}}$. As shown in Equation 8, K at the current speed is the ratio of the average theoretical value $\overline{d_{tn}}$ to the average actual value. Finally, this experiment will take the average K value as the final simulation value.

The following figure is a table that records all parameter abbreviations：

Table 4.2 List of Parameters

List of Abbreviations	Specification
V_0	Initial speed of ego vehicle
S_{lr}	Length of the road
S_{bd}	Length of the braking distance
$S_{sz} = S_{sd}$	Length of safety zone/safety distance
S_{phv}	Position of potential head vehicle
S_{cr}	Length of braking distance
d	Deceleration for potential head vehicle platooning
P_{tl}	Current phase of the traffic light
S_l	Length of the vehicle
S_d	Length of the vehicle distance
S_{tl}	Position of the traffic light
S_{ev}	Position of the ego vehicle
T_a	The time when traffic light turn green to other colors
T_l	The time gap between current time and absolute time
N_{id}	Potential head vehicle ID
K	Safety factor
S_{sl}	The stopping line’s position
T_{cd}	Communication delay

5. Simulation Results and Discussion

5.1 Preparation for the Simulation

As described in section 4.4, the first simulation is to find the appropriate compensation factor K. Therefore, the experiment needs to be carried out at different initial speeds, and the deceleration of each speed should be calculated respectively. Figure 5.1 and Figure 5.2 is the screenshot of the SUMO software interface, which shows the final state of the vehicles after the brake.

The specific steps of this simulation are as follows:

- Set the initial speed of the vehicle to 30, 35 and 40 km/h, respectively.
- Make the vehicle decelerate immediately when it reaches the braking distance corresponding to their speed.
- Mark the origin of time as the moment of braking.
- Calculate the average theoretical deceleration.
- Calculate the average actual deceleration.
- Make compensation for the deceleration loss by K

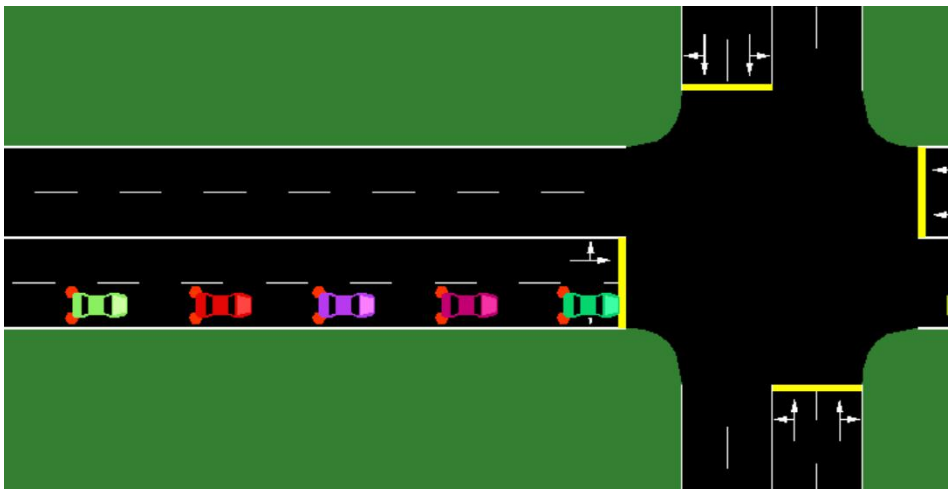


Figure 5.1 State of Vehicles After Braking - 1

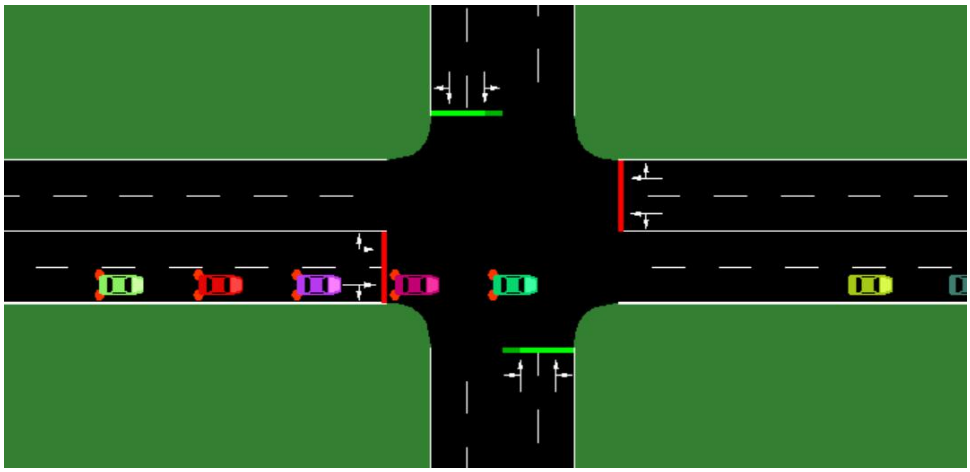
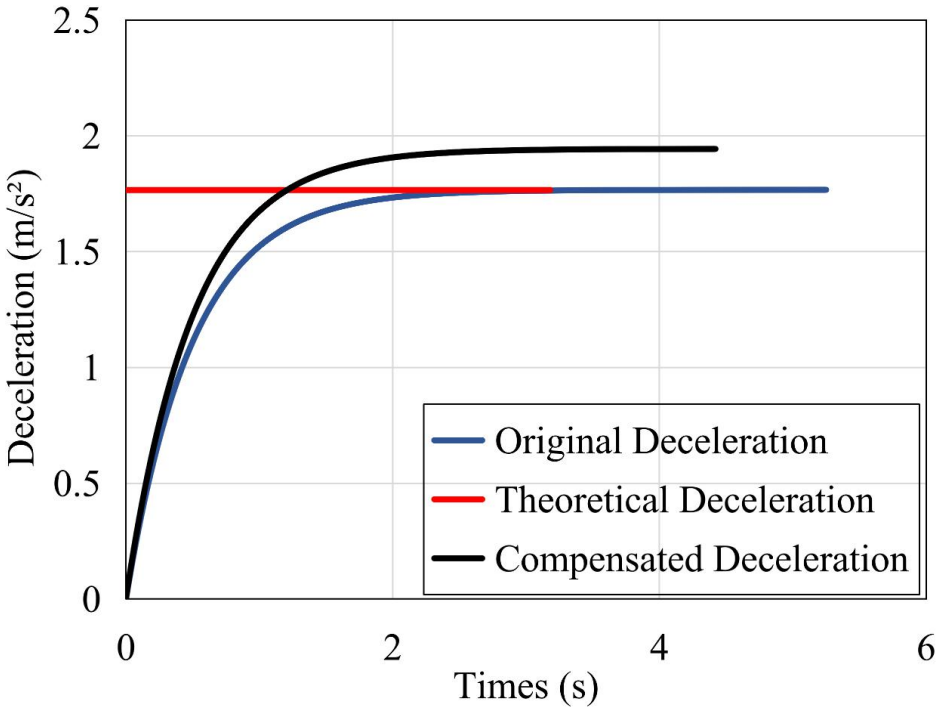


Figure 5.2 State of Vehicles After Braking - 2

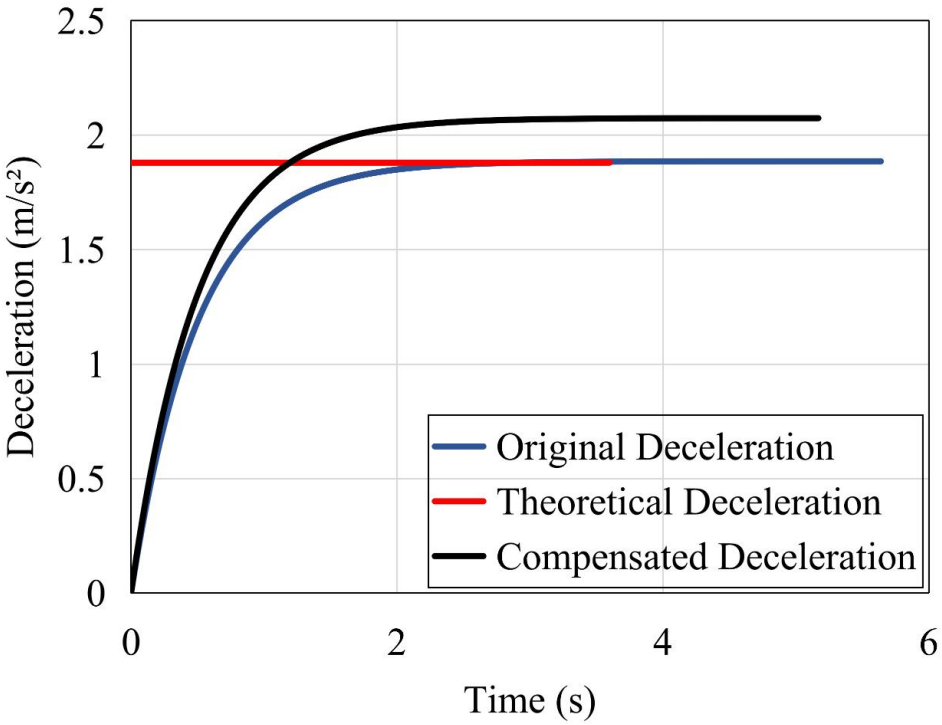
Table 5.1 List of Acceleration Values

	Speed=30 km/h	Speed=35 km/h	Speed=40 km/h
Average theoretical deceleration (m/s^2)	1.766	1.8857	2.0568
Average actual acceleration (m/s^2)	1.599	1.7155	1.867
K_n	1.10	1.09	1.1

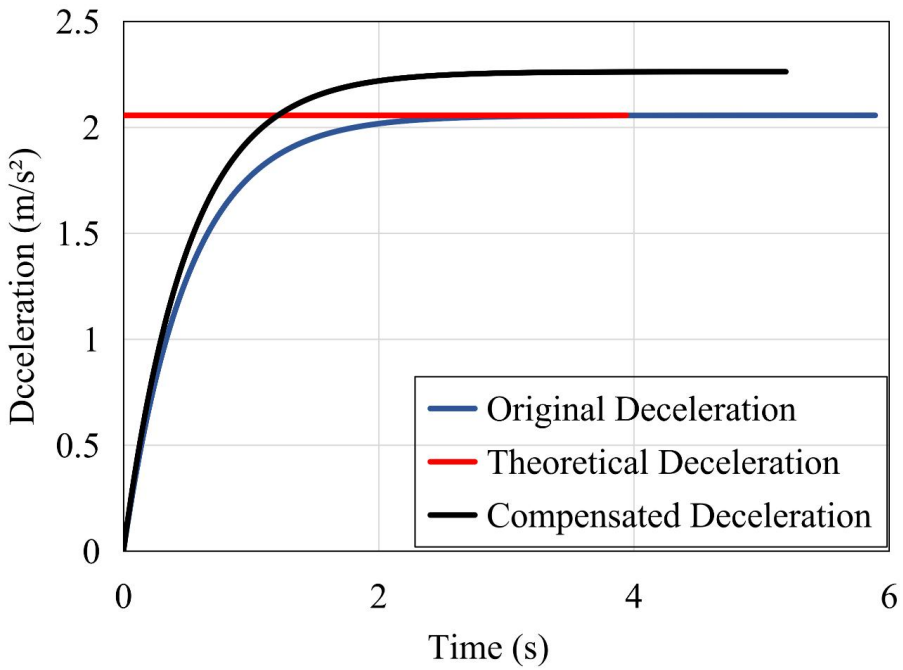
According to the above steps, the obtained data is listed in Table 5.1. As the speed increases, the absolute value of deceleration also increases further. For different initial speeds, the calculated compensation factor K_n is also about the same. By averaging K_n, the final K is got and the figure is 1.1.



(a) Speed = 30 km/h



(b) Speed = 35 km/h



(c) Speed = 40 km/h

Figure 5.3 Analysis of Different Decelerations

Figure 5.3 shows the analysis of the deceleration of the whole braking process under different speeds. Take Figure (a) as an example, the red line represents the theoretical deceleration. In an ideal situation, the deceleration will remain constant, and the vehicle will maintain a uniform deceleration state from the beginning of braking. As can be seen, the vehicle will stop moving after 3 minutes of braking operation.

The blue line in figure represents the actual acceleration curve. As described in section 4.4, considering the friction of the wheels, the efficiency of the vehicle's starting, etc., in real life, the acceleration of the vehicle will take a certain amount of time. It can be known from the figure that the acceleration reaches a maximum value around 2 second and remains constant afterwards. The maximum value is the same as the theoretical deceleration. Under such conditions, the vehicle will inevitably rush out of the braking line. As shown in Figure 5.2, this is because

because the braking time is too long, and the distance traveled by the vehicle is too large. The black line represents the actual acceleration of the vehicle after compensation by the factor K. It can be seen that the maximum acceleration value has a certain increase compared with the theoretical value, and there is a certain reduction in time. Although the ideal state cannot be achieved, taking into account the yellow light time, this time difference can be compensated. In Figure 5.3 (a) ~ (c), the maximum acceleration after compensation is between 0.1 G ~ 0.3 G(G is the acceleration of gravity). In this case, passengers will not feel discomfort due to acceleration, so the above experimental results show that the value of K is reasonable.

5.2 Simulation of Conducting V2X Communication at the Intersection

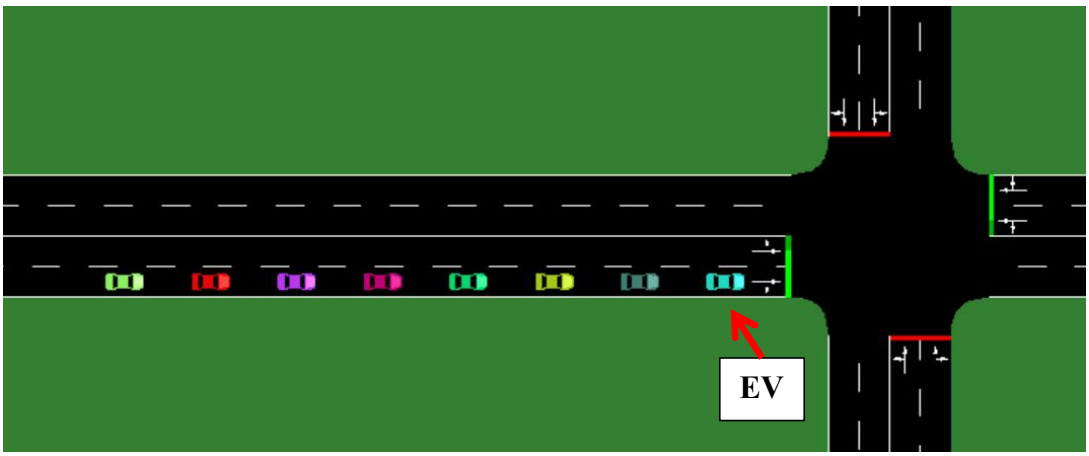
As described in Section 4.2, the experiment in this section mainly simulate V2X communication at the intersection. ADS needs to ensure the safety of platooning. As the initial EV platooning will be divided into two platoonings during the simulation process, so ADS needs to calculate all platoonings' safety zone in real time. At the same time, because there is only one lane in this experiment, **the width of the lane is ignored here, and the safety zone is used instead of the safety zone**. The vehicle length in this experiment is 4 meters, so the minimum safety distance is set to 0.3 ~ 0.5 vehicle length sections, and the median is about 1.5 meters long. The final safety distance obtained though simulation will be compared with 1.5 m. If it is in the range of 0 ~ 1.5 m, it is safe. If it is negative, it is not considered as a safe case. If it is greater than 1.5 m, that means it is absolutely safe. This situation will not be regarded as an ideal safe state.

The specific experimental steps of this experiment are as follows:

- Set the initial speed of the vehicle to 30, 35, 40 km/h.

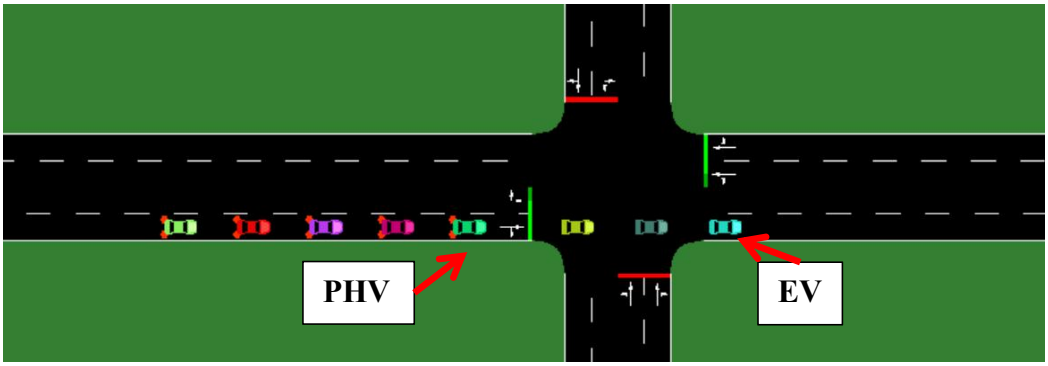
- Let the EV conduct V2X communication to obtain information about other vehicles.
- When the EV reaches the communication distance, V2I communication is performed to obtain the parameters of the traffic lights.
- Calculate the potential PHV and use V2X communication to inform the command.
- When the traffic light turns green again, PHV and its followers will pass the intersection.

The four pictures (a), (b), (c), and (d) in Figure 5.4 and 5.5 describe the process of conducting experiments at a speed of 30km/h. The first picture corresponds to "before passing the intersection substage" in Chapter 4.1. All vehicles will perform communication through V2X at this time. In Figure (b), the EV in the figure completes the calculation and transmits the output to the PHV. Figure (c) shows the braking process of PHV platooning. As to the initial speed, 6 out of 8 vehicles cannot pass through the intersection. The Figure (d) shows the final PHV platooning through the intersection.

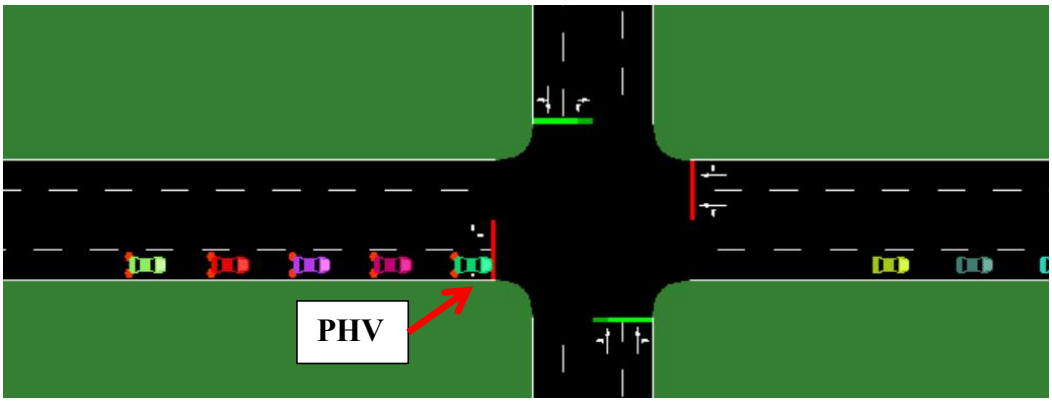


(a)

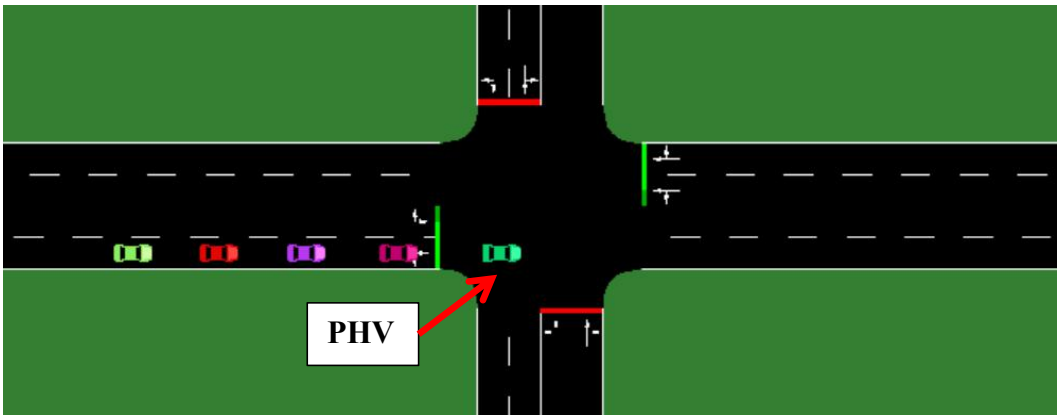
Figure 5.4 V2X Simulation Screenshots - 1



(b)



(c)



(d)

Figure 5.5 V2X Simulation Screenshots - 2

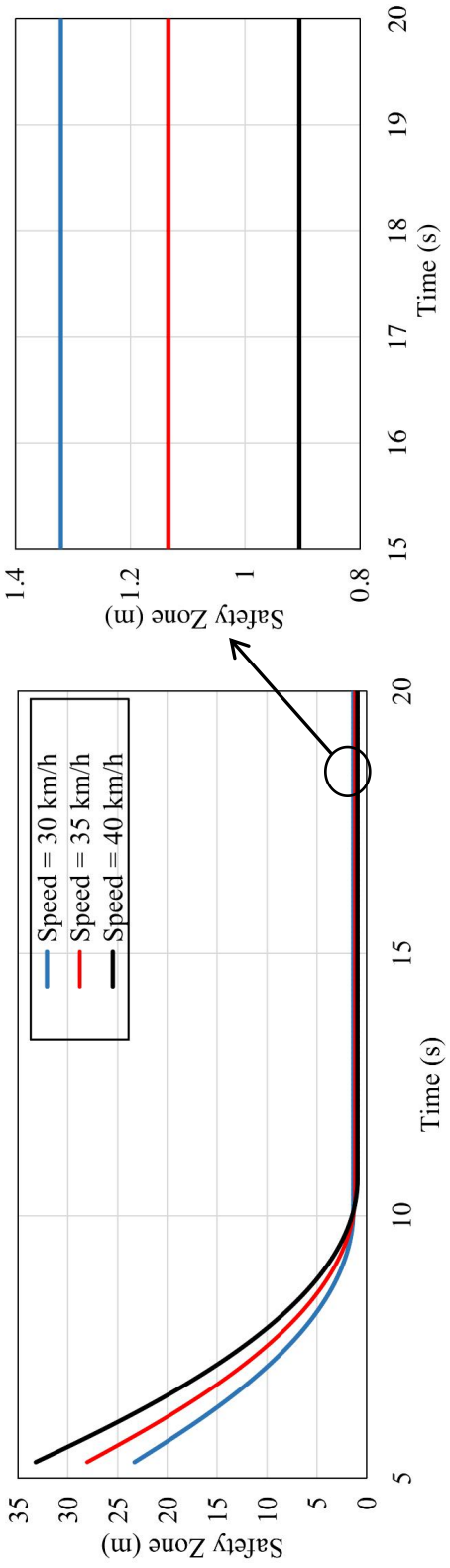


Figure 5.6 Safe Zone During Braking at Different Speeds

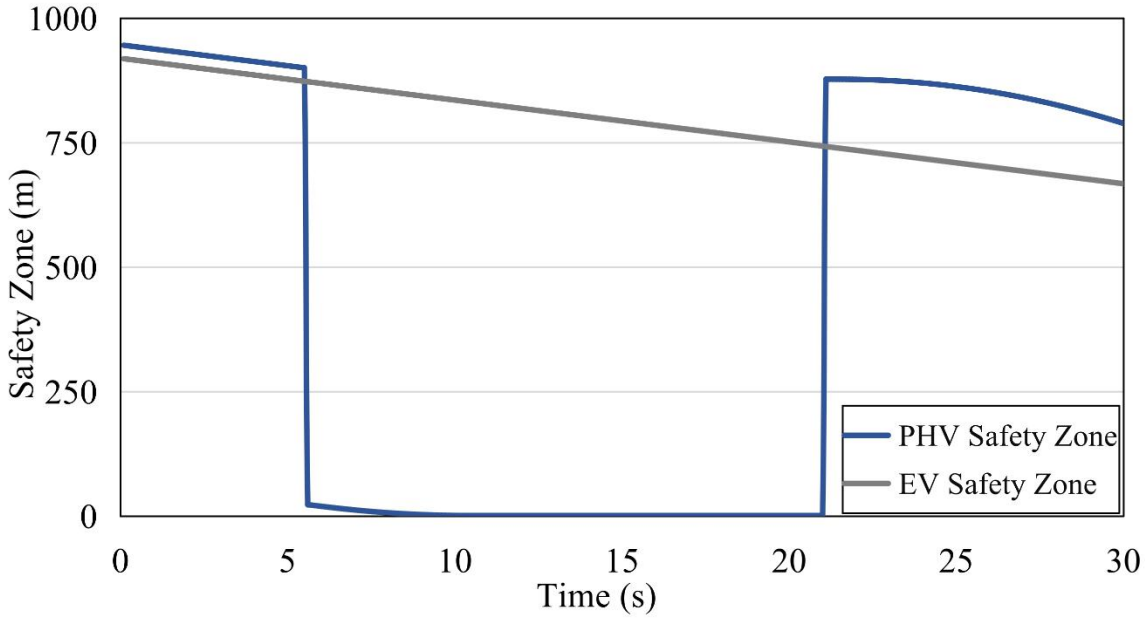


Figure 5.7 Analysis of Safety Zone Under the Speed of 30 km/h

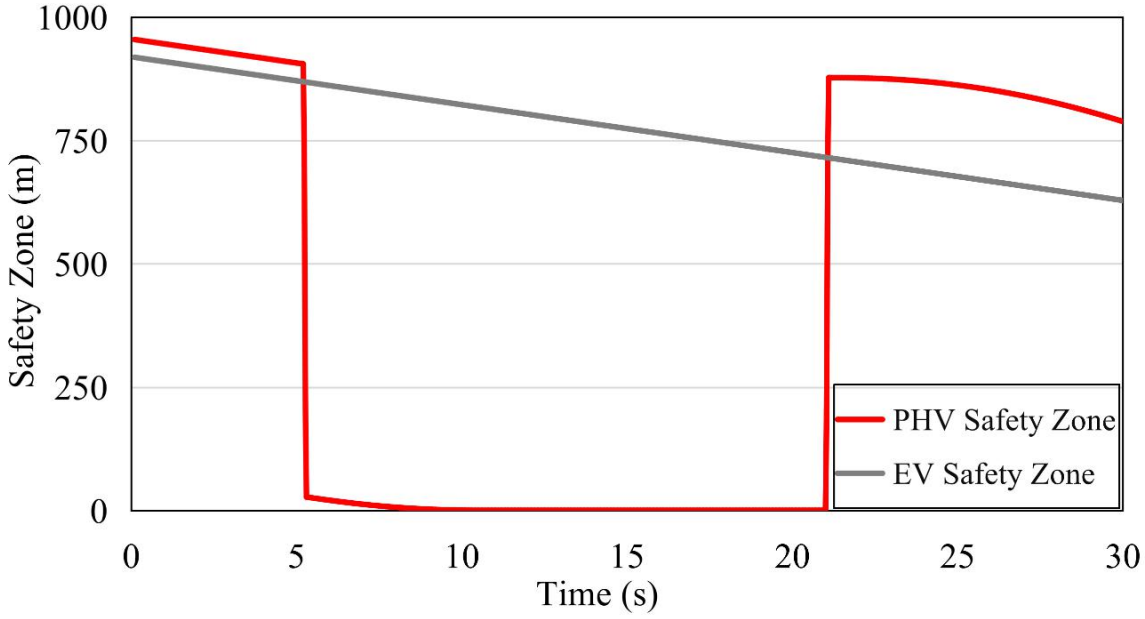


Figure 5.8 Analysis of Safety Zone Under the Speed of 35 km/h

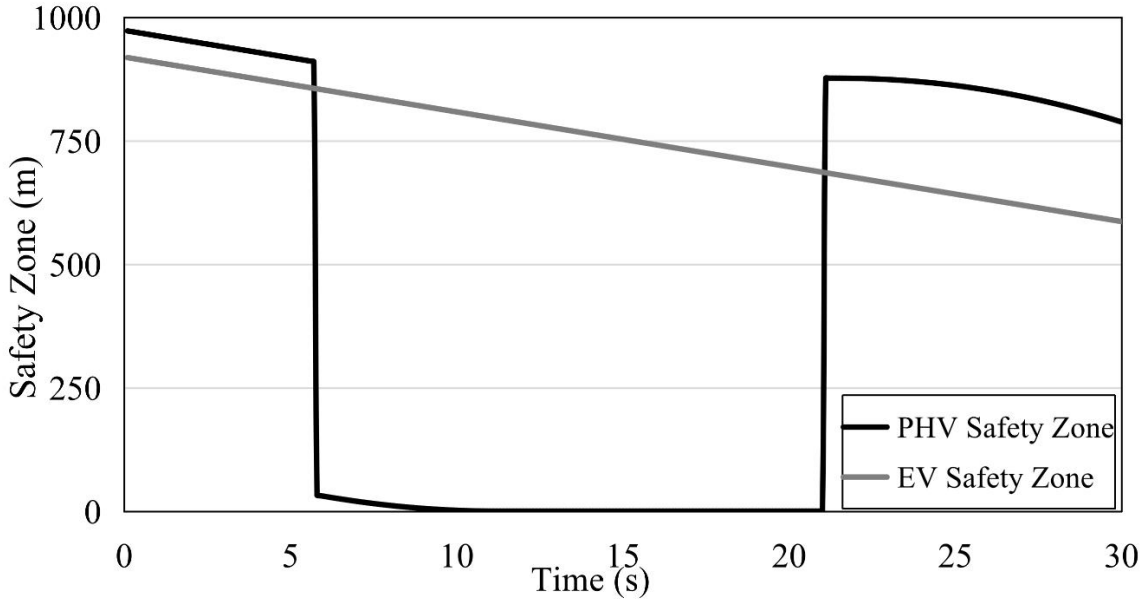


Figure 5.9 Analysis of Safety Zone Under the Speed of 40 km/h

The safety zone of EV platooning in Figures 5.7 ~ 5.9 are reduced at a constant speed and maintain a large value, because EV platooning can pass through the intersection. Figure 5.6 shows the safety zone in the braking state at different speeds. Take the speed of 30 km/h as an example, it is a blue line in the figure. After about 5 seconds, PHV platooning will start the braking operation. During this period, the speed of the vehicles will gradually decrease. After about 9.5 seconds, the PHV completes braking and enters the state of waiting for the green light. Corresponding to Figure 5.7, PHV platooning was at a constant speed at the beginning of the simulation, so the safety zone slowly became smaller. During this period, it exchanges information with the EV to obtain commands. Due to the braking operation, at the 5th second, the safety zone has a sudden change in the picture. Similarly, after the traffic light turned green, PHV Platooning got the EV's passing command, and safety zone changed again in the picture. It can also be found in Figure 5.6 that at different speeds, the PHV braking time is approximately the same, but the initial safety zone in the deceleration state is

different. The different values represent the braking distance corresponding to different speed.

Table 5.2 summarizes the minimum safety zone at different speeds. It can be seen that each value is positive and is in the range of 0 ~ 1.5 m, which shows that the safety of PHV platooning is guaranteed, the requirements of the experiment is met, and the feasibility of ADS is verified.

Table 5.2 List of Minimum Safety Zone (Distance)

Speed（km/h)	30	35	40
Minimum Safety Zone (m)	1.321	1.132	0.906

5.3 Modification of the Braking Time Node

Based on the above content, it can be known that the experimental data is relatively good. However, after careful study of the experiment video, it was found that the potential head vehicle would stop when the light was green. In reality, the vehicle will stop when the green light ends and the yellow light starts. So based on the previous experimental conditions, can the PHV vehicle be arranged to stop under the yellow light condition through the control of the EV? With this question, the logical structure of EV's control has been modified as follows in Figure 5.10. The PHV will start to park when the traffic light is exactly in yellow color.

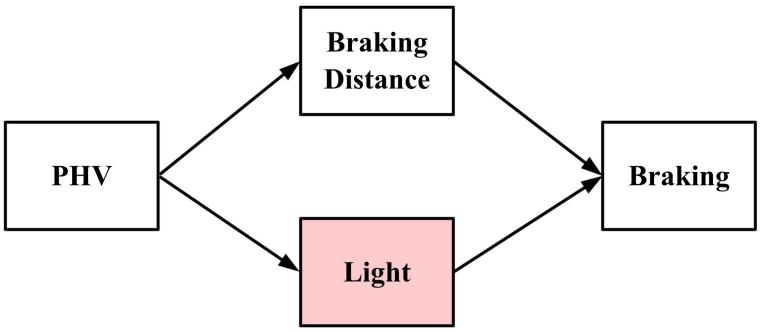
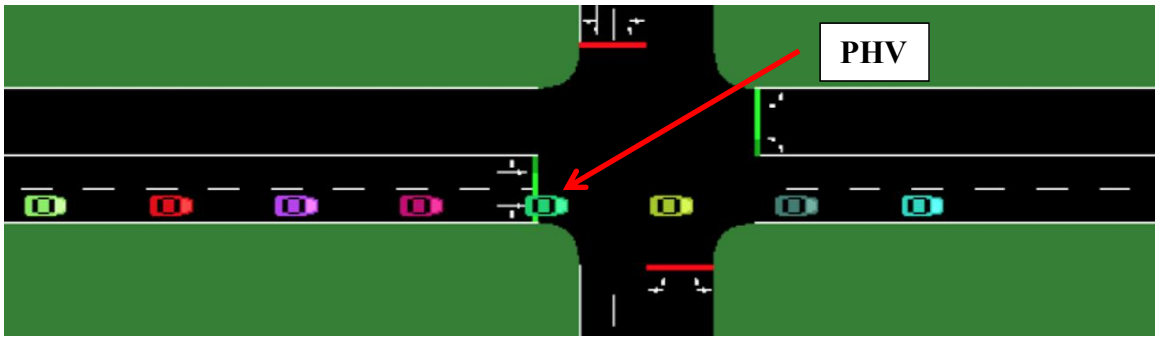
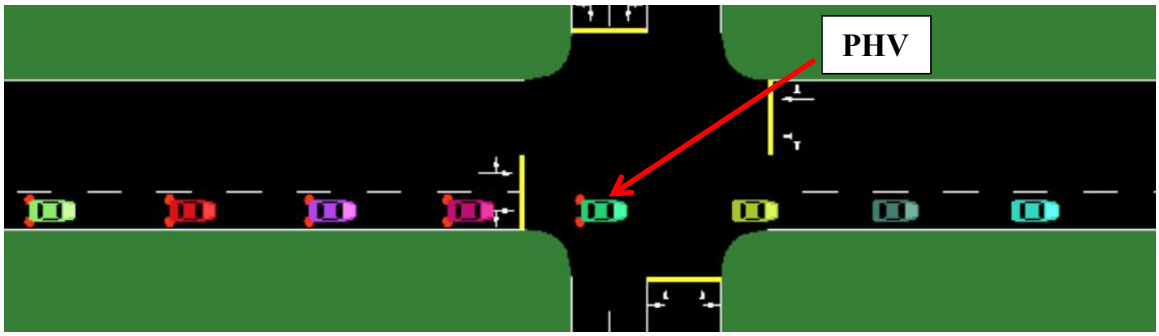


Figure 5.10 Analysis of PHV Braking Strategies



(a)



(b)

Figure 5.11 Yellow Light Braking Simulation Screenshots

The simulation screenshots are presented in the Figure 5.11. As can be seen in the picture (a) that the PHV can not pass the intersection, it have long vehicle distance (11m) with preceding vehicle, but the distance between PHV and stopping line is short at 10th second. When traffic light is in yellow color. PHV starts to brake, but it will crash out of the line. So the ADS will calculate which vehicle can not go through the intersection. And PHV will stop when it reaches the braking distance is better than stop when the light is exactly yellow. The former one will have suitable deceleration and make the driver feel comfortable. It may cause the PHV to stop even when the light is green indeed, but it's safer and this helps to reach the designated safety requirement of ADS.

On the other hand, no matter how long the vehicle distance is, the distance between PHV and stopping line could always have the possibility to be very short. Therefore the parameter that really matters is the braking distance, which can help to decelerate in advance to avoid risks.

5.4 Discussion

This chapter first simulates the vehicle's braking process, and compensates for the deceleration to ensure that the vehicle's braking model can meet the needs. Secondly, the simulation was carried out in the map model built in the first section of Chapter 4, and the results show that the safety of both platoonings can be guaranteed. In this simulation, the important parameters are the safety factor K and the initial speed of the vehicle. The communication distance and braking distance models set in this simulation have certain practicality in real life, but the specific application situation needs in-depth research.

At the same time, this simulation considered the communication delay in V2X communication process. In actual life, the quality of communication will vary with the change of network conditions. In the state of a single scene, the communication delay will not have a large impact, but in complex scenes, ADS needs to respond in real time. If there is a unexpected communication delay, the collected information will not be able to reasonably express the current physical environment around the vehicle[34-35].

Furthermore, the vehicle distance need more research. This study does consider the vehicle distance based on the definition of the platooning that the gap between each other may be small. In reality, the vehicle distance is expected to be longer to avoid unnecessary problems. The distance within 15 ~ 40 m may be simulated.

Besides, the map scene in this research is one type of intersection scenarios. If two intersections appear in a short distance, then the braking distance may inevitably be reduced in this case[36]. Therefore, collaboration of multiple infrastructures may be required, which will also affect the way ADS works. In future research, these factors will be taken into account.

6. Conclusion

In this paper, the architecture of an automated driving system with a perception system is defined using V2X communication method by SysML (System Modeling Language). The operational context of the system is described in detail. The internal components of the system, the activities within the system and the interfaces inside the system have also been modeled. The model defines the work sequence of the sensing system using sensors and V2X communication to obtain information. At the same time, the concept of safety zone was introduced to ensure the driving safety of the vehicle. Based on the above model, this research is simulated on the open source driving simulator sumo, programmed by Python language. In the process of simulation, the parameters of vehicle compensation factor, communication distance and braking distance are taken into consideration.

The first chapter illustrates the needs of automated driving and the current research expectations of automated driving system. Although the occurrence of traffic accidents has been greatly reduced in some developed countries, it has now entered a bottleneck period and needs new ways to help to improve traffic safety. On the other hand, traffic accidents often occur at the intersection, thus ADS need to be able to correctly perceive information at the intersection to ensure safety. The current V2X communication technology is developing rapidly. If this technology is applied to the ADS system, it will broaden the methods of collecting information, and may ensure that ADS obtains sufficient data to make decisions.

The second chapter describes the current research status of ADS. It illustrates the definition of SAE International (Society of Automotive Engineers) about the automation level of automated driving, and compares it with the standards of NHTSA (National Highway Traffic Safety Administration). Meanwhile, it also mentions the difference between Germany and China's definition regarding

automated driving. Secondly, it describes the architecture of the existing ADS system. Apollo and Autoware ADS are listed and explained their shortcomings and the necessity to introduce V2X to perceive the environment. Then it explained the content of SOTIF and ISO 26262 regarding the verification of automated driving system as well.

The third chapter describes the ADS model. It first introduces the context of automated driving, then illustrates the use case of ADS, introduces the internal systems of ADS, and details the activities of the perception system. Secondly, it takes the intersection as an example scenario to describe how the V2X communication system collects information and how the perception system calculates the safety zone. The fourth chapter specifies the parameters and designated scenarios of the simulation experiment, and lists some formulas that need to be used in the experiment. The experimental results of the chapter 5 are divided into two parts. The first part is to simulate the braking situation of the vehicles to get the compensation factor. After obtaining the safety parameters, the V2X communication at the intersection is simulated. The minimum safety zone value is positive and is in the range of 0 ~ 1.5 m, which shows that the proposed ADS system is feasible and successfully guarantees the safety. Also, the yellow light braking simulation is conducted and provides the truth that the braking distance is of benefit to maintaining the vehicle safety. The discussion section points out that the future research direction is to conduct V2X communication in complex scenarios like multiple intersections case, and consider the situation of longer vehicle distance and communication delay.

7. Bibliography

[1] Yunxuan Li,Mohamed Abdel-Aty,Jinghui Yuan,Zeyang Cheng,Jian Lu. Analyzing traffic violation behavior at urban intersections: A spatio-temporal kernel density estimation approach using automated enforcement system data[J]. Elsevier Ltd,2020,141.

[2] Yang Kaixi,Li Meiqi. Traffic Simulation, Optimization and Evaluation of Adjacent Intersections Based on VISSIM Model[J]. SCHOLINK INC.,2020,4(2).

[3] Sun Weiye. Study on traffic design of Urban Road intersection [J]. Urban architecture, 2020,17 (11): 171-173

[4] Krzysztof J. Szajowski,Kinga Włodarczyk. Drivers' Skills and Behavior vs. Traffic at Intersections[J]. MDPI,2020,8(3).

[5] Central Transport Safety Council, Situation of traffic accidents in 2008 and present state of traffic safety measures, 2008

[6] Zhaoyou Ma,Shouen Fang,Shuo Liu,Bowen Cai. Analysis of the Relationship between the Number of Traffic Accidents and the Traffic Flow & Section Location in Extra Long Tunnel[J]. Scientific Research Publishing,2020,12(02).

[7] Krzysztof J. Szajowski,Kinga Włodarczyk. Drivers' Skills and Behavior vs. Traffic at Intersections[J]. MDPI,2020,8(3).

[8] Wang Wanfeng. Statistical analysis of road traffic accidents in China [J]. Production safety in China, 2020,15 (03): 52-53

[9] Keisuke Yoneda,Ryo Yanase,Mohammad Aldibaja,Naoki Suganuma,Kei Sato. Mono-camera based vehicle localization using lidar intensity map for automated driving[J]. Springer Japan,2019,24(2).

[10] Martin Herrmann. Sensor Models for the Development and Validation of Automated Driving Functions[J]. Springer Fachmedien Wiesbaden,2019,14(6).

[11] Christian Merfels,Cyrill Stachniss. Sensor Fusion for Self-Localisation of Automated Vehicles[J]. Springer International Publishing,2017,85(2).

[12] Peng Cao,Walther Wachenfeld,Hermann Winner. Perception sensor modeling for virtual validation of automated driving[J]. De Gruyter Oldenbourg,2015,57(4).
[13] Ma min. discussion on sensor network technology based on Internet of vehicles [J]. Communication world, 2020,27 (06): 67-68
[14] Chen Xiaopeng, Zhang Jingjing. Development status and Prospect of intelligent networked vehicle [J]. Automotive practical technology, 2020 (10): 43-45
[15]. LG Electronics Inc.; Patent Issued for Method For Dropping Communication Based On Priority By Wireless Device Supporting WAN Communication And V2X Communication And, The Wireless Device Performing The Method (USPTO 10,652,911)[J]. Network Weekly News,2020.
[16] Zhang Xiaojun, Guo Jianrui, Guo Peng, Wang mengdan. Research on V2X test method for intelligent driving [J]. Automotive electronics, 2020 (05): 1-5
[17] Jung Chanyoung,Lee Daegyu,Lee Seungwook,Shim David Hyunchul. V2X-Communication-Aided Autonomous Driving: System Design and Experimental Validation.[J]. Pubmed,2020,20(10).
[18] Hidecazu Nishimura, Model based systems engineering and expectation for SysML. Design Engineering, Vol. 46, No. 5 (2011), pp. 241-246.
[19] Teoh Eric R. What's in a name? Drivers' perceptions of the use of five SAE Level 2 driving automation systems.[J]. Journal of safety research,2020,7
[20] Liu Jie. General situation and characteristics of automated driving policies in various countries [n]. People's Posts and telecommunications, December 14, 2018 (007)
[21] Lei Hongjun. Discussion on technical requirements and organization of automobile driving automation classification [J]. Automotive technologist, 2020 (05): 25-29 + 33
[22] Baidu Apollo enters the commercial era [J]. Auto horizon, 2019 (08): 54-55
[23] Sun Bing. How far is Apollo from auto Android? [J]. China Economic Weekly, 2018 (28): 66-67

[24] ON Semiconductor ’ s image sensors selected by Baidu for its Apollo Autonomous Driving Platform[J]. Worldwide Computer Products News,2018.
[25] ISO/PAS 21448. Road vehicles - Safety of the intended functionality. 2019.
[26] Philip Koopman. Safety and Validation of Autonomous Vehicles. 2019.
[27] Technical Report- Taxonomy of Scenarios for Automated Driving. V1.2. 2017.
[28] General Motors. Self-driving Safety Report. 2018.
[29] Riccardo Mariani. Can we trust autonomous systems? 2017.
[30] Mitsuo Nishimura, Hidekazu,Nishimura, Design and verification of traceable area based on architecture definition of automatic driving system, 2017.
[31] Yun songiru, Definition of Driving Safety Architecture for an Automated Vehicle to Enable Interaction with the Driver , 2020
[32] Zhonghui Pei,Wei Chen,Hongjiang Zheng,Luyao Du,Michele Garetto. Optimization of Maximum Routing Hop Count Parameter Based on Vehicle Density for VANET[J]. Mobile Information Systems,2020,2020.
[33] Pei Zhonghui,Chen Wei,Zheng Hongjiang,Du Luyao. Optimization of Maximum Routing Hop Count Parameter Based on Vehicle Density for VANET[J]. Mobile Information Systems,2020,2020.
[34] Yu Xiang, Chen Xiaodong, Wang Zheng, Shi Xueqin. Resource allocation algorithm for Internet of vehicles based on lte-v2x [J / OL]. Computer Engineering: 1-7 [2020-07-06] https://doi.org/10.19678/j.issn.1000-3428.0056935.
[35] Tian bin, Zhao Xiangmo, Xu Zhigang, Wang Miao, Zhang Yuqin. Traffic efficiency information adaptive distribution protocol of intelligent network connected Expressway under the condition of vehicle road coordination: nrt-v2x [J]. Chinese Journal of highway, 2019,32 (06): 293-307
[36] Liu Ning. Research on multi hop broadcast mechanism of Internet of vehicles based on LTE v2x [D]. Beijing University of Posts and telecommunications, 2019

Acknowledgement

This paper is completed under the guidance of Hidekazu Nishimura sensei and Tetsuya Toma sensei. During the study and research process, I encountered many problems that is hard to address. Fortunately, Nishimura sensei gave me a variety of useful suggestions and I could always benefit a lot from the communication in the ADS group meeting. If there is no Nishimura sensei and Toma sensei's instruction, the process of completing the project will be full of difficulties and obstacles. I would like to express my most sincere thank to them.

Secondly, I would like to thank SDM graduate school as well. As for me, graduation is just around the corner. Looking back on past experiences makes me feel extremely grateful. Thank you SDM for providing me with such a wonderful learning environment and excellent atmosphere. Even if I will come back to my home country in the future, the spent time here will always be the best memory in my heart.

Finally, I would like to thank my families. I have experienced many troubles in the past two years. They always offer me encouragement and great support in life, so that I can finish my studies without any burden. I would also like to thank all of you who have helped me in the past two years. In the future, I will also work harder and keep improving my personal ability to be a useful person towards the society.

Appendix

Source code - Map

```
<?xml version="1.0" encoding="UTF-8"?>
<tlLogic offset="0" programID="0" type="static" id="node_0_1">
<phase state="GGGgrrrrGGGgrrrr" duration="34"/>
<phase state="yyyyrrrryyyyrrrr" duration="2"/>
<phase state="rrrrGGGgrrrrGGGg" duration="5"/>
<phase state="rrrryyyyrrrryyyy" duration="1"/>
</tlLogic>

<?xml version="1.0" encoding="iso-8859-1"?>
<configuration>

    <input>
        <net-file value="test.net.xml"/>
        <route-files value="test.rou.xml"/>
    </input>

    <time>
        <begin value="0"/>
        <end value="81600"/>
        <step-length value="0.01"/>
    </time>

    <processing>
        <collision.action value="remove"/>
        <collision.stoptime value="10"/>
    </processing>

    <gui_only>
        <start value="true"/>
        <gui-settings-file value="freeway.gui.xml"/>
    </gui_only>

</configuration>

Version:1.0 StartHTML:000000228 EndHTML:000008741
StartFragment:000000256 EndFragment:000008714 StartSelection:000000256
EndSelection:000008714
<?xml version="1.0"?>
```

<routes>
<vType ploegH="0.5" ploegKd="0.7" ploegKp="0.2" ccAccel="1.5"
lanesCount="4" c1="0.5" xi="1" omegaN="0.2" tauEngine="0.5"
carFollowModel="CC" probability="1" color="1,0,0" maxSpeed="36"
minGap="0" length="4" sigma="0.5" decel="8" accel="2.5" id="vtypeauto"/>
<vType carFollowModel="IDM" color="0.16,0.33,0.42" maxSpeed="36.11"
minGap="0" length="4" sigma="0.5" decel="4.5" accel="1.5" id="passenger"
departSpeed="max" speedFactor="1" guiShape="passenger"
vClass="passenger"/>
<vType carFollowModel="IDM" color="0.16,0.33,0.42" maxSpeed="20"
minGap="0" length="4" sigma="0.5" decel="4.5" accel="1.5" id="passenger2"
departSpeed="max" speedFactor="1" guiShape="passenger"
vClass="passenger"/>
<route id="platoon_route" edges="gneE0 gneE3"/>
</routes>

<?xml version="1.0" encoding="UTF-8"?>
<location projParameter="!" origBoundary="-10000000000.00,-
10000000000.00,10000000000.00,10000000000.00" convBoundary="0.00,-
100.00,1300.00,101.07" netOffset="0.00,0.00"/>
<edge function="internal" id=":node_0_1_0">
<lane id=":node_0_1_0_0" shape="328.20,10.40 327.85,7.95 326.80,6.20
325.05,5.15 322.60,4.80" length="9.03" speed="9.7" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_1"><lane id=":node_0_1_1_0"
shape="328.20,10.40 328.20,-10.40" length="20.80" speed="9.7" index="0"/>
<lane id=":node_0_1_1_1" shape="331.40,10.40 331.40,-10.40" length="20.80"
speed="9.7" index="1"/>
</edge>
<edge function="internal" id=":node_0_1_3">
<lane id=":node_0_1_3_0" shape="331.40,10.40 332.15,5.15 333.00,3.73"
length="6.96" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_16">
<lane id=":node_0_1_16_0" shape="333.00,3.73 334.40,1.40 338.15,-0.85
343.40,-1.60" length="12.40" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_4">
<lane id=":node_0_1_4_0" shape="343.40,4.80 340.95,5.15 339.20,6.20
338.15,7.95 337.80,10.40" length="9.03" speed="9.7" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_5">

<lane id=":node_0_1_5_0" shape="343.40,4.80 322.60,4.80" length="20.80"
speed="9.7" index="0"/>
<lane id=":node_0_1_5_1" shape="343.40,1.60 322.60,1.60" length="20.80"
speed="9.7" index="1"/>
</edge>
<edge function="internal" id=":node_0_1_7">
<lane id=":node_0_1_7_0" shape="343.40,1.60 338.15,0.85 336.73,-0.00"
length="6.96" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_17">
<lane id=":node_0_1_17_0" shape="336.73,-0.00 334.40,-1.40 332.15,-5.15
331.40,-10.40" length="12.40" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_8">
<lane id=":node_0_1_8_0" shape="337.80,-10.40 338.15,-7.95 339.20,-6.20
340.95,-5.15 343.40,-4.80" length="9.03" speed="9.7" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_9">
<lane id=":node_0_1_9_0" shape="337.80,-10.40 337.80,10.40" length="20.80"
speed="9.7" index="0"/>
<lane id=":node_0_1_9_1" shape="334.60,-10.40 334.60,10.40" length="20.80"
speed="9.7" index="1"/>
</edge>
<edge function="internal" id=":node_0_1_11">
<lane id=":node_0_1_11_0" shape="334.60,-10.40 333.85,-5.15 333.00,-3.73"
length="6.96" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_18">
<lane id=":node_0_1_18_0" shape="333.00,-3.73 331.60,-1.40 327.85,0.85
322.60,1.60" length="12.40" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_12">
<lane id=":node_0_1_12_0" shape="322.60,-4.80 325.05,-5.15 326.80,-6.20
327.85,-7.95 328.20,-10.40" length="9.03" speed="9.7" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_13">
<lane id=":node_0_1_13_0" shape="322.60,-4.80 343.40,-4.80" length="20.80"
speed="9.7" index="0"/>
<lane id=":node_0_1_13_1" shape="322.60,-1.60 343.40,-1.60" length="20.80"
speed="9.7" index="1"/>
</edge>
<edge function="internal" id=":node_0_1_15">

<lane id=":node_0_1_15_0" shape="322.60,-1.60 327.85,-0.85 329.27,0.00"
length="6.96" speed="9.26" index="0"/>
</edge>
<edge function="internal" id=":node_0_1_19">
<lane id=":node_0_1_19_0" shape="329.27,0.00 331.60,1.40 333.85,5.15
334.60,10.40" length="12.40" speed="9.26" index="0"/>
</edge>
<edge id="-gneE0" priority="-1" to="node_0_0" from="node_0_1">
<lane id="-gneE0_0" shape="322.60,4.80 -0.00,4.80" length="322.60"
speed="13.89" index="0"/>
<lane id="-gneE0_1" shape="322.60,1.60 -0.00,1.60" length="322.60"
speed="13.89" index="1"/>
</edge>
<edge id="-gneE1" priority="-1" to="node_0_1" from="gneJ2">
<lane id="-gneE1_0" shape="328.20,101.07 328.20,10.40" length="90.67"
speed="9.7" index="0"/>
<lane id="-gneE1_1" shape="331.40,101.07 331.40,10.40" length="90.67"
speed="9.7" index="1"/>
</edge>
<edge id="-gneE2" priority="-1" to="node_0_1" from="gneJ3"><lane id="-
gneE2_0" shape="337.80,-100.00 337.80,-10.40" length="89.60" speed="9.7"
index="0"/
><lane id="-gneE2_1" shape="334.60,-100.00 334.60,-10.40" length="89.60"
speed="9.7" index="1"/>
</edge>
<edge id="-gneE3" priority="-1" to="node_0_1" from="node_0_2">
<lane id="-gneE3_0" shape="1300.00,4.80 343.40,4.80" length="956.60"
speed="9.7" index="0"/>
<lane id="-gneE3_1" shape="1300.00,1.60 343.40,1.60" length="956.60"
speed="9.7" index="1"/>
</edge>
<edge id="gneE0" priority="-1" to="node_0_1" from="node_0_0"><lane
id="gneE0_0" shape="-0.00,-4.80 322.60,-4.80" length="322.60" speed="9.7"
index="0"/>
<lane id="gneE0_1" shape="-0.00,-1.60 322.60,-1.60" length="322.60"
speed="9.7" index="1"/>
</edge>
<edge id="gneE1" priority="-1" to="gneJ2" from="node_0_1">
<lane id="gneE1_0" shape="337.80,10.40 337.80,101.07" length="90.67"
speed="9.7" index="0"/>
<lane id="gneE1_1" shape="334.60,10.40 334.60,101.07" length="90.67"
speed="9.7" index="1"/>
</edge>

<edge id="gneE2" priority="-1" to="gneJ3" from="node_0_1">
<lane id="gneE2_0" shape="328.20,-10.40 328.20,-100.00" length="89.60"
speed="9.7" index="0"/>
<lane id="gneE2_1" shape="331.40,-10.40 331.40,-100.00" length="89.60"
speed="9.7" index="1"/>
</edge>
<edge id="gneE3" priority="-1" to="node_0_2" from="node_0_1"><lane
id="gneE3_0" shape="343.40,-4.80 1300.00,-4.80" length="956.60" speed="9.7"
index="0"/>
<lane id="gneE3_1" shape="343.40,-1.60 1300.00,-1.60" length="956.60"
speed="9.7" index="1"/>
</edge>
<tlLogic id="node_0_1" offset="0" programID="0" type="static">
<phase state="rrrrGGGgrrrrGGGg" duration="10"/>
<phase state="yyyyyyyyyyyyyyyy" duration="2"/>
<phase state="GGGgrrrrGGGgrrrr" duration="7"/>
<phase state="yyyyyyyyyyyyyyyy" duration="2"/>
</tlLogic>
<junction id="gneJ2" shape="333.00,101.07 339.40,101.07 333.00,101.07"
type="dead_end" intLanes="" incLanes="gneE1_0 gneE1_1" y="101.07"
x="333.00"/>
<junction id="gneJ3" shape="333.00,-100.00 326.60,-100.00 333.00,-100.00"
type="dead_end" intLanes="" incLanes="gneE2_0 gneE2_1" y="-100.00"
x="333.00"/>
<junction id="node_0_0" shape="-0.00,0.00 -0.00,6.40 -0.00,0.00"
type="dead_end" intLanes="" incLanes="-gneE0_0 -gneE0_1" y="0.00"
x="0.00"/>
<junction id="node_0_1" shape="326.60,10.40 339.40,10.40 339.84,8.18
340.40,7.40 341.18,6.84 342.18,6.51 343.40,6.40 343.40,-6.40 341.18,-6.84
340.40,-7.40 339.84,-8.18 339.51,-9.18 339.40,-10.40 326.60,-10.40 326.16,-8.18
325.60,-7.40 324.82,-6.84 323.82,-6.51 322.60,-6.40 322.60,6.40 324.82,6.84
325.60,7.40 326.16,8.18 326.49,9.18" type="traffic_light"
intLanes=":node_0_1_0_0 :node_0_1_1_0 :node_0_1_1_1 :node_0_1_16_0 :node
_0_1_4_0 :node_0_1_5_0 :node_0_1_5_1 :node_0_1_17_0 :node_0_1_8_0 :node
_0_1_9_0 :node_0_1_9_1 :node_0_1_18_0 :node_0_1_12_0 :node_0_1_13_0 :no
de_0_1_13_1 :node_0_1_19_0" incLanes="-gneE1_0 -gneE1_1 -gneE3_0 -
gneE3_1 -gneE2_0 -gneE2_1 gneE0_0 gneE0_1" y="0.00" x="333.00">
<request index="0" cont="0" foes="0000000001100000"
response="0000000000000000"/><request index="1" cont="0"
foes="1111100011100000" response="1000000010000000"/>
<request index="2" cont="0" foes="1111100011100000"
response="1000000010000000"/><request index="3" cont="1"
foes="1110011011100000" response="1000011010000000"/>

<request index="4" cont="0" foes="0000011000000000"
response="0000011000000000"/><request index="5" cont="0"
foes="1000111000001111" response="0000111000001111"/>
<request index="6" cont="0" foes="1000111000001111"
response="0000111000001111"/><request index="7" cont="1"
foes="0110111000001110" response="0110111000001110"/>
<request index="8" cont="0" foes="0110000000000000"
response="0000000000000000"/><request index="9" cont="0"
foes="1110000011111000" response="1000000010000000"/>
<request index="10" cont="0" foes="1110000011111000"
response="1000000010000000"/><request index="11" cont="1"
foes="1110000011100110" response="1000000010000110"/>
<request index="12" cont="0" foes="0000000000000110"
response="0000000000000110"/><request index="13" cont="0"
foes="0000111110001110" response="0000111100001110"/>
<request index="14" cont="0" foes="0000111110001110"
response="0000111100001110"/><request index="15" cont="1"
foes="0000111001101110" response="0000111001101110"/>
</junction>
<junction id="node_0_2" shape="1300.00,0.00 1300.00,-6.40 1300.00,0.00"
type="dead_end" intLanes="" incLanes="gneE3_0 gneE3_1" y="0.00"
x="1300.00"/>
<junction id=":node_0_1_16_0" type="internal"
intLanes=":node_0_1_5_0 :node_0_1_5_1 :node_0_1_7_0 :node_0_1_8_0 :node_
0_1_9_0 :node_0_1_9_1 :node_0_1_13_0 :node_0_1_13_1 :node_0_1_15_0"
incLanes=":node_0_1_3_0 -gneE2_0 -gneE2_1" y="3.73" x="333.00"/>
<junction id=":node_0_1_17_0" type="internal"
intLanes=":node_0_1_1_0 :node_0_1_1_1 :node_0_1_3_0 :node_0_1_9_0 :node_
0_1_9_1 :node_0_1_11_0 :node_0_1_12_0 :node_0_1_13_0 :node_0_1_13_1"
incLanes=":node_0_1_7_0 gneE0_0 gneE0_1" y="-0.00" x="336.73"/>
<junction id=":node_0_1_18_0" type="internal"
intLanes=":node_0_1_0_0 :node_0_1_1_0 :node_0_1_1_1 :node_0_1_5_0 :node_
0_1_5_1 :node_0_1_7_0 :node_0_1_13_0 :node_0_1_13_1 :node_0_1_15_0"
incLanes=":node_0_1_11_0 -gneE1_0 -gneE1_1" y="-3.73" x="333.00"/>
<junction id=":node_0_1_19_0" type="internal"
intLanes=":node_0_1_1_0 :node_0_1_1_1 :node_0_1_3_0 :node_0_1_4_0 :node_
0_1_5_0 :node_0_1_5_1 :node_0_1_9_0 :node_0_1_9_1 :node_0_1_11_0"
incLanes=":node_0_1_15_0 -gneE3_0 -gneE3_1" y="0.00" x="329.27"/>
<connection dir="r" to="-gneE0" from="-gneE1" state="O" linkIndex="0"
tl="node_0_1" via=":node_0_1_0_0" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE2" from="-gneE1" state="O" linkIndex="1"
tl="node_0_1" via=":node_0_1_1_0" toLane="0" fromLane="0"/>

<connection dir="s" to="gneE2" from="-gneE1" state="O" linkIndex="2"
tl="node_0_1" via=":node_0_1_1_1" toLane="1" fromLane="1"/>
<connection dir="l" to="gneE3" from="-gneE1" state="o" linkIndex="3"
tl="node_0_1" via=":node_0_1_3_0" toLane="1" fromLane="1"/>
<connection dir="r" to="gneE3" from="-gneE2" state="O" linkIndex="8"
tl="node_0_1" via=":node_0_1_8_0" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE1" from="-gneE2" state="O" linkIndex="9"
tl="node_0_1" via=":node_0_1_9_0" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE1" from="-gneE2" state="O" linkIndex="10"
tl="node_0_1" via=":node_0_1_9_1" toLane="1" fromLane="1"/>
<connection dir="l" to="-gneE0" from="-gneE2" state="o" linkIndex="11"
tl="node_0_1" via=":node_0_1_11_0" toLane="1" fromLane="1"/>
<connection dir="r" to="gneE1" from="-gneE3" state="o" linkIndex="4"
tl="node_0_1" via=":node_0_1_4_0" toLane="0" fromLane="0"/>
<connection dir="s" to="-gneE0" from="-gneE3" state="o" linkIndex="5"
tl="node_0_1" via=":node_0_1_5_0" toLane="0" fromLane="0"/>
<connection dir="s" to="-gneE0" from="-gneE3" state="o" linkIndex="6"
tl="node_0_1" via=":node_0_1_5_1" toLane="1" fromLane="1"/>
<connection dir="l" to="gneE2" from="-gneE3" state="o" linkIndex="7"
tl="node_0_1" via=":node_0_1_7_0" toLane="1" fromLane="1"/>
<connection dir="r" to="gneE2" from="gneE0" state="o" linkIndex="12"
tl="node_0_1" via=":node_0_1_12_0" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE3" from="gneE0" state="o" linkIndex="13"
tl="node_0_1" via=":node_0_1_13_0" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE3" from="gneE0" state="o" linkIndex="14"
tl="node_0_1" via=":node_0_1_13_1" toLane="1" fromLane="1"/>
<connection dir="l" to="gneE1" from="gneE0" state="o" linkIndex="15"
tl="node_0_1" via=":node_0_1_15_0" toLane="1" fromLane="1"/>
<connection dir="r" to="-gneE0" from=":node_0_1_0" state="M" toLane="0"
fromLane="0"/>
<connection dir="s" to="gneE2" from=":node_0_1_1" state="M" toLane="0"
fromLane="0"/>
<connection dir="s" to="gneE2" from=":node_0_1_1" state="M" toLane="1"
fromLane="1"/>
<connection dir="l" to="gneE3" from=":node_0_1_3" state="m"
via=":node_0_1_16_0" toLane="1" fromLane="0"/><connection dir="l"
to="gneE3" from=":node_0_1_16" state="M" toLane="1" fromLane="0"/>
<connection dir="r" to="gneE1" from=":node_0_1_4" state="M" toLane="0"
fromLane="0"/>
<connection dir="s" to="-gneE0" from=":node_0_1_5" state="M" toLane="0"
fromLane="0"/>
<connection dir="s" to="-gneE0" from=":node_0_1_5" state="M" toLane="1"
fromLane="1"/>

<connection dir="l" to="gneE2" from=":node_0_1_7" state="m" via=":node_0_1_17_0" toLane="1" fromLane="0"/><connection dir="l" to="gneE2" from=":node_0_1_17" state="M" toLane="1" fromLane="0"/>
<connection dir="r" to="gneE3" from=":node_0_1_8" state="M" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE1" from=":node_0_1_9" state="M" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE1" from=":node_0_1_9" state="M" toLane="1" fromLane="1"/>
<connection dir="l" to="-gneE0" from=":node_0_1_11" state="m" via=":node_0_1_18_0" toLane="1" fromLane="0"/><connection dir="l" to="-gneE0" from=":node_0_1_18" state="M" toLane="1" fromLane="0"/>
<connection dir="r" to="gneE2" from=":node_0_1_12" state="M" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE3" from=":node_0_1_13" state="M" toLane="0" fromLane="0"/>
<connection dir="s" to="gneE3" from=":node_0_1_13" state="M" toLane="1" fromLane="1"/>
<connection dir="l" to="gneE1" from=":node_0_1_15" state="m" via=":node_0_1_19_0" toLane="1" fromLane="0"/>
<connection dir="l" to="gneE1" from=":node_0_1_19" state="M" toLane="1" fromLane="0"/>
</net>

Source code - Simulation

```
# -*- coding: utf-8 -*-
import os
import sys
from math import floor
import random
from utils import add_platooning_vehicle, start_sumo, running, communicate
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("please declare environment variable 'SUMO_HOME'")
import traci
from plexe import Plexe, ACC, CACC, RPM, GEAR, RADAR_DISTANCE
Vehicle_length = 4
Platooning_vehicle_num = 8
Platooning_num = 1
#Vehicle_speed = 8.4 m/s
```

```
#The speed is subject to change here

Vehicle_distance = vehicle_speed * 1.5 + 2
PHV = "v.0.0"
EV ="v.0.0"
Traffic_light_ID = "node_0_1"
Communication_range =30
Traffic_light_pos = 330
K = 1.1

def add_vehicles(plexe, n, n_platoons, real_engine=False):
	topology = {}
	p_length = n * Vehicle_length + (n - 1) * Vehicle_distance
	for p in range(n_platoonings):
		for i in range(n):
			vid = "v.%d.%d" % (p, i)
			add_platooning_vehicle(plexe, vid, 200 + (n-i+1) *
(Vehicle_length+Vehicle_distance), 0, Vehicle_speed, Vehicle_distance,
real_engine)
			plexe.set_fixed_lane(vid, 0, False)
			traci.vehicle.setSpeedMode(vid, 0)
			plexe.use_controller_acceleration(vid, False)
			if i == 0:
				plexe.set_active_controller(vid, ACC)
			else:
				plexe.set_active_controller(vid, CACC)
			if i > 0:
				topology[vid] = {"front": "v.%d.%d" % (p, i - 1),    "leader":
"v.%d.0" % p}
			else:
				topology[vid] = {}
	return topology
def main(demo_mode, real_engine, setter=None):

	random.seed(1)
	start_sumo("cfg/test.sumo.cfg", False)
	plexe = Plexe()
	traci.addStepListener(plexe)
	step = 0
	topology = dict()
	min_dist = 1e6
	split = False
```

```
    slow_down = False
    while running(demo_mode, step, 3000):
        traci.simulationStep()
        if step == 0:

            topology = add_vehicles(plexe, Platooning_vehicle_num,
Platooning_num, real_engine)
            tracked_veh = "v.0.%d" %(Platooning_vehicle_num -1)
            traci.gui.trackVehicle("View #0", tracked_veh)
            traci.gui.setZoom("View #0", 1000)
            leader_data = plexe.get_vehicle_data(EV)

        if leader_data.pos_x >= Traffic_light_pos - Communication_range:
           if not split:
            current_phase = traci.trafficlight.getPhase(Traffic_light_ID)
            if current_phase == 0:
               absolute_time = traci.trafficlight.getNextSwitch(Traffic_light_ID)
               time_left = absolute_time - traci.simulation.getTime()
               new_leader = int(floor((leader_data.speed * time_left -
Traffic_light_pos + leader_data.pos_x)/(Vehicle_length + Vehicle_distance)))

               if new_leader <= Platooning_vehicle_num -1:
                  if new_leader > 0:
                     new_leader_id = "v.0.%d" % new_leader
                     for i in range(new_leader+1, Platooning_num):
                        topology["v.0.%d" %i]["leader"] = new_leader_id
                        topology[new_leader_id] = {}
                     split = True
                  else:
                     new_leader = 0
                     new_leader_id = "v.0.%d" % new_leader

                  new_leader_data = plexe.get_vehicle_data(new_leader_id)

                     if 330 - new_leader_data.pos_x <= 30 and slow_down ==
False:

#The deceleration is tested here

#Deceleration 1: considering the yellow light.
            #decel = new_leader_data.speed**2 / (2* (TL_POS -
new_leader_data.pos_x - 1.7*new_leader_data.speed))
```

```
#Tested deceleration
                #decel = new_leader_data.speed**2 / (2* (TL_POS -
new_leader_data.pos_x ))

#Deceleration 2: considering the compensation
                #decel =    new_leader_data.speed**2 / (2* (320 -
new_leader_data.pos_x )) * K

#Tested deceleration
                #decel = new_leader_data.speed**2 / (2* (Communication_range +
new_leader * (Vehicle_length + Vehicle_distance)))
                plexe.set_fixed_acceleration(new_leader_id, True, -1 * decel)
                slow_down = True


#Tested data ----- for data output

            x = plexe.get_vehicle_data("v.0.0")
            #      print(x.acceleration)
            #      print(step)
            y = 0
            if step % 10 == 1:
              #if x.speed > 0 and x.acceleration == 0:
              #       y = 1000 - x.pos_x
              #if x.speed > 0 and x.acceleration < 0:
              #       y = 325 - x.pos_x
              #if x.speed == 0:
              #       y = 325 - x.pos_x
              #if x.speed > 0 and x.acceleration > 0:
              #       y = 1000 - x.pos_x
              print(1200 - x.pos_x)
            #x = plexe.get_vehicle_data("v.0.0")
            #y = plexe.get_vehicle_data("v.0.7")
            #print(x.pos_x - y.pos_x)
            #a = plexe.get_vehicle_data("v.0.3")
            #if a.acceleration < 0 :
              # print(325 - a.pos_x)
                #print(a.acceleration)

#Communication delay is defined here.

            if step % 10 == 1:
```

```
                communicate(plexe, topology)
            if real_engine and setter is not None:
                tracked_id = traci.gui.getTrackedVehicle("View #0")
                if tracked_id != "":
                    ed = plexe.get_engine_data(tracked_id)
                    vd = plexe.get_vehicle_data(tracked_id)
                    setter(ed[RPM], ed[GEAR], vd.speed, vd.acceleration)
            if split == True:
                    new_leader_data = plexe.get_vehicle_data(new_leader_id)
                    current_phase = traci.trafficlight.getPhase(Traffic_light_ID)
                    absolute_time = traci.trafficlight.getNextSwitch(Traffic_light_ID)
                    time_left = absolute_time - traci.simulation.getTime()
                    if Traffic_light_pos - new_leader_data.pos_x > 0 and
current_phase == 0 and time_left>=3 and new_leader_data.speed == 0:

#For each vehicle's data output

                    #V = plexe.get_vehicle_data("v.0.0")
                   # if V.speed == 0 and current_phase == 0:
                        plexe.set_fixed_acceleration(new_leader_id, True, 3)
                     #     plexe.set_fixed_acceleration("v.0.0", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.1", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.2", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.3", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.4", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.5", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.6", True, 3)
                     #     plexe.set_fixed_acceleration("v.0.7", True, 3)

            if step > 1:
                    radar = plexe.get_radar_data("v.0.1")
                    if radar[RADAR_DISTANCE] < min_dist:
                            min_dist = radar[RADAR_DISTANCE]
            step += 1
            #if step > 3000:
            #break
        traci.close()
if __name__ == "__main__":
    main(True, False)
```