POLITECNICO DI TORINO

Master's Degree in COMPUTER ENGINEERING

Master's Degree Thesis

Scenario Generation and Simulation for Autonomous Software Validation

Supervisors

Prof. ANDREA TONOLI

Prof. NICOLA AMATI

Prof. ANGELO BONFITTO

PhD Candidate STEFANO FAVELLI

PhD Candidate EUGENIO TRAMACERE

Candidate

PENG CAO

DECEMBER 2023

Abstract

Simulation and testing of Advanced Driving Assistance Systems (ADAS) and Autonomous Driving Systems (ADS) has gained in the recent years a lot of attention both from the automotive industry and academia. Testing this systems on the road is closer to their final real-world application and it is a desirable approach, but it is incredibly costly at the same time. Also, it is unfeasible or too dangerous to cover rare corner cases using such real-world testing. Thus, the common goal of the industry is to evaluate the systems' performance in some well-designed challenging scenarios, a.k.a. scenario-based testing, in a safe virtual environment. Testing in a virtual environment offers several advantages, including cost and time savings, as well as the ability to perform comprehensive tests that may not be feasible in the real world. Furthermore, it allows us to address the rare corner cases that are too challenging to cover on roads or proving grounds.

Simultaneously, virtual testing environment allows the data generation and collection from sensors within the simulation and the testing of algorithms. This approach provides a popular alternative for evaluating the performance of ADS and ADAS in well-designed challenging scenarios, also known as scenario-based testing.

This thesis primarily focuses on the design of the scenarios that cannot be effectively covered in a real-world environment. It begins by introducing some fundamental concepts related to ADAS and vehicle automation. Additionally, it discusses the regulations governing the testing of those systems in different countries. Following these regulatory guidelines, a set of relevant scenarios that meet provisions of relevant laws is proposed.

An important step of the activity, involved the selection of a suitable simulation software for testing. During this process, various simulation software products available in the market are analyzed, considering their advantages and disadvantages. A tool that aligns with the requirements to conduct the testing is selected and the design and validation of the scenarios for testing is performed. The final objective is to create a set of relevant scenarios and execute them in the virtual environment.

As a final step, the tool of scenario engineering is introduced to enhance the diversity and safety of real-road testing by integrating virtual and physical components in the testing loop.

Keywords: Scenario generation , Autonomous driving, Testing, Automated Driving System, Advanced Driver Assistance Systems

Table of Contents

Chapter 1

Introduction

1.1 Background

1.1.1 Autonomous Driving

Before delving into the assessment of Automated Driving (AD), it is essential to grasp the fundamental principles of driving automation. This section begins by introducing the definition of automated driving within the context of road vehicles. Subsequently, a brief overview of the history of AD summarizes its current stage of development. Finally, the structure of a state-of-the-art system and its components is outlined.

This thesis presents the background of Autonomous Driving, along with strict criteria for selecting papers. Terminologies are introduced along the way, additionally, provide a summary of all abbreviations utilized in this document within the Table 1.1.

Levels of Driving Automation

Autonomous driving invariably pertains to a self-driving car, also recognized as an autonomous car (AC), capable of traveling without human input. These vehicles undertake the tasks of perceiving the environment, monitoring crucial systems, and controlling navigation. The perception system processes visual and audio data from both external and internal sources, interpreting it to abstractly depict the vehicle and its surroundings. The control system subsequently executes actions to maneuver the vehicle, taking into account the designated route, road conditions, traffic signals, and obstacles.

Notation	Meaning
ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance System
ADS	Automated Driving System
AEB	Automated Emergency Braking
ALC	Automated Emergency Centering
ALC2	Automated Emergency Change
ARM	Autoregressive Model
BO	Bayesian Optimization
CLS	Coverage and Local Search
CNN	Convolutional Neural Network
DAS	Driving Automation System
DE	Differential Evolution
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DT	Decision Tree
EA	Evolutionary Algorithm
FCW	Forward Collision Warning
GA	Genetic Algorithm
GP	Gaussian Process
IL	Imitation Learing
IS	Importance Sampling
LDW	Lane Departure Warning
NPC	Non-Player Character
NN	Neural Network
ODD	Operational Design Domain
OEDR	Object and Event Detection and Response
PP	Pedestrian Protection
RL	Reinforcement Learning
SA	Simulated Annealing
TSR	Traffic Sign Recognition
TTC	Time to Collision

Table 1.1: Table of Notations

These self-driving cars possess the potential to profoundly influence various domains, encompassing the automotive industry, healthcare, welfare, urban planning, traffic management, insurance, the labor market, and more. The implementation of suitable regulations is imperative to ensure their safe integration.

Level	Name	Example	Sustained Control	Lat&Long	Complete OEDR	System Fallback	Unlimited Domain
0	no driving automation	automated emergency breaking	×	×	×	×	×
1	driver assistance	adaptive cruise control	√	×	×	×	×
2	partial driving automation	adaptive cruise control and lane centering at the sam time	√	√	×	×	×
3	conditional driving automation	traffic jam chauffeur	√	√	√	×	×
4	high driving automation	local driverless taxi	√	√	√	√	×
5	full driving automation	driverless taxi of all conditions	√	√	√	√	√

Table 1.2: Driving automation at different levels

As shown in Table 1.2, the SAE J3016(TM) "Standard Road Motor Vehicle Driving Automation System Classification and Definition" [1]categorizes driving automation systems into six levels from level0 to level5.

This report categorizes these systems as L0 to L5. L0 systems provide momentary intervention during potentially hazardous situations, such as Automated Emergency Braking (AEB) and Forward Collision Warning (FCW). L1 systems support sustained longitudinal or latitudinal control, including examples like Adaptive Cruise Control (ACC), Automated Lane Centering (ALC), and Automated Lane Change (ALC2). L2 systems enable both sustained longitudinal and sustained latitudinal driving controls simultaneously, as seen in systems with both ACC and ALC.

L3 systems integrate Object and Event Detection and Response (OEDR) functionality, allowing them to monitor the driving environment and execute appropriate responses. An example is a traffic jam chauffeur. L4 systems additionally support system fallback, ensuring no human interventions are required within a geo-fenced region. A typical example is a local driver-less taxi. L5 systems can operate under all conditions, exemplified by a driverless taxi capable of handling any situation.

Automated Driving System (ADS) refers to a highly automated system [1] (i.e., L3-L5). Advanced Driver Assistance System (ADAS) encompasses a broad range of features, including L0 and L1 features. In literature, it is commonly used to refer to an L0-L2 system.

Even if it might be overly ambitious to achieve L5 systems, many industrial companies are working hard to establish L4 systems. An architectural overview of a typical L4 Automated Driving System (ADS) is given in Figure 1.1 [2]. In order for a L4 system to understand its driving context, which includes its location (localization), traffic signs (perception), obstacles (perception), and their trajectories (prediction), it uses real-time sensor data (such as Cameras, Radar, LiDAR, and GPS) and High-Definition (HD) map data. To create short-term trajectories, the Planning module analyzes the information about the destination and the perceived driving environment. In order to manage the vehicle actuators, including the steering wheel and gas/brake, the manage module, at the end, converts the trajectory into orders for the Controller Area Network (CAN) bus.

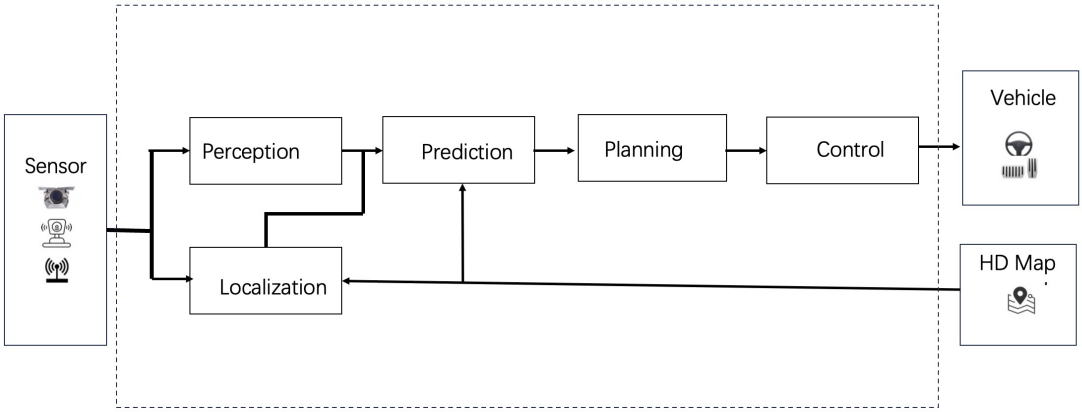

Figure 1.1: Overview of a typical L4 ADS

Middle-class and luxury vehicles now frequently have L0-L2 systems installed; Tesla's AutoPilot, for example, is an L2 system. Since cameras and radar are the main sensors used by L0-L2 systems, several of the elements shown in Figure 1.1 might not be present. For example, a deep neural network (DNN) might replace the entire middle block in an end-to-end lane follower (L2).

Automated Driving Systems (ADSs) usually have different designs than sophisticated Driver Assistance Systems (ADASs) since ADSs provide more sophisticated features. However, experts have seen similarities in the way that both ADS and ADAS are tested, especially when the test device is handled like a black box. As a result, They have chosen to incorporate both into our analysis. Systems ranging from L0 to L4 have been tested in the included works. And it can be straightforwardly categorized the distinct classes as follows [1]:

- **SAE level 0 – No Automation:** This characterizes vehicles without any automation at all. The human driver is responsible for all aspects of the driving task.

- **SAE level 1 – Driver Assistance:** As soon as modern driving assistance systems like ACC or Lane Keeping Assistance System (LKAS) can take over longitudinal or lateral control, respectively, the automation reaches level 1. A significant limitation is that either of these systems is allowed to be active simultaneously. The human driver is still respon- sible for the system and environment supervision and acts as a fallback. Also, ODD(Operational Design Domain) for such systems is limited.

- **SAE level 2 – Partial Automation:** Vehicles able to take over both lateral and lon- gitudinal control simultaneously qualify for level 2. However, the human driver is still responsible for monitoring the driving task and intervening at any time. The ODD(Operational Design Domain) is still limited to, e.g., highways. Vehicles that offer these capabilities for a limited time range are already available on the market.

- **SAE level 3 – Conditional Automation:** With level 3, the human driver is not responsible for supervising the environment and system anymore, while the machine is responsible for lateral and longitudinal control. However, he must still be present as a fallback by request, and the ODD is still limited.

- **SAE level 4 – High Automation:** Level 4 extends the previous layer's capabilities by removing the need for a human driver as a fallback. The limitation on the ODD still applies.

- **SAE level 5 – Full Automation:** Lastly, the ODD becomes unlimited in level 5. A fully automated vehicle is capable of driving in any environment without the need for a passenger at all.

History and Progress in Automated Driving

Since being recorded in history about a century ago, the idea and goal of driving automation have had a considerable trip [3]. The reality of this technology has frequently seemed to be just 20 years away, despite the ongoing expectation [4]. This section tries to shed light on how Automated Driving (AD) has historically changed from the past to the predicted future.The information mainly based [3] and [5].

The first autonomous vehicle was seen on Ohio's streets in Dayton in 1921, to much surprise [3]. Even though it was not completely autonomous and was still under human control, it raised awareness of the idea. The American Wonder, the first full-size driverless automobile with remote control, was originally launched by the Houdina Radio Control Company in 1925 [6].

The concept of automated driving (AD) was momentarily laid aside when the Great Depression struck between 1929 and 1941. General Motors Company LLC (GM) made a big advancement in 1958 when it switched from remotely operated to automatically guided vehicles. They created an autonomous guided vehicle that could drive itself and locate wires hidden in the road using sensors beneath the front end of the vehicle.

The "Stanford Cart" research project, which began in 1960, marked the first steps toward automation. The cart succeeded in 1966 when it used remotely run vision-based algorithms on a host computer to follow printed white lines on its own. These techniques were called guided because they did not correspond with the completely automated driving we expect today, even though they theoretically automated the driving work.

The "Stanford Cart" was the subject of a thorough investigation carried out by the same academics in 1979, which laid the groundwork for modern autonomous driving technology. With the "Stanford Cart," a little car with autonomous navigation capabilities, the driving task was first automated. The cart could navigate obstacles on its own and find a route to a predetermined destination. Stereo vision and sophisticated vision algorithms were used to plot the course using cameras that moved horizontally and were fixed on the track. It took the trolley five hours to finish the 20-meter obstacle course, though.

The 1986 launch of the PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety (PROMETHEUS) was a major turning point toward the automation of full-size cars on public roads. A research vehicle called VaMP [7] from

the University of Munich, part of the Bundeswehr, completed a 1,000-kilometer final demonstration in 1994 while traveling on Motorway 1 close to Paris-Charles de Gaulle Airport. VaMP was acknowledged as one of the first fully autonomous automobiles when the vehicle executed a number of autonomous duties during this highway journey, including lane changes and overtaking [8].An important turning point in urban autonomous driving was the 2007 Urban Challenge, which was run by the Advanced Research Projects Agency (DARPA) of the US Department of Defense [9]. Participating organizations used their research vehicles to put up parkour races in urban settings. The winners of this challenge were the Carnegie Mellon University (CMU) team known as "Tartan Racing" [10].

For the industry, particularly Original Equipment Manufacturers (OEMs), early research efforts demand substantial time and resources. The market launch of such technologies typically involves an extensive process of development, testing, and validation.

In the realm of automation in the automotive sector, the first vehicle meeting SAE Class 1 standards was developed by Toyota, featuring the inaugural laser-based automatic control system in 1997 [11]. Mercedes embraced the now widely used radio detection and ranging (radar)-based system in 1999. Nissan introduced the first Lane Keeping Assist System (LKAS) in 2001. Although the foundational systems for longitudinal or lateral guidance were introduced two decades ago, fully integrated SAE Level 2-compliant systems have been utilized in consumer cars since 2015. Since then, Tesla's Autopilot has demonstrated the capability to navigate motorways, assuming both lateral and longitudinal control [12]. The subsequent year saw other automakers, including Audi[13], BMW [14], and Daimler [15], releasing Level 2 production vehicles.

As of the current writing, this reflects the industry's progression. The release of Level 3 and higher vehicles is marked by announcements and commitments, contingent on the resolution of regulatory considerations. For instance, Audi's 2019 A8 is purportedly capable of Level 3, but the system is yet to be delivered [14]. Similarly, BMW has outlined its plans for 2021. Discussions on level 4 and level 5 remain distant, lacking further developments.

Progress toward the ambitious objective of achieving fully autonomous driving is underway. Its Technology Readiness Level (TRL) [16] has surpassed level 4 [17], indicating that it has entered the industrial development phase, necessitating the establishment of standards and regulations. Consequently, the assessment of its eventual market launch, being one of the many sub-topics of autonomous driving technology that is still unclear, remains an open question for current research.

1.1.2 Autonomous Driving software development

Autonomous driving software development refers to the process of designing, creating, and implementing software systems that enable vehicles to operate autonomously without human intervention. This field combines various technologies, algorithms, and components to achieve safe and efficient self-driving capabilities. Key aspects of autonomous driving software development include:

- **Perception and Sensor Fusion:** Autonomous vehicles are equipped with sensors such as LiDAR, Radar, Cameras, and ultrasonic sensors to perceive their environment. Software is developed to process data from these sensors and create a detailed understanding of the vehicle's surroundings.
- **Localization:** Software algorithms are used to determine the precise location and orientation of the vehicle in real-time, often through a combination of sensor data and high-definition maps.
- **Mapping:** High-definition maps are essential for autonomous driving, and software is used to create and maintain these maps. These maps provide information about road geometry, lane markings, traffic signs, and more.
- **Path Planning:** Path planning algorithms are responsible for generating a safe and efficient route for the vehicle to follow, taking into account the vehicle's current position, destination, and the surrounding environment.
- **Control Systems:** Control software ensures that the vehicle follows the planned path, adjusting throttle, brake, and steering inputs to navigate safely and smoothly.
- **Machine Learning and Artificial Intelligence:** Machine learning and AI techniques are often used to improve decision-making, adapt to changing road conditions, and handle complex scenarios.
- **Human-Machine Interface (HMI):** User interfaces are developed to communicate with passengers and provide information about the vehicle's status and intentions.
- **Safety and Redundancy:** Safety-critical software components and redundancy systems are integrated to ensure the vehicle's safety and the ability to handle unexpected situations.
- **Testing and Validation:** Extensive testing, including simulation and real-world testing, is a crucial part of software development to ensure the system's reliability and safety.

- **Regulatory Compliance:** Developers must adhere to regulations and standards specific to autonomous vehicles and their use on public roads.

Autonomous driving software development is a multidisciplinary field that combines expertise in software engineering, artificial intelligence, sensor technology, and vehicle dynamics. It aims to create safe and efficient self-driving systems that can navigate roads and interact with their environment autonomously.

1.2 Simulation for Autonomous Driving Software Development

Simulation plays a crucial role in the development of autonomous driving software. It provides a controlled environment for testing and validating various aspects of the software without the need for real-world scenarios. Here are key aspects of simulation for autonomous driving software development:

- **Scenario Testing:** Simulations allow developers to create and replicate diverse driving scenarios, including challenging and rare situations, to evaluate the software's performance.
- **Virtual Environments:** Developers can create virtual environments that mimic real-world conditions, such as different weather conditions, road types, and traffic scenarios, to assess the software's adaptability.
- **Sensor Simulation:** Simulating sensor inputs, including cameras, lidar, radar, and other perception sensors, helps test the software's ability to interpret and respond to the environment.
- **Traffic Simulation:** Simulating other vehicles and pedestrians in the environment enables testing of the software's ability to navigate complex traffic situations.
- **Behavior Prediction:** Simulations can be used to predict the behavior of other entities on the road, allowing the software to anticipate and respond to dynamic situations.
- **Edge Cases:** Simulating edge cases and extreme scenarios helps uncover potential vulnerabilities or weaknesses in the software and ensures robustness in real-world conditions.
- **Machine Learning Training:** Simulated environments are used for training machine learning models that are part of the autonomous driving software, providing a safe and controlled setting for learning.

- **Regulatory Compliance:** Simulations help assess the software's compliance with regulatory requirements and safety standards before real-world testing and deployment.
- **Cost-Efficiency:** Simulations offer a cost-effective means of testing and refining software algorithms without the expenses and risks associated with real-world testing.
- **Continuous Improvement:** Simulation allows for continuous testing and improvement of the software throughout the development lifecycle, addressing issues and enhancing performance iteratively.

By leveraging simulation, developers can accelerate the development process, enhance the safety and reliability of autonomous driving software, and ensure readiness for real-world deployment.

1.3 V-cycle software development

"V-cycle software development" refers to a software development process that is shaped like the letter 'V' when represented graphically. It is a model that emphasizes a systematic and structured approach to software development. This life cycle initiates with the requirements phase and progresses through multiple stages, culminating in the verification and validation phases.

Here's an overview of the V-cycle software development process:

- **Requirements Analysis (Left Side of the 'V'):** The left side of the "V" represents the initial stages of the project. It begins with requirements analysis, where the project team defines the system and software requirements. This phase involves close interaction with stakeholders to gather and document all necessary specifications.
- **System Design:** Once the requirements are established, the system design phase involves creating a high-level design that outlines the overall system architecture and how the software components will interact with each other and the hardware.
- **Software Design:** Following system design, the software design phase focuses on creating detailed software specifications and designs. This includes defining the data structures, algorithms, and interfaces that will be used in the software.
- **Coding and Implementation:** The coding and implementation phase involves writing the actual source code of the software based on the design specifications. This is where the software is developed.

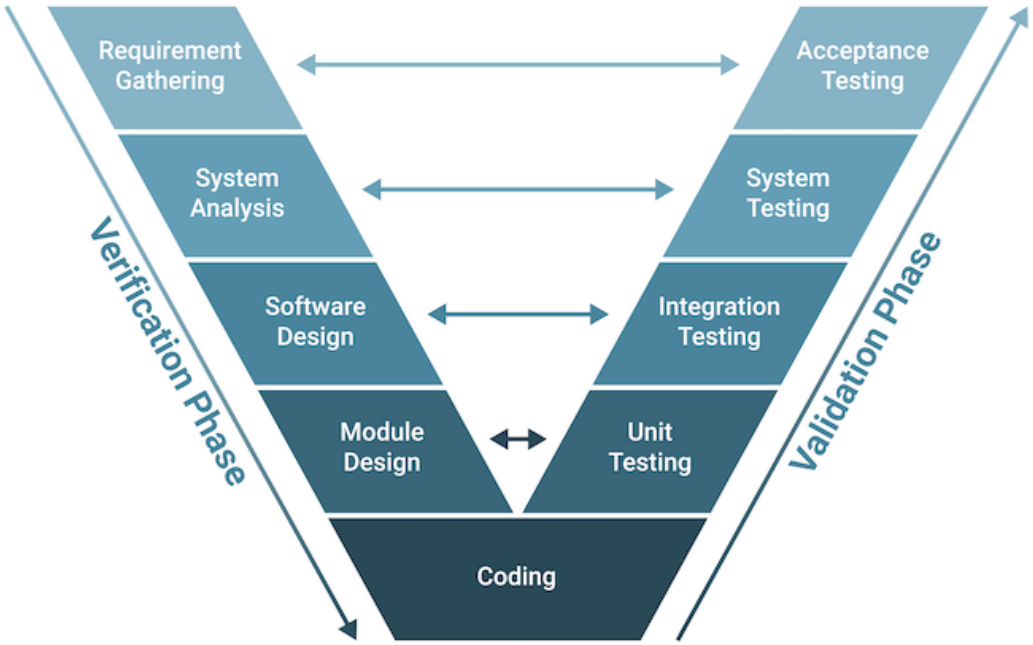

Figure 1.2: V-cycle software development process

- **Unit Testing:** After coding, unit testing is performed to validate individual software components (units). Each component is tested in isolation to ensure that it works correctly.
- **Integration Testing:** The right side of the "V" starts with integration testing, where the individual software units are integrated and tested as a whole to verify that they work together as intended.
- **System Testing:** System testing involves testing the entire system as a whole, ensuring that it meets the specified requirements and functions correctly.
- **Validation and Verification (Right Side of the 'V'):** The right side of the "V" represents the final stages of the project. Verification ensures that the software is built correctly (meeting the requirements), and validation ensures that the right software was built (meeting the user's needs). These phases often involve validation against customer needs, regulatory compliance, and safety standards.
- **Acceptance Testing:** In acceptance testing, the software is tested against the user's acceptance criteria. This phase ensures that the software satisfies

the end users' requirements and expectations.

- **Deployment and Maintenance:** Once the software passes all testing phases, it can be deployed for actual use. Maintenance involves ongoing support, updates, and bug fixes.

The V-cycle methodology is particularly suited for industries where safety and reliability are critical, as it emphasizes extensive testing and validation to ensure that the software meets the intended requirements and functions correctly. It provides a structured approach to software development, with an emphasis on thorough testing and verification throughout the entire project life cycle.

The V-cycle process is important in software development, particularly in the context of Automotive Software Development (AD), for several reasons:

- **Clarity and Structure:** The V-cycle provides a clear and structured approach to software development. It breaks down the development process into distinct phases, making it easier for development teams to understand and follow.

- **Early Detection of Defects:** The V-cycle emphasizes testing at each stage, from unit testing to system testing. This approach allows for the early detection and correction of defects, reducing the likelihood of more serious issues arising later in the development process or after deployment.

- **Traceability:** The V-cycle establishes a clear relationship between each development phase and its corresponding testing phase. This traceability ensures that each requirement is addressed and tested, providing a comprehensive verification process.

- **Efficient Use of Resources:** By catching defects early in the development process, the V-cycle helps in avoiding costly rework and modifications in later stages. This efficiency is crucial in automotive software development, where precision and reliability are paramount.

- **Validation against Requirements:** The V-cycle ensures that the final software product is validated against the initial requirements. This helps in confirming that the software meets the specified criteria and performs as intended in the automotive context.

- **Risk Mitigation:** The phased approach of the V-cycle allows for the identification and mitigation of risks at different stages of development. This is particularly important in safety-critical systems, such as those found in automotive applications.

- **Regulatory Compliance:** In the automotive industry, there are often strict regulatory standards and safety requirements. The V-cycle, with its emphasis on testing and validation, provides a framework for compliance with these standards.
- **Incremental and Iterative Development:** The V-cycle supports an incremental and iterative development approach, allowing for the progressive refinement of the software through multiple cycles. This is beneficial in addressing evolving requirements and accommodating changes in the development process.

In summary,the V-cycle process is important in Automotive Software Development because it promotes a systematic,thorough,and risk-mitigated approach,ultimately contributing to the development of reliable and high-quality software for automotive systems.

1.4 Regulations for AD software simulation and testing

The United States pioneered autonomous driving technology, becoming the first country to enact laws related to its implementation.In contemporary times, the concept of Autopilot, often associated with Tesla, has become widely recognized. Tesla, an American company, has played a prominent role in popularizing and implementing autonomous driving technologies. In September 2016, the U.S. National Economic Council and the U.S. Department of Transportation (USDOT) issued the Federal Automated Vehicles Policy. These standards outline how automated vehicles should respond in the event of technology failures, address passenger privacy concerns, and establish protocols for passenger safety in case of accidents. The purpose of these federal guidelines is to prevent a disparate set of state laws while ensuring they do not unduly inhibit innovation.And many states have enacted laws or issued executive orders pertaining to autonomous vehicles.

The European Union (EU) has implemented legislation governing the type approval of Automated Driving Systems (ADS) for fully automated vehicles. In July 2022, the "Vehicle General Safety Regulation" came into effect, establishing a legal framework for approving automated and fully driverless vehicles (Level 3 and above) within the EU. The EU Commission's delegated regulation includes specific requirements with regulatory distinctions among various types of fully automated vehicles. This reflects the stringent regulations in the EU concerning autonomous driving, emphasizing a dedicated legal framework for ensuring safety in autonomous driving technologies.several European countries have implemented legislative and

regulatory frameworks for self-driving cars and transportation systems. Notably, Germany has introduced the Federal Act Amending the Road Traffic Act and the Compulsory Insurance Act, while France has enacted the Mobility Orientation Law.

In summary, the majority of countries and regions have implemented laws regarding autonomous driving, with a particular emphasis on passenger safety. In the United States, safety is a paramount concern reflected in regulations that focus on rigorous testing and validation of autonomous technology to meet stringent safety standards. Ensuring the safety of self-driving cars remains a top priority. In the European Union (EU) and the United Nations Economic Commission for Europe (UNECE), collaborative efforts have been made to establish safety standards for autonomous vehicles, defining the conditions for their deployment. Consequently, developing robust test scenarios is crucial to effectively enhance the safety of autonomous driving. When it comes to scenario creation for simulation testing, it's often the responsibility of developers and manufacturers to ensure that their testing covers a diverse range of scenarios, including edge cases and challenging conditions. Adhering to safety and functional standards is crucial in this process. It's important to note that the field of autonomous vehicles is rapidly evolving, and new regulations and standards may be introduced. Organizations involved in autonomous vehicle development should stay informed about the latest developments and work in collaboration with relevant regulatory bodies to ensure compliance and safety.

1.5 Thesis Outline

This thesis works as follows :

- **Chapter 2** presents the theoretical background of the topics presented,with a particular focus on AD(Autonomous Driving) software simulation and testing.Following that, it outlines the simulation requirements and explores several widely-used simulation software options, providing a comparative analysis of their respective advantages and disadvantages.
- **Chapter 3** is devoted to detailing the design of the system architecture and the implementation of the proposed method. It begins with defining scenes and introducing tags, followed by the categorization of scenes. Subsequently, the report introduces various security metrics and proceeds to design scenarios aligned with these security metrics.
- **Chapter 4** presents the setup of the validation process for Autonomous Driving (AD) software simulation scenarios and the corresponding test results.

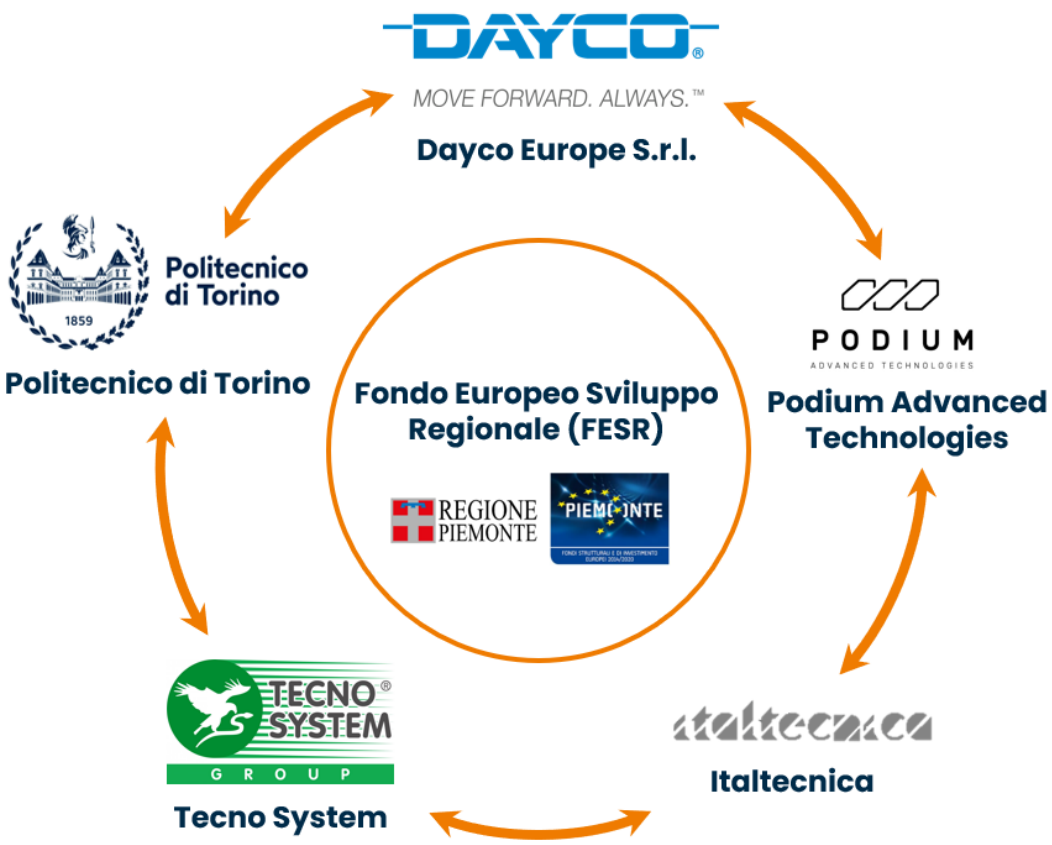

Figure 1.3: PITEF Project

- **Chapter 5** is the final chapter ,where conclusions and future works are reported.

Chapter 2

Theoretical Background

Before delving into the proposed method, this chapter is dedicated to providing a theoretical background for the thesis work. The autonomous driving industry has witnessed continuous development, leading to the emergence of various research findings and solutions. While this project does not aim for fully autonomous driving, its foundational principles align closely with environmental awareness and data processing.

To lay the groundwork, this thesis will first introduce the autonomous driving software pipeline. Subsequently, it will elaborate on the simulation requirements. Finally, it will delve into an examination of several popular simulation tools, comparing their respective advantages and disadvantages.

2.1 Autonomous Driving Software Pipeline

The Autonomous Driving Software Pipeline in the context of simulation for autonomous driving tests is connected to various components that collectively enable the simulation process. This pipeline encompasses software modules and algorithms designed for tasks such as perception, decision-making, and control, all crucial for simulating the behavior of an autonomous vehicle. The integration of these components within the pipeline allows for comprehensive testing and evaluation of autonomous driving capabilities in a simulated environment.

Here is a high-level overview of the typical components in an autonomous driving software pipeline:

Perception:

- **Sensor Input:** In this stage, data is collected from various sensors such as Cameras, Lidar, Radar, and other environmental sensors.

- **Sensor Fusion:** The data from different sensors is integrated and processed to create a comprehensive and accurate representation of the vehicle's surroundings.
- **Object Detection and Recognition:** Algorithms analyze the sensor data to identify and classify objects in the environment, such as other vehicles, pedestrians, and obstacles.

Localization:

- **Mapping:** The vehicle's position is determined by comparing the sensor data with pre-existing maps or by creating a real-time map of the environment.
- **Sensor-based Localization:** Combining sensor data with mapping information helps the vehicle accurately determine its position in relation to the surroundings.

Path Planning:

- **Route Planning:** The system determines the optimal route or path based on the destination, traffic conditions, and other relevant factors.
- **Trajectory Planning:** Once the route is established, the system plans a trajectory that considers the vehicle's dynamics and the surrounding environment.

Control:

- **Vehicle Control:** The control system executes the planned trajectory by adjusting the vehicle's speed, steering, and other control parameters.
- **Adaptive Cruise Control (ACC) and Lane Keeping Assist (LKA):** These systems assist in maintaining a safe distance from other vehicles and staying within the lanes.

Decision-Making:

- **Behavior Planning:** The vehicle makes high-level decisions, considering traffic rules, safety requirements, and the overall driving strategy.
- **Risk Assessment:** The system assesses potential risks and adapts the driving behavior accordingly.

Human-Machine Interface (HMI):

- **Feedback and Communication:** The HMI provides feedback to the vehicle occupants and communicates the vehicle's intentions to pedestrians and other road users.

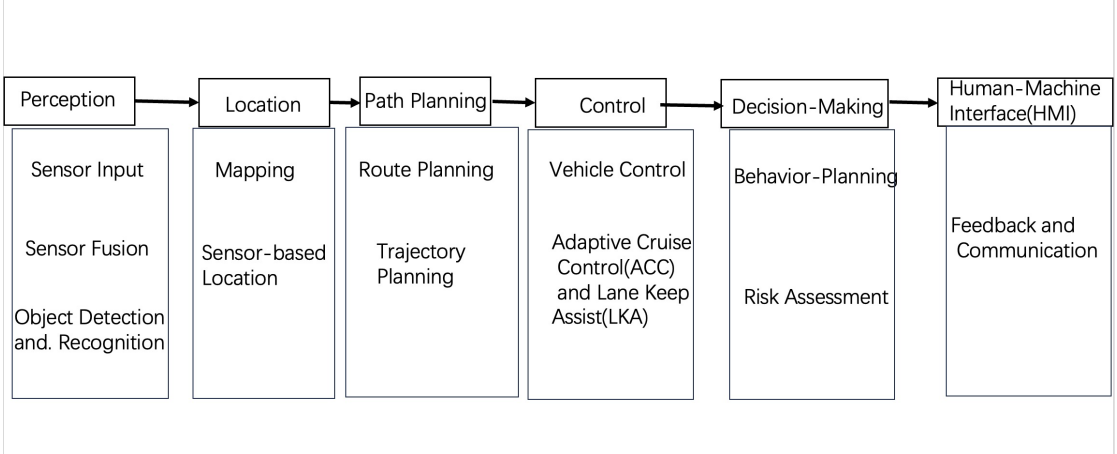

Figure 2.1: Autonomous Driving Software Pipeline

The architecture and components of the autonomous driving software pipeline can vary among different manufacturers and developers of autonomous vehicles. These components work together to enable self-driving vehicles to navigate, make decisions, and respond to their environment without human intervention.

2.2 Simulation Requirements

Autonomous Driving Software Simulation Requirements refers to the requirements necessary for developing and testing autonomous driving software in a simulated environment. These requirements encompass hardware, software, data, and other aspects to ensure that the simulation environment accurately replicates real-world road scenarios and various conditions for the development and testing of autonomous driving software.

Here are some potential requirements for autonomous driving software simulation:

- **High-Performance Computing Hardware:** The simulation environment requires powerful computing hardware, including high-performance CPUs and GPUs, to process a large amount of sensor data and complex algorithms in real-time.

- **Sensor Simulation:** The simulator needs to be capable of simulating various sensors such as cameras, LiDAR, radar, and ultrasonic sensors to generate accurate perception data.

- **High-Definition Maps and Road Data:** The simulation environment needs accurate high-definition map data, including road layouts, lane markings, traffic signals, and road signs, to simulate real road scenarios.
- **Weather and Lighting Simulation:** The simulator should be able to simulate different weather conditions (e.g., rain, snow, fog) and various lighting conditions (e.g., day and night) to test the robustness of the autonomous driving software.
- **Traffic Flow and Pedestrian Simulation:** The simulation environment needs to simulate the behavior of other vehicles, traffic flow, and dynamic pedestrian behavior to test the performance of autonomous vehicles in complex traffic environments.
- **Real-Time Physics Engine:** The simulator needs a real-time physics engine to accurately simulate the vehicle's dynamics, braking, and suspension characteristics.
- **Data Recording and Playback:** The simulation environment should be able to record data from simulations for subsequent analysis and improvement.
- **Diversity and Edge Cases:** The simulation environment should be capable of generating a variety of scenarios, including extreme and edge cases, to test the software's responsiveness.
- **Virtual Vehicle and Scene Generation Tools:** Developers need the ability to create virtual vehicles and road scenes to test software performance in different scenarios.
- **Safety and Privacy:** Safety and data privacy should be critical considerations in the simulation environment to ensure data is secure and protected from unauthorized access and risks.

These requirements help ensure that autonomous driving software undergoes comprehensive testing and validation in a simulated environment, ultimately improving the reliability and safety of autonomous driving systems.

2.3 Simulation Tools

Along with the development of autonomous driving, simulation test software for autonomous driving is also booming. The following section introduces a few popular test software and compares their advantages and disadvantages.In the following section, this thesis will provide a comprehensive analysis of three simulation software

platforms: SCANeR, CARLA, and MATLAB. The assessment will encompass various aspects, including API compatibility, ROS integration, and the support for an Autonomous Driving sensor suite. This exploration is particularly relevant as one of the primary objectives of this article is to experiment with data transfer between the Robot Operating System (ROS) and simulation scenarios. This experimentation aims to enhance the testing capabilities of Advanced Driver Assistance Systems (ADAS).

2.3.1 CARLA

CARLA Simulator is an open-source simulator for autonomous driving research.It provides a platform for testing and developing autonomous vehicle algorithms in a realistic and controlled environment. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation and much more.

CARLA has its highlighted features:

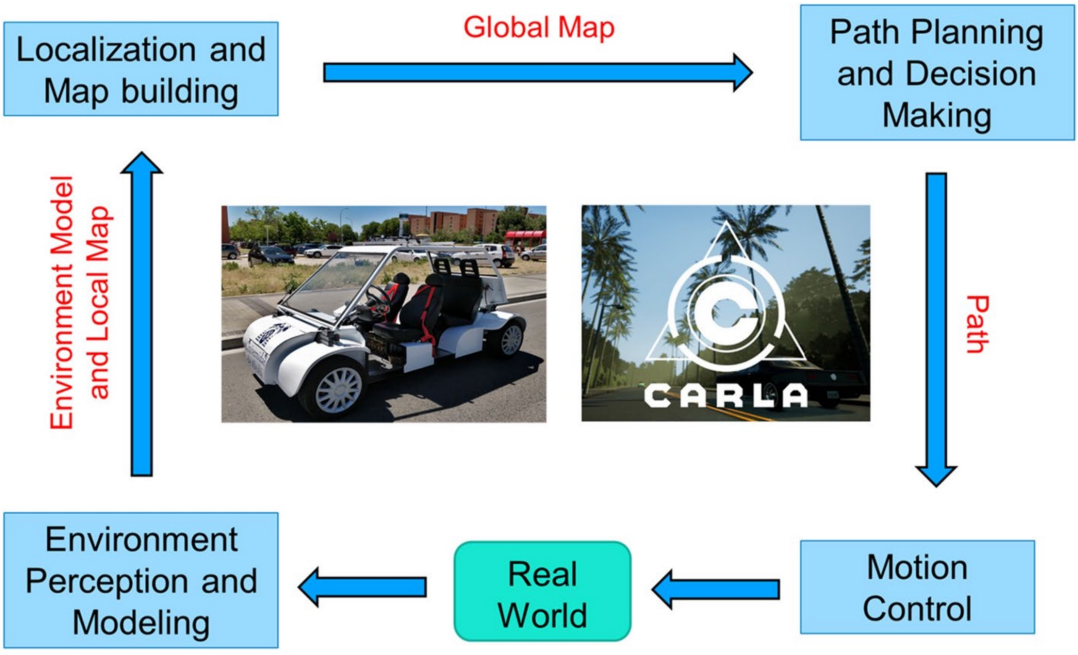

Figure 2.2: CARLA

- **Scalability via a server multi-client architecture:** multiple clients in the same or in different nodes can control different actors.

- **Flexible API:** CARLA exposes a powerful API that allows users to control all aspects related to the simulation, including traffic generation, pedestrian behaviors, weathers, sensors, and much more.
- **Autonomous Driving sensor suite:** users can configure diverse sensor suites including LIDARs, multiple cameras, depth sensors and GPS among others.
- **Fast simulation for planning and control:** this mode disables rendering to offer a fast execution of traffic simulation and road behaviors for which graphics are not required.
- **Maps generation:** users can easily create their own maps following the OpenDrive standard via tools like RoadRunner.
- **Traffic scenarios simulation:** our engine ScenarioRunner allows users to define and execute different traffic situations based on modular behaviors.
- **ROS integration:** CARLA is provided with integration with ROS via our ROS-bridge
- **Autonomous Driving baselines:** we provide Autonomous Driving baselines as runnable agents in CARLA, including an AutoWare agent and a Conditional Imitation Learning agent.

CARLA's Core features:

- **Quickstart:** Getting started with CARLA is easy, this guide will show you how to install and run the simulator.
- **Actors:** CARLA's actors are entities that interact within the simulation like vehicles, pedestrians and traffic signals. Get to know them here.
- **Sensors:** CARLA boasts an impressive array of models of real world sensors like cameras, LIDAR and RADAR. The simulator also gives access to privileged information such as ground truth semantic segmentation and depth information.
- **Traffic Manager:** CARLA's Traffic Manager controls NPCs to challenge your autonomous driving agent.
- **ROS bridge:** CARLA's ROS bridge enables seamless connection with the Robot Operating System.

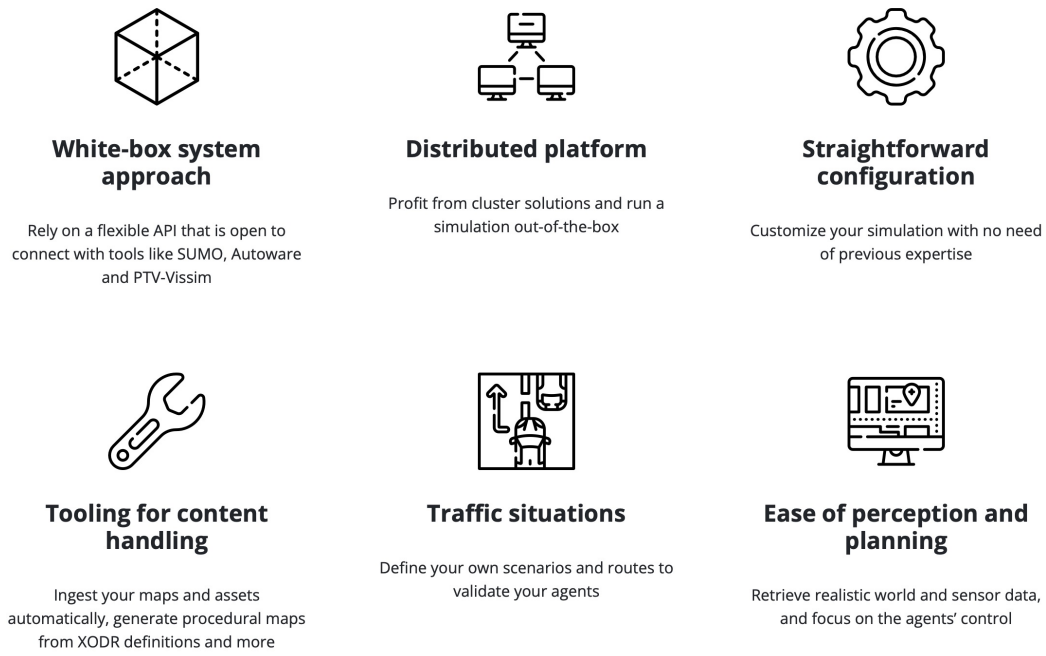

Figure 2.3: CARLA's advantages

Figure 2.4: MATLAB

2.3.2 Driving Scenario designer from MATLAB

Matlab is a high-level programming language and an integrated development environment (IDE) primarily used for numerical and scientific computing.

Matlab, short for Matrix Laboratory, is a high-performance programming language and environment primarily used for numerical computing, data analysis, and visualization. It is widely used in academia, industry, and research for a variety

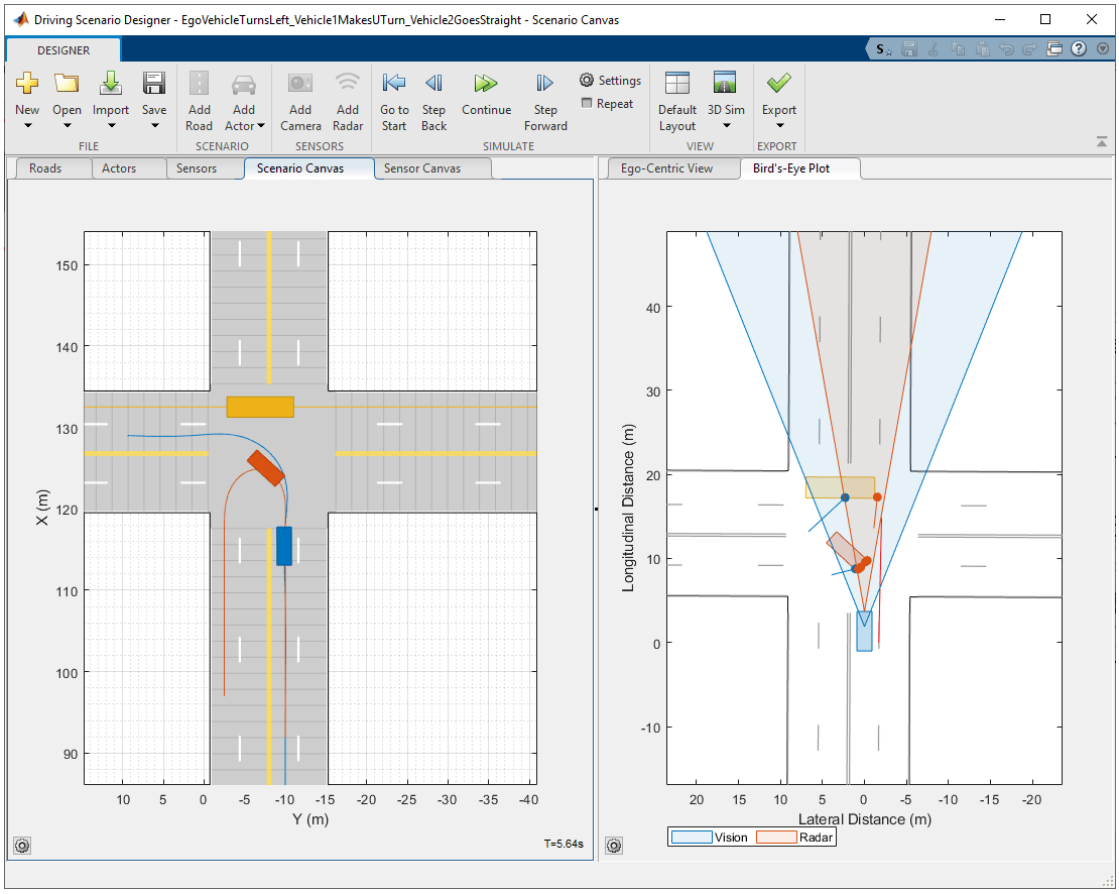

Figure 2.5: Driving Scenario Designer app of MATLAB

of applications, including signal processing, image processing, machine learning, control systems, and more.

The key point is that MATLAB offers a simulation environment for Autonomous Driving. This environment enables the design of driving scenarios, the configuration of sensors, and the generation of synthetic data.

The Driving Scenario Designer app enables you to design synthetic driving scenarios for testing your autonomous driving systems.

Using the app, you can:

- Create road and actor models using a drag-and-drop interface.
- Configure vision, radar, lidar, INS, and ultrasonic sensors mounted on the ego vehicle. You can use these sensors to generate actor and lane boundary detections, point cloud data, and inertial measurements.

- Load driving scenarios representing European New Car Assessment Programme (Euro NCAP) test protocols and other prebuilt scenarios.
- Export road network and static actors to the RoadRunner HD Map file format.
- Export synthetic sensor detections to MATLAB
- Generate MATLAB code of the scenario and sensors, and then programmatically modify the scenario and import it back into the app for further simulation.
- Generate a Simulink model from the scenario and sensors, and use the generated models to test your sensor fusion or vehicle control algorithms.

But also every software has its drawbacks and limitations.Here is the limitations:

Clothoid Import/Export Limitations

- Driving scenarios presently support only the clothoid interpolated roads. When you import roads created using other geometric interpolation methods, the generated road shapes might contain inaccuracies

Sensor Import/Export Limitations

- Driving scenarios presently support only the clothoid interpolated roads. When you import roads created using other geometric interpolation methods, the generated road shapes might contain inaccuracies

Euro NCAP Limitations

- Scenarios of speed assistance systems (SAS) are not supported. These scenarios require the detection of speed limits from traffic signs, which the app does not support.

Heading Limitations to Road Group Centers

- When you load a drivingScenario object containing a road group of road segments with specified headings into the Driving Scenario Designer app, the generated road network might contain inaccuracies. These inaccuracies occur because the app does not support heading angle information in the Road Group Centers table.

Parking Lot Limitations

- The importing of parking lots created using the parkingLot function is not supported. If you import a scenario containing a parking lot into the app, the app omits the parking lot from the scenario.

Figure 2.6: SCANeR with 3D UXD engine

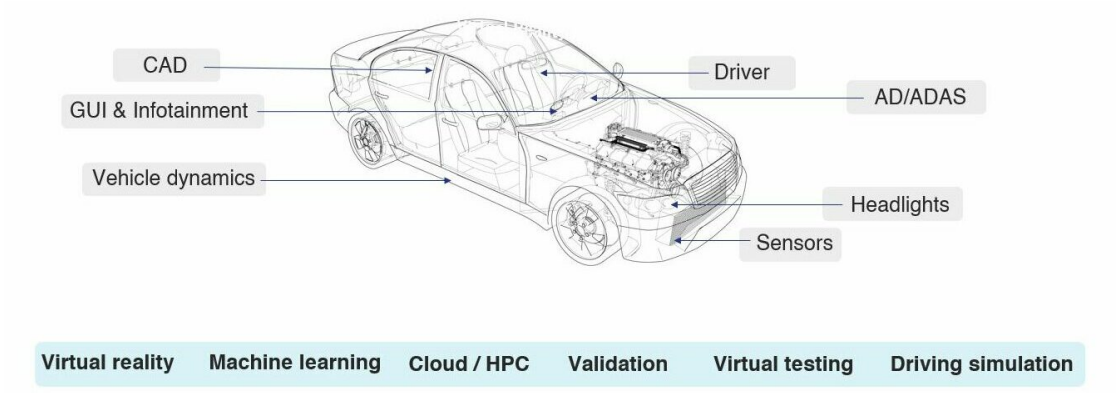

Figure 2.7: Different Application Cases with SCANeR

2.3.3 SCANeR

SCANeR studio is a simulation software developed by the company AVSimulation. It is used in the automotive industry for virtual testing and simulation of various aspects of vehicle behavior and performance. The software allows users to create realistic virtual environments, simulate driving scenarios, and test the functionality of different vehicle systems.

Key features of SCANeR studio include the ability to simulate real-world driving conditions, assess vehicle safety, test advanced driver assistance systems (ADAS), and evaluate the performance of autonomous vehicles. It is a valuable tool for automotive engineers and researchers to conduct virtual tests before physical prototypes are built, saving time and resources in the product development process.

It is the most complete solution on the market with its full graphical environment: it allows users to configure, prepare, run simulations and analyze results. Flexible and versatile, it can be used by different teams working on different aspects of the same project (headlights, AD/ADAS, HMI, etc.). Moreover, SCANeR is the only simulation software that can be used throughout the V-cycle: from the validation of the requirements expression to acceptance tests ViL (Vehicle in the Loop).

The standardization around SCANeR is a competitive advantage. Indeed, the generalization of its use favors the sharing and reuse of experiments by different teams. Moreover, experience shows that the use of SCANeR eliminates the development and maintenance of tools that are supposed to allow teams to share and exchange data or simulation results between them. Finally, the widespread use of SCANeR ensures that best practices are shared, employee skills are increased and efficiency is improved.

SCANeR is an open software, the development kit is provided with all configurations at no extra cost. Thanks to this kit, customers can adapt SCANeR to their needs, to their existing tools or to third-party software (compatible with Windows and Linux). Moreover, it can be easily updated

XiL testing with SCANeR enables the early detection and resolution of issues before physical hardware becomes available. This approach results in cost savings by minimizing the necessity for testing on real hardware. Additionally, hazardous or uncontrollable scenarios can be safely tested within a simulated environment.

While there are many benefits to sensor fusion XIL Test with SCANeR, it serves as a crucial component of our testing process. Firstly, it facilitates the connection between software simulation and hardware simulation data. Additionally, it allows for component and sub-system level functional tests, in-lab optimization before test drives, and reliability and regression testing for ADAS functions. Furthermore, it offers customization for ADAS sensor configurations, boasts a future-proof modular architecture for evolving sensors, and provides support for evolving standards such as EuroNCAP and SOTIF, among others.

In summary, SCANeR software perfectly aligns with our requirements. It stands

out as the only simulation software applicable throughout the entire V-cycle. Given that this thesis follows a V-cycle approach for the process, SCANeR facilitates testing and validation of scenarios seamlessly.

Chapter 3

Methodology

This section aims to clarify the terms employed in this report, reducing ambiguity in their usage. Initially, the concept of a scenario is introduced in Section 3.1. Subsequently, Section 3.2 delves into scenario categories, which represent the abstraction of scenarios. As detailed in Section 3.3, scenario categories are characterized by a set of tags. Section 3.4 presents the tags utilized to describe the scenario categories defined in Section 4. Finally, Section 3.5 introduce the existing assessment metrics, it is of paramount importance to ADAS test scenarios.

3.1 Scenario

In defining a scenario, this thesis presents the term's definition, which is more applicable to the context of assessing Autonomous Vehicles (AVs). Before presenting the definition of the term "scenario," a few key concepts are introduced.

- **Ego vehicle:** The term "ego vehicle" pertains to the perspective from which the world is observed. Typically, it denotes the vehicle that perceives the world through its sensors or the vehicle tasked with a specific function. The ego vehicle is commonly denoted as the system-under-test or the vehicle-under-test (VUT) – in our context, the Autonomous Vehicle (AV)-under-test.

- **Activity:** An activity refers to the behavior of a specific mode within a system. For instance, an activity could be described by labels such as 'braking' or 'changing lanes'.

- **Event:** An event marks the instant in time when a transition of state occurs, signifying that before and after the event, the state corresponds to two distinct activities. For instance, an event could be described by the label 'initiate braking'.

- **Actor:** An actor is an element within a scenario that operates on its own behalf. Examples of actors in a scenario include the ego vehicle and other road users.
- **Static environment:** The static environment encompasses elements in a scenario that remain unchanged throughout its duration. This includes geo-spatially stationary elements, such as the infrastructure layout, road configuration, and road type. Additionally, the presence of buildings near the roadside, serving as view-blocking obstructions, is considered part of the static environment.
- **Dynamic environment:** In contrast to the static environment, the dynamic environment pertains to the elements in a scenario that undergo changes within its time frame. The dynamic environment is delineated by activities, illustrating how the state of actors evolves over time. In practical terms, the dynamic environment predominantly comprises moving actors (excluding the ego vehicle) that are pertinent to the ego vehicle.
- **Conditions :** Crucial to the scenario description are the weather and lighting conditions, as they significantly impact the ego vehicle. For instance, precipitation can markedly affect sensor performance and vehicle dynamics. Lighting conditions also play a crucial role in sensor performance; for example, cameras may struggle to detect and classify objects during nighttime in the absence of artificial light. While one may debate whether light and weather conditions are dynamic, it is reasonable to assume that these conditions, in most cases, do not undergo significant changes during the scenario's time frame.

The definition of "Scenario" is taken from [18]:

Definition 3.1(Scenario). A scenario is a quantitative description of the ego vehicle, its activities and/or goals, its dynamic environment (consisting of traffic environment and conditions) and its static environment. From the perspective of the ego vehicle, a scenario contains all relevant events.

3.2 Scenario category

While a scenario is inherently a quantitative description, there also exists a qualitative counterpart known as a scenario category. The qualitative description serves as an abstraction of the quantitative scenario.

Scenarios are classified into scenario categories, with multiple scenarios potentially falling into a single category, and vice versa—a scenario may belong to one

or multiple categories. To illustrate, consider all scenarios occurring during the nighttime, constituting the scenario category "Nighttime". Similarly, all scenarios featuring sunny are categorized under "Sunny" as depicted in Figure 3.1. A scenario occurring during the daytime without sunny does not align with any defined scenario categories. Conversely, a scenario transpiring during daytime with rain falls into both the "Nighttime" and "Sunny" scenario categories.

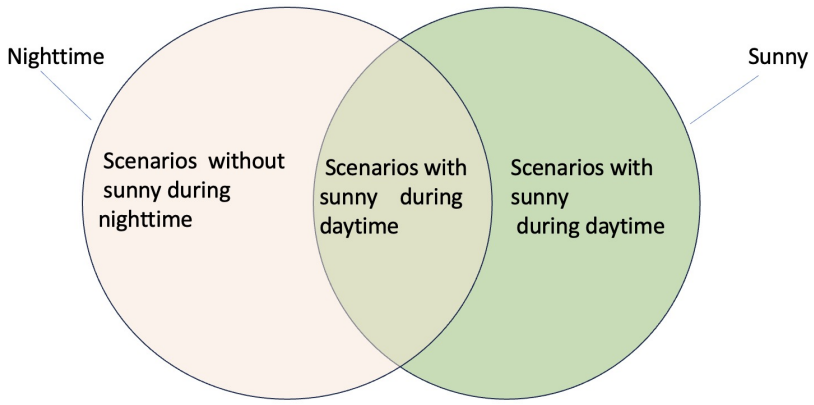

Figure 3.1: Scenario Categories (Nighttime and Sunny)

Furthermore, a scenario category can encompass another scenario category. For instance, in our prior example involving the scenario categories "Nighttime" and "Sunny", these categories include the scenario category "Nighttime and sunny".

3.3 Tags

It is proposed to associate scenarios with relevant tags that qualitatively describe the scenario. The tags assigned to a scenario determine its inclusion in specific scenario categories. For instance, if a scenario unfolds during nighttime, it will be annotated with the property “tags=Nighttime”. Consequently, the scenario is automatically identified as an instance of the scenario category “Nighttime” distinguished by the singular tag “Nighttime”.The use of these tags offers several advantages:

- Scenarios are not required to be directly categorized, saving time, especially with a large number of scenario categories.
- When a scenario category with known tags is added to the scenario category database, it becomes easy to identify which scenarios belong to it by inspecting the tags of the scenarios..

- Selecting scenarios from a scenario database or library becomes straightforward using tags or a combination of tags.

The balance between generic and specific scenario categories, resulting in variability among scenarios within a category, is crucial. Different systems may have varying interests, with some focusing on specific scenarios and others on a diverse set. To address this, tags are organized in trees , where each layer represents a different abstraction level.

The report proposes a first list of tags and trees of tags on Figure 3.2.Certainly, incorporating numerous labels enhances the comprehensiveness of the test, but it also escalates the workload significantly. In this thesis, we strive to judiciously select the necessary labels to set up the scenarios.In the following section we describe the tags we have selected.

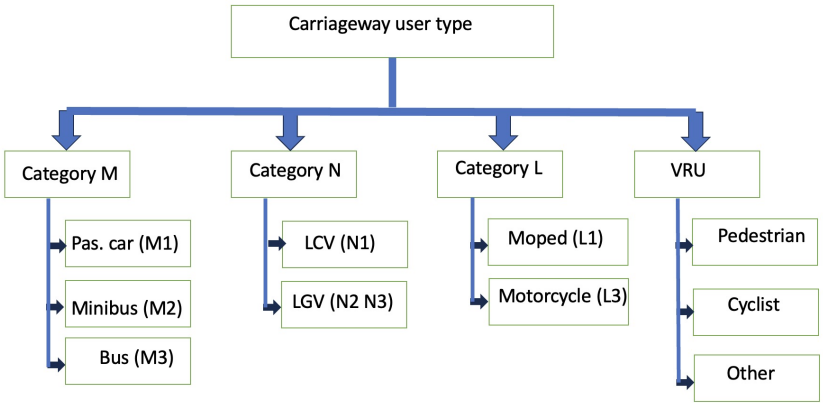

Figure 3.2: Tag Trees

3.4 Selection of tags and trees of tags

The definition of tags and trees of tags will be presented subsequently for the dynamic environment, for the static environment , and finally for the conditions.This thesis delves into practical scenario design, considering factors such as manpower. To streamline the scenarios, we aim to simplify them as much as possible, yet ensuring that the designed scenarios meet project requirements. Consequently, this thesis primarily focuses on the ego car and leader car, with the possibility of adding, at most, one other vehicle. Moreover, the behavior of these additional vehicles is limited to cut-in and cut-out maneuvers.As shown in the figure 3.3.

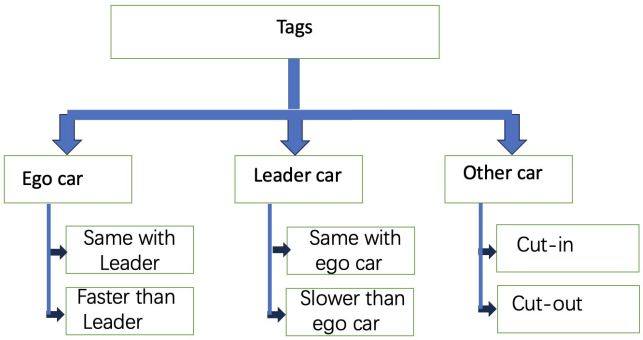

Figure 3.3: Tags Chosen

3.5 Existing Assessment Metrics

Following the introduction of a test case, the assessment of results becomes crucial. A key aspect of assessing autonomous driving (AD) is the definition of metrics. In the subsequent sections, we elaborate on existing metrics from the literature used to assess driving tests. Given that accident risk is paramount for evaluating the safety of an AD system, this thesis places a primary focus on such metrics. To provide a comprehensive overview, we also briefly touch upon insights into the other two categories of metrics.

3.5.1 Safety Related Assessment Metrics

Numerous assessment metrics pertaining to traffic safety have been identified in the literature, with many focusing on measuring the time until an impending incident within a given scenario. The subsequent sections present various existing metrics from the field of Situation Threat Assessment (STA).

Time-to-Collision (TTC)

The primary and widely used metric involves measuring the time remaining until an impending collision between two vehicles. To illustrate, a vehicle following scenario is illustrated in Figure 3.4.In this scenario, the blue vehicle, with a velocity denoted as v_{follow} and acceleration as a_{follow}, is trailing the orange vehicle, which has a velocity v_{lead} and acceleration a_{lead}. The Time-to-Collision (TTC) is calculated as follows:

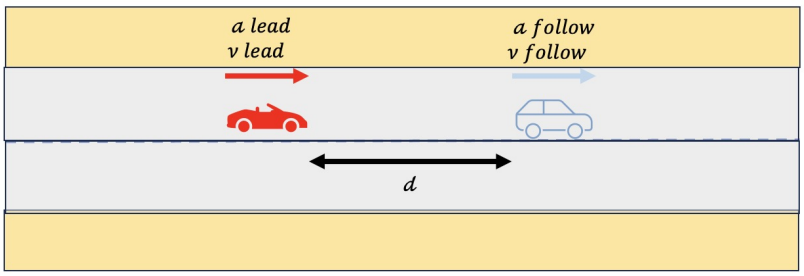

Figure 3.4: TTC

$$v_{rel} = v_{follow} - v_{lead} \tag{3.1}$$

$$TTC = \frac{d}{v_{rel}} \qquad v_{rel} \geqslant 0\frac{m}{s} \tag{3.2}$$

Notably, the TTC is only defined for a positive relative velocity v_{rel}, as there will never be a collision if the following vehicle is slower. The simplified formula in Equation 3.2 only applies to this exact scenario. If the distance d calculates along the vehicles' predicted paths to the collision point, the formula applies to arbitrary scenes. Still, only a single future path for TO (Traffic Object)s is considered, and uncertainties in their driving intention are ignored.

The Time-to-Collision (TTC) is specifically defined for a positive relative velocity v_{rel} indicating that a collision is only relevant when the following vehicle is moving faster than the lead vehicle. The simplified formula in Equation 3.2 is tailored to the described scenario. For a more general application to arbitrary scenes, the distance is calculated along the predicted paths of the vehicles to the collision point. It's important to note that this formula considers only a single future path for the Time Origins (TOs) and doesn't account for uncertainties in their driving intentions.

Enhanced Time-to-Collision (ETTC)

The Enhanced Time-to-Collision (ETTC) extends the standard TTC by incorporating the acceleration of the participating vehicles, aiming to enhance the precision of the metric. The calculation of ETTC is based on the standard TTC and is expressed as follows:

$$a_{rel} = a_{follow} - a_{lead} \tag{3.3}$$

$$ETTC = \frac{\sqrt{v_{rel}^2 + 2a_{rel}d} - v_{rel}}{a_{rel}} \qquad v_{rel}^2 > 2a_{rel}d \tag{3.4}$$

Again, a condition on the relative velocity v_{rel} and acceleration a_{rel} must be satisfied for the Enhanced Time-to-Collision (ETTC) to be defined. Despite its increased accuracy, the same drawbacks as those of TTC apply.

Worst-Time-to-Collision (WTTC)

Another extension to the standard Time-to-Collision (TTC) is its worst-case estimation, known as Worst-Time-to-Collision (WTTC). In this metric, not only is a single path of the vehicles considered, but a whole set of physically drivable trajectories is predicted using a simple vehicle model. Consequently, a broader set of TTCs emerges from collision checking, and its worst case is selected to represent the scene. It can be interpreted as the time until a collision, with every traffic participant attempting to achieve a collision as fast as possible. This metric is the first to consider multiple possible Time-to-Collision scenarios and, therefore, includes uncertainties in human intention. However, as it overestimates the scene's risk due to the worst-case consideration, it is useful for filtering large datasets for potentially critical scenes rather than for detailed assessment.

Time-Headway (THW)

The Time-Headway (THW) metric can be considered as a simplification of the Time-to-Collision (TTC). It only contemplates the velocity v_{follow} of the following vehicle and is hence defined for a broader range of velocities. The calculation is done as follows:

$$THW = \frac{d}{v_{follow}} \qquad v_{follow} > 0 \tag{3.5}$$

The THW is used in the German Straßenverkehrsordnung (StVO) to calculate fines for tailgating [118]. Being a simplification of TTC, the same drawbacks apply. It is a good measure for traffic congestion and flow rather than for safety.

Time-to-React (TTR)

Now, as all of the above-mentioned evasion strategies reveal different reaction times, the TTR is defined as the maximum of those, hence the evasion strategy leaving the most reaction time for the EGO vehicle acceleration. As depicted through the purple maneuver in Figure 3.5, acceleration is not as powerful as

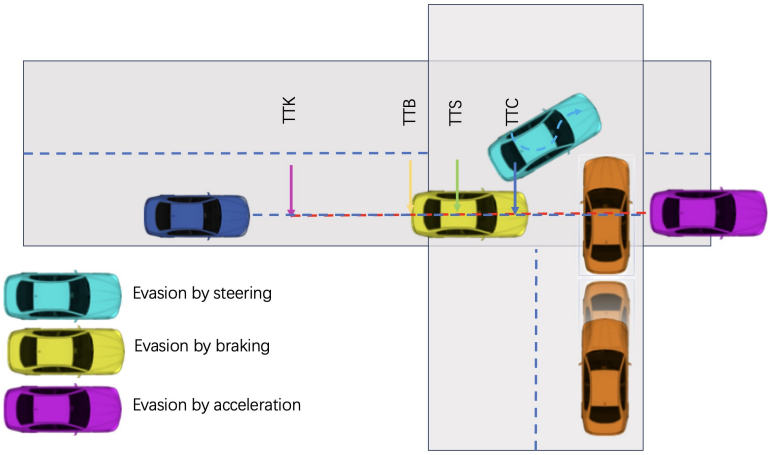

Figure 3.5: Description of TTR alongside a crossroad scenario. The EGO vehicle pictures in blue and the TO in orange. Evasion plans are shown in light blue for steering, yellow for braking, and purple for acceleration.

braking, and the maneuver needs to be initiated earlier to have the same effect. Thus, the Time-to-Accelerate (TTA) is longer than the TTB. While the literature on the TTC focuses on time, the literature on the TTR extends its perspective to space. The focus in this report is on time metrics, thus only the scenario in Figure 3.5 is of interest. All considered measures are purely preventive, aiming to avoid collisions altogether, and are limited by the capabilities of the EGO vehicle.

Now, as all of the above-mentioned evasion strategies reveal different reaction times,[19] indicates that he TTR is defined as the maximum of those, hence the evasion strategy leaving the most reaction time for the EGO vehicle acceleration.For upcoming collisions, three basic evasion options are defined and illustrated in Figure 3.5.

In a crossroad scenario, the trajectory of the blue vehicle is jeopardized by an orange vehicle that disregards the red light. Without intervention, a collision would occur after the expiration of the time defined by the TTC (Time-to-Collision). As the primary avoidance measure, thorough investigation into full braking is conducted. This allows the orange vehicle to clear the intersection before the blue vehicle enters the critical zone. The maneuver is depicted in yellow. The remaining time until full braking prevents the accident and is estimated as the Time-to-Break (TTB).Algebraic solutions for the TTB are proposed in [20].A second option involves traversing the critical area ahead of the orange vehicle by

employing full acceleration. As illustrated by the purple maneuver in Figure 3.5, acceleration is not as potent as braking, necessitating an earlier initiation of the maneuver, consequently leading to a lower Time-to-Kickdown (TTK). Steering to the left or right stands as the final option. In the illustrative scenario, evading through full left steering is represented in light blue. This provides the maximum time for avoidance, as defined by the Time-to-Steer (TTS).

Now, considering the diverse reaction times associated with the aforementioned evasion strategies, the Total Time to React (TTR) is defined as the maximum among them. Consequently, the chosen evasion strategy allows the EGO vehicle the maximum available reaction time.

$$TTR = max(TTB, TTK, TTD) \tag{3.6}$$

The TTR provides an enhanced comprehension of risk by incorporating the EGO's options to avert an imminent accident. Nevertheless, it is crucial to note that uncertainties regarding the intentions of other traffic objects are still not taken into account.

Predicted-Minimum-Distance (PMD) and Time-to-PMD (TPMD)

After highlighting the limitations of static Space-Time Analysis (STA) calculations like Time-to-Collision (TTC) and Time Headway (THW) for next-generation Advanced Driver Assistance Systems (ADAS), reference [21] introduces Predicted-Minimum-Distance (PMD) along with Time-to-PMD (TPMD). In this approach, a future collision on predicted vehicle paths is not obligatory. Instead, the algorithm initially seeks the shortest distance between the EGO vehicle and other Traffic Objects TOs (Traffic Object) or obstacles in the scene, resulting in the PMD. Subsequently, TPMD is defined as the time until this minimum distance is reached.

An improvement of this metric lies in its ability to measure risk even in "close call" situations, where vehicles pass each other too closely to be considered safe. However, it's important to note that this approach only incorporates a single future trajectory for uncertain Traffic Objects.

Metrics play a crucial role in the context of Autonomous Driving (AD) software simulation and testing for several reasons:

- **Safety Verification:** Metrics provide a quantitative way to assess the safety of autonomous systems. Safety is paramount in AD, and metrics help evaluate how well the software performs in various scenarios, including edge cases.

- **Performance Evaluation:** Metrics allow developers to measure the performance of AD software in terms of accuracy, efficiency, and responsiveness. This includes assessing how well the system detects and responds to objects, pedestrians, and other vehicles.

- **Validation of Simulation Scenarios:** Metrics help validate the effectiveness of simulation scenarios. By quantifying the outcomes of simulated events, developers can ensure that the scenarios are representative of real-world conditions.

- **Generalization Capability:** Metrics assist in evaluating the generalization capability of AD software. A well-performing system should be able to handle a wide range of scenarios, and metrics provide insights into how the software adapts to new and diverse situations.

- **Robustness Testing:** Metrics support robustness testing by measuring how well the AD software performs under adverse conditions, uncertainties, and unexpected scenarios. This is crucial for ensuring the system's reliability in real-world environments.

- **Adherence to Regulations:** Metrics help demonstrate compliance with regulatory standards and guidelines. Regulatory bodies may require specific performance metrics to ensure that autonomous vehicles meet safety and operational requirements.

- **Benchmarking:** Metrics provide a basis for benchmarking different AD software solutions. Comparative analysis through metrics allows developers to identify strengths and weaknesses in their systems compared to industry standards or competitors.

- **Continuous Improvement:** Metrics enable continuous improvement by providing feedback on system performance. Developers can use metrics to identify areas that need enhancement or refinement, leading to iterative improvements in the software.

- **Risk Assessment:** Metrics contribute to assessing and quantifying risks associated with AD software. This is essential for identifying potential areas of concern and implementing risk mitigation strategies.

- **Simulation Validation:** Metrics play a role in validating the effectiveness of simulation as a testing methodology. By comparing simulated results with real-world outcomes, developers can ensure that simulations accurately represent the complexities of on-road scenarios.

In summary, metrics are essential in AD software simulation and testing as they offer a quantitative means of evaluating safety, performance, compliance, and the overall robustness of autonomous systems. They provide actionable insights for developers to enhance the capabilities and reliability of AD software in diverse and challenging environments.

Chapter 4

Simulations and Results

4.1 SC1:Driving straight

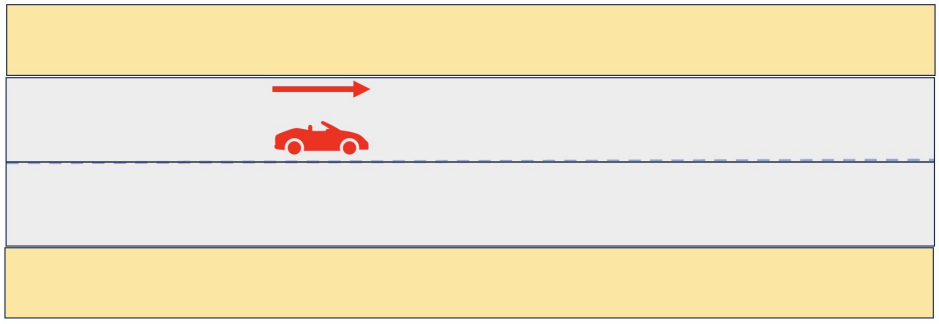

Figure 4.1: Schematic representation of SC1 : Driving straight.

4.1.1 General description

The scenario is schematically shown in 4.1. The ego vehicle is driving on a straight road. It is assumed that the initial objective of the ego vehicle is to continue driving straight in the same direction. The road, however, might be slippery, for example because of oil on the road or an excess of water. It might, therefore, be possible that the ego vehicle needs to divert its path to avoid the oil or water on the road. The presence of bad, unclear or aged line markings is considered a subclass of this scenario class.

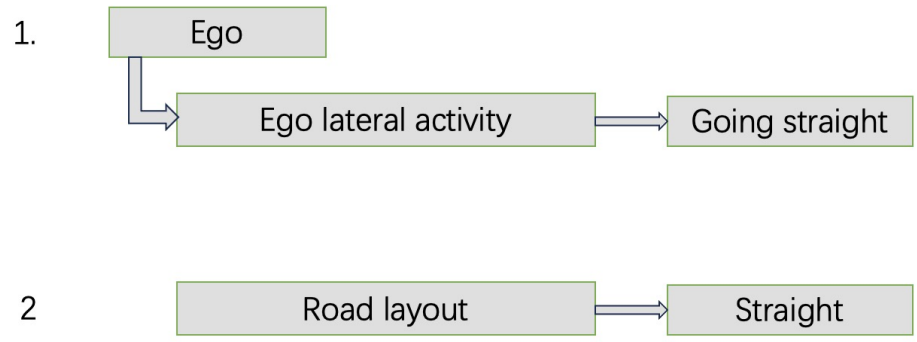

Figure 4.2: Tags of SC1 : Driving straight.

4.1.2 Formal description

- Static environment The static environment consists of a straight road. Parts of the road might be slippery, having a lower friction coefficient.
- Ego vehicle The initial objective of the ego vehicle is to drive straight.
- Dynamic environment For this scenario category, no dynamic environment is considered.

4.1.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.1 are described by at least the parameters mentioned as follows :

- The ego car accelerates from 0 to 120 km/h and maintains a constant speed thereafter
- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure4.5.
- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.
- Camera‘s Detailed table of camera detection targets and distances Figure.

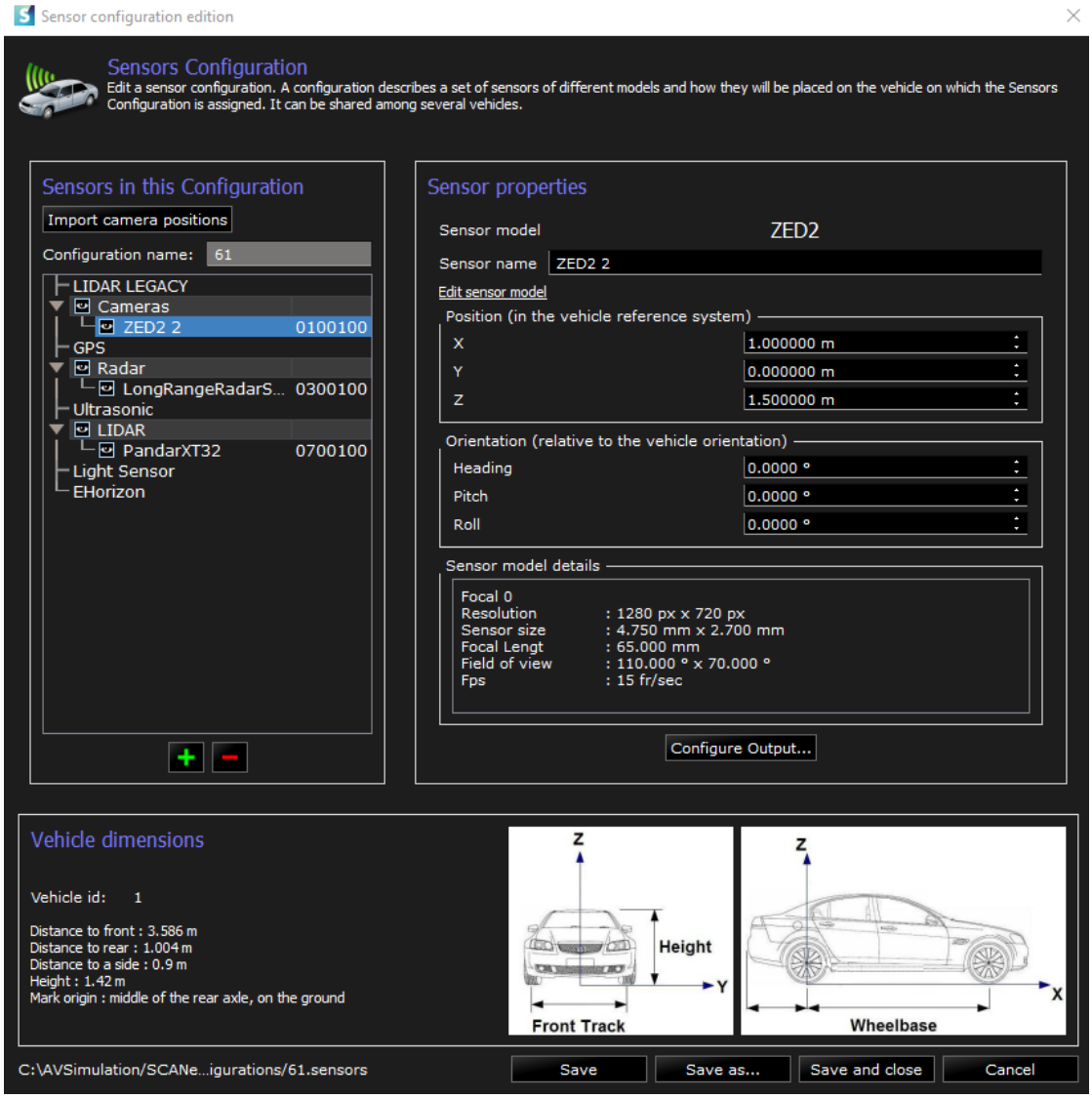

Figure 4.3: Sensors setup

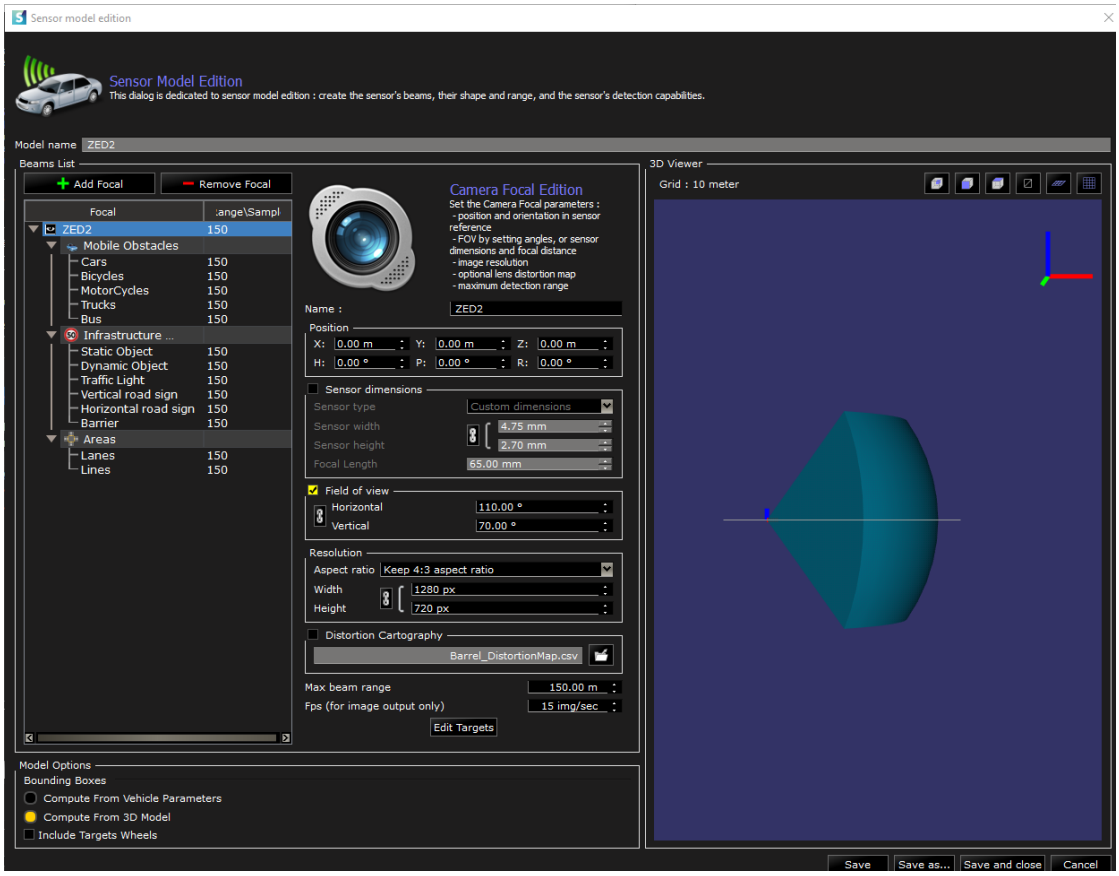

Figure 4.4: Camera setup

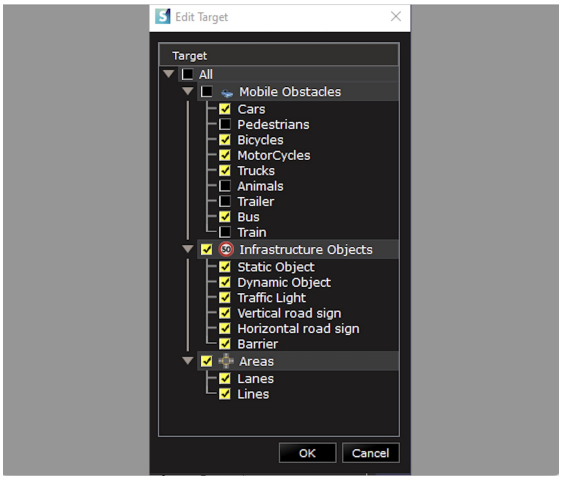

Figure 4.5: Camera setup for target

Focal	Distance
Cars	150
Bicycles	150
Motor Cycles	150
Bus	150
Trucks	150
Dynamic Object	150
Traffic light	150
Vertical road sign	150
Horizontal road sign	150
Barrier	150
Lanes	150
Lines	150

Table 4.1: Table of Camera Detection Targets and Distances

4.1.4 Simulation Result

Figure 4.8: Output of Simulation for SC1 : Driving straight

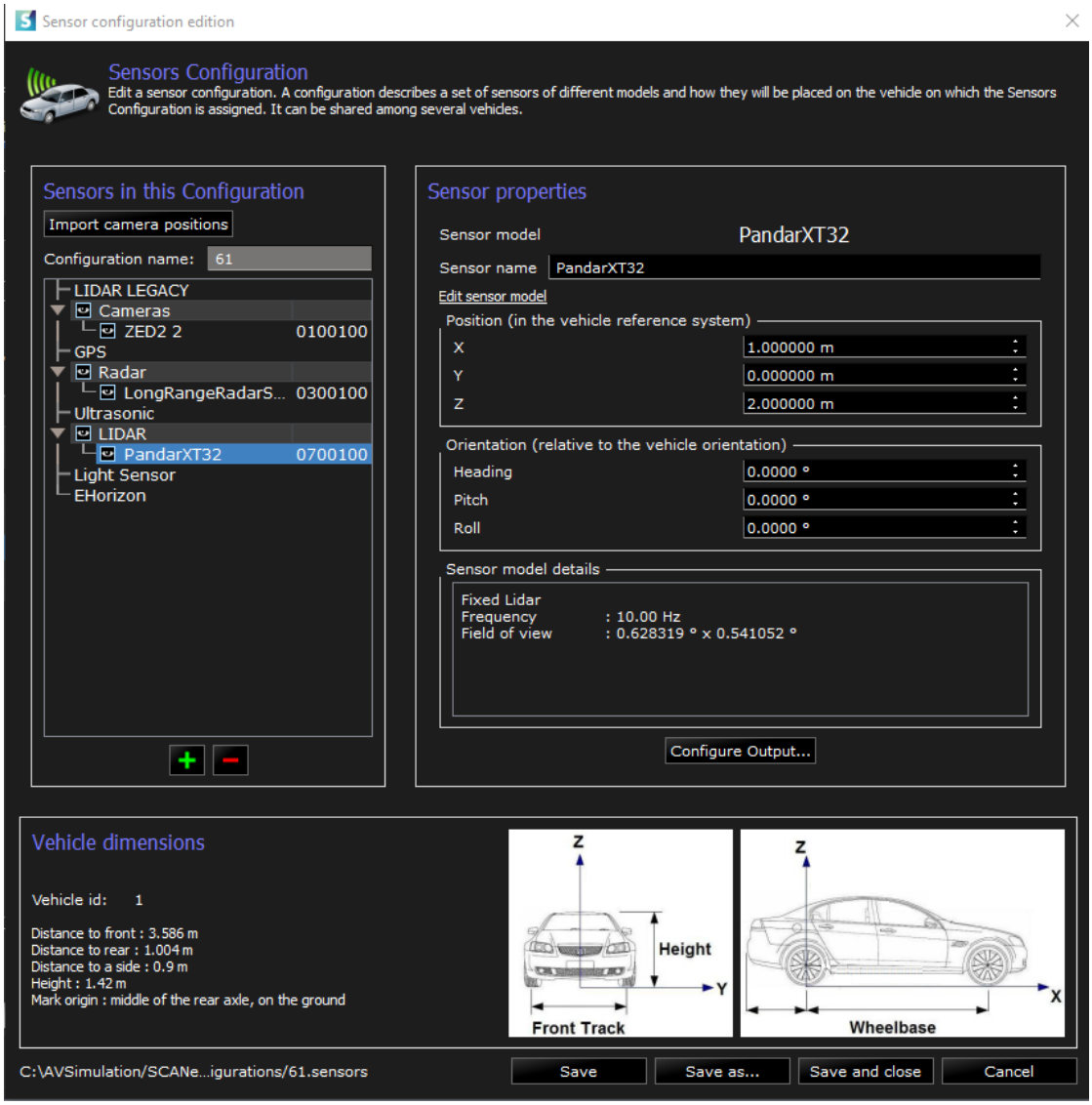

Figure 4.6: LiDAR setup

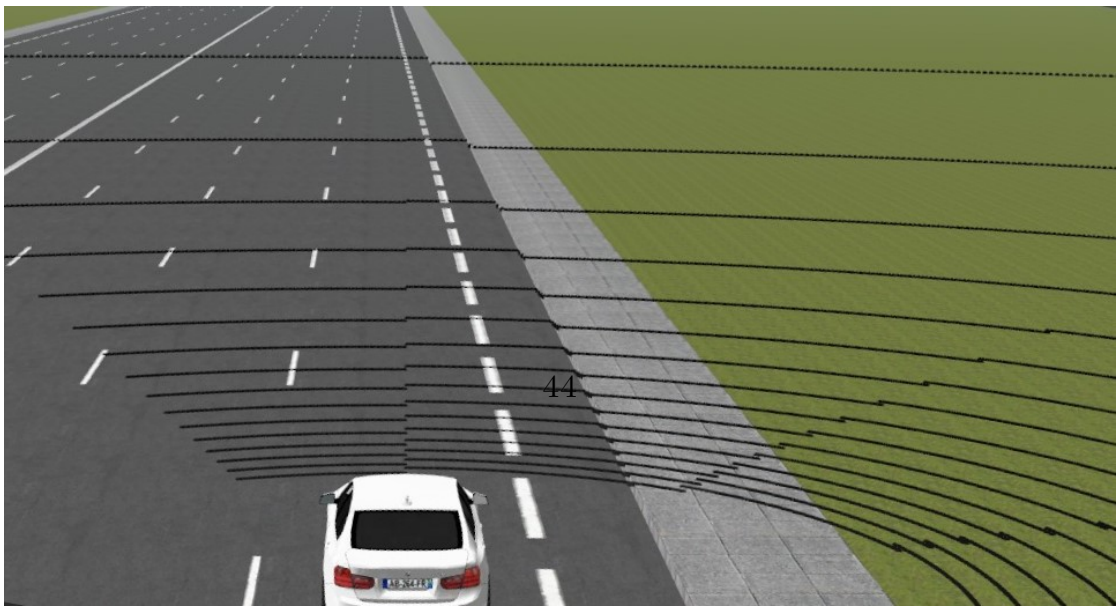

Figure 4.9: Output of Simulation for SC1 : Driving straight

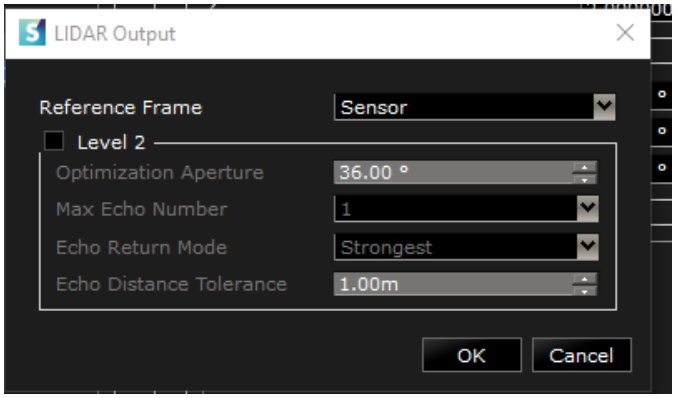

Figure 4.7: LiDAR Output setup

Figure 4.8 contains Camera's bounding box map. Figure 4.9 contains the LiDAR point cloud proximity map for scenario 1 .

4.2 SC2: Cut-in in front of the ego vehicle

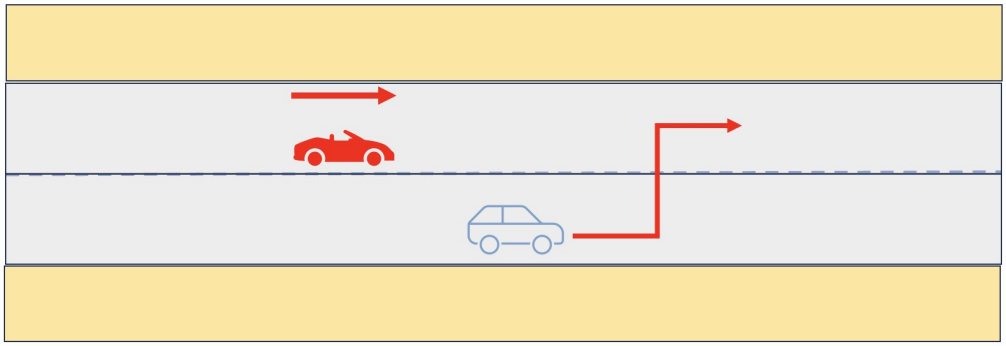

Figure 4.10: Schematic representation of SC2: Cut-in in front of the ego vehicle

4.2.1 General description

The scenario is schematically shown in Figure 4.10. Another vehicle is driving in the same direction as the ego vehicle in an adjacent lane. The other vehicle makes a lane change, such that is becomes the lead vehicle from the ego vehicle's perspective. The reason for the other vehicle to perform the lane change is principally not

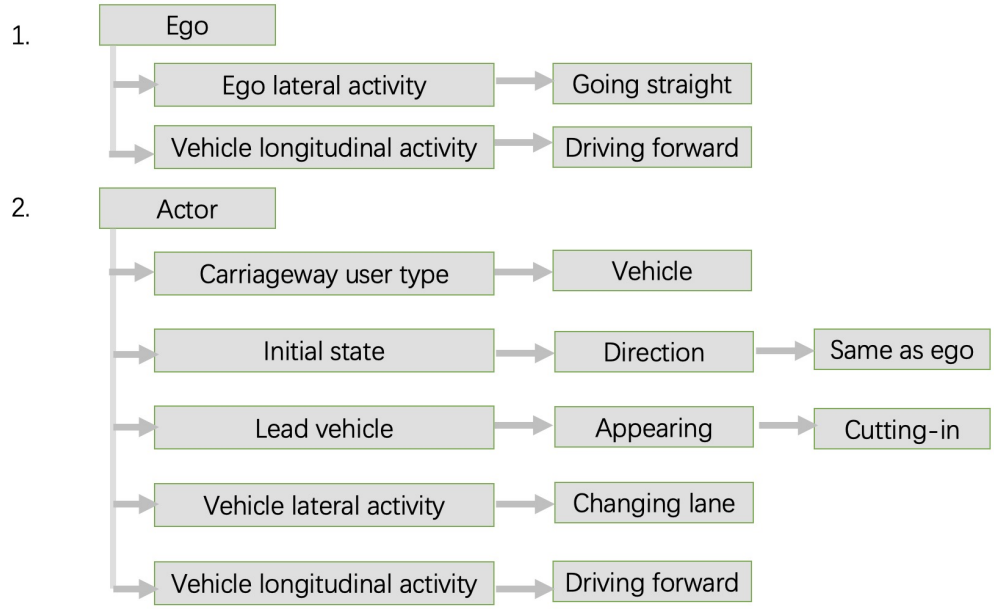

Figure 4.11: Tags of SC2 : Cut-in in front of the ego car.

important to the scenario. As shown in Figure 4.10, the reason for the other vehicle to change lane is for instance that the lane of the other vehicle is merged with the ego vehicle lane.

4.2.2 Formal description

- Static environment The static environment consists of a road with at least two lanes at the starting location of the ego vehicle.
- Ego vehicle The objective of the ego vehicle is to continue driving in the same direction.
- Dynamic environment The dynamic environment consists of another vehicle that starts at a lane adjacent to the ego vehicle lane. The other vehicle performs a lane change towards the ego vehicle lane.

4.2.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.10 are described by at least the parameters mentioned as follows :

- The ego car accelerates from 0 to 120 km/h and maintains a constant speed thereafter

- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure4.5.

- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.

4.2.4 Simulation result

Figure 4.12 contains Camera's bounding box map. Figure 4.13 contains the LiDAR point cloud proximity map for scenario 2 .

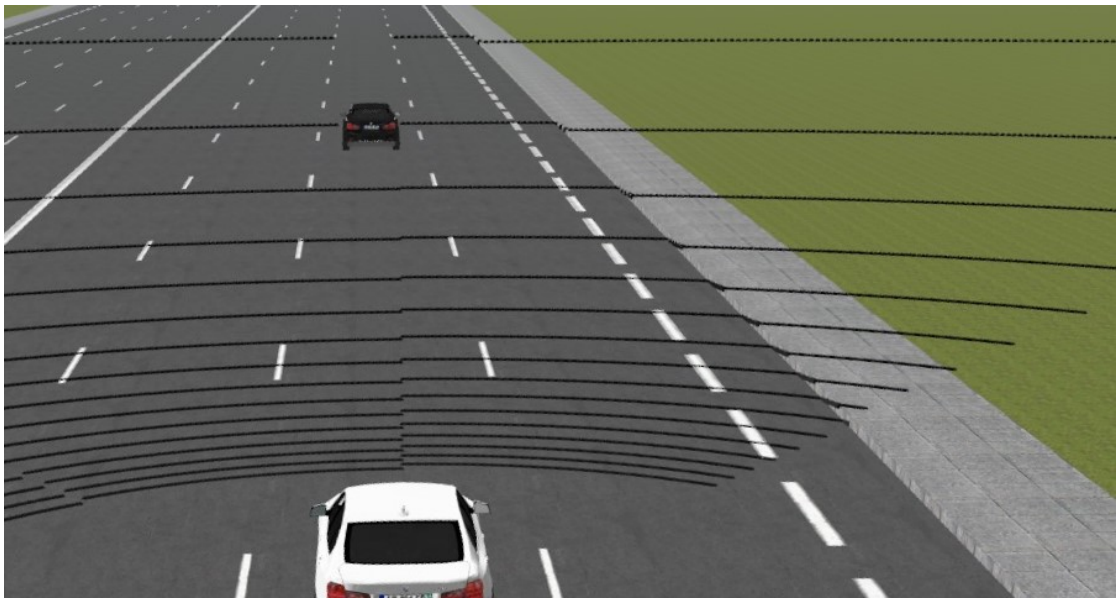

Figure 4.12: Output of Simulation for SC2 : Cut-in in front of the ego vehicle.

Figure 4.13: Output of Simulation for SC2 : Cut-in in front of the ego vehicle.

4.3 SC3: Following the leader car

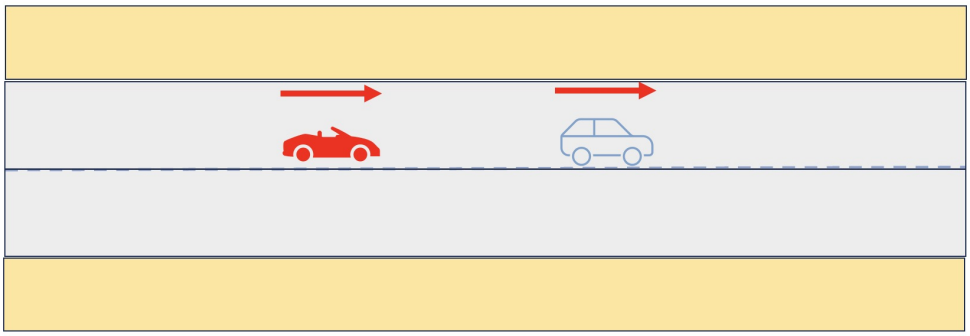

Figure 4.14: Schematic representation of SC3: follow the lead vehicle

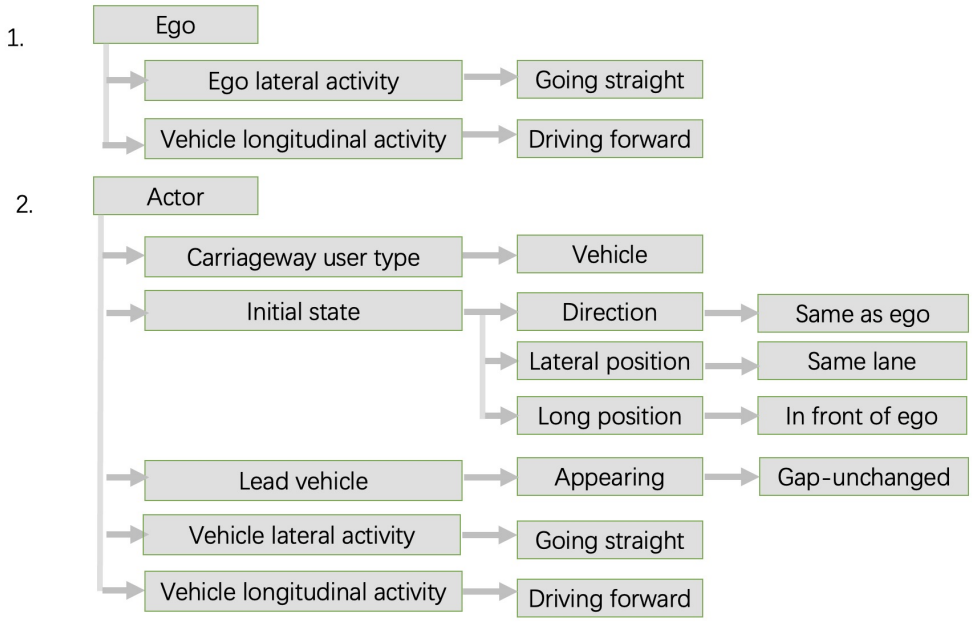

Figure 4.15: Tags of SC3 : follow the lead vehicle.

4.3.1 General description

The scenario is schematically shown in Figure 4.14. Another vehicle is driving in the same direction as the ego vehicle in an same lane. The other vehicle is positioned as the lead vehicle from the perspective of the ego vehicle. Both the other vehicle and the ego vehicle are moving at a relatively constant speed. The other vehicle is traveling in the same direction as the ego vehicle, maintaining a consistent relative distance and speed, as depicted in the diagram.

4.3.2 Formal description

- Static environment The static environment consists of a road with at least two lanes at the starting location of the ego vehicle.

- Ego vehicle The objective of the ego vehicle is to continue driving in the same direction.

- Dynamic environment The dynamic environment features an additional vehicle in the same starting lane as the ego vehicle. This other vehicle is moving within the same lane as the ego vehicle, maintaining a constant speed.

4.3.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.14 are described by at least the parameters mentioned as follows :

- The ego car and lead car accelerates from 0 to 120 km/h and maintains a constant speed thereafter

- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure 4.5.

- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.

4.3.4 Simulation result

Figure 4.16 contains Camera's bounding box map. Figure 4.17 contains the LiDAR point cloud proximity map for scenario 3 .

Figure 4.16: Output of Simulation for SC3: follow the lead vehicle

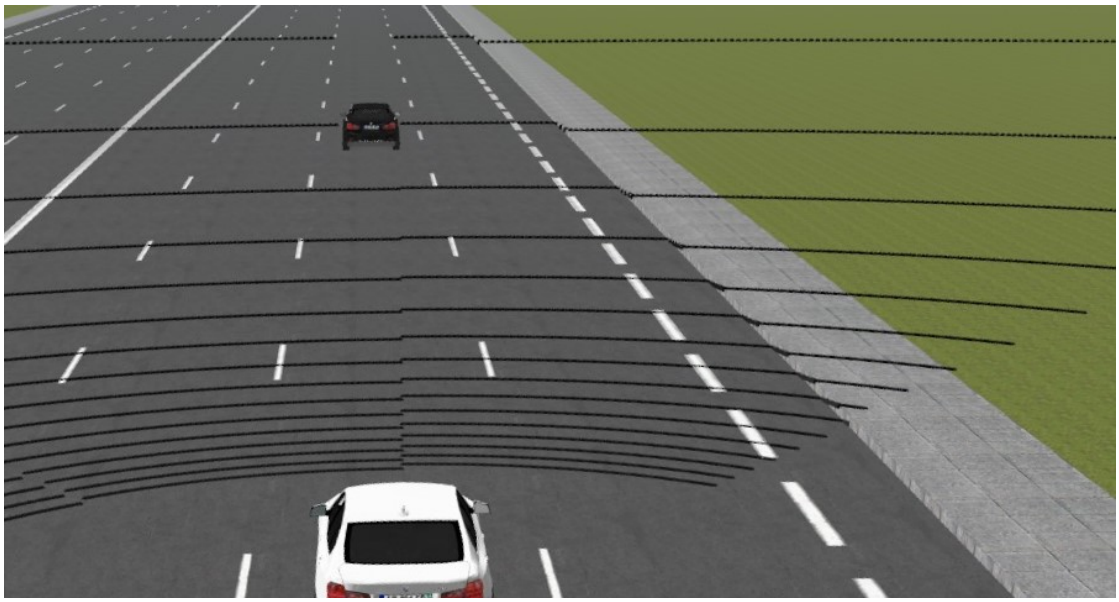

Figure 4.17: Output of Simulation for SC3: follow the lead vehicle

4.4 SC4: Following the two vehicles

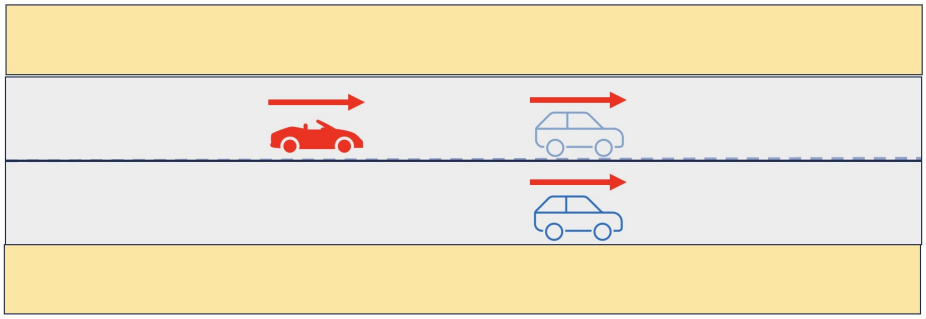

Figure 4.18: Schematic representation of SC4: follow the two vehicles

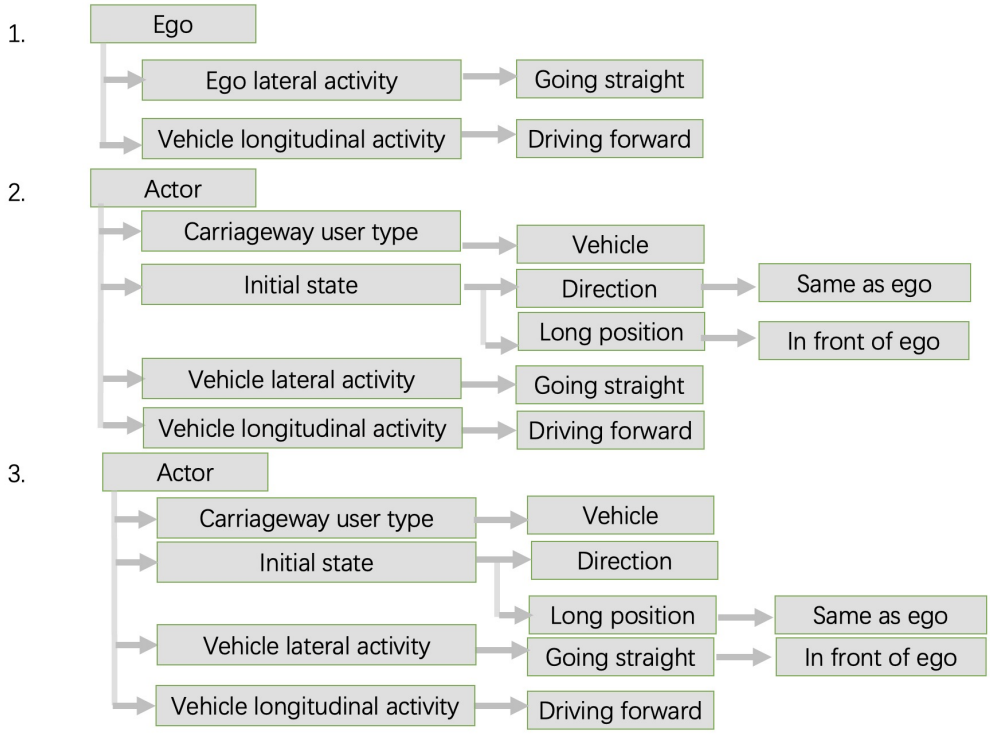

Figure 4.19: Tags of SC4 : follow the two vehicles.

4.4.1 General description

The scenario is schematically shown in Figure 4.18 .The leader car is driving in the same direction as the ego vehicle in an same lane. Another vehicle driving in the same direction as the ego vehicle in an adjacent lane. Both the other vehicles and the ego vehicle are moving at a relatively constant speed. The other vehicles are traveling in the same direction as the ego vehicle, maintaining a consistent relative distance and speed, as depicted in the diagram.

4.4.2 Formal description

- Static environment The static environment consists of a road with at least two lanes at the starting location of the ego vehicle.
- Ego vehicle The objective of the ego vehicle is to continue driving in the same direction.
- Dynamic environment The dynamic environment features an additional vehicle in the same starting lane as the ego vehicle. This other two vehicles are moving within the same lane as the ego vehicle, maintaining a constant speed.

4.4.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.18 are described by at least the parameters mentioned as follows :

- The ego car and other two cars accelerates from 0 to 120 km/h and maintains a constant speed thereafter
- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure 4.5.
- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.

4.4.4 Simulation result

Figure 4.20 contains Camera's bounding box map. Figure 4.21 contains the LiDAR point cloud proximity map for scenario 4 .

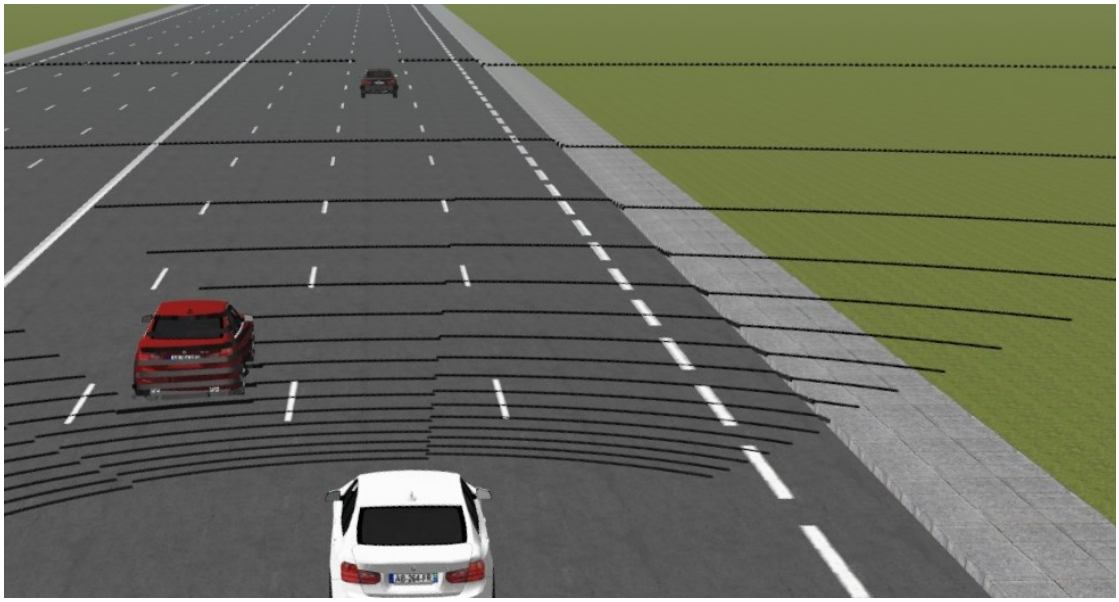

Figure 4.20: Output of Simulation for SC4: follow the two vehicles

Figure 4.21: Output of Simulation for SC4: follow the two vehicles

4.5 SC5: Cut-out in the front of the ego vehicle

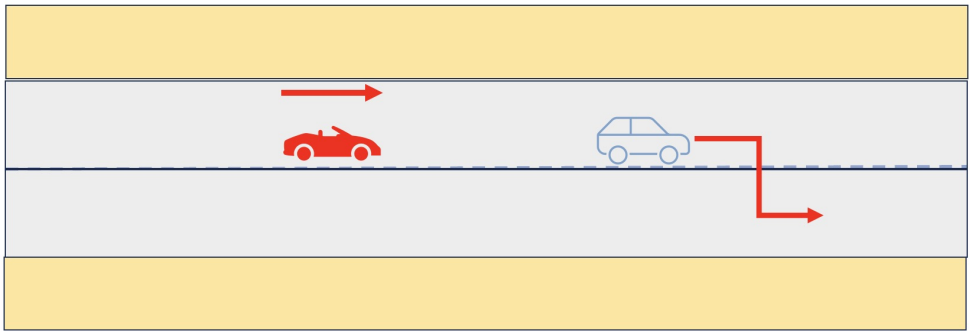

Figure 4.22: Schematic representation of SC5: Cut-out in front of the ego vehicle

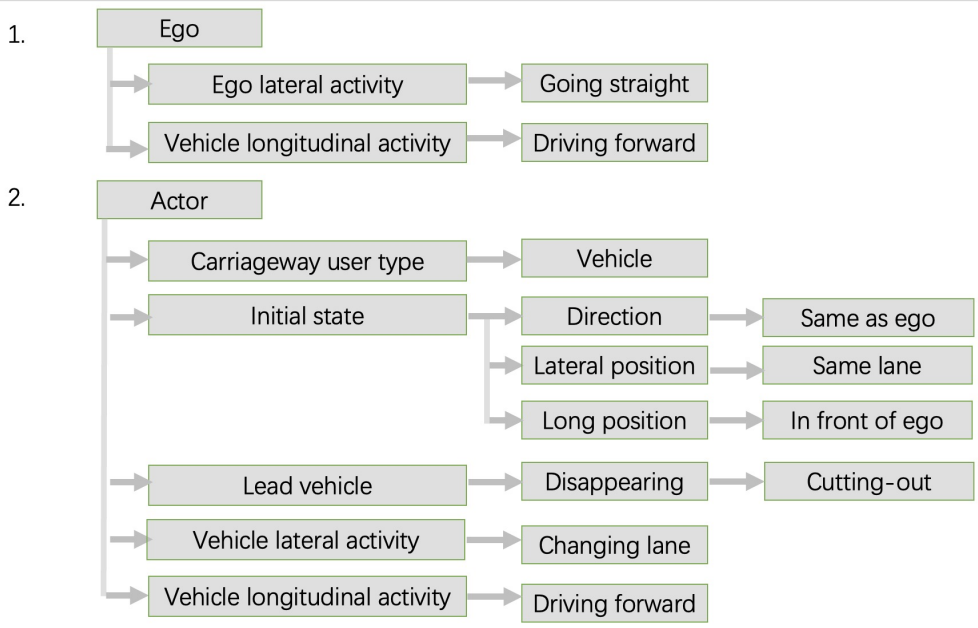

Figure 4.23: Tags of SC5 : Cut-out in front of the ego vehicle.

4.5.1 General description

The scenario is schematically shown in Figure 4.22 .The leader car is driving in the same direction as the ego vehicle in an same lane. The leader vehicle makes a lane change, then drives in an adjacent lane, the scenario is similar to scenario 4.1.It is schematically shown in 4.1 .

4.5.2 Formal description

- Static environment The static environment consists of a road with at least two lanes at the starting location of the ego vehicle.
- Ego vehicle The objective of the ego vehicle is to continue driving in the same direction.
- Dynamic environment The dynamic environment features an additional vehicle in the same starting lane as the ego vehicle. This other vehicle is moving within the same lane as the ego vehicle, maintaining a constant speed.and then the leader vehicle makes a lane change, then drive in an adjacent lane in the constant speed.

4.5.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.22 are described by at least the parameters mentioned as follows :

- The ego car and the leader car accelerates from 0 to 120 km/h and maintains a constant speed thereafter,
- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure 4.5.
- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.

4.5.4 Simulation result

Figure 4.24 contains Camera's bounding box map. Figure 4.25 contains the LiDAR point cloud proximity map for scenario 5 .

Figure 4.24: Output of Simulation for SC5: Cut-out in the front of the ego vehicle

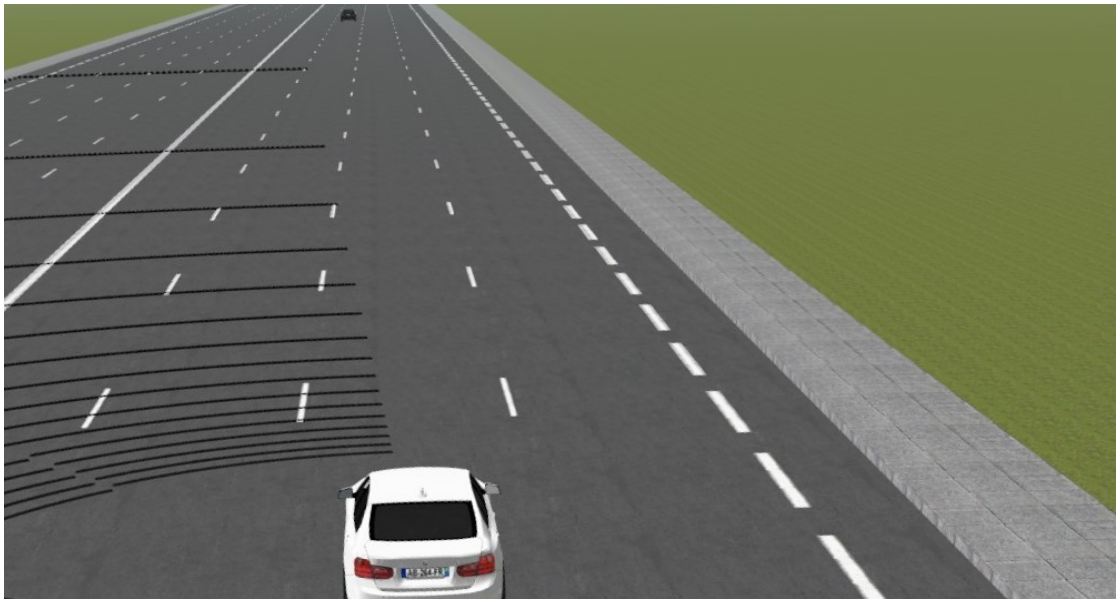

Figure 4.25: Output of Simulation for SC5: Cut-out in the front of the ego vehicle

4.6 SC6:following the two car,cut-in in the front of the ego vehicle

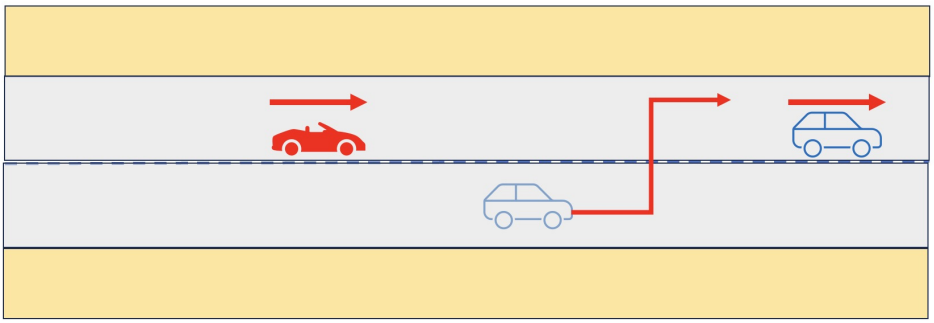

Figure 4.26: Schematic representation of SC6: following the two car,cut-in in the front of the ego vehicle

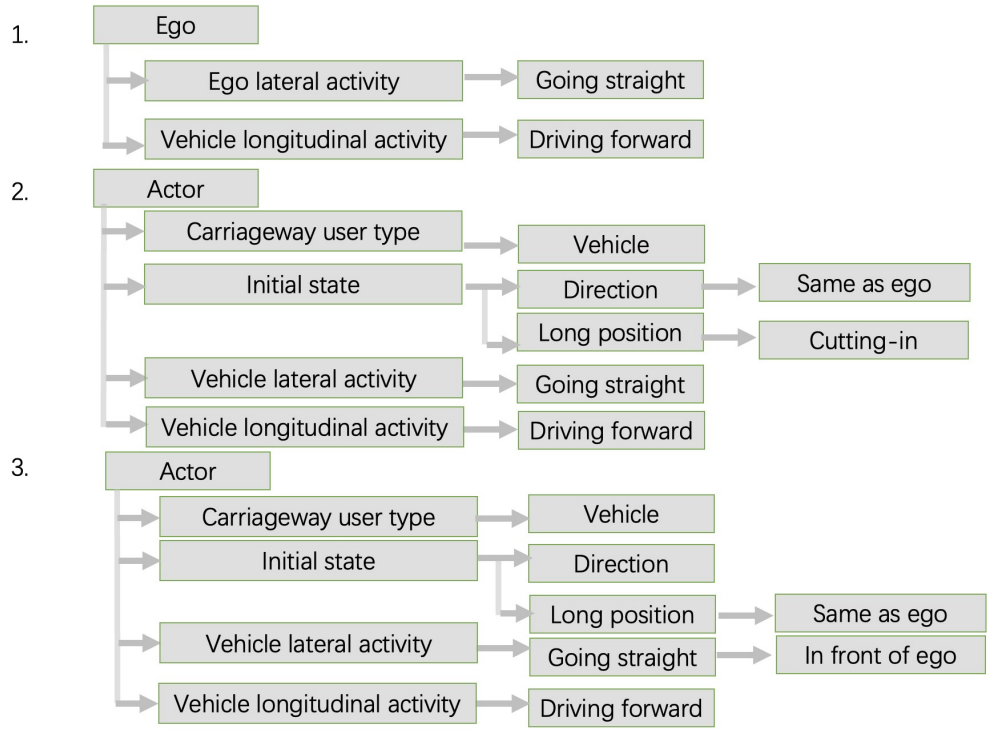

Figure 4.27: Tags of SC6: following the two car,cut-in in the front of the ego vehicle

4.6.1 General description

The scenario is schematically shown in Figure 4.26 .The leading car is moving in the same direction as the ego vehicle within the same lane. The other car, traveling at the same speed, initiates a lane change, moving into an adjacent lane also in the same direction, and subsequently merges back into the original lane with the ego vehicle.

4.6.2 Formal description

- Static environment The static environment consists of a road with at least two lanes at the starting location of the ego vehicle.
- Ego vehicle The objective of the ego vehicle is to continue driving in the same direction.
- Dynamic environment The dynamic environment features an additional vehicle in the same starting lane as the ego vehicle. This other vehicle is moving within the adjacent lane near the ego vehicle, maintaining a constant speed.and

then the other vehicle makes a lane change, then drive in an same lane with the ego car.

4.6.3 Parameters

The scenarios that belong to the scenario category depicted in Figure 4.22 are described by at least the parameters mentioned as follows :

- The ego car and the other two cars accelerates from 0 to 120 km/h and maintains a constant speed thereafter,

- Camera's configuration is set up as Figure 4.3 4.4 and 4.5 shown here.The position of the Camera is X=1 m,Y=0 m, Z=1.5 m 4.3.Horizontal and Vertical of the Filed of view are 110.00° and 70.00° individually .And Width and Height of the resolution is 1280 px and 720 px separately .At the same time, the Focal is added that the Camera could identify Cars ,Bicycles and so on in 150 meters .The Detecting Targets could be added in this interface Figure 4.5.

- LiDAR's configuration is set up as Figure 4.6 shown here. The position of the LiDAR is X=1 m,Y=0 m, Z=2 m.The Frequency is 10.00HZ and the Field of view is 0.628319° X 0.541052 °.

4.6.4 Simulation result

Figure 4.28 contains Camera's bounding box map. Figure 4.29 contains the LiDAR point cloud proximity map for scenario 6 .

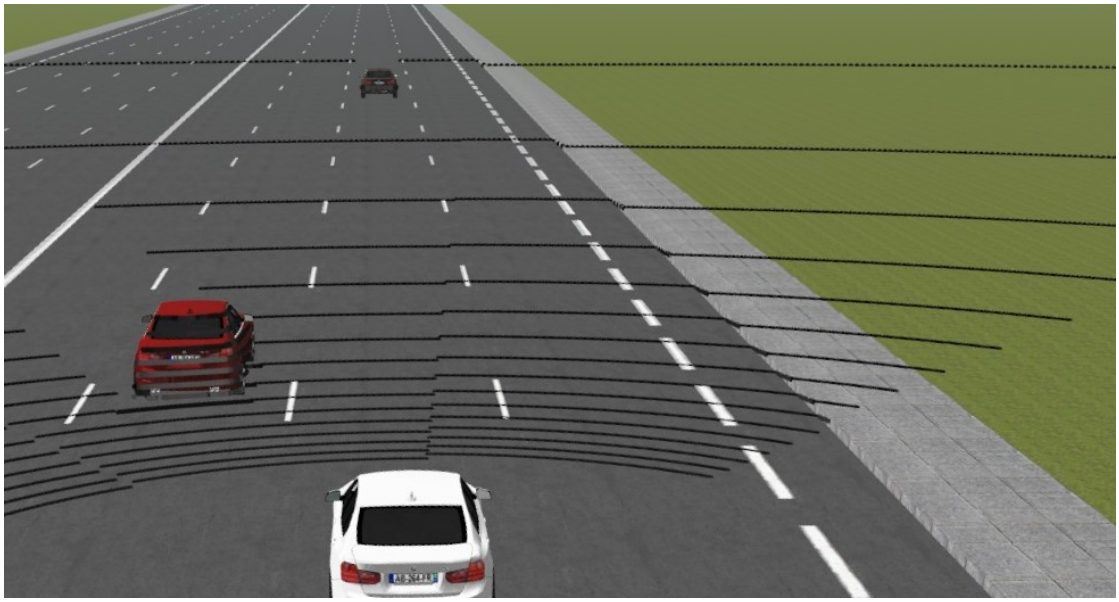

Figure 4.28: Output of Simulation for SC6: following the two car,cut-in in the front of the ego vehicle

Figure 4.29: Output of Simulation for SC6: following the two car,cut-in in the front of the ego vehicle

Chapter 5

Conclusions and Future Works

With the increasing development of artificial intelligence and the automotive industry, the capability of vehicles for Efficient and Complete Testing Enables ADAS Releasing has become a crucial aspect of ADAS. This work aims to investigate the feasibility of meeting project requirements for the environmental awareness task in a real ADAS project and propose a practical solution.

The study begins by investigating the theoretical background of each related field, which involves ADAS(Advanced Driver Assistance Systems), XiL, and simulation testing techniques. By gaining a deep understanding of these concepts, the groundwork is laid for the subsequent development of an effective solution. Then the focus shifts to the design of hardware and simulation software , leveraging existing hardware equipment available. Next, we build a reliable software working environment that enables seamless integration between the hardware and software components. Then some basic scenarios are developed to realize virtual environment testing, which aims to effectively combine data from various virtual sensors, such as LiDAR and camera, with potential scalability, and output meaningful result according to system requirements, as depicted in Figure 5.1 5.2, enabling a comprehensive perception of the vehicle's surrounding environment.

Once the scenarios are tested in the virtual environment successfully, which will reduce the cost in the real-world test vehicle. Real-world testing conducting in diverse application scenarios to evaluate the performance and capabilities of the system is very expensive.By testing in the virtual environment,the same effect will be achieved Throughout the virtual environment testing phase, data is collected and analyzed to assess the system's performance, reliability, and adherence to project requirements. This evaluation serves as a foundation for identifying areas of improvement, refining algorithms, enhancing the overall performance of the system,

and Demonstrating the effectiveness and reliability economics of system integration with ADAS systems.

Based on the test results, several key conclusions can be drawn. First, While testing in virtual environments cannot fully replace real-world testing, certain aspects of virtual testing can serve as viable substitutes for real-world scenarios. This not only results in cost and time savings but also streamlines the testing process for Advanced Driver Assistance Systems (ADAS).Additionally,Utilizing simulation software and the ROS environment, one can test the adaptability of the actual machine within the simulated scenario. This marks the next step in the exploration of this area.

This study outlines the process of exploring, designing, implementing,and evaluating Autonomous Driving (AD) software simulation scenarios in SCANeR.The proposed framework introduces a flexible and resource-efficient method for testing ADAS using the simulation software SCANeR, demonstrating its potential as a promising solution for XiL testing in simulation. The collected data and analysis provide valuable insights for further development of the ADAS system, offering an opportunity to enhance the safety and capabilities of vehicles in real-driving scenarios by leveraging virtual-driving scenarios.

In conclusion, this study provides a solid testing platform for future development in the ADAS system. Upgrading simulation software, improving, and optimizing test scenarios can contribute to enhancing the system's performance, safety, and overall user experience. Continuous innovation and advancements in this field will contribute to the realization of more sophisticated and effective ADAS systems in the future.

Our future work involves creating an efficient platform for enhanced data transfer between hardware and software, thereby improving the testing of scenarios.In this case, our specific work starts with the creation of more scenarios.the tool of scenario engineering is introduced to enhance the diversity and safety of real-road testing by integrating virtual and physical components in the testing loop.

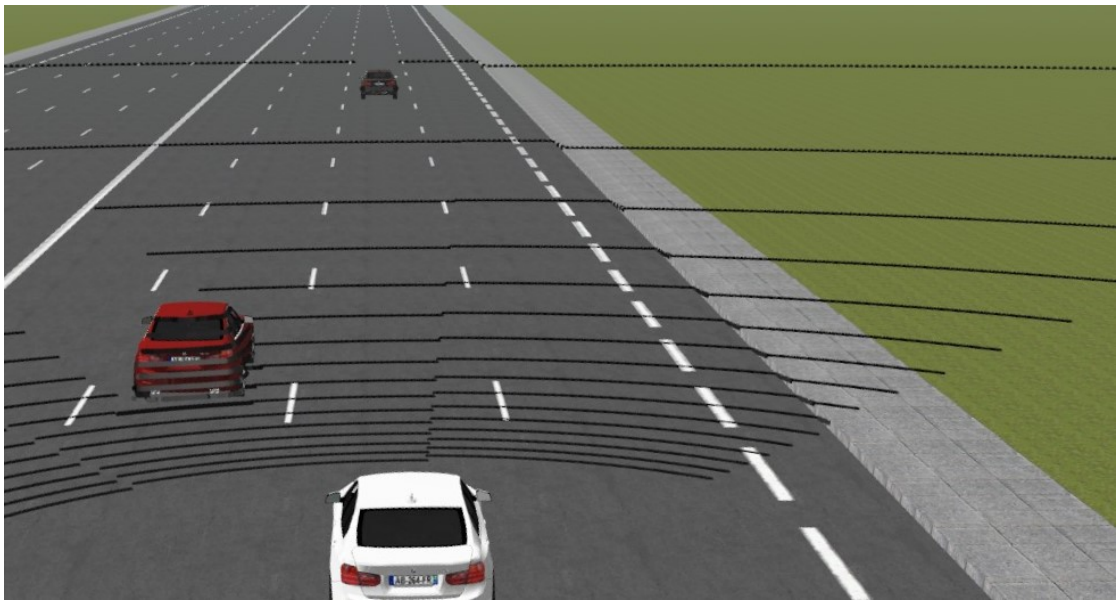

Figure 5.1: Visualization of Simulation result

Figure 5.2: Visualization of Simulation result

List of Tables

List of Figures

Bibliography

[1] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. URL: https://www.sae.org/standards/content/j3016_202104/. (accessed: 30.04.2021) (cit. on pp. 3–5).

[2] *apollo: Open source autonomous driving,* URL: https://github.com/ApolloAuto/apollo. (accessed: 14.07.2023) (cit. on p. 4).

[3] *Automated Driving in Its Social, Historical and Cultural Contexts*. URL: https://link.springer.com/chapter/10.1007/978-3-662-48847-8_3. (accessed: 2016) (cit. on p. 6).

[4] J Wetmore. «Driving the dream. The history and motivations behind 60 years of automated highway systems in America». In: *Automotive History Review* 7 (2003), pp. 4–19. DOI: 10.1007/978-3-662-45854-9_3 (cit. on p. 6).

[5] *Prof. Schmidhuber's highlights of robot car history*. URL: https://people.idsia.ch/~juergen/robotcars.html. (accessed: 2019) (cit. on p. 6).

[6] *History of Autonomous Cars*. URL: https://www.tomorrowsworldtoday.com/artificial-intelligence/history-of-autonomous-cars/. (accessed: 2021) (cit. on p. 6).

[7] J Wetmore. «A system architecture for autonomous visual road vehicle guidance». In: *IEEE* (Nov. 1997). DOI: 10.1109/ITSC.1997.660538 (cit. on p. 6).

[8] Ernst D. Dickmanns. *Dynamic Vision for Perception and Control of Motion*. Springer London, 2007. DOI: https://doi.org/10.1007/978-1-84628-638-4 (cit. on p. 7).

[9] Sanjiv Singh Martin Buehler Karl Iagnemma. *The DARPA Urban Challenge:Autonomous Vehicles in City Traffic*. Springer Berlin, Heidelberg, 2009. DOI: https://doi.org/10.1007/978-3-642-03991-1 (cit. on p. 7).

[10] Dave Ferguson Tugrul Galatali Chris Geyer Michele Gittleman Sam Harbaugh Martial Hebert Tom Howard Alonzo Kelly David Kohanbash Maxim Likhachev Nick Miller Kevin Peterson Raj Rajkumar Paul Rybski Bryan Salesky Sebastian Scherer Young Woo-Seo Reid Simmons Sanjiv Singh Jarrod Snider Anthony Stentz William "Red" Whittaker Chris Urmson Joshua Anhalt Drew Bagnell Christopher Baker Robert Bittner John Dolan Dave Duggins and Jason Ziglar. «Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge». In: *ResearchGate* (Apr. 2007) (cit. on p. 7).

[11] *TOYOTA Motor — 75 Years of TOYOTA — Technical Development — Electronics Parts*. URL: https://www.toyota-global.com/company/history_of_toyota/75years/data/automotive_business/products_technology/technology_development/electronics_parts/index.html. (accessed: 2019) (cit. on p. 7).

[12] *Tesla beams down 'autopilot' mode to Model S*. URL: https://www.autonews.com/article/20151014/OEM06/151019938/tesla-beams-down-autopilot-mode-to-model-s (cit. on p. 7).

[13] *2019 Audi A8 L Review | Brilliant engineering in an unassuming wrapper*. URL: https://www.autoblog.com/2018/10/16/2019-audi-a8-l-review-first-drive/ (cit. on p. 7).

[14] *The history of autonomous driving at the BMW Group*. URL: https://www.autonomousvehicleinternational.com/features/the-history-of-autonomous-driving-at-the-bmw-group.html (cit. on p. 7).

[15] *Mercedes-Benz E-Class 2016 first drive: The Einstein of luxury cars*. URL: https://www.pocket-lint.com/cars/reviews/mercedes-benz/136965-mercedes-benz-e-class-2016-first-drive-the-einstein-of-luxury-cars/ (cit. on p. 7).

[16] European Space Agency. «Technology Readiness Levels Handbook for Space Applications». In: *ESA CSC* (Sept. 2008). URL: https://connectivity.esa.int/sites/default/files/TRL_Handbook.pdf (cit. on p. 7).

[17] European Space Agency. «ERTRAC - Automated Driving Roadmap». In: *ESA CSC* (July 2015). URL: https://www.ertrac.org/wp-content/uploads/2022/07/ERTRAC-CAD-Roadmap-2019.pdf (cit. on p. 7).

[18] Hala Elrofai Jan-Pieter Paardekooper Erwin de Gelder Sytze Kalisvaart Olaf Op den Camp. «SCENARIO-BASED SAFETY VALIDATION OF CONNECTED AND AUTOMATED DRIVING». In: *StreetWise* (July 2018). URL: https://www.scipedia.com/public/Elrofai_et_al_2018a (cit. on p. 29).

[19] A. M. Spieker J. Hillenbrand and K. Kroschel. «A multilevel collision mitigation ap- proach—Its situation assessment, decision making, and performance tradeoffs». In: *IEEE* 7 (Dec. 2006). DOI: 10.1109/TITS.2006.883115 (cit. on p. 35).

[20] K. Kroschel J. Hillenbrand and V. Schmid. «Situation assessment algorithm for a collision prevention assistant». In: *IEEE* (June 2005). DOI: 10.1109/IVS.2005.1505146 (cit. on p. 35).

[21] A. Polychronopoulos; M. Tsogas; A. Amditis; U. Scheunert; L. Andreone; F. Tango. «Dynamic situation and threat assessment for collision warning systems: the EUCLIDE approach». In: *IEEE* (June 2004). DOI: 10.1109/IVS.2004.1336458 (cit. on p. 36).