TUM School of Computation, Information and Technology
Technical University of Munich

Bachelor’s Thesis in Informatics

Intelligent Safety Validation for Autonomous Vehicles

Supervisor	Prof. Dr.-Ing. habil. Alois C. Knoll
Advisor	Dr. rer. nat. Hu Cao
Author	Selim Mert Kastan
Date	February 20, 2025 in Munich

Disclaimer

I confirm that this Bachelor's Thesis is my own work and I have documented all sources and material used.

[illegible]

Munich, February 20, 2025

(Selim Mert Kastan)

Abstract

Ensuring the safety of autonomous vehicles before widescale deployment is essential. However, traditional safety validation methods, such as real-life testing, are not only inefficient, but also very costly economically. Therefore, scenario-based safety validation methods are promising alternatives for safety validation. However, the main problem in scenario-based testing is that scenarios are written by humans. ChatScene tries to overcome this obstacle by automating the scenario generation using reasoning capabilities from large language models. However, algorithms used in autonomous driving technologies lack formal verification guarantees. Therefore, risk estimation while driving is life-critical for autonomous vehicles. We bring an RSS-based risk estimation methodology into ChatScene to further enhance its safety validation capabilities. By adding risk estimation, this approach enables AVs to assess potential dangers and improve decision-making. The research demonstrates that RSS can be utilized within ChatScene to quantify collision risk, offering a structured method for evaluating AV safety.

Contents

List of Figures

Chapter 1

Introduction

An autonomous vehicle (AV) refers to a vehicle that makes use of advanced driver assistance technologies to eliminate the necessity of a human operator. To replace human perception, AVs employ various remote-sensing technologies such as radar, lidar, GPS, and cameras in order to create a 3D map of the environment. The environment is defined by roads, other vehicles, pedestrians, traffic lights, road signs, and much more. The gathered data is afterwards processed by computers with various machine learning algorithms. Based on the information, the AV can then decide the vehicle operations, such as adjusting steering, controlling cruising speed, acceleration, and braking [Enc23].

Autonomous vehicles are classified into six levels of autonomy by the Society of Automotive Engineers (SAE), ranging from Level 0 to Level 5 (full automation). Levels 0 to 2 could be collectively referred to as low-level automation, while Levels 3 to 5 could be categorized as high-level automation. The main difference is the level of human interaction needed. Low-level automation systems such as Advanced Driver Assistance Systems (ADAS) require constant driver supervision. These systems include features like automatic emergency braking, lane centering, and adaptive cruise control systems. In contrast, high-level autonomy eliminates the need for human intervention. At level 3, systems like automated lane-keeping systems can drive without human input under specific conditions. If a situation requires intervention, the driver is supposed to take control. For levels 4 and 5, no human input is needed, and the AVs are supposed to driver on their own. The example for such systems is local driverless taxis. What's more is that levels 0 to 3 are already widely deployed on the roads. Ford's BlueCruise system and Tesla's Autopilot offer automations up to level 3 [SAE21] [Enc23].

Since we almost reached level 4 automation with rapid advancements in autonomous driving systems (ADS), we are on the brink of a transportation revolution. ADS promises to greatly reduce traffic collisions by eliminating human factors responsible for accidents. Furthermore, these systems offer a more comfortable travel experience, greater mobility for individuals with disabilities or elderly populations, and a more efficient mode of transportation for all users. Beyond the benefits to users, ADS can contribute to society much more with decreased traffic congestion through improved traffic flow and incident management, enhanced energy efficiency through optimized routing, and dynamic routing that adapts in real-time to changing road conditions. Therefore, ADS has the potential to make transportation safer, more efficient, and more sustainable [Cha17].

However, research has shown us that the benefits of AVs are still far away. The disengagement rate, which describes the rate of situations where the driver has to take control, is ten times higher than the crash probability in the United States [Fen+23]. Furthermore, it is known that even the low-level automations are error-prone. Tesla's Autopilot and Full Self-Driving (FSD) systems have been involved in my accidents. Even simple parking assist systems cause accidents while parking [Vol22]. Therefore, it should be clear that safety validation of ADS is a critical necessity before allowing them to drive on public roads. Furthermore, an AV is a black box to an average user. In order to gain wide-scale acceptance, it has been put forward that AVs should be at least 4 times safer than human drivers [Gel+21].

It should be made clear that testing autonomous driving systems is not a straightforward task. These systems consist on many algorithms and deep learning models that are responsible for various tasks such as object detection, trajectory planning, anomaly detection etc. This increases the complexity of test validation, as every single system should be tested in theory separately and together for maximum safety guarantees. Formal safety verification is clearly out of the question since deep learning models do not allow this due to their inherent nonlinearity. Thus, most common testing approaches are a combination of software simulation, closed test track, and real-world testing [Fen+23]. Real-world testing is economically costly and time intensive, since millions of kilometers in real-world driving are necessary for accurate estimation for human-level safety [ZXL24].

Furthermore, the concepts known as the curse of dimensionality and curse of rarity make this even harder to produce good results. Autonomous driving is a high-dimensional problem as driving environments are spatially and temporally complex. The variables to define such environments are high-dimensional. Since the total space grows exponentially with every added dimension, the computational complexity is growing exponentially too. The curse of rarity describes that most of the points in this high-dimensional space are also not safety critical, as the occurrence probability of accidents is very low. Therefore, these points do not contribute to safety validation at all [Fen+23]. Thus, to conclude, we can say that traditional safety validation methods are hugely inefficient, and alternative methods are necessary due to the high dimensional nature of the problem.

This is where software simulation comes in. Simulations allow to recreate the real environment like vehicles, pedestrians sensor and allow the vehicle logic to experience the real world as close as possible. In this thesis, we will be using the Carla simulator as our choice. It is very efficient to run driving scenarios on simulations, and it can be done in parallel. Furthermore, the main choice for driving logic is reinforcement learning algorithms for this thesis. Reinforcement learning is a machine learning algorithm where an agent takes actions to maximize a reward.

However, creating safety-critical scenarios by hand is very time consuming. Such scenarios are either extracted from real world data or created artificially. ChatScene paper intended to solve this problem by making use of the reasoning capabilities of large language models to generate safety-critical scenarios. By combining LLMs with a code snippet database, they allow diverse safety critical scenario generation. Furthermore, they also provide reinforcement learning model checkpoints that are trained on such scenarios [ZXL24].

Another important aspect of safety validation is quantitative risk estimation in real-time. Risk estimation is a good measure of safety validation. In this thesis, we will be using a risk estimation method that was introduced by Guo et al. (2022). It takes RSS safety model as a foundation. RSS safety model is a mathematical model that has definitions of concepts related to safety validation for autonomous systems [GZW22] [Sha17]. Its details will be discussed in the following sections.

In this thesis, we explore the integration of RSS safety model into ChatScene framework to enhance its safety validation capabilities with risk estimation methodology. This approach aims to create a unified framework for testing and improving AV systems, addressing both the technical and safety challenges that currently hinder widespread deployment.

1.1 Motivating Example

Before going further with the topics, we should first make clear why real-time risk estimation technologies is essential for autonomous vehicles. The underlying problem is the fact that formal verification is not possible for machine learning the technologies being used. Almost all autonomous driving systems use Deep Neural Networks to sense the environment, to make future trajectory prediction for other vehicles, to plan the next move with reinforcement learning controllers. However, these machine learning methods struggle with unforeseen driving situations that are caused by the black-box nature and poor generalization of DNNs within AD systems. A deep learning model learns the distribution from an input dataset. If the input in test time is from a different distribution, the model will not work on the test time. To illustrate the point, one can think of the environment sensing model being trained on day time and inferenced during night time. Thus, its necessary to always have additional guidelines and protective measures that decide whether the decisions from DNNs violate road safety.

Let us consider a concrete scenario to illustrate this point. Lets imagine a classic neighbourhood in the real world. Usually, near school districts the traffic speed limit is very low for safety of children and pedestrians. In our imagined scenario, one child puts a sticker on top of a stop sign near pedestrian crossing. An AV is driving on this road. Normally, all human drives are supposed to slow down after they see the stop sign. The problem is that AVs use deep neural networks for image classification tasks such as recognizing the traffic signs. Due to the image of the stop sign being altered slightly, there is a non-zero chance that the autonomous vehicle will mislabel the stop sign as something else. Therefore, due to the wrong recognition of the traffic sign, the decision module of the AV could potentially decide non-safe actions such as not slowing down or even accelerating further, thus risking the safety of other vehicles and pedestrians [ZXL24].

All in all, such weakness of the deep learning models can be mitigated by introducing quantitative risk estimation systems alongside decision making modules of an AV and verifying the current risk of all times. To help achieve this, this thesis aims to bring RSS model into the chatscene framework to further enhance its safety validation capabilities. Ultimately, this research with this controbution seeks to improve the reliability and safety of AV decision-making and wants to ensure that AVs can navigate complex, unforeseen scenarios with minimal risk to all road users.

1.2 Structure

- **Related Work:** Reviews related papers that are relevant for this thesis.
- **Methodology:** Reviews the technologies being used and talks about the integration steps taken
- **Results:** Shows the results from the integration through applying it on pretrained reinforcement learning agents provided within ChatScene
- **Conclusion:** Summarizes what has been achieved
- **Appendix:** Shows extra graphs and figures from ChatScene runtime

Chapter 2

Literature Review

Before continuing further, we want to have a look at what has been achieved in the field of safety validation for autonomous vehicles. We will begin with reviewing one paper that makes use of reasoning capabilities of LLMs and follow with reviewing RSS safety model and ChatScene paper. At the end, we will conclude with 2 differing approaches towards quantitative risk estimation for autonomous vehicles. We believe these steps are necessary for a good understanding of this thesis.

2.1 Usage of LLMs in the field

Wang et al. (2025), in their paper "Empowering Autonomous Driving with Large Language Models: A Safety Perspective" introduce a novel approach to Autonomous Vehicle safety, focusing on integration of large language models (LLMs) into autonomous driving systems [Wan+24]. This research represents a significant advancement in addressing safety challenges in autonomous driving, such as poor generalization of Deep Neural Networks (DNN), by using common-sense knowledge and reasoning abilities of LLMs [Wan+24]. The authors propose 2 possible ways to make use of LLMs in their paper. We will briefly go over them here.

The first approach suggests introducing LLMs as a behavior planner that provides safety constraints to the trajectory planner module, which is responsible for sending control inputs after calculating the trajectories of the vehicles around. The method begins with sending information such as scenario descriptions and its previous interaction with the LLM to the LLM. Afterwards, the LLM outputs a behaviour decision based on its understanding of the driving scenario. Consequently, the decisions are then interpreted as safety constraints for the trajectory planning module, thus acting as a safety verifier. This is made possible by the reasoning capabilities of LLMs [Wan+24]. It should be noted here that the input to the LLM is the current information in one single frame.

To overcome this incapability that the method looks at only one frame, second method proposes an LLM-based behavior planner for autonomous driving that enhances safety and decision-making by integrating a state machine, memory module, and reflection module. The state machine enforces structured behavior transitions, reducing the decision space and improving interpretability. The memory module tracks historical interactions to predict surrounding vehicles' intentions, while the reflection module ensures compliance with safety constraints and facilitates in-context learning. Compared to the first approach, this method considers multiple consecutive steps [Wan+24].

Ultimately, this research tries to transfer the reasoning capabilities and common knowledge of large language models into autonomous driving systems. In the context of this thesis, which aims to improve the safety capabilities of the ChatScene framework, the capabilities should be noted since ChatScene provides scenario generation capabilities with the usage of LLMs.

2.2 Responsibility-Sensitive Safety (RSS) model

Shalev-Shwartz et al. introduced the RSS safety model first in their paper "On a Formal Model of Safe and Scalable Self-driving Cars" to create a mathematical model in order to formalize the human judgment in relation to all driving scenarios [Sha17]. It proposes some guidelines and some minimum distances between vehicles that should be kept at all times to avoid collisions. The model claims that there will be no accidents if every traffic participant follows the principles. Furthermore, this model is called responsibility aware as the blame belongs to the party that violates this model. The overall intention is to bring some sort of safety guarantee to accelerate the development of autonomous vehicles [Sha17]. Below we can see an illustration of safety distances, which is the mathematical foundation of the model. All the following explanations of the various safety distances can be read while taking the below illustration into consideration.

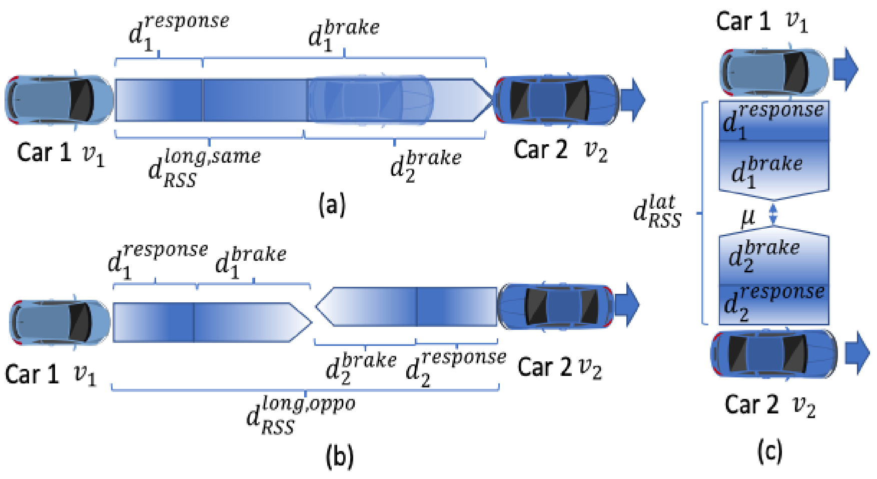


Figure 2.1: Illustration of RSS safe distances. (a) Longitudinal Distance in the same direction (b) Longitudinal Distance in th opposite direction (c) Lateral Distane [GZW22].

Longitudinal Safe Distance – Same Direction

The longitudinal distance in the same direction $\mathbf{d}_{\text{rss}}^{\text{long,same}}$ is defined for 2 cars that are driving in the same direction, namely the car-following scenario. Assume the leading vehicle travels at velocity v_1 and the following vehicle moves at velocity v_2. In order to guarantee maximum safety, we image the worst case possible. The leading car, noted car 2 in the figure 2.1 (a) , applies full braking force $a_{\text{max, brake}}$, and eventually comes to a sudden full stop. At the start of this sudden deceleration, the distance between the two vehicles is denoted as $d_2{}^{\text{brake}}$. The time it takes the following car, noted as car 1 in the figure 2.1 (a), to react to this sudden braking is defined as response time p. During response time p, it's further assumed that the following car accelerates with maximum before taking any action. The distance during the response time p is defined as ${d_1}^{\text{response}}$. After the response time, the following vehicle brakes with minimum deceleration $a_{\text{min, brake}}$. The braking distance of the rear car is defined as ${d_1}^{\text{brake}}$. Here it should be noted that the following vehicle brakes weaker than the lead car. So the longitudinal distance between two vehicles that go the same direction is considered safe if it's smaller than the safety distance, and its calculation is shown below [GZW22]:

$$\begin{aligned} d_{\text{rss}}^{\text{long,same}}(v_1, v_2) = {} & d_1^{\text{response}}(v_1, p, a_{\text{maxaccel}}) \\ & + d_1^{\text{brake}}(v_1, p, a_{\text{maxaccel}}, a_{\text{minbrake}}) \\ & - d_2^{\text{brake}}(v_2, a_{\text{maxbrake}}) \end{aligned} \tag{2.1}$$

Safe Longitudinal Distance for Opposing Directions

The longitudinal distance in the opposing direction is also defined for 2 cars that are driving in the opposite directions. Suppose that two vehicles, car 1 and car 2 from the figure 2.1 (b), are driving in adjacent lanes at opposing directions with velocities v1 and v2, respectively. One of the cars, car 1, wants to overtake a slower car that drives in front of it in the same lane. Therefore, it needs to enter the opposing lane. After car 1 enters the opposing lane, we assume for the sake of the worst-case situation to guarantee maximum safety, car 1 speeds up with its maximum acceleration after entering the opposing lane for the duration of its response time p1. We further suppose that it takes response time p2 for car 2 to react to car 1, during which it also speeds up with its maximum acceleration. After the response times, both cars are braking with minimum deceleration until coming to a full stop. ${d_1}^{\text{response}}$ and ${d_2}^{\text{response}}$ define the distance they drive during their corresponding response times, and ${d_1}^{\text{brake}}$ and ${d_2}^{\text{brake}}$ are the braking distance for each car. It should be here known that since the car 2 is driving in its correct lane, it is expected the car 2 brakes harder relative to car 1. The example image can be seen in the above figure. Consequently, the longitudinal distance in the opposite direction between two vehicles is considered safe if the actual distance is smaller than the safe distance, and the safe distance formula can be seen below [GZW22]:

$$\begin{aligned} d_{\text{RSS}}^{\text{long,oppo}}(v_1, v_2) = {} & d_1^{\text{response}}(v_1, p, a_{\text{maxaccel}}) \\ & + d_1^{\text{brake}}(v_1, p, a_{\text{maxaccel}}, a_{\text{minbrake}}) \\ & + d_2^{\text{brake}}(v_2, p, a_{\text{maxaccel}}, a_{\text{minbrakecorrect}}) \\ & + d_2^{\text{response}}(v_2, p, a_{\text{maxaccel}}) \end{aligned} \tag{2.2}$$

Lateral Safe Distance

The last safety distance is the lateral safe distance. The lateral safe distance is defined for 2 vehicles that can drive in the same direction or in the opposing directions. Here we also assume the worst, like before, for the distance calculation. Assume two vehicles, car 1 and car 2 in the figure 2.1 (c), are diving in the same direction with the velocities v1 and v2. Here we consider that car 1 is coming into the lane of car 2, and a lateral collision is possible. As soon as car 1 drives closer to car 2, we assume that the response time p begins for each cars. During the response time, both cars are speeding up with their respective maximum velocities. After the response time, both cars are braking with their minimum deceleration and come to a full stop. ${d_1}^{\text{response}}$ and ${d_2}^{\text{response}}$ denotes the driving distance during response times while ${d_1}^{\text{brae}}$ and ${d_1}^{\text{brake}}$ denote the driving distance during the braking phase. Furthermore, cars usually perform small lateral movements during driving [GZW22]. Therefore, this definition also takes this into account by adding a margin distance u. The lateral distance is considered safe if the actual distance value is bigger than the lateral safety distance, and it is calculated as following below [GZW22]:

$$\begin{aligned} d_{\text{RSS}}^{\text{lat}}(v_1, v_2) = u &+ d_1^{\text{response}}(v_1, p, a_{\text{maxaccel}}) \\ &+ d_1^{\text{brake}}(v_1, p, a_{\text{maxaccel}}, a_{\text{minbrake}}) \\ &+ d_2^{\text{response}}(v_2, p, a_{\text{maxaccel}}) \\ &+ d_2^{\text{brake}}(v_2, p, a_{\text{maxaccel}}, a_{\text{minbrake}}) \end{aligned} \tag{2.3}$$

It is important to mention that the maximum acceleration and minimum brake are different from car to car. Mobileye argues that the values should be decided by government and industry together [Mob].

Furhter Concepts

Based on the mathematical definitions above for safety distance, RSS model defines dangerous situations. A situation is considered dangerous in case both the longitudinal and lateral distances between them are not safe. If only one of them is non-safe, then the situation is classified as not dangerous. The responsible action against dangerous situations is called proper response, and both vehicles in interaction are supposed to take evasive maneuvers in order to avoid collisions. In case of an accident, RSS assigns blame to the party that violates its principles. That's why RSS is also called the responsibility-aware safety model [Mob].

2.3 ChatScene

Zhang et al. (2024), in their paper "ChatScene: Knowledge-Enabled Safety-Critical Scenario Generation for Autonomous Vehicles", present a novel approach leveraging large language models to generate safety-critical scenarios for autonomous driving [ZXL24]. Their study is particularly relevant for assessing and enhancing the safety of autonomous vehicles through simulation-based scenario generation, aligning closely with the broader focus of this thesis on risk evaluation in autonomous driving.

Scenario generation, the cornerstone of their approach, begins with gathering unstructured language instructions. These instructions are sent to an LLM, which can also be generated through LLM by prompting it with "Please, generate me some safety-critical scenarios.". Afterwards, LLM is prompted again with these sentences to extract information about scenarios that align with critical scenario components, such as adversarial behaviors of nearby vehicles. Later on, ChatScene encodes these components into embeddings to retrieve corresponding scenic snippetes from a pre-constructed database. Scenic is a domain-specific language that allows the definition of scenarios programmatically, and we will talk about it later in the methodology section. The last step is assembling the retrieved snippets into a complete and executable Scenic script that can be run in the Carla simulator [ZXL24].

By automating the generation of diverse and adversarial driving scenarios, ChatScene enhances the robustness of autonomous vehicle testing, addressing limitations in traditional scenario design methods. Its ability to translate high-level textual descriptions into executable simulations enables more comprehensive safety assessments, ensuring that AVs are exposed to a wider range of challenging conditions. It's also highly relevant for this thesis because we will integrate the RSS safety model into the framework provided by ChatScene.

2.4 Closely Related Approaches

In this section, details of two approaches applied to risk estimation for safety validation, are discussed. The approaches are based on very different stuff. Readers should note that this is not an exhaustive list, but an attempt to bring closer what has been tried before in regards to risk estimation.

2.4.1 Data- driven risk estimation

De Gelder et al. (2021), in their paper "Risk Quantification for Automated Driving Systems in Real-World Driving Scenarios" undertake a comprehensive study that analyzes the applicability of statistical methods for risk measurement for autonomous vehicles [Gel+21]. This study is highly relevant in the context of risk estimation for autonomous vehicles, a theme that resonates with the wide focus of this thesis.

The authors argue that the risk estimation methods in the widely accepted safety standards ISO 26262 and ISO/DIS 21448 are lacking. There is no complete method provided to quantify risk, and they depend on subjective judgments of safety experts. The result is a qualitative risk estimation. Therefore, the author's aim is to develop a method to estimate the risk of an ADS in a more quantitative and objective manner. For this, the authors propose a novel data-driven method for assessing and quantifying the risk of an ADS considering real-world driving scenarios. To provide a more objective method for risk quantification compared to existing approaches, the proposed method uses real-world data and relies less on the judgments of experts.

The proposed method works with decomposing the quantified risk into 3 aspects of risk as mentioned in the standards: exposure, severity, and controllability. The first step is identifying the relevant scenarios in the Operational Design Domain (ODD) of the ADS and then grouping them into scenario categories. The exposure is simply the expected number of encounters per unit of time of a scenario belonging to a scenario category. Thus, simply the data needs to be counted. Afterwards, a scenario from the broad scenario category needs to be simulated. These simulated scenarios are identified from real-world driving data that are collected in field operational tests or naturalistic driving studies. The aim is to represent real-world driving conditions. Severity is the probability of injury. It is calculated as the mean result over a large number of simulations. Controllability is the ratio of the probabilities of an injury with and without a backup operator, such as a human. The risk is calculated at the end as the multiplication of exposure, severity, and controllability. The output of the method is the expected number of injuries in a potential collision. Some factors, such as heavy rain and poor lighting, could cause hazardous behavior, and they are called triggering conditions.

In conclusion, the study by De Gelder et al. (2021) stands as a pivotal contribution to the field of safety validation for autonomous vehicles, particularly in the application of statisticsbased risk measurement for collision risk. The findings and future research directions highlighted in this paper establish an important foundation for advancing the field, providing valuable insights for future studies. However, it should also be noted that since this method estimates the risk based on data, the results might not represent the real world.

2.4.2 RSS-based risk quantification

Guo et al. (2022) address a critical challenge in the adoption of autonomous vehicles (AVs): ensuring traffic safety through responsibility-aware risk assessment. Their work builds on the Responsibility-Sensitive Safety (RSS) model, a framework designed to define interpretable safety boundaries for AVs. This new idea addresses the limitations in applicability of other risk quantification methods that fit a probability distribution to gathered data. The proposed approach evaluates the risk of being responsible for an accident in interaction scenarios by applying the safe distances from the RSS model. To complement this risk assessment, the authors introduce a safety level evaluation method that quantifies a vehicle's ability to navigate various risk levels. Additionally, this method provides insights into the relative safety of an AV's maneuvering strategy, comparing its driving behavior to that of other vehicles [GZW22].

In the context of this thesis, which aims to evaluate the risk values of RL-trained agents with the help of LLMs, this risk measuring method is at the core. We will incorparete this method to the existing ChatScene framework and analyse the safety of control. The results of this paper establish a basis for future research, like the one presented in this thesis, to further progress the field.

Chapter 3

Methodology

In the following section, the technical approach used to achieve the objectives of this thesis is outlined. Specifically, the integration of the Responsibility-Sensitive Safety (RSS) model into the ChatScene framework is described. Additionally, the methodology for using RSS to quantify risk in various driving scenarios is presented. We will begin with a brief explanation about the Carla simulator and Scenic language and follow with an overview of the ChatScene framework. Afterwards, we will talk about the integration of risk estimation based on the paper by Guo et al. into the ChatScene framework to evaluate how safe the reinforcement learning agents that are trained based on ChatScene drive.

3.1 CARLA Simulator

CARLA (Car Learning to Act) is an open-source autonomous driving simulator. It was built on the Unreal Engine 4 to run realistic driving simulations. Its extensive use in academia and industry makes CARLA a benchmark platform for autonomous vehicle research, particularly in scenario testing and reinforcement learning applications [Tea25a].

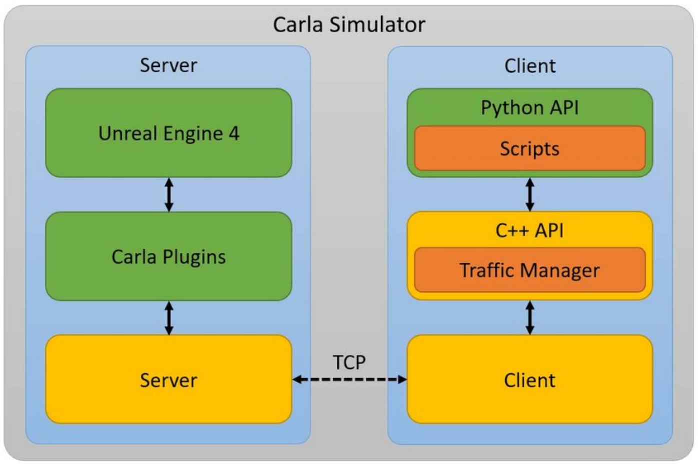


Figure 3.1: CARLA client-server architecture [Pir+21].

Carla simulator consists of scalable client-server architecture. The server takes care of everything that has to do with the simulation itself: sensor renderings, graphical renderings, physics computations, world-state updates, and much more. The client side consists of a group of client modules that are responsible for controlling the actors and setting the world conditions, such as weather. This is achieved by making use of the CARLA API that is available in Python. In the following, we will explain the important things to know about Carla simulator [Tea25a].

World is an object that represents the simulation. It acts as the abstraction layer containing various methods. Through this object, the client spawns actors and changes the weather [Tea25a].

Actors are the things that perform actions within the simulation. They have the ability to influence other actors as well. For example, vehicles, pedestrians, sensors, traffic signs, and traffic lights are all actors. Within the simulator, they are created through a blueprint library and spawned in a specified location in the map [Tea25a].

Sensors are actors that retrieve data from the environment. They are usually attached to the ego vehicle. Carla provides a wide range of sensors, such as cameras, radar, lidar, GPS, collision sensors, and also custom sensors like RSS sensor. The data is then accessible through callback functions in Python, and further processing can be done within those functions. For this thesis, the cameras and collision sensors are highly relevant as the ChatScene framework makes use of them for reinforcement learning training [Tea25a].

3.2 Scenic

Scenic is a domain-specific probabilistic programming language. It is used mainly for modeling the environments of cyber-physical systems like robots and autonomous vehicles. Scenic has an interface to Carla. Through the interface, it allows the creation of scenarios that can be run in the CARLA simulator. It is very important for scenario-based testing because of its scenario generation capabilities [Tea25b].

Scenic allows users to define dynamic objects, known as agents, which take actions over time within a simulated environment. These agents can represent vehicles, pedestrians, or other interactive entities. One of Scenic's key features is its ability to specify initial conditions, such as the positions, velocities, and orientations of agents, using probabilistic and geometric constraints [Tea25b].

Additionally, Scenic supports the definition of agent behaviors, which are sequences of actions that an agent executes throughout a scenario. These behaviors can include low-level controls, such as adjusting a vehicle's steering angle, acceleration, or braking, enabling the simulation of complex driving maneuvers. The flexibility of Scenic in describing both static scene layouts and dynamic interactions makes it a powerful tool for generating diverse and realistic test cases [Tea25b].

The following example demonstrates the capabilities of Scenic in defining agent behaviors and spatial relations within a traffic scenario. We have two behavior definitions, EgoBehavior and LeadingCarBehavior. The ego car is supposed to act the EgoBehavior, and EgoBehavior is just FollowLaneBehavior, so the car will just follow the lane after spawning at the specified speed. After that, we see an instruction to interrupt the ego behavior and make the ego vehicle brake if another car comes within a specified distance of ego. Leading car behavior and its following interrupt is similar to ego vehicle's. After that, we can see the definition of spatial relations; we spawn a trash can in the middle of the lane, and we spawn the vehicles on the road. The ego is following the lead car, and as soon as the lead car encounters trash, it will brake, and as soon as the lead car brakes, the ego will brake when distance requirements are fulfilled. At the end, we can see the termination specification of the following code. Otherwise, the scenario could go on indefinitely. This example shows that we can consider scenic to consist of some building blocks, and by mixing these building blocks correctly, we can achieve more diverse scenarios [Tea25b].

```
## DEFINING BEHAVIORS
behavior EgoBehavior(speed=10):
try:
do FollowLaneBehavior(speed)

interrupt when withinDistanceToAnyCars(self, EGO_BRAKING_THRESHOLD):
take SetBrakeAction(BRAKE_ACTION)

behavior LeadingCarBehavior(speed=10):
try:
do FollowLaneBehavior(speed)

interrupt when withinDistanceToAnyObjs(self, LEADCAR_BRAKING_THRESHOLD):
take SetBrakeAction(BRAKE_ACTION)

## DEFINING SPATIAL RELATIONS
lane = Uniform(*network.lanes)

obstacle = new Trash on lane.centerline

leadCar = new Car following roadDirection from obstacle for Range(-50, -30),
with behavior LeadingCarBehavior(LEAD_CAR_SPEED)

ego = new Car following roadDirection from leadCar for Range(-15, -10),
with blueprint EGO_MODEL,
with behavior EgoBehavior(EGO_SPEED)

require (distance to intersection) > 80
terminate when ego.speed < 0.1 and (distance to obstacle) < 30
```

Listing 3.1: Traffic Scenario 02 in Scenic [Tea25b].

3.3 Chatscene Framework

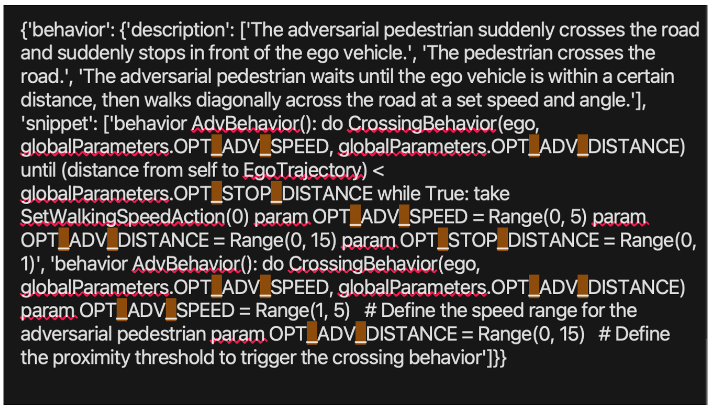
ChatScene is the cornerstone of this research, as they provide a method to generate new scenarios using LLMs, to train new reinforcement learning agents based on these scenarios, and to evaluate the trained models on scenarios. Furthermore, they provide us with pre-trained model checkpoints that we can use to evaluate our risk estimation approach [ZXL24].

As a basis, Chatscene makes use of the framework called SafeBench that was published by Xu et al. (2022) in a paper called "SafeBench: A Benchmarking Platform for Safety Evaluation of Autonomous Vehicles" [Xu+22]. This framework allows us to train reinforcement learning agents on some Scenic scenarios and evaluate the agents on these scenarios as well. However, the main problem here is that these scenarios have to be written by humans. ChatScene aims to improve this inefficiency by generating new scenarios methodically [ZXL24]. In the following, we will see how exactly this generation process works.

The first step for the generation of new scenarios is gathering adversarial scene descriptions. These scene descriptions can be written in natural language by humans or be received from LLMs through prompts. Afterwards, we send this set of scene descriptions into the LLM and ask it to bring these into a certain format with clear behavior, geometry, and spawn position descriptions for all the nearby adversarial vehicles, pedestrians, and objects. In the following, the descriptions are encoded into a high-dimensional space, and the embeddings are used as keys to retrieve the relevant code snippets from the pre-built retrieval database. Afterwards, the snippets will be assembled into a complete Scenic scenario script and will be ready to run in the CARLA simulator [ZXL24].

In these scenarios, there are variables that define some property of the scene, such as adversarial vehicle speed, etc. Chatscene samples these values from a probability distribution many times and adjusts the sampling range after each scenario run, as some sampling range result in more safety-critical scenarios. This is called scenario training in ChatScene and is done before training agents usually [ZXL24].

Below, in the first figure, we can see a section of their retrieval database. It has 2 behaviors and their corresponding scenic snippets. The figure below shows the natural language inputs that go into the LLM as initial scenario descriptions.



```
{'behavior': {'description': ['The adversarial pedestrian suddenly crosses the road
and suddenly stops in front of the ego vehicle.', 'The pedestrian crosses the
road.', 'The adversarial pedestrian waits until the ego vehicle is within a certain
distance, then walks diagonally across the road at a set speed and angle.'],
'snippet': ['behavior AdvBehavior(): do CrossingBehavior(ego,
globalParameters.OPT_ADV_SPEED, globalParameters.OPT_ADV_DISTANCE)
until (distance from self to EgoTrajectory) <
globalParameters.OPT_STOP_DISTANCE while True: take
SetWalkingSpeedAction(0) param OPT_ADV_SPEED = Range(0, 5) param
OPT_ADV_DISTANCE = Range(0, 15) param OPT_STOP_DISTANCE = Range(0,
1)', 'behavior AdvBehavior(): do CrossingBehavior(ego,
globalParameters.OPT_ADV_SPEED, globalParameters.OPT_ADV_DISTANCE)
param OPT_ADV_SPEED = Range(1, 5)   # Define the speed range for the
adversarial pedestrian param OPT_ADV_DISTANCE = Range(0, 15)   # Define
the proximity threshold to trigger the crossing behavior']}}
```

Figure 3.2: Examples from the retrieval database [ZXL24].

```
The ego vehicle is driving on a straight road when a
pedestrian suddenly crosses from the right front and
stops as the ego vehicle approaches.
The ego vehicle is driving on a straight road when a
pedestrian suddenly crosses from the left front and
stops as the ego vehicle approaches.
The ego vehicle is driving on a straight road
when a pedestrian suddenly crosses from the left
front, behind a bench, and stops as the ego vehicle
approaches.
The ego vehicle is driving straight through a
four-way intersection when a pedestrian suddenly
crosses from the left front on the zebra crossing
and stops as the ego vehicle approaches.
The ego vehicle is driving straight through a
four-way intersection when a pedestrian suddenly
crosses from the right front and stops as the ego
vehicle approaches.
```

Figure 3.3: Example scenario description prompts from ChatScene [ZXL24].

3.4 Integration

Now that we are informed about the building blocks, we will begin with the actual integration. We don't have to implement the RSS safety method ourselves as Intel provides a library called ad-rss-lib written in C++ that implements the functionality of the Responsibility Sensitive Safety model (RSS).

The ad-rss-lib library is provided as a standalone implementation of the RSS module. It receives sensor information from the simulator as input and is capable of providing the actuator with driving restrictions as output. Input to the library consists of information about all road agents in the surrounding environment of the ego vehicle. For each object, the library creates a description called situation that describes the properties of the object-ego vehicle pair. Necessary RSS safety checks are performed for each situation, and a proper response is calculated in the following regarding each situation. Afterwards, individual responses are combined into one overall response. This response is the restricting actuation command. It limits the longitudinal and lateral acceleration of the ego vehicle [Cor25].

This library is provided as a standalone implementation of the RSS module. For its integration into autonomous driving simulators, simulators have to interface with it. They are supposed to convert the AV sensor data from the simulator into the format the library expects. Furthermore, simulators are also responsible for converting the outputted proper response back into actuation commands [Cor25].

Luckily, this RSS library is already interfaced into the Carla simulator. The figure below shows the integration. The interfacing happens through RSS Sensor and RSS Restrictor. For us, only the RSS sensor is of importance, as we will use it for risk estimation. The sensor has to be created in the CARLA simulator and be attached to the ego vehicle; it provides us with information after the RSS checks are done in the ad-rss-lib library. While it allows us to access the raw RSS safe distances and actual distances, it also processes them and gives us information on the longitudinal margin, lateral margin, and whether a situation is dangerous. In the below figure, the exact communication of the RSS library and the CARLA simulator can be seen. The sensor receives the world data from the server and extracts a world model as the ad-rss-lib expects. Then, the library extracts the situations and does the calculations per situation. The calculated response is then sent to the client or directly to the RSS restrictor, which we omitted in our work [Tea25a].

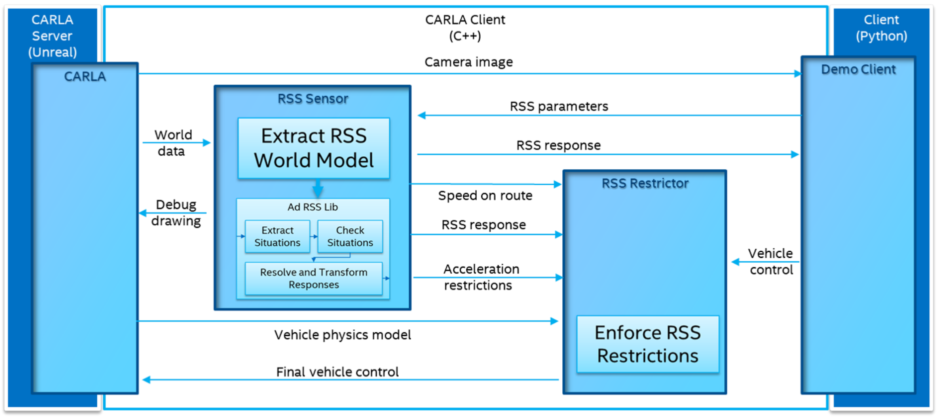


Figure 3.4: RSS Sensor in CARLA Simulator [Tea25a].

Now that we have all the necessary tools, we will next talk about the actual integration. In the figure below, one can see a high level view of the integration, which we will go into a bit more detail in the next section.

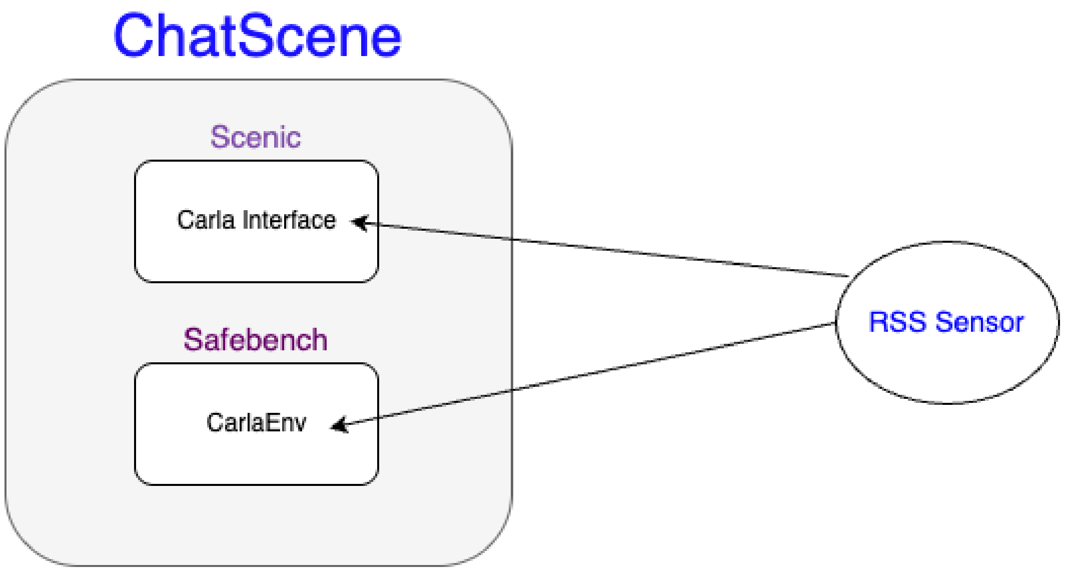


Figure 3.5: Overview of the integration.

3.4.1 Integration into the Scenic

Our first approach is initializing the RSS sensor in the Scenic codebase. As we discussed, Scenic is responsible for the initialization of the scenario objects and for controlling these objects. The idea is to initialize it while Scenic is initializing the scene objects and to attach it to the ego vehicle.

On the one hand, this approach suffered from some drawbacks due to the way ChatScene is making use of Scenic. ChatScene initializes the scenario many times before running the actual scene to adjust the scenario parameter sampling. This causes performance issues and sensor instability for the RSS sensor, as the sensor is being initialized and destroyed constantly.

On the other hand, the advantage of this approach lies outside of the ChatScene. Since we integrated it into the Scenic, anyone who makes use of Scenic to generate and manage scenarios can directly perform a similar integration and have access to the benefits of the RSS safety model.

3.4.2 Integration into the Safebench

Our second approach is initializing the RSS sensor within the Safebench platform. Chatscene uses this as the foundation. Here, we observe that some sensors, such as camera and lidar are initialized and attached to the ego vehicle within the CarlaEnv class, an OpenAI-gym style interface for CARLA simulator. Therefore, we just initialize the RSS sensor alongside other sensors in this class and attach it to the ego vehicle. This method also does not suffer from the same instabilities as the first method.

3.4.3 Risk Estimation Using the RSS Sensor

Now that we have successfully integrated the RSS model into ChatScene, we can think of the quantitative risk assessment we want. Here, we use the risk estimation methodology that was introduced by Guo et al. (2022) that we reviewed in the literature review section of this thesis [GZW22]. We can estimate risk with the safe distance information provided by the RSS sensor. The quantitative risk definition according to their paper is as follows:

$$\text{risk}_{\text{abs}}(d_{\text{current}}, d_{\text{safe}}) = \begin{cases} 0, & \text{if } d_{\text{current}} \geq d_{\text{safe}} \\ 1 - \frac{d_{\text{current}}}{d_{\text{safe}}}, & \text{if } d_{\text{current}} < d_{\text{safe}} \end{cases} \tag{3.1}$$

Figure 3.6: Risk estimation function [GZW22].

Chapter 4

Results

In this section, we will present a demonstration of the results of this thesis. ChatScene provides us with pre-trained reinforcement learning agent checkpoints. We will, in the following, evaluate how the agent drives in a given scenario. We will only go through one concrete scenario to illustrate what we gained by demonstrating the capabilities of the integration. Afterwards, we will discuss the shortcomings of the methodology and its reasons. The source code can be found in this link: https://github.com/wardstoneX/ChatScene_rss

4.1 Example

The scenario we chose for the demonstration can be found in the ChatScene repository, and we added the Scenic script in the appendix. It is a lane change scenario. The ego vehicle and the 2 other vehicles are driving in the same direction. The ego vehicle is behind a leading car and next to a parallel driving car. The ego vehicle intends to change to the right lane as the leading car is driving slowly. However, a parallel car is blocking its path to the right. While the ego vehicle is controlled by the RL agent, the other vehicles are controlled by Scenic.

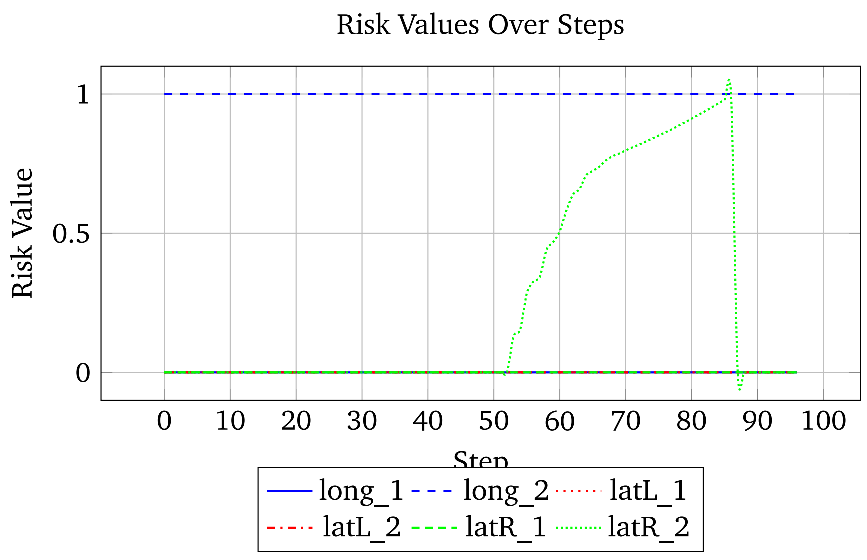


Figure 4.1: Risk Values for Car 1 and Car 2 (Longitudinal, Lateral Left, and Lateral Right).

We ran the scenario and put its results into a graph format. In the graph above, we can see the 3 types of risks that our integration provides us: longitudinal risk, lateral left risk, and lateral right risk. In the scenario, we have 2 other vehicles. Therefore, we have 3 risks for each object. The risk values for the leading car, car 1, are named with 1 subscript, and those for the parallel car, car 2, are named with subscript 2.

During the entire scenario, the lateral left risk regarding car 1 and car 2 is 0, as these vehicles are not to the left of the ego vehicle. As the ego vehicle maintains a safe longitudinal distance to the leading vehicle, the longitudinal risk is 0 regarding the leading vehicle. However, the longitudinal risk regarding the parallel car is 1 during the entire scenario, as they drive practically side by side longitudinally. Lastly, the lateral right risk to the leading vehicle is also 0 during the entire scenario, as they drive in the same lane. In contrast, we can see the most interesting part of the scenario run. As the ego vehicle tries to merge into the right lane, we can observe an increase in lateral right risk to the parallel car, indicating a dangerous decision-making. This was also observed from the accident that happened at the end.

In the following, we graphed when the situation became dangerous. As we know, both lateral and longitudinal distances have to be unsafe for a situation to be considered dangerous. The below graph reflects that clearly. As both lateral and longitudinal risks are above 1, the situation became dangerous.

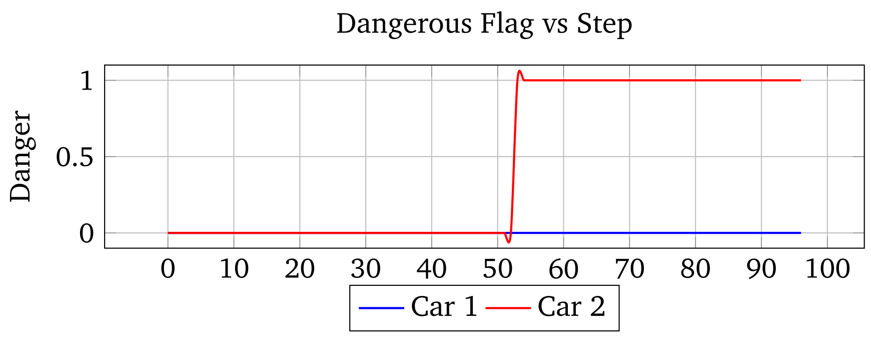


Figure 4.2: Dangerous Flag vs Step for Car 1 and Car 2.

In the next figure, we graph the longitudinal distance, safe and current. We observe that longitudinal current distance is 0 during the scenario. However, RSS clearly tells us that the ego vehicle needs more longitudinal distance to the parallel vehicle. And during the lane change, RSS recommends more distance to be kept.

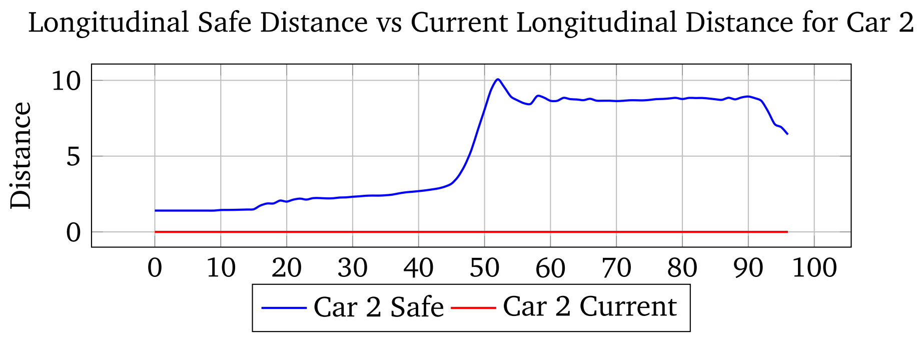


Figure 4.3: Longitudinal Safe vs Current Longitudinal Distance for Car 2.

And the last figure is the lateral right distances. It shows how the lateral distances change during the scenario. What's noteworthy here is that as the ego vehicle starts merging to the right, RSS signals that it violated the safe distance.

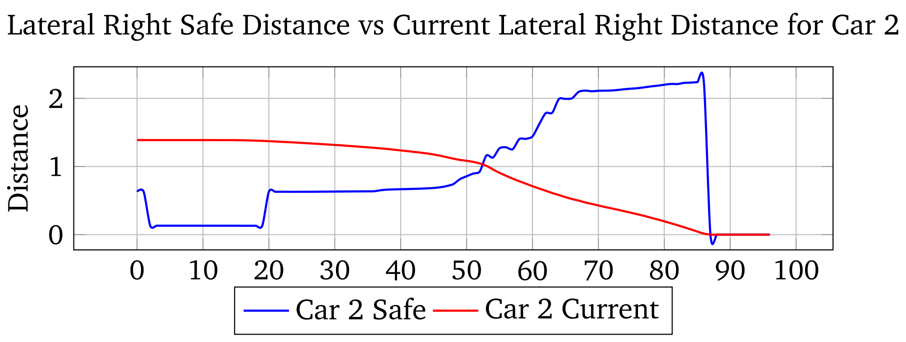


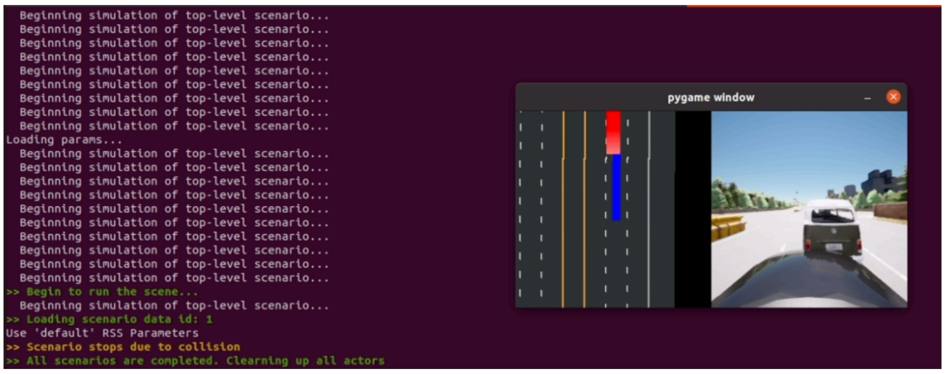
Figure 4.4: Lateral Right Safe vs Current Lateral Right Distance for Car 2.

Above, we included the most interesting graphs for the scenario and walked through them. However, we have more data available, but they do not offer significant information. We still added them into the appendix for the sake of completeness.

4.2 Observation

In the CARLA simulator, other sensors have a variable called sensor tick. It controls the time between each consecutive data capture through sensors. This does not exist for the RSS sensor. Therefore, we cannot specify its value. We have observed sometimes that the sensor data is lagging behind the actual frame of the simulation.

We will demonstrate the delay with the following scenario. The ego vehicle drives in the same lane as the leading vehicle in the same direction. It accelerates and collides with the back of the leading vehicle. ChatScene ends the scenario due to the collision and cleans up the scenario, which means our RSS sensor is destroyed. In the figure below, the collision at the end can be seen.



```
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
Loading params...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
  Beginning simulation of top-level scenario...
>> Begin to run the scene...
  Beginning simulation of top-level scenario...
>> Loading scenario data id: 1
Use 'default' RSS Parameters
>> Scenario stops due to collision
>> All scenarios are completed. Clearning up all actors
```

Figure 4.5: Scenario stops due to collision.

In this scenario, we want to highlight the sensor delay. Therefore, we only focus on the longitudinal distance information to the leading vehicle. We graph the data that was provided by the RSS sensor below. In the graph, we observe that the current longitudinal distance in red is going down in the second part of the graph. Therefore, the margin between the safe distance and the current distance is decreasing. We know that there was a collision. Thus, the current longitudinal distance should have been declined until zero. But that is not the case. We suspect the reason is that we can't control the tick of the RSS sensor. There could be other reasons causing this issue, but we are not aware of it currently.

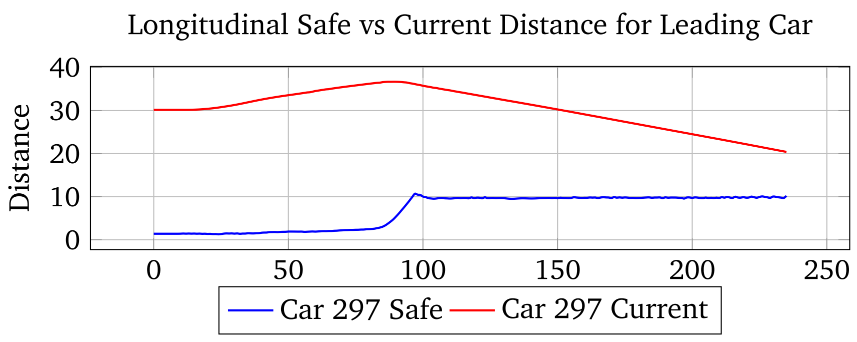


Figure 4.6: Longitudinal Safe vs Current Distance for Leading Car.

Chapter 5

Conclusion

Ensuring the safety of autonomous vehicles (AVs) is a very difficult problem, as traditional validation methods are often inefficient and costly. This thesis explored the integration of the Responsibility-Sensitive Safety (RSS) model into the ChatScene framework to enhance its safety validation capabilities. This work aimed to contribute to the field of autonomous driving safety by combining a mathematical safety model with the scenario generation capabilities of ChatScene, thus creating a unified framework where we can observe the driving risk and train reinforcement agents.

Through our integration, we demonstrated that the RSS safety model can provide risk estimation within the ChatScene framework, offering a systematic way to assess the safety of AV decision-making. By quantifying risk, AVs can better evaluate dangerous situations and make safer driving choices. The results suggest that adding RSS-based risk estimation into AV development can improve both testing methodologies and real-world deployment strategies.

While this research successfully achieved its goals, there is more work to do in the future. One promising direction is adding risk values into reinforcement learning reward functions to encourage safer driving behavior. Additionally, extending risk estimation to predict future risk using trajectory planning methods could further improve AV decision-making. By anticipating potential dangers, autonomous vehicles can proactively adjust their actions, improving overall safety in the traffic.

In conclusion, this thesis contributes to the field of AV safety validation by integrating the RSS safety model into the ChatScene framework, demonstrating its practical applications for risk estimation, and highlighting pathways for future research. As AV technology continues to improve, such advancements will be very important in bringing theoretical safety models into the real world, thus enabling safer autonomous transportation systems.

Appendix A

Appendix

```
Town = globalParameters.town
EgoSpawnPt = globalParameters.spawnPt
yaw = globalParameters.yaw
lanePts = globalParameters.lanePts
egoTrajectory = PolylineRegion(globalParameters.waypoints)
param map = localPath(f'../maps/{Town}.xodr')
param carla_map = Town
model scenic.simulators.carla.model
EGO_MODEL = "vehicle.lincoln.mkz_2017"

behavior AdvBehavior():
while True:
take SetVelocityAction(*ego.velocity)

param OPT_LEADING_DISTANCE = Range(0, 30)
param OPT_LEADING_SPEED = Range(1, 5)
param OPT_GEO_Y_DISTANCE = Range(0, 5)

ego = Car at EgoSpawnPt,
with heading yaw,
with regionContainedIn None,
with blueprint EGO_MODEL

NewSpawnPt = egoTrajectory[20]
LeadingAgent = Car following roadDirection from NewSpawnPt for
    globalParameters.OPT_LEADING_DISTANCE,
with regionContainedIn None,
with behavior
    FollowLaneBehavior(target_speed=globalParameters.OPT_LEADING_SPEED)

laneSec = network.laneSectionAt(EgoSpawnPt)
advLane = laneSec._laneToRight.lane
IntSpawnPt = OrientedPoint following roadDirection from EgoSpawnPt for
    globalParameters.OPT_GEO_Y_DISTANCE
projectPt = Vector(*advLane.centerline.project(IntSpawnPt.position).coords[0])
advHeading = advLane.orientation[projectPt]
AdvAgent = Car at projectPt,
with heading advHeading,
with regionContainedIn None,
with behavior AdvBehavior()
```

Listing A.1: Lane Change Scenario 02 in Scenic [**scenic_doc**].

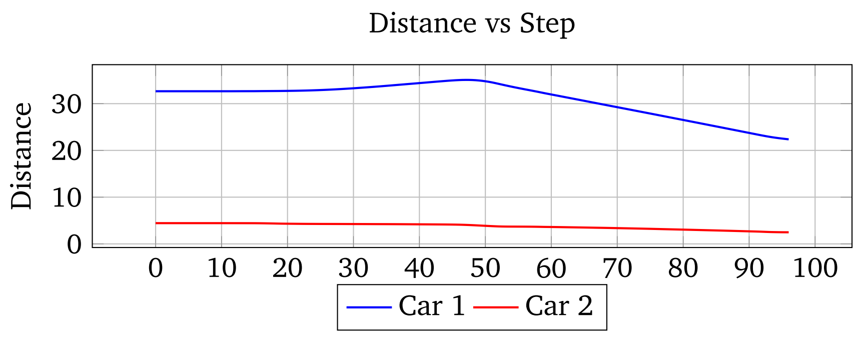


Figure A.1: Distance vs Step for Car 1 and Car 2.

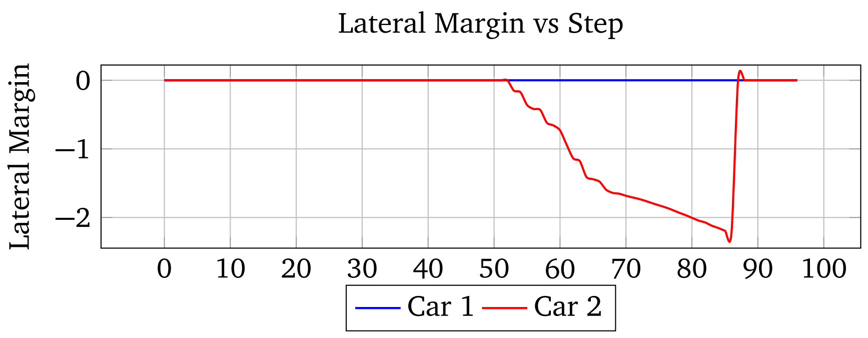


Figure A.2: Lateral Margin vs Step for Car 1 and Car 2.

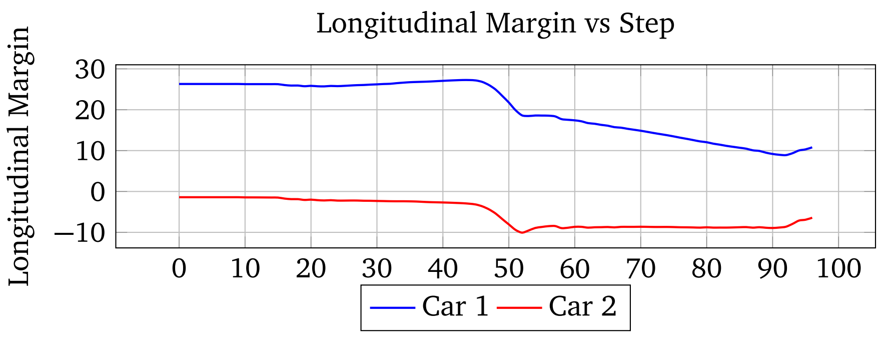


Figure A.3: Longitudinal Margin vs Step for Car 1 and Car 2.

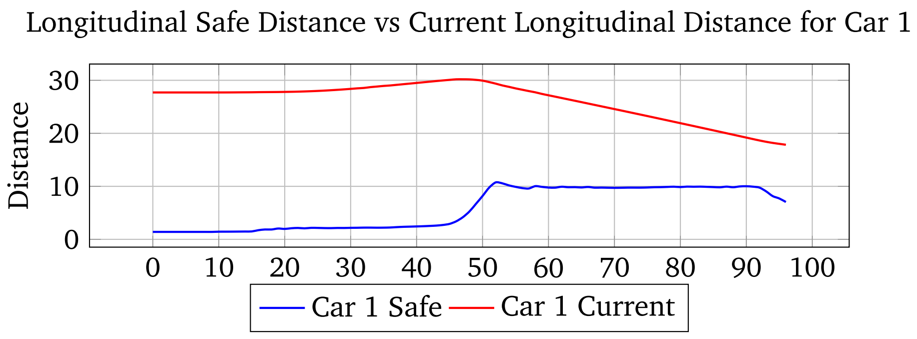


Figure A.4: Longitudinal Safe vs Current Longitudinal Distance for Car 1.

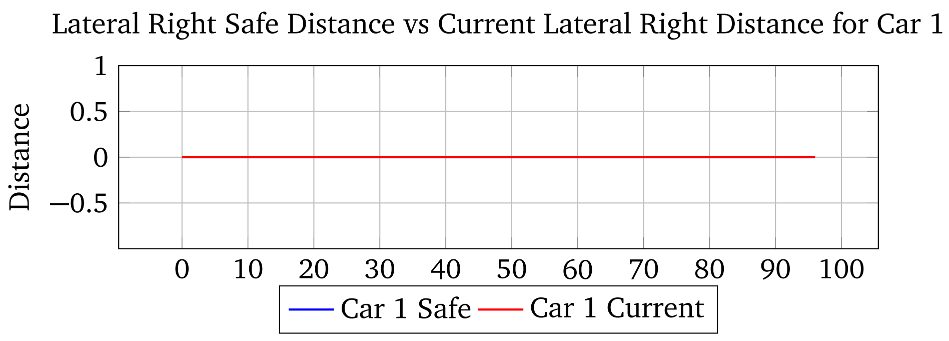


Figure A.5: Lateral Right Safe vs Current Lateral Right Distance for Car 1.

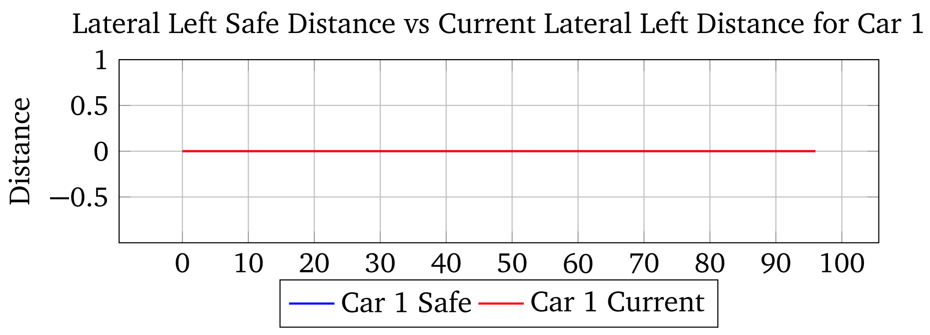


Figure A.6: Lateral Left Safe vs Current Lateral Left Distance for Car 1.

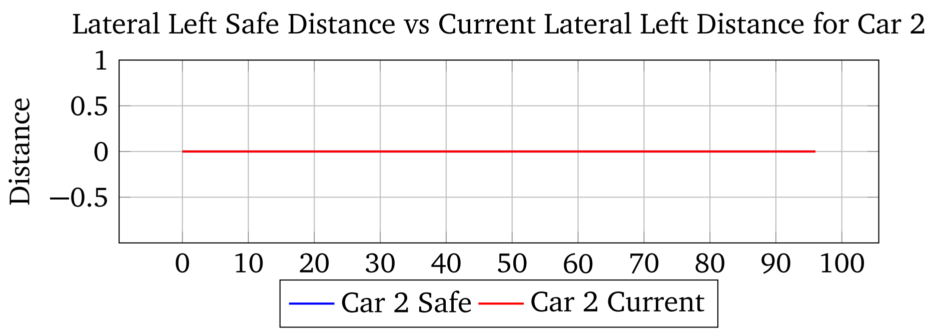


Figure A.7: Lateral Left Safe vs Current Lateral Left Distance for Car 2.

Bibliography

[Cha17] Chan, C.-Y. “Advancements, Prospects, and Impacts of Automated Driving Systems”. In: *International Journal of Transportation Science and Technology* 6.3 (2017). Available online 28 July 2017, Published by Tongji University Press, pp. 208–216. DOI: 10.1016/j.ijtst.2017.07.008. URL: https://doi.org/10.1016/j.ijtst.2017.07.008.

[Cor25] Corporation, I. *AD-RSS-Lib: Library implementing the Responsibility Sensitive Safety model (RSS) for Autonomous Vehicles*. Accessed: 2025-02-13. 2025. URL: https://github.com/intel/ad-rss-lib.

[Enc23] Encyclopaedia Britannica. *Autonomous Vehicle*. 2023. URL: https://www.britannica.com/technology/autonomous-vehicle.

[Fen+23] Feng, S., Sun, H., Yan, X., Zhu, H., Zou, Z., Shen, S., and Liu, H. X. “Dense Reinforcement Learning for Safety Validation of Autonomous Vehicles”. In: *Nature* (2023). Published online: 22 March 2023. DOI: 10.1038/s41586-023-05732-2. URL: https://doi.org/10.1038/s41586-023-05732-2.

[Gel+21] Gelder, E. de, Elrofai, H., Saberi, A. K., Paardekooper, J.-P., Camp, O. O. D., and Schutter, B. D. “Risk Quantification for Automated Driving Systems in Real-World Driving Scenarios”. In: *IEEE Access* 9 (2021), pp. 168953–168965. DOI: 10.1109/ACCESS.2021.3136585.

[GZW22] Guo, P., Zhu, Q., and Wu, X. “Responsibility-Sensitive Collision Risk Assessment and Maneuvering Safety Evaluation”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 978–984. DOI: 10.1109/ITSC55140.2022.9922228.

[Mob] Mobileye. “Implementing the RSS Model on NHTSA Pre-Crash Scenarios”. In: (). URL: https://static.mobileye.com/website/corporate/rss/rss_on_nhtsa.pdf.

[Pir+21] Pirri, P., Pahl, C., Ioini, N. E., and Barzegar, H. R. “Towards Cooperative Maneuvering Simulation: Tools and Architecture”. In: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021. DOI: 10.1109/CCNC49032.2021.9369507.

[SAE21] SAE International. *SAE Levels of Driving Automation: J3016 Update*. Accessed: 2025-01-03. 2021. URL: https://www.sae.org/blog/sae-j3016-update.

[Sha17] Shai Shalev-Shwartz Shaked Shammah, A. S. “On a Formal Model of Safe and Scalable Self-driving Cars”. In: *Mobileye* (2017). DOI: 10.48550/arXiv.1708.06374v6. URL: https://arxiv.org/abs/1708.06374v6.

[Tea25a] Team, C. *CARLA Simulator Documentation*. Accessed: 2025-02-13, also includes the github repository. 2025. URL: https://carla.readthedocs.io/en/latest/.

[Tea25b] Team, S. *Scenic Documentation*. Accessed: 2025-02-13, , also includes the github repository. 2025. URL: https://docs.scenic-lang.org/en/latest/.

[Vol22] Volpe, M. A. *Man Crashing Into Another Car While Using Park Assist Delights Viewers*. Accessed: 2025-02-13. 2022. URL: https://www.newsweek.com/man-crashing-car-park-assist-viral-video-delights-viewers-1765364.

[Wan+24] Wang, Y., Jiao, R., Zhan, S. S., Lang, C., Huang, C., Wang, Z., Yang, Z., and Zhu, Q. "Empowering Autonomous Driving with Large Language Models: A Safety Perspective". In: *arXiv:2312.00812v4 [cs.AI]* (2024). URL: https://arxiv.org/abs/2312.00812.

[Xu+22] Xu, C., Ding, W., Lyu, W., Liu, Z., Wang, S., He, Y., Hu, H., Zhao, D., and Li, B. "SafeBench: A Benchmarking Platform for Safety Evaluation of Autonomous Vehicles". In: *Proceedings of arXiv*. 2022. DOI: arXiv:2206.09682v4.

[ZXL24] Zhang, J., Xu, C., and Li, B. "ChatScene: Knowledge-Enabled Safety-Critical Scenario Generation for Autonomous Vehicles". In: *arXiv preprint arXiv:2405.14062* (2024). URL: https://arxiv.org/abs/2405.14062.