Politecnico di Torino

Institut für Kraftfahrzeuge (ika)
RWTH Aachen University

Master's Degree in Automotive Engineering

Master of Science Thesis

Combining Scenario Elements for Scenario-based Testing

Supervisors
Prof. Francesco Paolo Deflorio
M.Sc. Christoph Glasmacher

Candidate
Mohammad Javad Ahmadi

November, 2024

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Ensuring the functionality and safety of automated vehicles (AVs) is crucial for validating their performance in the complex and dynamic environment of modern traffic systems. This process involves not only assessing the vehicle's performance but also evaluating how it interacts with various road users, including pedestrians, cyclists, and other vehicles. Additionally, the presence of these road users, as well as infrastructural elements and environmental factors, presents numerous challenges that AVs must navigate safely and efficiently.

Initially, Hauer et al. discuss the challenges of testing autonomous driving systems [7], showing that testing AVs in real-world conditions might be the most straightforward approach. Real-world testing provides valuable insights by exposing vehicles to actual road environments and spontaneous human behaviors. However, this approach presents significant limitations. It is computationally expensive and requires extensive time and resources to run exhaustive real-world scenarios. Additionally, it may expose participants to unnecessary risks and lack the ability to cover edge cases and rare but critical situations comprehensively, making it impossible to test all real-world scenarios. Also, reproducibility and limited control are the two other cons of real-world testing due to the lack of consistent and controlled testing and controlling of the real-world variables.

To address these challenges, it is essential to test AVs in scenarios that realistically simulate the wide range of conditions they may encounter. This shows how the vehicle can manage different situations, aiming at improving overall traffic safety. The design and implementation of such scenarios are fundamental to advancing the safety and reliability of AV technologies.

In light of these limitations, an alternative and more efficient approach is required. This research will focus on the concept of scenario generation, which involves creating controlled, repeatable scenarios based on real-world data. These scenarios are processed and validated to ensure they accurately reflect real-world conditions and challenges. The scenarios are then systematically used in the testing procedures of automated vehicles, offering a more practical, scalable, and safe method for validating vehicle performance and safety.

To address this challenge, a novel merging is developed. This method combines elements of existing scenario-generation techniques with advanced data processing and validation procedures. This approach will create a diverse set of controlled, repeatable scenarios that more accurately reflect the complexities of actual driving conditions. The resulting scenarios are systematically integrated into the testing procedures for automated vehicles, offering a more comprehensive and nuanced evaluation of vehicle performance and safety.

Chapter 2

State of the Art

Introduction: In this chapter, the driving scenarios and their types as a basis for the definition of the scenarios will be discussed, then the data source that is used for the scenario and a definition of the similarity and how it functions will be discussed. Finally, several evaluation methods as metrics for the scenario analysis will be addressed.

Following, levels of automation are shown in Figure 2.1 according to the SAE J3016 [14]. Subsequently, they will be explained briefly, and as a point of this work, this is a step towards achieving higher levels of automation.

2.1 Levels of Automation

As shown in Figure 2.1, the SAE J3016 [14] illustrates the various levels of automation in automated driving systems, ranging from Level 0 (no automation) to Level 5 (full automation). These levels, as defined by the Society of Automotive Engineers (SAE), categorize the extent to which an automated driving system can control the vehicle and the responsibilities that fall upon the human driver. A detailed description of each level is provided below:

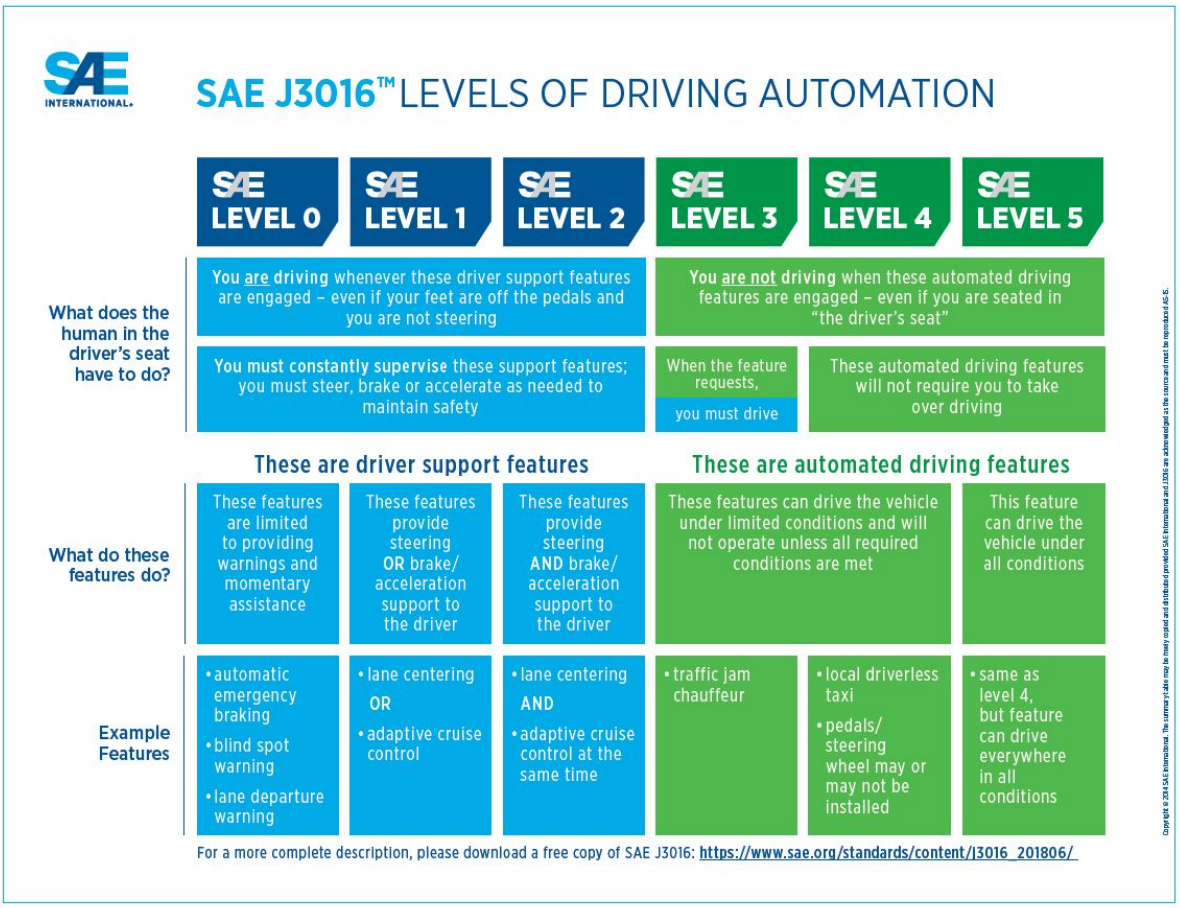


Figure 2.1: Levels of Automation [14]

Level 0: No Automation: There is no automation at this level. All aspects of the driving task are controlled by the human driver: steering, accelerating, braking, and observing the surrounding environment. Warning systems may be present in the vehicle to alert the driver to potential hazards, but these systems do not perform any driving functions.

Level 1: Driver Assistance: The lowest level of automation covers basic driver assistance systems found in level 1 vehicles. The vehicle could have a system to help with steering (like lane-keeping assistance) or acceleration and braking (adaptive cruise control), but not both at once. The majority of driving responsibilities continue to rest with the human driver, who must observe the environment and be ready to resume control of the vehicle at any time.

Level 2: Partial Automation: At this level, vehicles can have integrated automated functions like steering and acceleration that can be used at the same time but only under specific circumstances. The vehicle keeps its position in the lane and modifies its speed according to other vehicles. The human driver remains responsible for the driving task at all times during operation and must monitor the road and surroundings and take control when the system fails to respond correctly.

Level 3: Conditional Automation: This level is a conditional one and corresponds to the level of automated driving, where a vehicle can control all aspects of driving within certain conditions or environments (for instance, on a highway). At this level the human driver does not

need to keep his eyes on the environment. Nevertheless, the driver has to be present and ready to resume control when the system demands this. This level of automation implies that the system is capable of responding to most driving situations, but there may be certain situations that demand the intervention of the human driver.

Level 4: High Automation: vehicles can carry out all driving tasks and environmental monitoring with no human intervention given defined conditions (e.g. within a geo-fenced area or under certain weather conditions). If these conditions are not met, the vehicle may require the driver to take over. If the driver does not respond, the vehicle can safely terminate the mission and pull over. The system is able to operate and respond to emergencies and unexpected situations on its own within its ODD.

Level 5: Full Automation: The final level, Level 5, means full automation and such a vehicle can drive under any condition and can steer, brake, or turn without the need to have a human driver. They can function without a steering wheel, pedals, or even a human inside to drive and can travel in any environment, any time, and in any situation. It can manage complex traffic conditions, various kinds of weather, and unexpected events and provides the necessary safety and optimization during dehumanized navigation.

2.2 Scenario-based Testing

The way in which vehicle automation progresses makes the human driver's responsibility transfer to automated systems more critical with increasing levels of automation. The shift underscores the critical importance of validating these systems to operate safely and effectively across a wide range of driving conditions. Scenario-based testing has been proposed as a viable alternative. The Pegasus project [4] highlighted below, broke down the broader traffic environment into several manageable scenarios, which can be systematically evaluated, and recommended this approach. The method of scenario-based testing includes the identification, definition, and execution of certain traffic scenarios that correspond to typical critical events that AVs may face. The scenarios are intended to encompass various conditions: regular driving processes and unusual or dangerous incidents.

The scenario-based testing approach offers several advantages over traditional real-world testing. Firstly, it allows for a more controlled testing environment, where specific variables can be manipulated to assess the behavior and performance of AD systems under various conditions. This method enables the reconstruction of scenarios that are difficult or dangerous to reproduce in real life, such as emergency maneuvers or severe weather conditions, without endangering human participants.

Furthermore, scenario-based testing can contribute to the functional safety and robustness of automated driving functions. By isolating and testing individual scenarios, it becomes possible to identify weaknesses and areas for improvement in the ADS. Moreover, this approach facilitates repeatability and consistency in testing, ensuring that results are reliable and comparable across different systems and testing cycles.

Consequently, scenario-based testing represents a methodology for the validation and verification of ADS. By systematically simulating real-world traffic conditions in a controlled environment, this approach mitigates the limitations of traditional testing methods, paving the way for higher levels of vehicle automation with improved safety and reliability. The following sections will further explore the methodologies for scenario generation, the implementation of these scenarios in testing, and the implications for the broader automotive industry.

2.3 Driving Scenarios

A Driving scenario is a sequence of scenes [4]; these scenarios emulate the complexity and diversity of real-world driving conditions and thus enable the possibility that automated vehicles (AVs) can operate safely and reliably in numerous ways. The use of driving events and sequences helps researchers and engineers to systematically investigate how an automated vehicle reacts to different stimuli and challenges in a controlled environment. A driving scenario typically comprises severals key components, each representing different aspects of the traffic environment [4]:

Scene: The scene defines the intrinsic and evolving elements which configure the context of the driving action. This comprises the spatial configuration and detailed elements such as road layout, paint markings, vehicle placement, pedestrian location, traffic signals, indicators, and other critical elements of the transportation system. The scene presents a momentary picture of the environment while interacting within it.

Events: A sequence of actions and changes that occur over time is known as events. Such events include vehicles swapping leads, pedestrians crossing the road, or an instruction from a traffic light turning red from green. These events are designed to be a threat to the automated driving system, which must detect and interpret these dynamic events and give an appropriate response.

Ego Vehicle: In this thesis, the ego vehicle is the vehicle being tested within the scenario. It is equipped with an automated driving system that is under evaluation. The ego vehicle is not a part of the scenario, but it is added to the scenario to see its behavior and the surrounding objects. The ego vehicle's responses to various events and interactions within the scene provide critical data on the effectiveness and safety of the automation features in addition to other Road Users (RUs), which is the focus of this thesis as the scenario. These components(scenes, events) work together to simulate real-world driving conditions that are essential for evaluating the performance and safety of AVs. By structuring scenarios around specific scenes, events, and the role of the ego vehicle, researchers can create targeted tests that replicate common driving situations as well as rare or hazardous events that may pose significant challenges.

2.3.1 Scenario Levels:

Functional Scenario: A functional scenario models typical driving behaviors and their interactions to represent natural traffic dynamics at a high level of abstraction. This type of scenario describes events through narrative language: For example, a vehicle makes a left turn at an intersection while another vehicle waits at a red light that later turns green. This kind of scenario focuses on the sequence of action that is expected, not on some specific technical parameters.

The use of functional scenarios is crucial for the development and testing of automated vehicle systems to guarantee they can operate safely and as expected in typical traffic conditions. They provide a starting point for creating more comprehensive test cases and serve to synchronize automated vehicle responses to human-like driving modes that follow established traffic regulations for safe operation.

Abstract Scenario: What makes an abstract scenario different from a functional scenario is that it is more formal and more structured and is designed to be machine-readable. It gives a detailed technical description of driving situations using standardized formats and parameters that can be used in simulations and automated testing environments. Such scenarios are vital for the systematic testing and validation of automated driving systems and for ensuring that the testing is consistent and repeatable across different test cases.

Logical Scenario Class: Here is a logical scenario class that defines the attributes and parameters of abstract scenarios and their variations and dependencies. It provides a blueprint for creating various instances of scenarios, defining how parameters can differ and how they are related to one another. This approach enables systematic testing by testing different configurations within a given scenario framework.

Logical Scenario Instance: A real scenario instance is a set of parameters with specific values and declared parameters that have been filled with real-world knowledge or data. It is a concrete configuration of a scenario that can be used to put automated driving systems to the test and validate them in real conditions.

Concrete Scenario: A concrete scenario represents a particular instance of a logical scenario in which all parameters are defined by specific, precise, concrete values. This kind of scenario is utilized for practical, real world testing and verification of automated driving systems, giving a clear and exact definition of a configuration for assessing performance.

2.3.2 6-Layer Model

The 6-Layer Model [15] is associated with the Pegasus project and provides a structured framework for organizing driving elements and traffic scenarios. This model is used to structure the components and provide comprehensive coverage of different traffic conditions and interactions. This model is described in this thesis, and the methodology will be done in layer 4. The following outlines the six layers of this model.

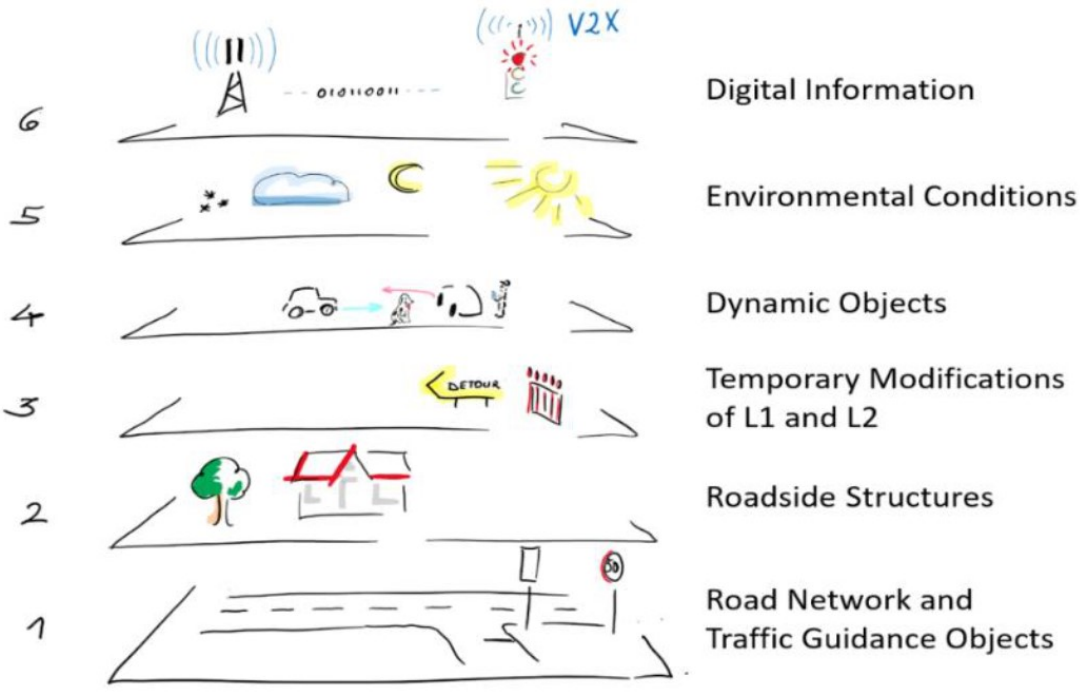


Figure 2.2: 6-Layer Model for a Structured Description of Urban Traffic [4]

As shown in Figure 2.2, the 6 Layers are defined as follows:

Layer 1 (Road Network and Traffic Guidance Objects): This layer contains all elements required for the navigation and direction of road users' traffic, including lanes, road signs, traffic signals, and other crucial infrastructure for traffic direction.

Layer 2 (Roadside Structures): The layer refers to stationary objects that can affect vehicle perception and includes trees, buildings, poles, and barriers. These elements can pose challenges to the visibility and sensor readings of automated driving systems.

Layer 3 (Temporary Modifications of Layers 1 and 2): This layer shows the temporary changes in the road network or in the roadside, for example construction sites, or temporary traffic cones or detour signs. Such changes may cause a change in the traffic flow and such needs to be taken into account by automated systems.

Layer 4 (Dynamic Objects): In this layer, the movement of elements within the traffic environment, including vehicles, bicycles and pedestrians and their paths and trajectories are considered. It describes the behaviour and interactions of these dynamic entities.

Layer 5 (Environmental Conditions): This layer is due to natural conditions such as weather (rain, fog, snow), lighting conditions (day and night) and other environmental factors that can influence both driving and sensor performance.

Layer 6 (Digital Information): This layer includes all digital communication and information exchange within the traffic system, including vehicle to vehicle (V2V) communication, vehicle to infrastructure (V2I) communication and other vehicle to everything (V2X) communication, as well as digital signals of traffic lights, etc.

2.3.3 Definitions and Classifications of Scenarios

The VVM project [4] is used to systematically evaluate the performance of automated driving systems in the following sections, which are definitions of different types of driving. These are regular driving operations, emergency maneuvers, scenarios in which the system interacts with other vehicles, pedestrians, and road infrastructure, as well as scenarios that are designed to assess the system's performance in response to unexpected events and changes in the environment. These are the types of scenarios that help in the development of a complete testing framework that makes sure that ADS are able to operate safely and efficiently within their ODD.

Replay to Simulation: The Replay to Simulation (RtS) approach uses historical real-world driving data to reconstruct scenarios without extensive modifications. The method provides precise replication of actual driving environments because it applies data from genuine traffic settings. The high fidelity of these scenarios establishes RtS as a valuable validation tool to check automated vehicle behavior in realistic conditions. During an RtS scenario, original sequences of events and environmental conditions, together with behaviors of other road users, stay in their originally recorded formats. The method guarantees that the scenario maintains its high resemblance to actual events by simulating traffic system complexities alongside driver behaviours and environmental interactions. The effectiveness of the RtS approach depends strongly on both the quality and extent of the captured recorded data. The RtS approach has an essential limitation that stems from its fixed nature. The scenario's predefined nature, based on recorded data, creates problems because any different behavior from the ego vehicle produces issues. For instance, if the ego vehicle responds differently to the recorded vehicle from a specific acceleration move to a deceleration action, this results in inconsistent behavior that might create unrealistic or even invalid interactions within the scenario. The method's effectiveness for evaluating automated vehicle responses to real-time traffic variations or unexpected changes becomes limited due to these deviations.

Advanced Replay to Simulation: The Advanced Replay to Simulation (ARtS) framework extends the basic Replay to Simulation (RtS) method through adaptive behaviors which improve simulation realism and practicality. The basic Replay to Simulation system depends entirely on replaying historical data records but ARtS implements adaptive strategies which simulate how the ego vehicle and other road users behave hence generating a dynamic testing environment. At the core of ARtS the recorded data establishes the scenario but the system provides functionality which surpasses simple replication. It examines the connections between vehicle behaviours along with their modify potential in relation to other traffic members. Through this method, real-time deviations in the simulation receive accurate responses which produce realistic and valuable evaluations of automated driving systems. ARtS adjusts other vehicles'

responses and road users' behaviour when the ego vehicle produces choices that diverge from original recorded movements while preserving scenario validity and coherence. ARtS improves scenario capabilities to accurately represent actual driving conditions through its integration of driver behaviour models and dynamic event synchronization which goes beyond traditional RtS capabilities. This adaptive aspect is particularly valuable for testing the automated system's response to unexpected system changes and interactions that were not present in the initial recording. It serves as a Scenario configuration to assess how the vehicle preserves safety while rendering proper decisions whenever alterations occur in traffic due to its actions or external factors. ARtS incorporates adaptive behaviour models to overcome several restrictions which are present in RtS. ARtS demonstrates superior flexibility compared to RtS when the ego vehicle's behaviour differs from recorded data because it accommodates these deviations thus minimizing the chance of unrealistic or invalid scenario results. ARtS shows superior effectiveness and comprehensive coverage as a scenario-based testing solution because its capability closely replicates genuine dynamic elements found in real-world driving environments. The simulation gains greater flexibility through Advanced Replay to Simulation because it adapts to variations in vehicle conduct along with their mutual interactions. The simulated environment becomes more authentic which enables automated driving systems to receive evaluations that resemble actual road conditions. ARtS functions as an essential component of scenario-based testing methods which make automated vehicles safer and more reliable.

2.3.4 Base Scenario:

According to the VVM project [4], Base scenario is the scenario that describes typical and important traffic conditions which can be used for further expansion of other scenario tests. These scenarios are developed from basic driving maneuvers that are often seen in typical traffic conditions, including halting at a red light, maintaining a following distance from the vehicle in front, or giving way to a pedestrian at a zebra crossing. Each base scenario simplifies real-world driving conditions and represents them using a set of predefined parameters which makes it easier to work with automated driving systems. Base scenarios are limited to a particular driving task or interaction. They focus on the specific aspects of driving behaviour or the interaction with other road users, which enables the evaluation of the response of the automated vehicle to certain conditions. For example, the base scenario of stopping at a red light would mainly assess the ego vehicle's performance on whether it can detect the traffic signal, slow down, and stop at the right point.

The same approach that requires following a preceding vehicle would evaluate the vehicle's ability to keep a safe distance from the leading vehicle while modifying speed according to the leading vehicle's actions and respond to alterations in traffic dynamics. Because base scenarios offer a controlled testing environment that can be both easily duplicated and modified they have the advantage of being utilized. Base scenarios establish a standardized framework to evaluate automated driving functions by defining parameters such as vehicle speed, traffic light timing, pedestrian crossing time and distance to other vehicles. Due to the fact that these scenarios can be performed multiple times with identical parameters the results obtained are precise and enable comparison of results between different test runs. The initial phases of automated driving system testing depend heavily on base scenarios because they concentrate on particular defined driving tasks. They give an initial assessment of the core capabilities of the system, including its ability for object detection, signal recognition, and fundamental control. These scenarios are indispensable to confirm that the automated vehicle can perform routine driving

tasks safely and properly before moving on to more complicated and less routine scenarios. Scenario-based testing frameworks depend fundamentally on base scenarios as an integral part of their structure. They provide a systematic approach to testing automated driving systems in standard traffic situations to confirm that fundamental operations are reliable and effective. Further developments on these base scenarios and their integration into broader testing strategies will be presented in the following sections.

2.4 Data Sources for Scenarios

As a prerequisite for the scenario generation, which will be discussed in the chapter methodology 4, as the real-world recorded data is utilized, these data will be used as input for the methodology, in which the scenarios are included which will be discussed in scenario data section in more detail, in addition to that, the layout of the road network where the scenario takes place will be mentioned in InD maps section.

2.4.1 Scenario Data

Traffic or real-world driving data is recorded to create reference data for testing automated driving systems. These recordings provide the foundational information, like the trajectory data, needed to extract specific driving scenarios, accurately reflecting real-world conditions. To handle data from various sources and formats, interfaces are introduced, such as object-based trajectories formatted in the OMEGA standard [13]. This file format contains a scenario called OpenSCENARIO, which illustrates the interaction between vehicles, pedestrians, and other dynamic objects. This data includes the vehicles' trajectories, actor behavior, and others, which focus on dynamic events and their actions and control where, when, and how an actor moves and interacts based on the scenario configuration. Automated algorithms are employed to analyze the raw data, identify relevant scenarios, and compute the necessary parameters for these scenarios, ensuring that key aspects of real-world traffic behavior are captured. These data are mostly used in OpenSCENARIO-based simulation environments like CARLA, in our case, [5] for matters of dynamic behavior and interactions. This format allows for a consistent representation of vehicle movements and interactions. As for the methodology of this report, the Dynamic behavior of the road users from layer four will be used for the methodology.

2.4.2 InD Maps

InD maps define the road network or maps that show the layout of the roads, lanes, and junctions and describe the physical infrastructure of the environment, including road geometry, lane markings, road curvature, elevation, and lane connectivity. The InD dataset, OpenDRIVE data, is used to define detailed maps for the design of intersections and urban streets in simulations. As part of this work, which focuses on the scenarios that take place in intersections, InD maps have been used. The dataset's utility is further enhanced by its ability to represent diverse traffic situations across different types of intersections, making it a robust foundation for scenario-based testing. Below is an example of an inD map, taken from The inD Dataset [3]. Also, the analysis of the recorded traffic data from four different intersections, as shown below, is used to consider diverse data.

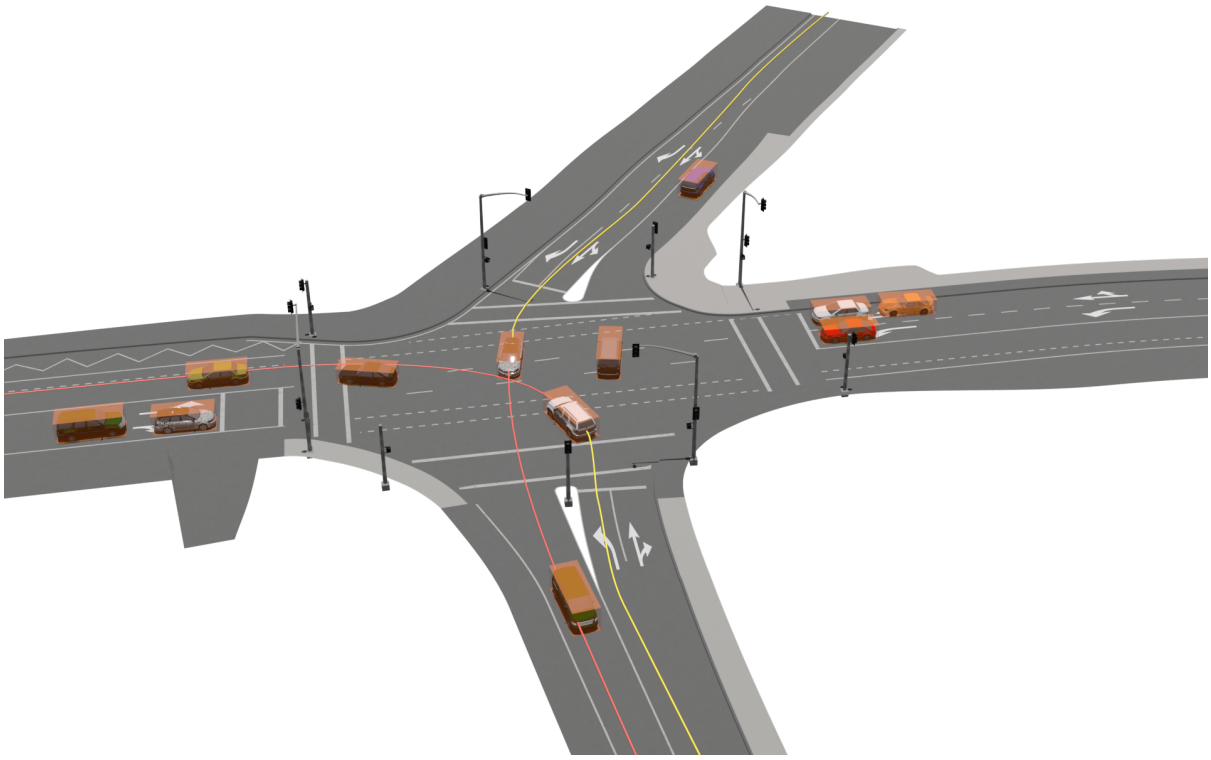
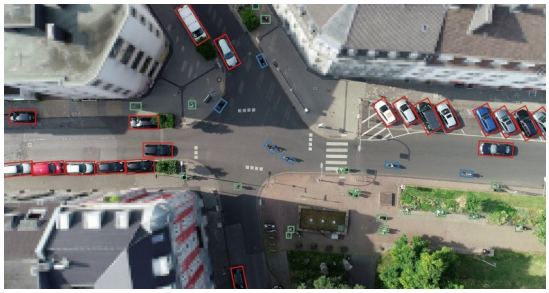
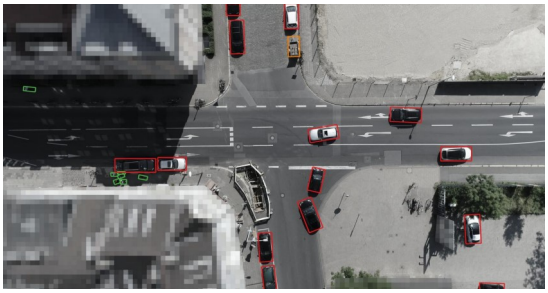


Figure 2.3: An example of Intersection map from inD Maps



(a) Frankenburg, Aachen, Germany



(b) Bendplatz, Aachen, Germany

(c) heckstraße, Aachen, Germany

(d) Neukollner, Aachen, Germany

Figure 2.4: Intersections from Aachen City in Germany, InD Maps [3]

The scenario data used in this study consists of real-world recordings captured from various intersections in Aachen, Germany, provided by the InD dataset [2]. This dataset offers raw traffic data that can be processed to extract, manage, and generate traffic scenarios for simulation and analysis [16]. The four intersections included are distinctly different from one another; there are no intersections identical in design or traffic rules.

In Figure 2.4a, it can be observed that at a three-way intersection, the vehicles on the main road have the right of way, and the vehicles entering from the side road can be seen slowing down or stopping at the yield or stop sign. Side road vehicles have a speed range of 10-25 km/h based on signs and traffic density, while side road vehicles have a range of 30-60 km/h. In regard to the main road, drivers have lesser expectations of side road traffic, and hence, there is less probability of interference with their flow. The side road traffic makes fewer decisions but has to check for oncoming traffic before turning right or left to the main road. This being the case, pedestrians are more regulated and tend to use designated crosswalks on the side road, thus reducing conflict with traffic.

The image which is labelled as Figure 2.4b shows a complex four-way intersection of an urban area with several buildings around it. Some of the vehicles are halter or in motion near the pedestrian crossings. The presence of pedestrian zones and nearby buildings indicate that it is a dense built-up area with several lanes for vehicles. It is a intersection of two roads meeting at right angle to each other forming four corners. Drivers are more careful to some extent especially if there are things like stoplights or signs. The typical speeds at which vehicles approach the intersection are 10 to 30 km/h during peak hours, but at uncontrolled intersections or during low traffic, vehicles may approach at 40 to 50 km/h but will reduce speed within the intersection. In regard to the social aspect, there is more communication between vehicles from all directions, and this is especially true for unsignalized intersections. Turning left is even more difficult and hazardous due to the possibility of colliding with oncoming or cross traffic. Such intersections present problems for pedestrians who have numerous crossing points, only to find that they may come into conflict with turning vehicles, and this is particularly so where there are no marked crosswalks.

Figures 2.4c and 2.4d depict the Heckstraße and Neuköller intersections, respectively. The former is characterized by lower urban density, fewer buildings, and greater open space, while the latter is a more complex intersection that includes various lanes for different directions, as well as potential public transportation integration.

The VVM project describes an enveloping scenario in further detail through scenario 235414. A traffic scenario occurs within spatially and temporally confined boundaries known as an enveloping scenario which focuses primarily on the “ego vehicle”. The framework contains logically connected infrastructure elements, including intersections and roundabouts, which maintain these features as whole units instead of creating artificial fragmentation. The scenario starts when the ego vehicle first moves through this bounded space and ends when it exits, demonstrating the full contextual movement of the vehicle throughout this boundary.

Figure 2.5 illustrates how enveloping scenarios are created by dividing extended traffic data into logical, spatial, and temporal segments. Segmenting scenarios in this manner allows longer journeys to be divided into manageable sections for analysis while continuity in vehicle behavior patterns is maintained. By structuring these scenarios, researchers can perform systematic clustering and calculate scenario distributions more effectively, enabling the generation and testing of traffic scenarios.

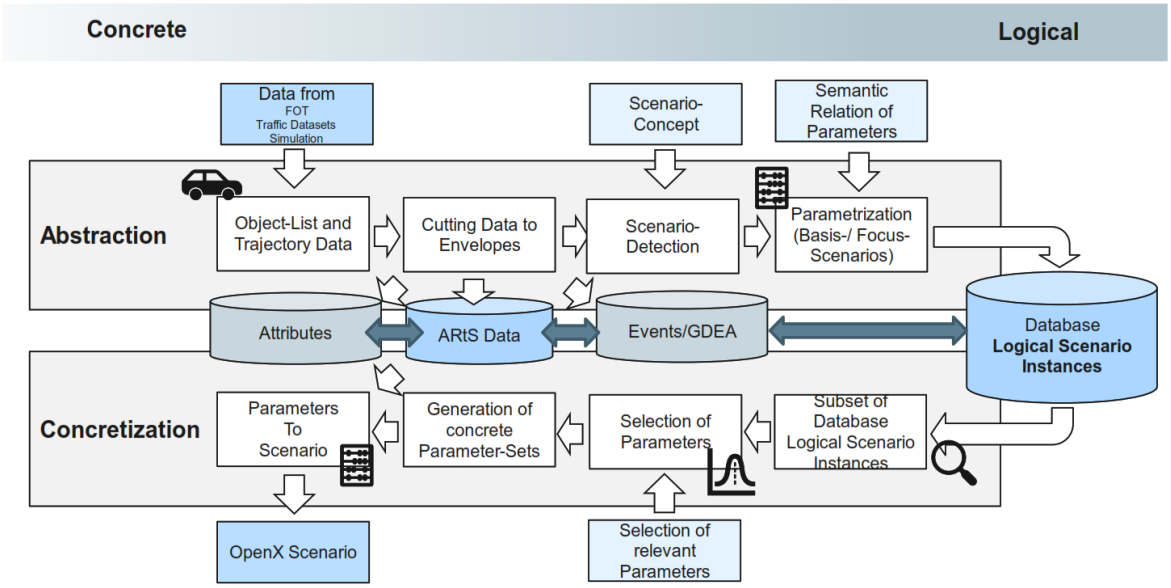


Figure 2.5: Flow Chart of From Data to Scenario

2.5 Similarity of the Scenarios and Classification Fields

In the development of a framework for similarity analysis of Urban Driving Scenarios [10], It is important to determine the similarity of different driving scenarios for the assessment and verification of automated driving systems. Scenario similarity can be defined as the closeness of different driving situations in terms of certain parameters or events such as vehicle trajectories, interactions with other road users and environmental conditions. This is important in scenario-based testing, where it is important to know how to classify and group similar scenarios and how to make distinctions for dissimilar scenarios. This helps in achieving thorough testing coverage by concentrating on the different and important driving situations. For instance, two scenarios may be said to be similar if the vehicles in the two scenarios are moving in the same direction or if the vehicles are responding to the same traffic signal in the same manner. On the other hand, scenarios that have different paths, interactions, or outcomes should be classified as dissimilar. Thus, the identification of similarities can help researchers to improve the testing process, decrease the redundancy and concentrate on the evaluation of the automated driving system's performance in critical and diverse situations. Also, the effective classification of scenarios according to their similarity is helpful for scenario prioritization, which means that targeted testing can be used to reveal potential weaknesses or areas for improvement in AD systems.

Types of Similarity:

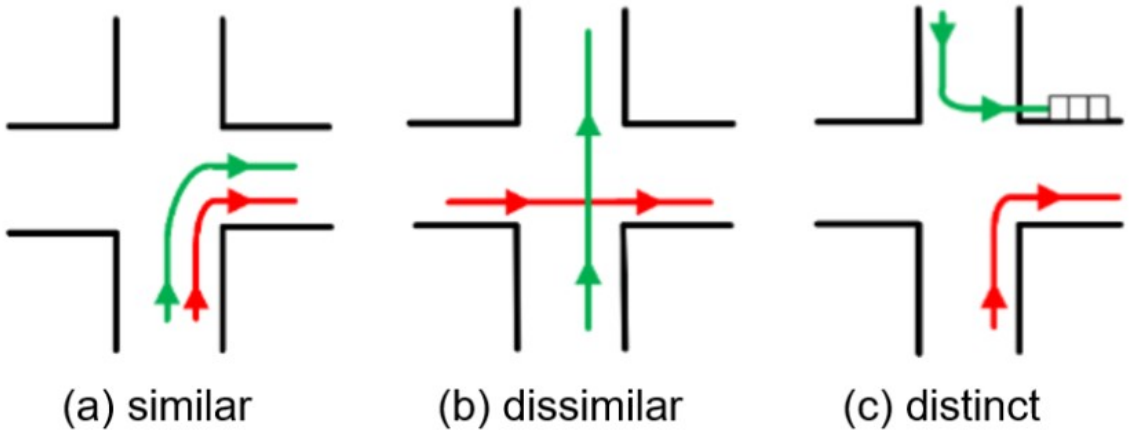


Figure 2.6: Types of similarity [10]

Similar Scenarios: In this classification, Figure 2.6, scenarios are mostly alike but not identical. They share common elements, such as road user behaviors or vehicle trajectories, allowing for meaningful interactions when combined. For example, integrating road users from one similar scenario into another can create realistic testing conditions where the interactions between elements make sense and reflect potential real-world situations.

Dissimilar Scenarios: These scenarios lack similarity and do not share significant common elements. They differ in key aspects, such as trajectories, vehicle behaviors, or environmental conditions, making it impractical to combine them meaningfully for testing purposes.

Distinct Scenarios: These are completely different and independent scenarios. Although it is possible from a technical point of view to join elements of the road from different scenarios, it would not be useful or realistic to do so, because the interactions would be meaningless. This has already been mentioned: the idea of scenario similarity is important for the first stage of combining scenarios. This is because, when different driving scenarios are compared with respect to their trajectories, and then merged, it is possible to merge scenarios effectively.

2.6 Metrics for Scenario Assessment

The development of automated driving systems relies heavily on scenario validation because it ensures that tested scenarios provide thorough, realistic, and effective evaluation of system performance. Through the validation of scenarios, it can be shown that they can sufficiently mimic actual world scenarios and their interactions thus providing a solid foundation for testing the automated system's capabilities. Achieving this requires examining the validity of the scenarios and the behaviour of the road users. Quantitative measures for assessing the performance of an automated driving system, specifically the ego vehicle within a given scenario, are provided by metrics which play a key role in this process. Metrics enable evaluation of different system response aspects including safety, efficiency, comfort and traffic rule compliance. The following sections present some of the most important metrics used in scenario validation for automated driving systems and how they are used to assess their performance.

Time to Collision (TTC): As discussed in Criticality Metrics for Automated Driving [12], TTC is a valuable metric for scenarios where one vehicle is following another. It measures the time remaining before a collision occurs if both vehicles continue on their current trajectories without any changes in speed or direction. TTC is highly sensitive in simple car-following scenarios, making it a crucial metric for assessing the risk of rear-end collisions. However, its sensitivity decreases in intersection scenarios, where the paths of the vehicles are more varied. TTC gives indications about collision risks during driving operations.

Post Encroachment Time (PET): PET [12] is particularly suitable for intersection and crossing scenarios. It provides a measure of the temporal safety margin in urban driving environments, especially at intersections. PET calculates the time gap between one vehicle leaving a conflict area and another vehicle entering that same area. This metric helps evaluate the safety of crossing paths and is useful for assessing the situational risk in complex traffic scenarios where vehicles and pedestrians frequently interact.

2.7 Controller

In this section, the controller used in my thesis will be applied to change the behavior of the ego vehicle and consequently analyze the behavior of other road users. The controllers are defined in the OpenSCENARIO file and are essential for defining and managing the behavior of ego vehicles within a scenario. Controllers interpret and execute the actions specified in the OpenSCENARIO file, ensuring that the ego vehicle responds appropriately to various events and triggers. They interpret and follow the actions defined in the OpenSCENARIO file, which means that the ego vehicle's actions are in line with the scenario specifications and the ego vehicle does not respond incorrectly to scenario events and triggers, thus increasing the realism and control of vehicle behavior. OpenSCENARIO provides several types of controllers for ego vehicles which provides flexibility and flexibility in behavior control. If no controller is defined for the vehicle, then a default controller is set to the vehicle. This controller adheres strictly to the OpenSCENARIO actions; it extends the standard behaviours of KeepVelocity and KeepLateralOffset. These actions represent basic, standardized vehicle control but can be extended to perform more sophisticated behaviors as required by the scenario. For more complex or specialized behavior, an example such as "simple vehicle control" offers enhanced control capabilities. Advanced controllers allow for the implementation of specific autonomous driving behaviors, providing greater flexibility in scenario design and the ability to test various control strategies within the same scenario. Controllers offer scenario design flexibility by allowing switching between different control strategies, enabling the testing of various autonomous driving algorithms or behavioral models within a single scenario. They can be coupled with control systems or driver models, so that they can be used in conjunction with simulation environments without a problem. This integration also allows for Hardware-in-the-loop (HIL) and multiple simulation platforms for testing. Controllers thus play an important role in OpenSCENARIO by providing the flexibility that is required for the evaluation of a comprehensive set of autonomous driving behaviors and making sure that the ego vehicle acts in a manner that is predictable and realistic in the given simulated conditions. List 2.1 describes the location of the controller in the OpenSCENARIO file.

Listing 2.1: Controller Properties

```
<ScenarioObject name="Ego">
    <CatalogReference catalogName="VehicleCatalog" entryName="
        car_mini_cooper"/>
    <ObjectController>
        <Controller name="Ego">
            <Properties>
                <Property name="AgentProfile" value="MiddleClassCarAgent"/>
            </Properties>
        </Controller>
    </ObjectController>
</ScenarioObject>
```

The following two controllers will be explained in detail, including how they function. They were developed at IKA, Aachen, Germany, and will be used for the Ego vehicle.

Approaching Controller: This vehicle controller is engineered to autonomously navigate a vehicle along a predefined path, prioritizing safety by braking if another vehicle is detected ahead. Leveraging ground truth data instead of real-time sensor input, the controller continuously adjusts both speed and direction to follow a designated sequence of waypoints. To maintain safety, it actively monitors critical safety metrics such as Time-to-Collision (TTC) and Time Headway (THW) to assess collision risk. Should these metrics fall below-specified thresholds, the controller initiates braking to avoid a collision. The controller's configuration enables the specification of parameters like target speed, lookahead distance and safety thresholds to match the conditions of the traffic. This flexibility is consequently utilised to support safe and efficient navigation, enabling the vehicle to respond to the presence of other road users dynamically and to keep safe distances, thus improving overall traffic safety.

Intersection Controllers: The intersection assistant controller is intended to enhance safety and to improve the smooth flow of traffic through intersections. It incorporates several core functions: It is an AEB system that can halt the vehicle when there is a threat of collision, as well as gentle driving modes that control acceleration and braking on intersection by intersection basis. It derives advanced safety metrics like Time-to-Collision (TTC), Distance Headway (DHW) and Predicted Post Encroachment Time (PPET) to assess traffic dynamics and respond to it accordingly. Based on the traffic rules such as "right-before-left," the controller determines the positions and movements of other road users and whether the vehicle should stop, slow down or proceed through the intersection. It controls speed through a sequence of predefined waypoints to dynamically adjust to meet traffic compliance aims and minimize collision risks. In hooked mode, from ground truth data from a simulated environment, the controller directly manages the vehicle's speed and direction and thus provides precise, rule-based navigation and safe manoeuvring within intersection zones.

2.8 CARLA

This section shows two different modes of syncing in CARLA: Synchronous and Asynchronous. This will be studied in chapter methodology 4 as the syncing time between the Ego and the elements in CARLA affects the performance accuracy and the scenario-based testing.

Synchronous Mode: In synchronous mode, the simulation advances in discrete steps or "ticks." Each tick represents a fixed update to the environment, and the next tick does not commence until all processes, vehicle updates, sensor readings, and environmental changes are fully processed. This mode ensures strict control over time progression. Its benefits are time control, which provides deterministic behavior, making it ideal for testing scenarios where precise time management is essential, such as sensor fusion, perception testing, or when replaying recorded scenarios, and reproducibility, which since the cycling time is fixed, synchronous mode, enables experiments to be precisely reproduced. Are only advanced one frame of simulation procedure is initiated once all vehicles and pedestrians are in update state because the simulation does not miss any interactions based on the time constraints. In regards to the drawbacks, performance constraints can increase the computational requirement of the simulator, The simulator has to wait for all sensors and actors to finish their updates, which can decrease the simulation speed, especially in the case of complex scenarios, and a fixed rate of update which can be limiting as each component has to wait for the slowest process in the cycle, which may well increase the cycling time. In CARLA, cycling time in synchronous mode is usually consistent, but it depends heavily on the computational resources available and the complexity of the scenario. For instance, scenarios with multiple sensors, high-resolution images, or numerous actors will have longer cycling times because every component's data must be processed each cycle before advancing.

Asynchronous Mode: In asynchronous mode the simulation cycles operate autonomously without regard for restricted time periods. Each sensor and actor gets updated as soon as it becomes ready to run without forcing other entities to finish their computations. Time moves straight ahead without stopping to let each actor finish its actions. The advantage it provides is performance which becomes more effective for situations that do not demand precise time management. It allows the simulator to operate at increased speed which makes it suitable for real-time operations or information system processing tasks that require fast data analysis. The lack of synchronization barriers between updates allows individual actors to process data without affecting cycling time which results in faster processing and throughput. However, there are some drawbacks to this approach: Non-deterministic behaviour: Actors work at different times independently of one another. The outcome of the simulation can differ from one run to another as a result of this approach which makes it harder to capture exact results. For example, in autonomous vehicle systems regression testing becomes problematic due to inconsistent outcomes. Potential skipping of Interactions: High-priority processes could potentially skip' or overlook interactions like collision events or sensor readings based on timing differences especially when the CPU or GPU load is high. The asynchronous cycling times produce vastly different results because they avoid constant time intervals. The slowest actor or sensor normally determines the simulation speed, but since there is no strict waiting mechanism, the overall cycle times are typically faster than in synchronous mode, where everything runs at once.

In automated driving, synchronous mode is often preferred for tasks like perception and decision-making, as precise control over the scenario is essential. For example, when testing an autonomous vehicle's response to a pedestrian suddenly crossing the street, it is critical to ensure that the simulation progresses in tightly controlled time steps so that the exact position and behavior of the pedestrian and vehicle are captured for every cycle. Later, the effects of both modes will be taken into consideration, including how they can affect the scenario performance.

Chapter 3

Research Objectives

Introduction: This chapter presents the objectives of the research. It begins by outlining a set of requirements for scenario combination and provides a brief comparison of different scenario generation methodologies. Subsequently, the main research questions are introduced, each linked to these requirements. The approach to addressing these questions is also discussed.

3.1 Research Gap

Recent advancements in the automotive industry require testing automated driving systems to make sure they function correctly based on the scenario they face. Current methods for testing the functionality of automated driving systems include real-world testing and track-based testing. While real-world testing can provide valuable insights, it is often inefficient due to its time-consuming nature and the need to collect extensive amounts of data over long periods [11] [9]. Fin Malte Heuer at Scenario Generation for Testing of Automated Driving Functions based on Real Data [8], introduced scenario generation for testing autonomous driving systems, which involves using real-world traffic data to simulate realistic driving behaviors in controlled environments. Parameters are derived from expert knowledge, optimization, or clustering-based methods, ensuring scenarios closely resemble real-world interactions. This approach enables efficient, scalable testing of driving functions under various conditions. Yao Deng discussed a robust, automated framework for generating and testing driving scenarios based on traffic rules [17]. It improves the efficiency of testing autonomous vehicles by identifying critical issues early in the development process. As mentioned from the [8] and [17], for the first one it is indicated that parameterization of the simulation is expert-based and the system may struggle to capture rare but critical driving scenarios that are infrequent in real-world data, and regarding

the [17] which is based on the traffic rules and in reality there might be scenarios that do not follow trajectory rule which these trajectory rules govern the movement and interactions of vehicles by defining their starting positions, connecting lanes, and endpoints through waypoints and mapped routes. Trajectories are derived based on road network configurations (e.g., intersections, roundabouts), vehicle behaviors (e.g., turning, lane-changing), and compliance with traffic rules such as yielding or stopping. Moreover, deep learning techniques address perception, decision-making, and motion planning for autonomous vehicles, whereas, for challenges, ML methods have a dependency on large datasets, poor generalization to rare events, and lack of interpretability, making validation difficult for safety-critical scenarios [6]. As the findings are interesting, there is a need to have complex scenarios where there are limited amounts of data available to test, which, as a part of this work, complex and rare scenarios can be generated based on real data.

3.2 Research Questions and Approach

Based on what has been discussed in the research gap section, to answer the gap, a certain number of questions must be answered. The following aims to answer the main important questions:

Q1: What approaches can generate complex scenarios, and how might merging scenarios enhance the process??
Q2: Are the generated scenarios robust and resemble the real-world traffic scenario?
Q3: To what extent is the scenario flexible to the behavior of the ego test vehicle?

For the matter of generation complex scenarios, there is a need to define a methodology that defines the process of generating scenarios from which data and how they are processed to approach a complex scenario. For this, There are input data, which are recordings of the real-world traffic, and then based on these data, the scenarios can be found. Later, these scenarios taken from recording go through the similarity concept [10], meaning that based on the available recorded data from the traffic, which is our input data source, two similar scenarios will be found based on the trajectory of the relevant ego vehicle and the object that they are following. The similarity is important here because the aim is to replace the ego with a different scenario but not a completely different ego trajectory, as the ego vehicle must be adapted to the traffic intersection and the trajectory it follows. Therefore, there would be two similar scenarios in terms of ego-object behavior but different behavior. Then these two scenarios will be merged together, as will be discussed in the methodology chapter 4, in which, in principle, the generated scenario has elements of ego-object from one scenario but different other RUs. For the matter of robustness, the methodology will try to change the behavior of the ego vehicle by assigning the controller and based on how the ego vehicle behaves and follows the generated trajectory based on the controller output, how other RUs react, and what extent they can show flexibility in terms of stopping, deviate from the trajectory and other factors. So, in short, compared to reality, if one vehicle changes its behavior, the other vehicle will respond and change its behavior. As a part of the analysis, I look forward to this situation by analyzing how many accidents happened, which can be a good tool to evaluate the robustness of the scenario and its reality.

In this research, the below tools and data will be used:

- InD maps
- OMEGA File
- ASE Engine
- CARLA
- Python
- Data Analysis tools
- Actor Controllers

The real-world scenarios will be taken from OMEGA[1] files, which contain detailed information about the traffic scenarios. The ASE engine[2] will be used as a required tool to modify the data and generate new, complex scenarios. These scenarios will then be simulated in the CARLA[3] environment, with Actor Controllers, used to control ego vehicle behavior and its effects on the overall traffic scenario performance. Python will facilitate the entire workflow, from data processing to scenario generation and analysis.

[1]The OMEGA-Format is an HDF5-based data-format designed to store reference data in an object-list-based structure together with map and weather information. A unified data format that includes information on dynamic objects, map data, and weather information tailored to the needs of the VVM-Project

[2]A powerful tool used to modify recorded data, focusing primarily on layer 4 of scenario modification, which involves altering the behavior of road users. The ASE engine will also be used to generate OpenSCENARIO files for simulation in the CARLA environment.

[3]An open-source simulator designed for testing autonomous driving functions. CARLA will be used to simulate the generated scenarios and observe how vehicles and other actors behave under varying conditions.

Chapter 4

Methodology

Introduction: This chapter will provide an in-depth explanation of the methodology that will be utilized in this study, detailing each step within the workflow that will be undertaken to achieve the research objectives 3 being how to generate complex scenarios and how they are robust to real-world scenarios. The workflow will encompass the framework design, selection of input data, approach to data analysis, and the overall implementation strategy. Each of these components is integral to the methodology, ensuring a structured and practical approach to the research. Additionally, this chapter will provide a comprehensive explanation of each section within the framework design, clarifying how each part contributes to the overall process and supports the study's goals, where in the end, it ends with a method that provides a complex scenario that also shows robustness similar to real-world data.

4.1 Framework Design

In this section, the framework of the methodological procedure will be concisely discussed. As illustrated in the figure 4.1, the process begins with the careful selection of input data, a step that will be elaborated on in subsequent sections in which data shows the scenarios. Based on this input data, the similarity between the chosen scenarios will be assessed, providing a foundation for enriching scenarios by merging and generating complex scenarios. Following this, a controller will be applied to the Ego vehicle, taking parameter variations into account as a method to observe its impact on the scenario. Simulating this configured scenario will yield valuable data, such as trajectories. This data will then be analyzed using dedicated tools developed at IKA, Aachen, Germany, to evaluate and validate the generated scenarios effectively.

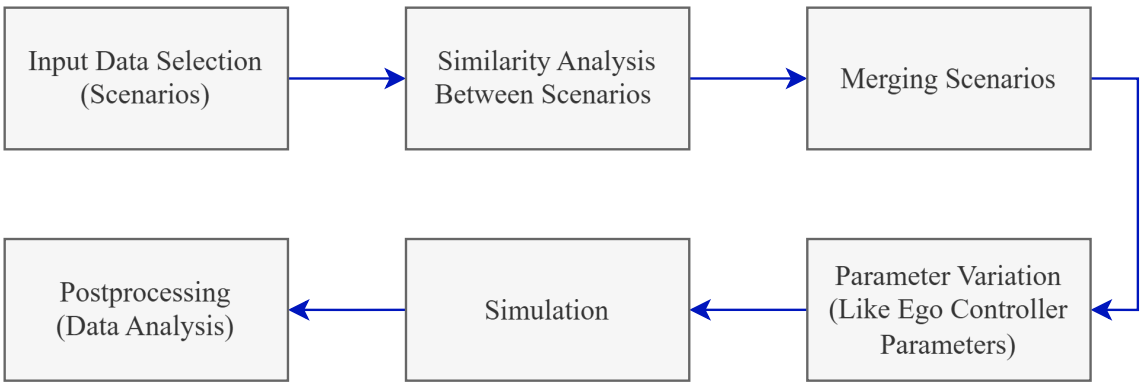


Figure 4.1: Methodology Framework

4.2 Input Data Selection

As discussed in Chapter 2, scenario data serve as the input data for our methodology. For the purposes of this study, four different intersections were considered, each with multiple recordings that represent data captured at varying times of the day and on different dates. The rationale for having several recordings is to capture a variety of traffic behaviors, as there are different road users whose trajectories, positions, and speeds vary over time. These variations lead to distinct scenario dynamics at different times and dates, influenced by the behavior of road users, though not necessarily all road users in each scenario.

In my analysis, the variation in traffic parameters, such as vehicle speed, sudden braking or acceleration events, and the trajectories of road users, is crucial. These parameters collectively define what can be termed the ”behavior” of road users. Later in this thesis, scenario generation based on these recordings will enable us to assess the impact of additional vehicles within each scenario. Furthermore, variations in the types of road users, including vehicles, bicycles, motorcycles, and pedestrians, introduce additional complexity to the scenarios. This diversity of road users and their behaviors provides a broader understanding of interactions within urban traffic environments.

The next step involves establishing a clear definition of the ”envelope” as it pertains to the recording process. As previously noted in the discussion on why multiple recordings are taken into consideration, Figure 2.5 illustrates that enveloping scenarios are created by dividing extended traffic data into logical, spatial, and temporal segments. These enveloping scenarios constitute the foundational structure for subsequent analyses, where vehicle behaviors are examined and annotated, and essential events and base scenarios are identified. The recorded trajectory data of each vehicle captures and models their movements, allowing these scenarios to be further divided into fundamental segments, or ”base scenarios,” ready for parameterization. Once parameterized, these base scenarios are prepared for simulation, providing a realistic, data-driven environment in which vehicle behaviors can be rigorously tested and validated.

4.3 Similarity Analysis

In this section, the application of similarity analysis in the generation of complex scenarios will be explained. As a need for testing the ego vehicle in a new complex scenario, in this thesis, the placement of the ego vehicle is done by finding a similar Ego vehicle but different behavior in the new scenario, where the new replaced Ego vehicle can challenge the scenario in scenario-based testing, For this aim, the similarity between two scenarios for finding similarity based on the Ego but different scenario is done. As initially outlined in Chapter 2, under the section on similarity. This analysis requires a specific approach to selecting input data, which in this context involves comparing two distinct ”envelopes” derived from the recorded data. A preliminary consideration in this process is determining the criteria for envelope selection. The conditions for choosing these envelopes are essential to ensure meaningful results in the similarity analysis.

The first condition stipulates that the similarity analysis must focus on envelopes within the same intersection. When comparing envelopes, it is essential that they share the same intersection to maintain contextual consistency. This requirement arises from limitations in scaling intersections for different conditions, such as variations in lane width or intersection geometry. The inaccuracy that may result from requiring geometric adaptation of a road user's behaviour from one intersection to another is likely due to the fact that this is not an issue which this thesis is concerned with, as it does not address the issue of the scalability of the approach across intersections. The focus is, therefore, on picking envelopes from recordings that are collected at the same intersection in order to maintain analytical precision.

The second condition is about the potential of having similar or even the same scenarios in a given time period, which can affect the accuracy of the results. In the case of two scenarios, if they are very similar or even the same in terms of vehicle trajectories and behavior, then the similarity analysis may provide duplicate or low-fidelity results. This redundancy arises because similar or identical behaviors from road users can introduce bias or noise in the analysis. Two solutions are proposed to address this issue, which will be discussed in detail in subsequent sections. The first one is based on selecting distinct temporal subsections from a unique envelope, and the second one is based on selecting distinct temporal subsections from a unique intersection, where these solutions aim to refine the process of similarity analysis by reducing redundancy and enhancing the accuracy of the comparative outcomes.

Distinct temporal subsections: As illustrated in Figure 4.2, the recording database contains data from four distinct intersections. Any recording within this scenario database may be used for analysis, provided that during the envelope extraction process, two distinct dataset segments are derived from different subsections of the time frame. These two temporal subsections do not necessarily need to be extracted from the very beginning and end of the recording. Instead, they should be selected from sufficiently different points within the recording to minimize the likelihood of capturing nearly identical envelopes. This ensures that each dataset subset is a unique segment of time, reducing redundancy and enabling more meaningful similarity analysis between envelopes. Using different temporal subsections, we prevent situations in which envelope similarities are so high as to call the reliability of the similarity analysis into question. It also means that every envelope provides different context and behaviour information which can be compared robustly and giving a more accurate base for scenario analysis.

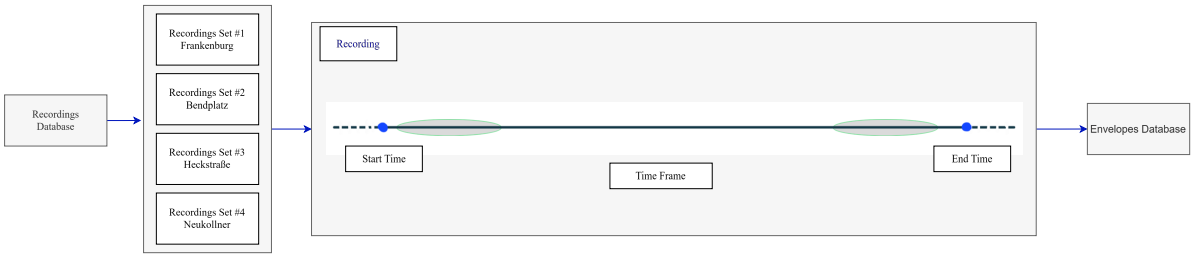


Figure 4.2: Envelope Selection Method 1: Recording Datasets to Envelope Scenarios

Distinct intersection: As depicted in Figure 4.3, we have a database containing recordings from four distinct intersections. To proceed, I select a specific subset of recordings, such as those from the Frankenburg intersection. Within this subset, recordings are available from various dates and times, which reduces the likelihood of encountering identical envelopes. This temporal and contextual diversity across recordings enhances the uniqueness of each envelope, ensuring that each captures distinct traffic patterns and behaviors. By loading these envelopes from the selected recordings, we can establish a dedicated database specifically for the envelopes. This envelope database then serves as the input for similarity analysis, allowing for systematic comparison and study of traffic scenarios under varied conditions without redundancy. This approach maximizes the reliability and relevance of the similarity analysis by ensuring each envelope represents a unique data instance within the broader dataset.

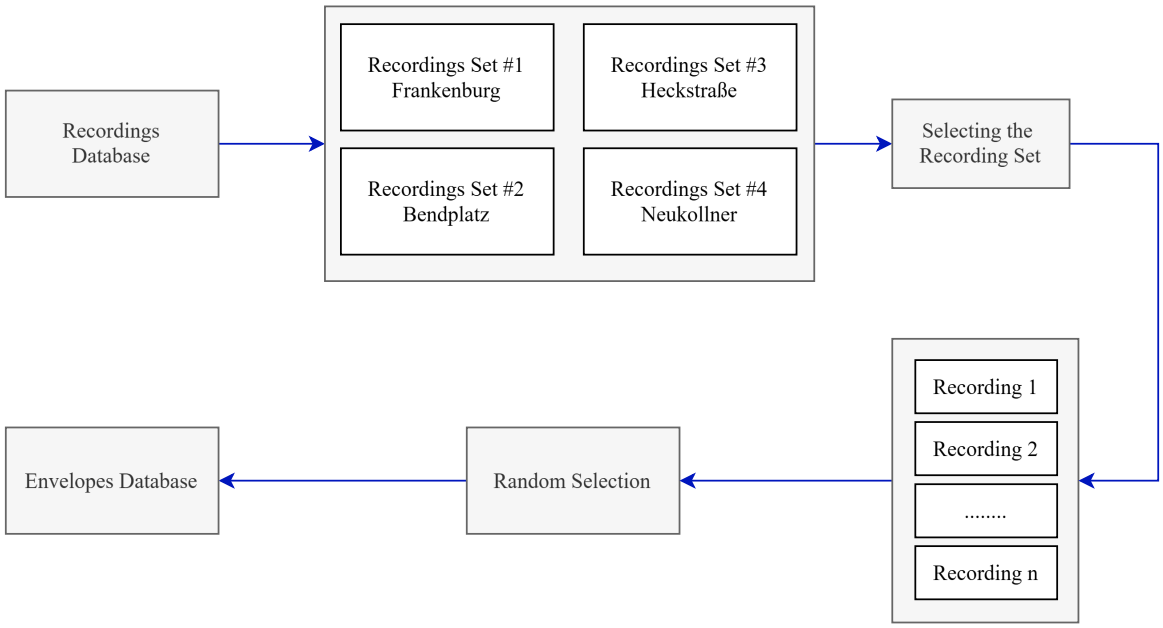


Figure 4.3: Envelope Selection Method 2: Recording Datasets to Envelope Scenarios

Considering the selection method from the recordings based on the First method, the process involves identifying two similar envelopes, as illustrated in Figure 4.4. To clarify, two terms require definition here: the **"From"** envelope and the **"To"** envelope. These envelopes represent the core elements subject to similarity comparison and serve as the foundational basis

for generating merging scenarios. Specifically, they are sets of **”From”** and **”To”** envelopes, from which one is chosen from each set. The process proceeds by examining the relationship between the ”ego” vehicle and other objects within each selected envelope, followed by the extraction of a base scenario. At this stage, a determination is made as to whether these base scenarios are ”similar” by applying a predefined similarity threshold. If the base scenarios meet the similarity criteria, further analysis is conducted to evaluate whether the trajectories of the ego vehicle and object in the **”From”** and **”To”** envelopes exhibit comparable patterns. If both these similarity conditions of base scenarios and trajectory patterns are satisfied, then the two envelopes are considered similar. This similarity is a prerequisite for the merging procedure, ensuring that the selected envelopes have sufficient contextual alignment for scenario integration as there is a similar Ego object base relation with different Ego parameters, where the new Ego vehicle can challenge the new scenario and the scenario itself can be assessed.

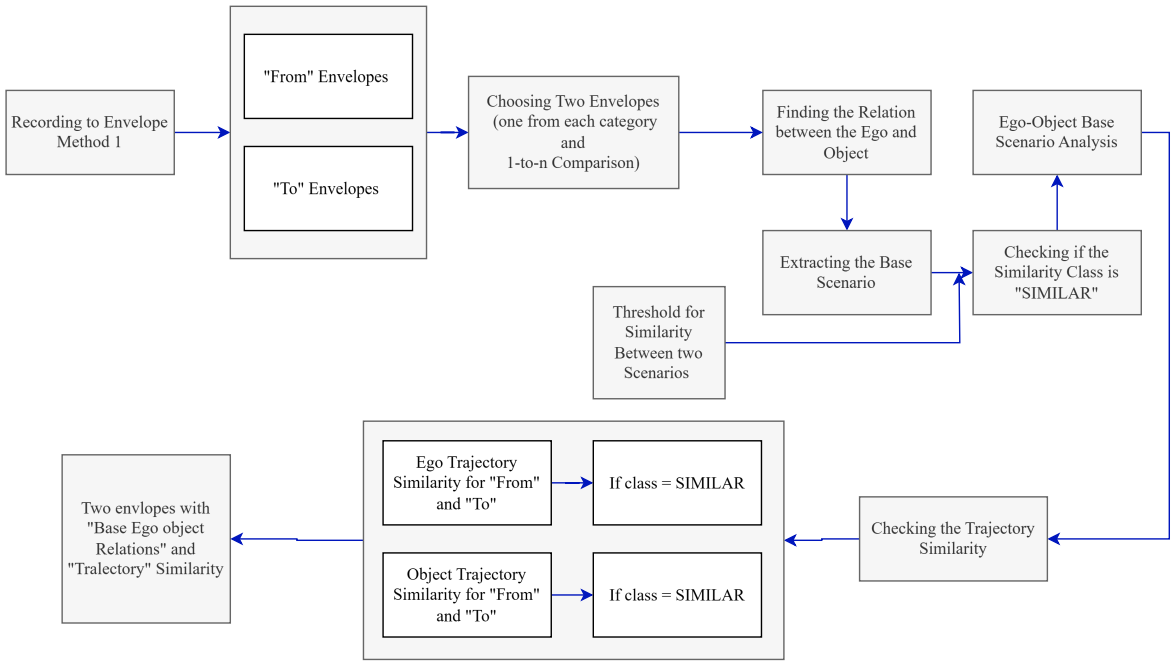
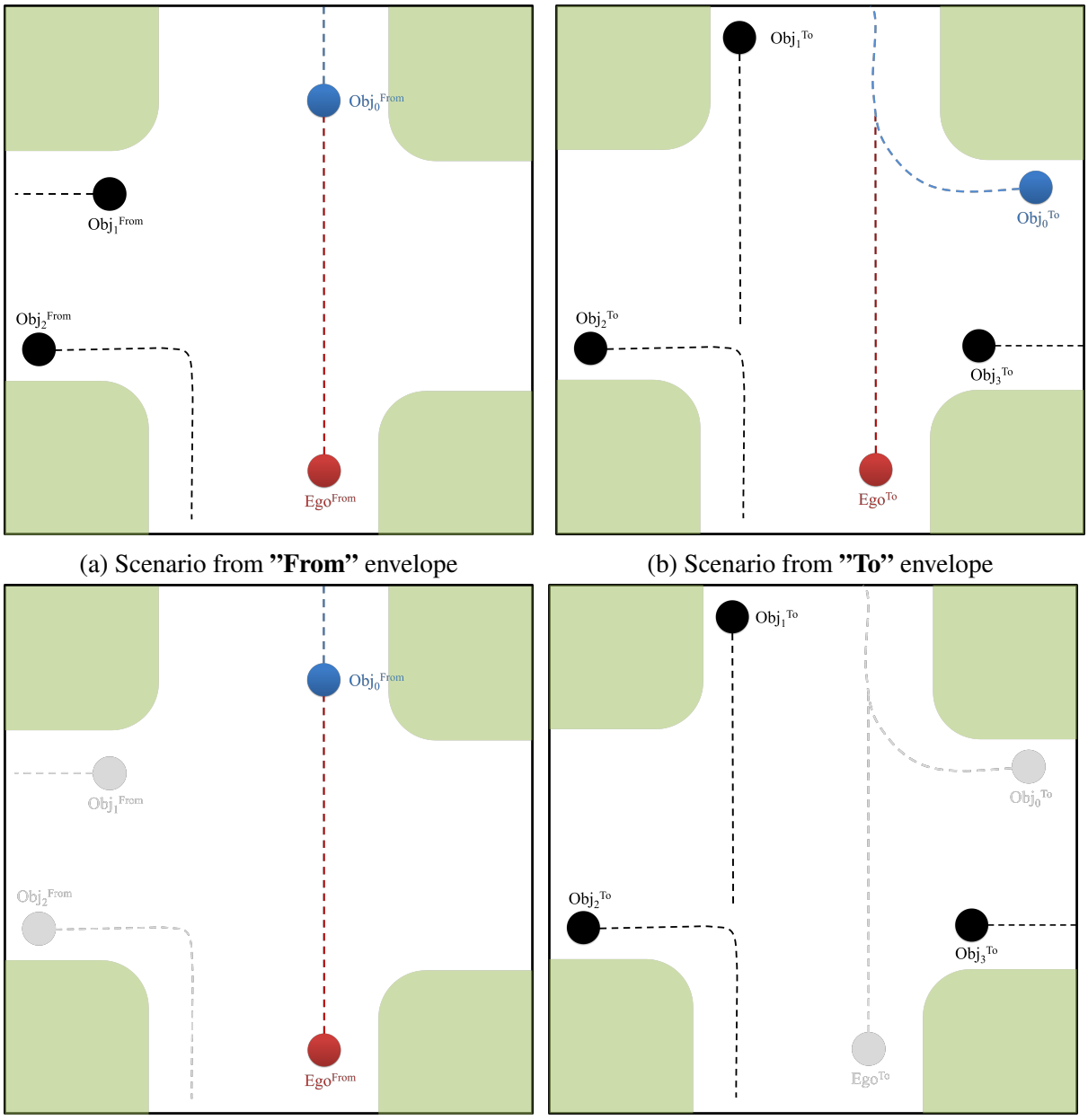


Figure 4.4: Similarity Analysis Procedure

4.4 Scenario Merging

In this section, the framework for scenario merging is examined in figure 4.5. It shows a visual illustration of the steps of how a new complex scenario is created.



(a) Scenario from **"From"** envelope

(b) Scenario from **"To"** envelope

(c) Ego Object from **"From"** envelope

(d) Ego Object elimination from **"To"** envelope

Figure 4.5: Illustration of Cross Intersection Scenario Of the Similar Scenarios

Before proceeding with scenario merging, the fundamental concept is outlined. Figure 4.5 illustrates two distinct scenarios that share a similar relationship between the ego and object scenario, as indicated. The core idea is to substitute the ego object found in the **"To"** envelope with the ego object from the **"From"** envelope, ensuring minimal deviation in the ego vehicle's trajectory between the two scenarios. This finding of a similar Ego vehicle from two different envelopes is a sample, and in fact, the method is having an Ego vehicle or an Ego-object vehicle and replacing it in a new scenario, whereas, from an Ego's perspective, it is a complex scenario.

Here, the ego vehicle serves as a reference point, allowing its behavior to be adjusted to test the scenario's robustness to changes while other road users retain their unique trajectories and behaviors. The grey objects in picture (d) in Figure 4.5 show similar objects in both similar envelopes. This approach examines how variations in the ego vehicle's behavior impact the scenario's overall dynamics. The figure demonstrates the procedure of identifying similar road users within the two scenarios, removing the original ego object, and then importing the ego object from the **"From"** envelope into the **"To"** scenario. This process retains other road users in their initial state, meaning the traffic environment comprises diverse vehicle types, varied locations, velocities, and behaviors. These four factors, vehicle type, position, speed, and behavior, play a critical role in the scenario generation process and will be further detailed in the replacement procedure.

The concept of scenario merging is introduced, followed by an exploration of its contribution to generating complex scenarios. Scenario merging is a systematic approach that integrates two or more similar scenarios into a single, cohesive entity, significantly enhancing the creation of intricate and realistic scenarios. This process enables the study of complex interactions within traffic environments. After selecting an appropriate recording and deriving similar envelopes, the merging process proceeds. This approach builds on the results of the similarity analysis, where compatible **"From"** and **"To"** envelopes are identified. By integrating these envelopes, a unified scenario is created, combining elements of both while preserving key behavioral patterns and spatial-temporal relationships. The merged scenario allows for a more comprehensive scenario dataset to simulate real-world traffic dynamics and support robust vehicle responses under varied and complex conditions. Scenario merging expands the scope of scenario generation beyond isolated cases to produce a more layered and interconnected representation of traffic behavior. The approach also enhances the database of test scenarios, allowing for detailed analysis and testing of the multifaceted nature of real-world traffic interactions.

The final Merged Scenario is presented in Figure 4.6, encompassing both the ego vehicle and associated objects. This scenario is constructed based on the ego-object relationship, using the results of the base scenario similarity analysis. Specifically, it integrates elements from the **"From"** envelope for the ego and relevant objects from the **"To"** envelope. This combination ensures that the final scenario reflects the alignment of behavior patterns and spatial-temporal relationships identified during the similarity analysis, providing a cohesive and representative merged scenario for further analysis and testing.

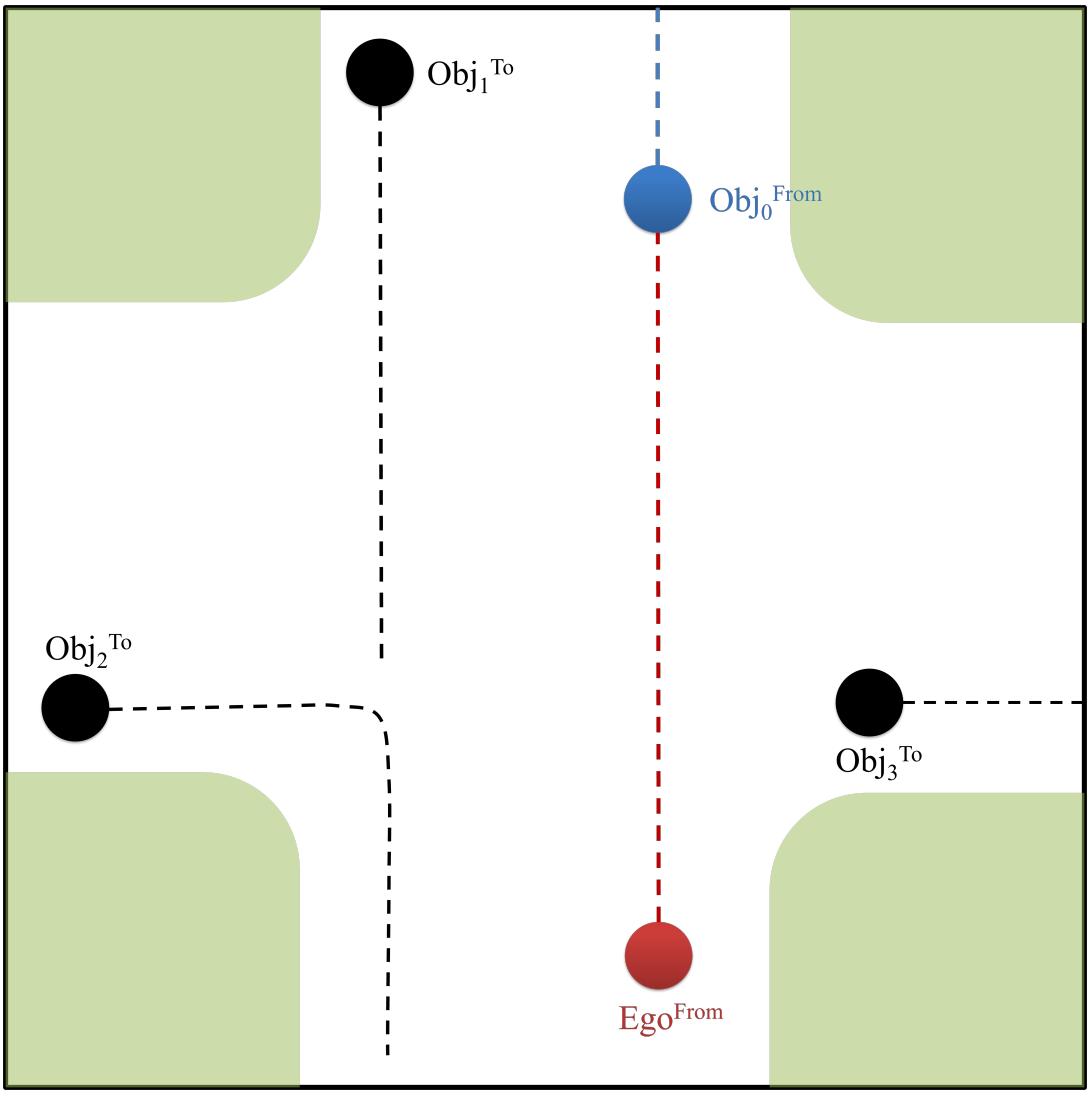


Figure 4.6: Complex Scenario illustration Sample

Figure 4.7 illustrates the abstract procedure involved in scenario combination, encompassing each stage from envelope selection and similarity analysis to the combination process involving replacement, which will be further detailed in the following paragraph. The final outcome of this procedure is an enriched scenario that integrates key elements from both the **"From"** and **"To"** envelopes, preserving contextual continuity and enhancing the complexity of the traffic scenario for subsequent analysis.

4.5 Replacement Procedure

In this section, the primary step of the methodology, the replacement of scenario elements, for scenario-based testing will be explained. It consists of procedures figured out in Figure 4.5. Figure 4.8 represents a detailed procedure of the replacement procedure. In the initial steps, as the **"From"** and **"To"** envelopes are found based on the similarity analysis, the ego and object are then identified based on the similar relations for both the **"To"** and **"From"** envelopes. Thereafter, upon replacing these objects, the scenario includes a new ego object unfamiliar to the other road users (RUs). In this case, the behavior of other RUs must be recognized with respect to the imported object, forming the initial step. As discussed previously, the definition of ARtS in 2 is then applied, where the ARtS object is updated based on the **"From"** and **"To"** envelopes. The prioritized objects which are referred to as "prioritized objects" and are defined as selected agents (for example, vehicles, pedestrians) in a simulation scenario that individual agents have to take into account for interaction or response. This prioritization enables the simulation to focus on significant entities while ignoring less relevant objects, optimizing real-time capabilities. Selection considers basic traffic rules and avoids conflicts unrelated to the ego vehicle's behavior, reducing complexity in scenario simulation. These prioritized objects must be identified from the **"From"** envelope and recognized in the **"To"** envelope. The third block addresses additional relations, an optional relation focusing on a road user, where ARtS attributes are updated for that object; this third block might be important if an object does not prioritize a focused object and can be added for the purpose of additional relations, not necessarily critical step. Completing these three blocks updates the behavior of other road users in the **"To"** envelope based on changes in the behavior of the ego vehicle, allowing other road users to adjust their behavior and reactions accordingly.

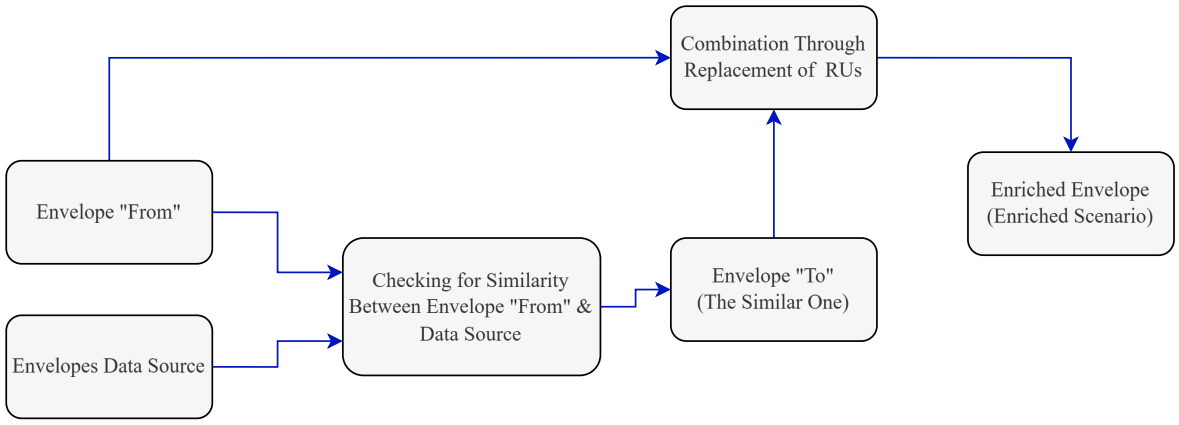


Figure 4.7: Scenario Merging Abstract Diagram

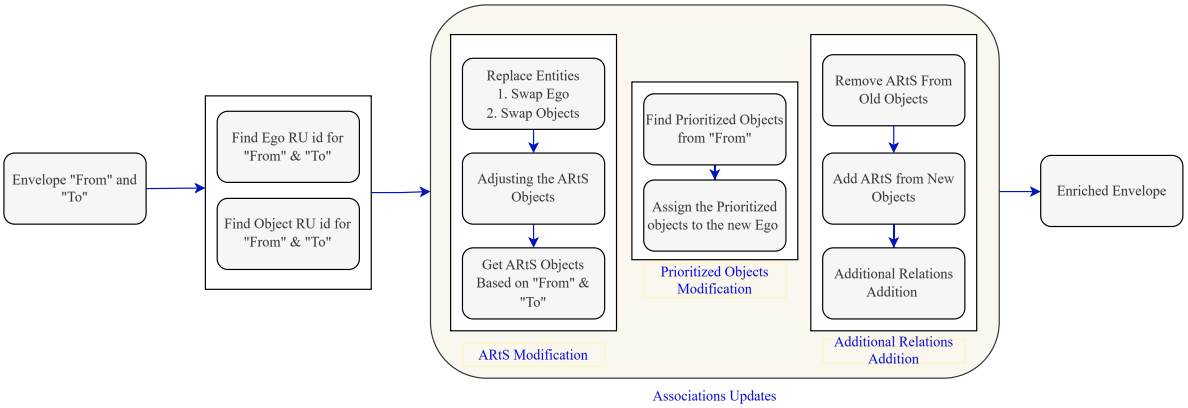


Figure 4.8: Scenario Elements Replacement Diagram

4.6 Simple Scenario

As discussed in the scenario merging section, the similar ego object has been replaced based on the ego-object relationship, with associations updated to reflect the behavior of other road users. This has resulted in the creation of an enriched or merged scenario. Another type of scenario termed the ”simple scenario,” or, in another term, ”reduced scenario,” is defined here as one that includes only the ego and object elements from the **”From”** envelope. The purpose of generating a reduced scenario is to enable comparative analysis between the merged and reduced scenarios. It can also be considered a baseline of performance in terms of cycling time and behavioral analysis compared to complex ones. For the Analysis section, the following data are considered for comparison:

Cycling Time: The cycling time of the ego vehicle is critical, especially when a controller is used. In this context, cycling time refers to the interval required by the controller to update itself. In one case, this timing is influenced by the number of road participants, as the ego controller must analyze the traffic situation and apply relevant functions based on behavior. Evaluating controller cycling time between the reduced and merged scenarios can yield insights into performance under different traffic complexities.

Behavioral Analysis: The impact of other road users on the controller’s functioning varies based on whether a controller is present. This section is focused on analyzing how the presence of other road users in the merged scenario affects the ego vehicle’s behaviour, rather than the controller’s design itself.

Metrics Evaluation: Additional metrics, such as the number of crashes and Time to X (TTX) metrics, provide more data points for comparison. Such metrics, expanded upon in the Data Analysis section, will be used to compare the performance and safety of both reduced and merged scenarios, and they give insight into the complexity of scenario based testing and failure case analysis in the event of a crash in scenarios.

4.7 CARLA

Figure 4.9 outlines the purpose of using the CARLA simulator and its contribution to the research results. CARLA is an open-source simulator specifically designed for autonomous driving research, providing a highly realistic and customizable environment that supports the testing and development of self-driving vehicle technologies. Its advanced capabilities enable the simulation of complex traffic scenarios, allowing for comprehensive analysis of vehicle behaviors in controlled settings. Below is an illustration of the CARLA simulator environment featuring a sample intersection from the Bendplatz intersection dataset. This environment replicates real-world conditions and supports scenario-based testing, making it an ideal tool for examining the interactions between the ego vehicle and other road users under various conditions. Through CARLA, detailed data on vehicle performance, safety metrics, and response times can be collected and analyzed, facilitating the validation of the scenarios described in this study.

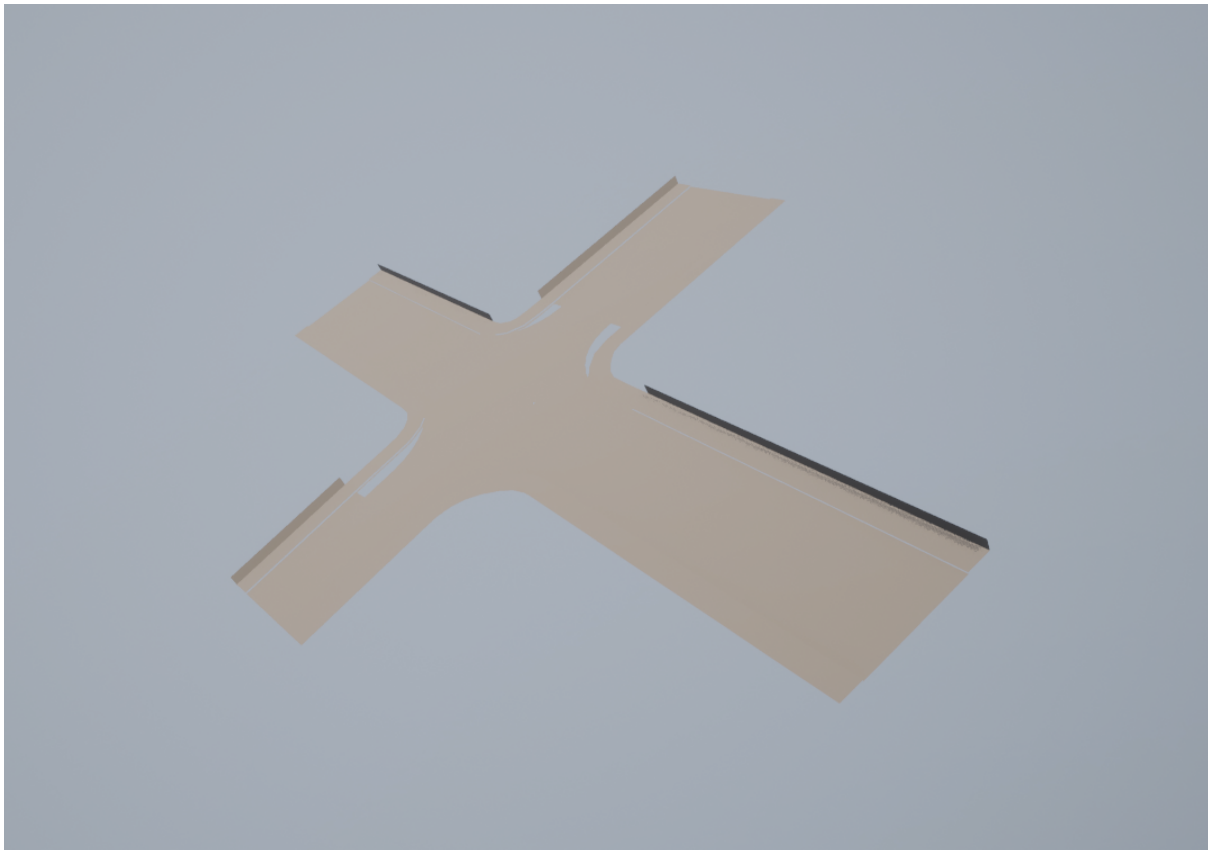


Figure 4.9: CARLA Environment, Bendplatz Intersection

Simulation and Postprocessing: Figure 4.10 illustrates the process from Omega files to the CARLA simulator. The procedure begins with identifying envelopes through similarity analysis and merging these envelopes to create scenarios. For simulation purposes, OpenSCENARIO and OpenDRIVE files are generated, providing the structural and behavioral definitions required for CARLA. Additionally, a driving function may be incorporated if necessary to model the ego vehicle's interactions within the scenario. All elements, the envelopes, OpenSCENARIO, OpenDRIVE, and any added driving function are integrated within the scenario runner, which interfaces directly with the CARLA simulator. This setup facilitates the execution of the merged scenarios in a simulated environment. Upon completion of each simulation, the trajectories of all road users (RUs) are recorded, producing test results essential for subse-

quent data analysis. These recorded trajectories allow for an in-depth examination of vehicle behavior, interactions, and performance metrics.

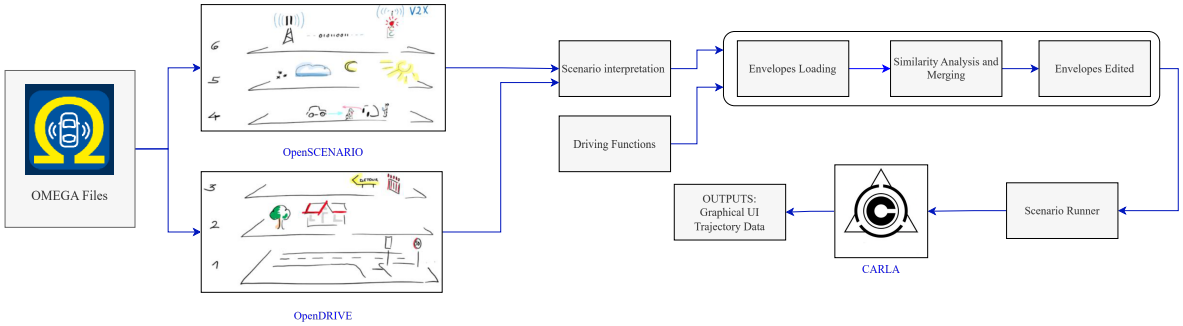


Figure 4.10: From Omega Files to CARLA

In the CARLA simulator, synchronous and asynchronous modes [1] directly influence the simulation's cycling time, which refers to the time taken per cycle to process all relevant simulation updates. The distinction between synchronous and asynchronous cycling time is crucial for different testing scenarios, particularly when evaluating autonomous vehicles and complex environmental interactions.

4.8 Data Analysis

In this section, the data analysis process is discussed and developed at IKA, Aachen, Germany. Beginning with the trajectory files obtained from the CARLA simulator. Figure 4.11 outlines the data analysis procedure, starting with input files in JSON format from the CARLA simulator. These files contain trajectory data for each road user, including position coordinates (x, y), velocity, and yaw. To prepare this data for analysis, the trajectories are interpolated, allowing for a consistent representation of movement over time. The Scenario Analysis stage involves loading and structuring trajectory data for each road user, facilitating a systematic analysis of behaviors.

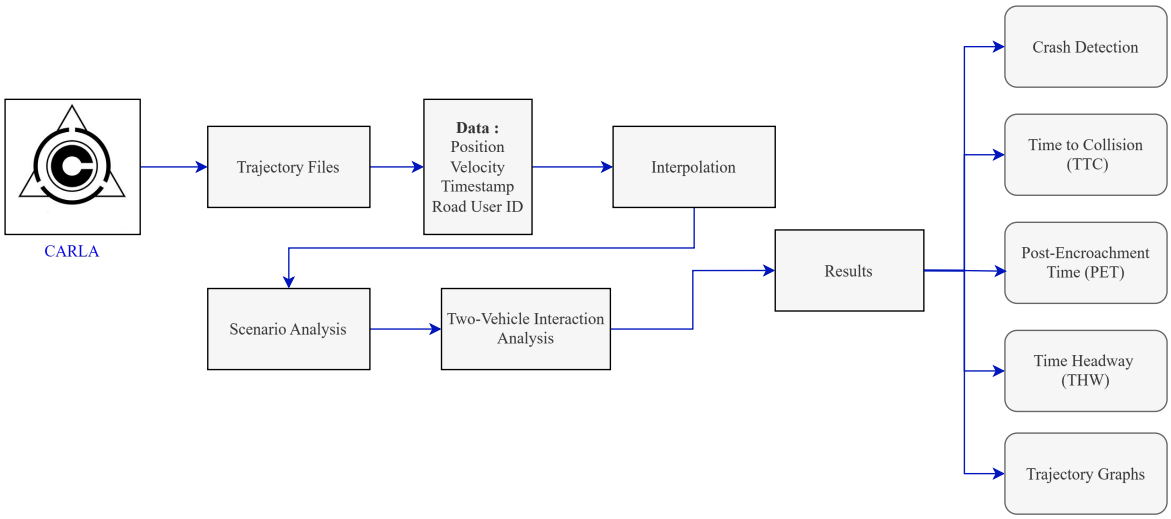


Figure 4.11: Data Analysis Procedure

This stage uses each road user's speed and direction to project future paths and to conduct behavior analysis to generate relevant statistics; optionally it can assess interactions between road users, a starting point for further evaluation of individual behavior, interaction potential, and risk identification. The Two-Vehicle Interaction Analysis stage is focused on the interactions between two road users and the assessment of the potential collision risks through the detection of trajectory overlaps. The metrics of Time to Collision (TTC), Time Headway (THW) and Post-Encroachment Time (PET) are calculated in order to quantify the safety of the scenario. These metrics give a view of the proximity, following distances and timing in shared spaces and can be used to assess the crash risks, safe passing scenarios, and near miss events. This detailed analysis supports a robust evaluation of safety in autonomous driving systems, highlighting proximity based metrics to enhance risk detection and response strategies.

Chapter 5

Results

5.1 Introduction

This chapter presents the output of the implemented methodology. Table 5.1 represents two main subsections, one is (ScS1 to ScS4), which shows the sets of complex scenarios as the result of the methodology in chapter 4 and later, they are divided into four subsections which mainly focuses on how each parameter like (ARtS/RtS), and the existence of the controller can affect the robustness of the scenario. For covering the second research question in chapter Research Objectives 3, where later for each scenario sets the percentage of crashes will be reported where based on those values, an appropriate set of parameters can be chosen, as scenario sets, which can be considered the scenario sets with parameters contributing more robust scenarios. Again, the remaining scenario sets (ScS5 to ScS8) are related to the simple scenario, only the ego and following object, from the ego object relation. These sets of scenarios, again, are simple but only consider the ego objects relation objects contributing to the complex scenario where in case the reason for the crash is unknown in a complex scenario, it can be inspected in the simple scenario whether there is a failure there or not, where the existence of crash in the simple scenario can bring the same problem in the merged scenario. This is where the generated scenarios in which the Ego and Object were replaced in a new, complex scenario. This complexity arises because both of them can behave differently, challenging the scenario itself in terms of how other Road Users (RUs) can react, thereby demonstrating the robustness of the RUs in scenarios in how they react in terms of breaking, changing the speed, or following in their path. Accordingly, after obtaining the trajectory results of the generated scenario, they are evaluated using metrics discussed in Chapter 2, which assess how the RUs behave in the scenario according to the new objects(ego and newly imported object) to catch if

they were robust enough to prevent from the crash or not. The results include datasets detailing the scenario contents, with analysis conducted on four different intersections and over 300 scenario files. To evaluate the scenario, it has undergone different variations, such as the presence of the controller for the ego vehicle. Table 5.1 illustrates these scenario variations, showing the presence of different elements affecting scenario performance. Initially, simple scenarios are presented as reduced ones, followed by combined scenarios in a complex form. Another important factor affecting scenario variation is the presence of the controller used for the ego vehicle, influencing the scenario's behavioral performance. Additionally, the inclusion of the ARtS serves as an element for achieving a more robust scenario.

Scenarios	Combined	Reduced	ARtS	RtS	With Controller	No Controller
Scenario Set 1 (ScS1)	✓		✓		✓	
Scenario Set 2 (ScS2)	✓		✓			✓
Scenario Set 3 (ScS3)	✓			✓	✓	
Scenario Set 4 (ScS4)	✓			✓		✓
Scenario Set 5 (ScS5)		✓	✓		✓	
Scenario Set 6 (ScS6)		✓	✓			✓
Scenario Set 7 (ScS7)		✓		✓	✓	
Scenario Set 8 (ScS8)		✓		✓		✓

Table 5.1: Scenario Sets Table

Furthermore, based on the defined parameters presented in Table 5.1, a comprehensive analysis will be conducted. The evaluation begins with key safety metrics, such as the number of crashes, to assess the overall safety performance of the scenarios. Following this, a detailed failure case analysis is performed for scenarios involving crashes to identify the underlying causes of failure. This in-depth examination aims to uncover factors such as unexpected road user behaviors or deficiencies in scenario design that contribute to failures. To achieve this, we collect and analyze trajectory data which combined with velocity-trajectory plots, Time-to-Collision (TTC) and Time Headway (THW) metrics. The insights gained help to reveal vehicle dynamics and potential collision points over time, which enables the identification of patterns and anomalies that may not be immediately evident from isolated metrics. In addition to trajectory based metrics, other parameters such as the cycling time of the ego controller and its updates in CARLA are crucial in influencing road user (RU) behaviour and scenario robustness for the ego vehicle. The cycling time becomes particularly significant in complex scenarios with a higher number of road participants, as it affects how frequently the ego vehicle updates itself in response to dynamic interactions. Moreover, the ego vehicle's cycling time is analyzed across scenarios with different controller parameters to evaluate its influence on interactions with other road users. This assessment helps determine how the ego vehicle's update frequency affects the responses and behaviors of surrounding road users, ultimately shaping the overall realism and effectiveness of the scenario.

5.2 Crash Analysis

As illustrated in the table 5.2, out of over 300 scenarios in total for all scenario sets, the table shows the number of crashes in terms of the percentage. Here, the crashes are focused on the ego vehicle and object road user.

Scenarios	Number of Scenarios	Crash Occurrence Percentage
Scenario Set 1 (ScS1)	39	25.64%
Scenario Set 2 (ScS2)	39	23.08%
Scenario Set 3 (ScS3)	39	25.64%
Scenario Set 4 (ScS4)	39	23.08%
Scenario Set 5 (ScS5)	41	12.20%
Scenario Set 6 (ScS6)	41	0%
Scenario Set 7 (ScS7)	41	9.76%
Scenario Set 8 (ScS8)	41	2.44%

Table 5.2: Scenario Sets Crash Percentages Table

Table 5.2 shows that the highest crash percentage, 25.64 %, occurs in the combined scenarios. This percentage reflects crashes in RtS and ARtS scenarios, as well as in scenarios where a controller is applied to the ego vehicle. Interestingly, better performance is observed when the controller is absent, as expected. This suggests that the controller's presence, depending on its operational efficiency, may have contributed to collisions involving the ego vehicle and other road users. Additionally, the table reveals that ScS1 and ScS3 share the same number of crashes, as do ScS2 and ScS4. This pattern suggests that crashes in these scenarios may stem from similar underlying factors, raising questions about potential limitations in scenario configurations. To investigate these issues further, a failure-case analysis table has been created to document the specific causes behind each crash occurrence. Table 5.3 provides detailed insights into the factors leading to crashes, such as controller malfunctions and misinterpretations of road user behavior. By systematically categorizing these failure cases, we can identify common trends and root causes that require further examination. Following this, a detailed analysis of the failure cases is conducted. This includes examining trajectory data, such as velocity-trajectory plots, Time-to-Collision (TTC), and Time Headway (THW) metrics, to better understand the dynamics behind each crash. By analyzing these interactions, this study aims to uncover how the ego vehicle and other road users contributed to the crash scenarios. This in-depth examination is crucial for identifying weaknesses in either the controller's performance or the scenario configurations, providing valuable insights for improving the safety and effectiveness of future scenarios.

Table 5.3 presents the causes of several crashes based on manual inspection. It was found that in three instances, a teleported object unexpectedly appeared near the ego vehicle, leading to an unavoidable collision. This sudden intrusion disrupted the normal flow and control of the ego vehicle, making it impossible to avoid the crash.

In another case, the ego vehicle's speed was significantly higher than that of a newly introduced object, resulting in a collision because the ego vehicle couldn't adjust its speed in time. To assess the effectiveness of the controller in such situations, the scenario was retested with the controller in operation. The implementation of a different controller, referred to as the test controller, successfully regulated the ego vehicle's speed, preventing any collisions. This outcome demonstrates the controller's ability to improve scenario safety by effectively adjusting the ego vehicle's speed.

Additionally, another scenario exhibited backward-forward oscillations of the ego vehicle, ultimately leading to a crash. The full failure analysis table is available in the appendix 8.1. By categorizing these unexpected behaviors as anomalies, which are not directly related to the core methodology—and omitting them from the dataset, the crash percentage is significantly reduced from 25.64 % to below 5% for scenarios involving ARtS and different configurations. This substantial reduction highlights the method's strong performance, suggesting that most crashes resulted from external factors rather than fundamental issues within the methodology. These irregularities can be mitigated by:

A. Refining the TTX metrics within the ARtS approach for cases related to ARtS limitations.

B. Preventing vehicle spawns in locations already occupied by the ego vehicle.

C. Enhancing the concept of adaptive teleporting to minimize unexpected object appearances.

This analysis reinforces the methodology's reliability and robustness, demonstrating its effectiveness under normal conditions without unforeseen anomalies.

Scenarios for Combined, ARtS	**Crash Reason, No Controller**	**Crash Reason, IAC**
inD10/env13/RU56	No Crash Detected	High object's speed from back side caused a crash with ego
inD24/env2/RU45	Teleporting Object	Teleporting Object
inD32/env4/RU74	Bad Ego Behaviour	Bad Ego Behaviour
inD32/env16/RU49	No Crash Detected	No Crash Detected
inD32/env54/RU72	Teleporting Object	Teleporting Object
inD32/env60/RU33	Crash due to high Ego speed	No Crash Detected

Table 5.3: Failure Case Analysis For Scenario with Crashes, Combined

Additionally, a detailed trajectory analysis will be conducted to further examine the previously identified crash causes. This analysis provides deeper insights into the dynamics that lead to collisions, offering a clearer understanding of the factors influencing vehicle interactions and movements. Such an in-depth investigation is essential for refining scenarios and enhancing overall safety and performance. Scenarios that involve a combination of ARtS and Intersection Assistance Control are further analyzed to uncover the specific causes of crashes within these configurations. The findings reveal that, similar to previous cases, crashes occurred in the same scenarios, with the exception of one newly identified scenario. This new scenario was closely linked to how the object interacted with the ego vehicle, highlighting that the object's behavior directly influenced the outcome. Specifically, the object exhibited behaviors that were not adequately anticipated by the ego vehicle's control system, resulting in a collision. To systematically categorize and understand these failure cases, a comprehensive failure case analysis table is included in the annex. This table serves as a valuable tool for identifying patterns and recurring issues across different scenarios. By thoroughly analyzing each failure case, adjustments can be made to scenario parameters, control algorithms, and interaction protocols to mitigate these risks. This proactive approach ensures that the methodology remains effective and continues to evolve in response to identified challenges, ultimately improving autonomous vehicle safety and performance.

Table 5.4 shows the failure analysis for the scenarios with IAC and Reduced ones.

Scenarios for Reduced, ARtS	**Crash Reason, IAC**
inD24/env02/RU45	High Speed of the Ego
inD24/env20/RU40	Bad Braking When Approaching the Cyclist
inD32/env4/RU74	Bad Behaviour of Object
inD32/env16/RU49	High Speed of the Object From Back

Table 5.4: Failure Analysis For Scenario with Crashes, Reduced

Table 5.4 represents the same scenarios as those presented in the combined dataset, with one exception: the scenario "inD24/env20/RU46" does not have an equivalent entry in the combined database. This discrepancy arises from a spawning issue where certain actors cannot be placed at their designated spawn locations. Such issues typically occur when the specified locations are occupied, blocked by other objects, or rendered inaccessible due to map constraints that prevent safe spawning in those areas. As a result of the tables 5.2, 5.3, and 5.4, two main reasons for the crashes are the teleportation and limitation of the flexibility of the Road User objects which were witnessed in the scenarios with the no controller and also the controller itself made some challenges due to its internal parameters that caused the crash. These issues can be addressed by modifying the envelope extraction method, enhancing the adaptability of ARtS, and adjusting teleportation mechanisms based on distance and spawning constraints.

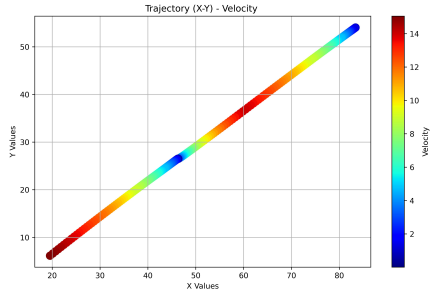
To further validate the methodology in scenarios without failures, two approaches are employed. First, the same crash-inducing scenarios are tested with different controllers or controller parameter adjustments to evaluate their impact. This is based on the logic that safe distances between vehicles, derived from Time-to-X (TTX) metrics, influence how the con-

troller responds to the TTX threshold and how robust the controller itself is. An analysis of Tables 5.2, 5.3, and 5.4 highlights two primary causes of crashes: Teleportation issues, where objects are introduced in positions that disrupt the scenario. Limitations in the flexibility of Road User (RU) objects, affecting how they adapt to scenario variations. These issues were observed in scenarios without a controller and in cases where the controller was not the direct cause of failure. Instead, crashes were linked to specific internal parameters that restricted scenario adaptability. To confirm these findings, two validation methods are proposed: Reusing the same crash-inducing scenarios while testing different controllers or modifying controller parameters to assess their influence on scenario outcomes. Exploring the relationship between safe distance and controller performance using Time-to-X (TTX) metrics to evaluate how effectively the controller reacts within the defined TTX thresholds and maintains safe vehicle distances. By applying these methods, the study aims to enhance the reliability and robustness of scenario-based testing, ensuring controllers can effectively respond to dynamic conditions and prevent crashes.

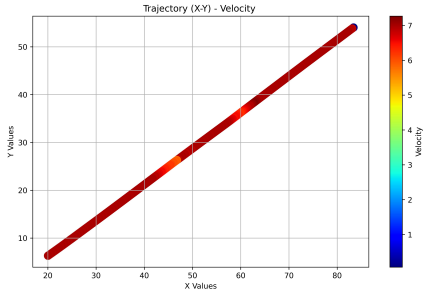
Effect of changing the controller on the Crashes: Furthermore, another type of controller, the test controller, as discussed in Chapter Research Objectives 2, was applied to scenarios involving crashes with the IAC controller. In scenarios with the IAC controller, several cases were identified where the sudden appearance of an object in front of the ego vehicle caused an inappropriate reaction from the ego controller. While this may explain the crashes in those cases, it does not necessarily imply the same reasoning applies to crashes with the test controller. The aim of using another type of controller, in this case, the test controller, is to make sure the reasoning behind the crash is the controller itself but not the scenario elements themselves. After applying the test controller, the results show a similar effect, with crashes occurring due to the sudden appearance of an object. However, in one scenario, the ego vehicle proceeded straight while the object turned left. This interaction resulted in a collision with the IAC controller but did not lead to a crash with the test controller. It again proved the scenario, the hypothesis with limitation with controller functionality is true, and the scenario itself functions correctly. This specific scenario can be analyzed from two perspectives:

Controller Perspective: The test controller demonstrated a positive effect, indicating that changing the controller mitigated the crash and also showed the scenario is efficient, but the controller has limited functionality. It can be analyzed that the scenario elements reacted correctly, like waiting, which is the correct behavior in the real world, but the controller did not behave like a human road user in reality because the controller is limited in this case (no AEB activated).

Scenario and Road User Behavior Perspective: The scenario reveals that crashes can occur if an inappropriate controller is used. These findings highlight the importance of the ARtS object's flexibility in adapting to the ego vehicle's behavior, particularly in terms of not only speed changes but also trajectory adjustments. The change in the controller did not have any significant effect on scenarios involving the sudden appearance of objects. Therefore, it can be concluded that modifying the parameters of the IAC controller may not have a substantial impact on preventing crashes in such cases. However, it does influence scenarios with crashes by affecting how other Road Users (RUs) react to the ego vehicle's behavior. Below is the velocity profile over the trajectory for the scenario rec32/env21 using both the IAC and test controllers.



(a) Ego Velocity over Vehicle Trajectory, Test Controller



(b) Ego Velocity over Vehicle Trajectory, IAC

Figure 5.1: Controller Effects on Preventing High Ego Speed Approaching a Turnover Object

Figure 5.1 illustrates the effects of using the Test Controller and IAC Controller. As indicated in the accompanying table 5.5, in the third row, no crash occurred with the Test Controller. Plot (a) demonstrates the variation in the ego vehicle's speed, where a reduction in speed at the midpoint effectively prevented a crash during the object's turning maneuver. In contrast, Plot (b) reveals that the ego vehicle maintained a high velocity, which led to a crash with the object under the IAC Controller. This analysis again represents the limitation from the controller side, and the scenario elements behaved correctly; nevertheless, there would be a crash in the scenario with the test controller. So, in short analysis, natural behavior from the other objects in the scenario but unreal behavior from the ego behavior if utilized with an inappropriate controller.

Table 5.5 shows the crash reasons with the test controller for combined ARtS scenarios.

Scenarios: Combined, ARtS, TC	**Crash Reason**
inD32/env04/RU74	Object Bad Behaviour
inD32/env16/RU49	Sudden appearance of the Obj
inD32/env21/RU60	Smooth reaction of Ego with respect to tuning object
inD32/env54/RU72	Sudden appearance of the Obj

Table 5.5: Failure Analysis For Scenario with Crashes, Test Controller

The failure case analysis in table 5.5 has been conducted by reasoning through scenario inspection and illustrating the relevant values using trajectory plots. Attention now shifts to a sample working scenario that does not exhibit the previously observed effects. In this scenario, complex behaviors such as Autonomous Emergency Braking (AEB) or emergency braking will be applied to the ego vehicle to evaluate their impact on other road users.

Furthermore, a sample working scenario without crashes for either the ego vehicle or the object was selected from the envelopes, as an example, recording 32, identified as Rec32/Combined/ARtS/IAC. To introduce additional challenges to the scenario, certain parameters were modified as arguments in the OpenSCENARIO file based on the controller being used, which are reported in appendix 8. This work is done due to the fact that in Rec32/Combined/ARtS/IAC, the scenario is not challenged enough, as there is a far distance between the objects and the possibility of the crash goes to zero, and not many interactions with other RUs, for assessing the scenario in more challenge mode, the parameters in ego controller is changed to see how the scenario objects can behave accordingly. The mentioned scenario is illustrated in its default mode and will be analyzed in two challenging modes: one is high speed for Ego vehicle and Deactivated AEB, and one in high speed for Ego vehicle and Activated AEB; these changes aim to make other RUs more prone to the ego vehicle and see how the scenario behave accordingly. The result of the three mentioned modes will be pictured and explained.

Mode 1 (Default Mode) : This is the default scenario without Automated Emergency Braking (AEB) activated; controller parameters are defined in appendix list 8.1.

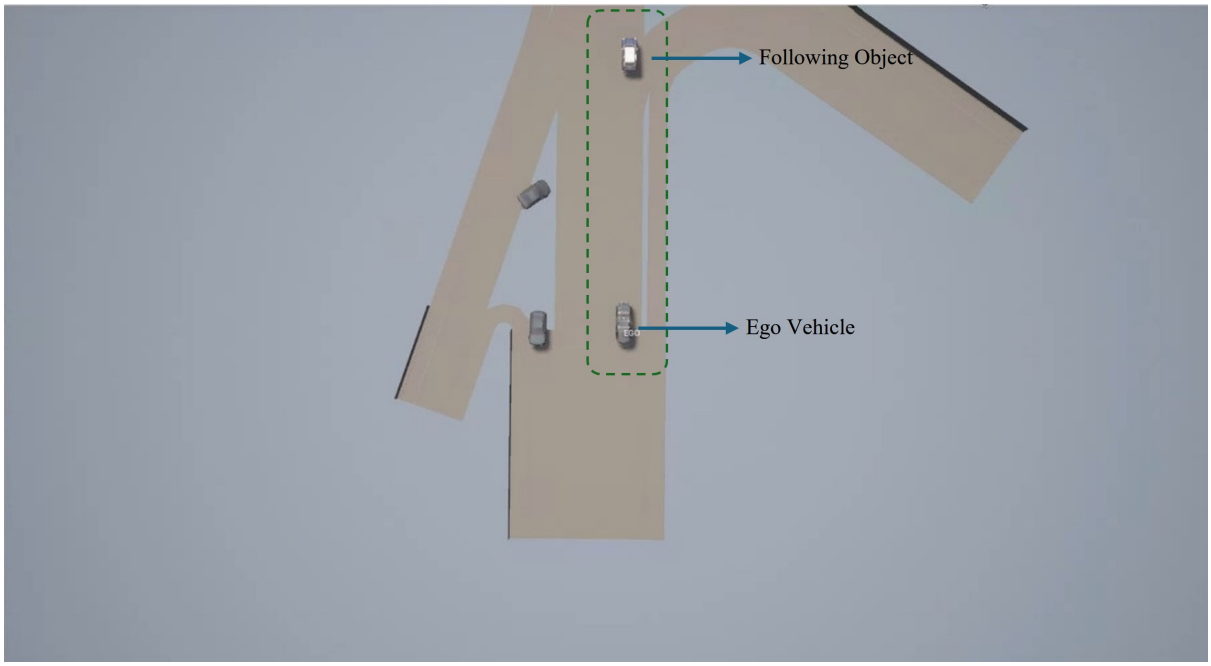


Figure 5.2: Mode 1: Snapshot 1

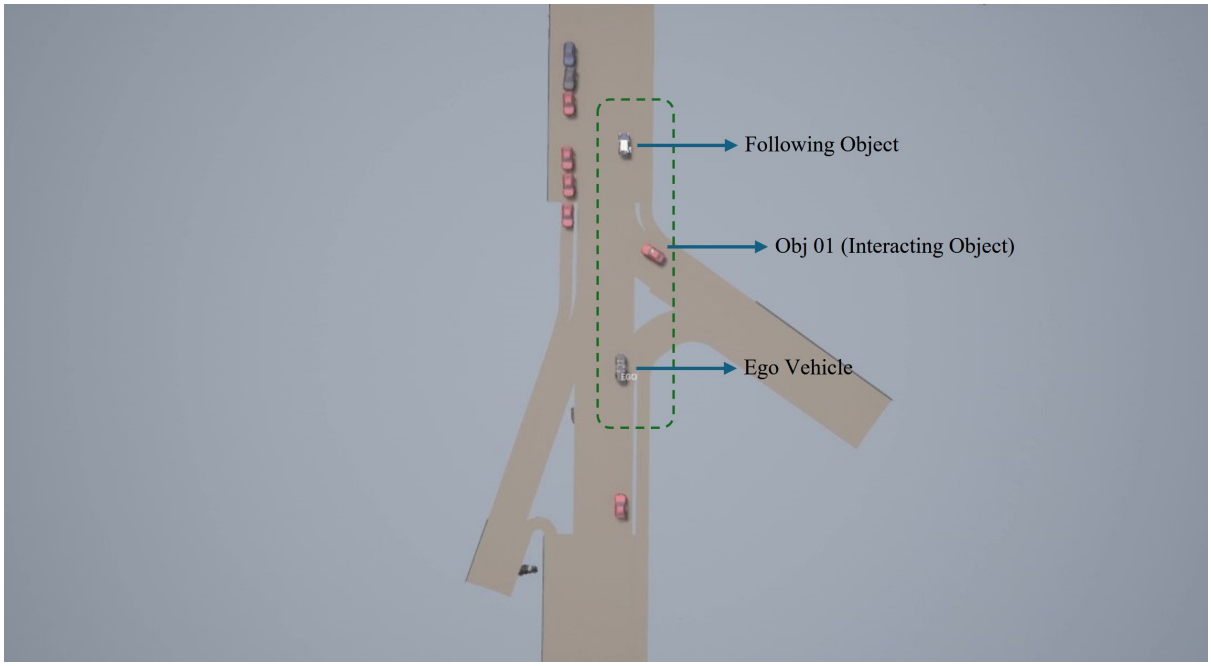


Figure 5.3: Mode 1: Snapshot 2

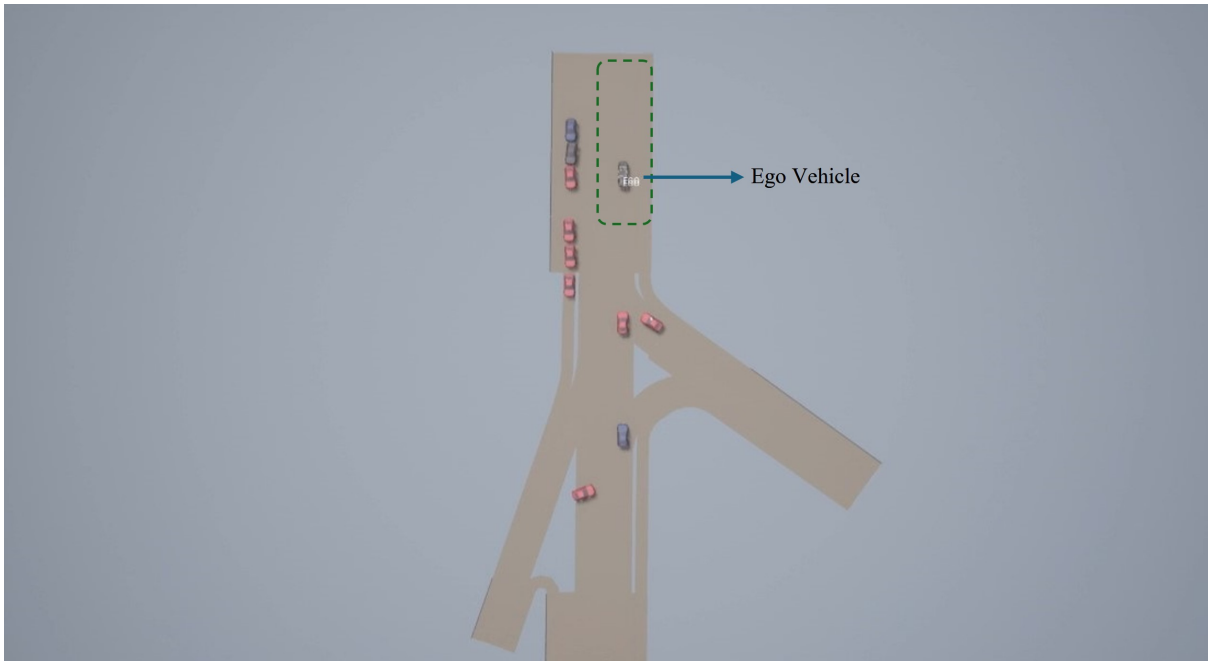


Figure 5.4: Mode 1: Snapshot 3

Figures 5.18, 5.3, 5.4 illustrate three main steps of the scenario where the ego vehicle controller's parameters are in Mode 1 as indicated in list 8.1. Initially, snapshot 1 shows the Ego vehicle, which follows the following object at a far distance, and there is no other challenging vehicle to interact with the Ego vehicle. Later, in snapshot 2, as both the Ego vehicle and the following object still have approximately the same distance as in snapshot 1, it suggests they have approximately the same speed. Moreover, obj 01 in snapshot 2 shows a stopped object that already stopped apart from how the Ego vehicle reacted, where, in this case, it does not contribute as a challenger element to the scenario. In total, mode 1 shows a scenario where both the Ego and object have very limited interaction with other RUs. For scenario-based testing, it is needed also to verify and validate that other RUs can react correctly in a challenging scenario.

Mode 2 (High Speed, AEB activated) : In this mode, the controller parameters are set to High Speed for Ego vehicle, and also the AEB is activated as shown in appendix list 8.2. Snapshot 1 Figure 5.5 again shows the Ego vehicle Following the Following object, whereas Snapshot 2 Figure 5.6 shows that the Ego vehicle reaches the following object at the intersection compared to mode 1.

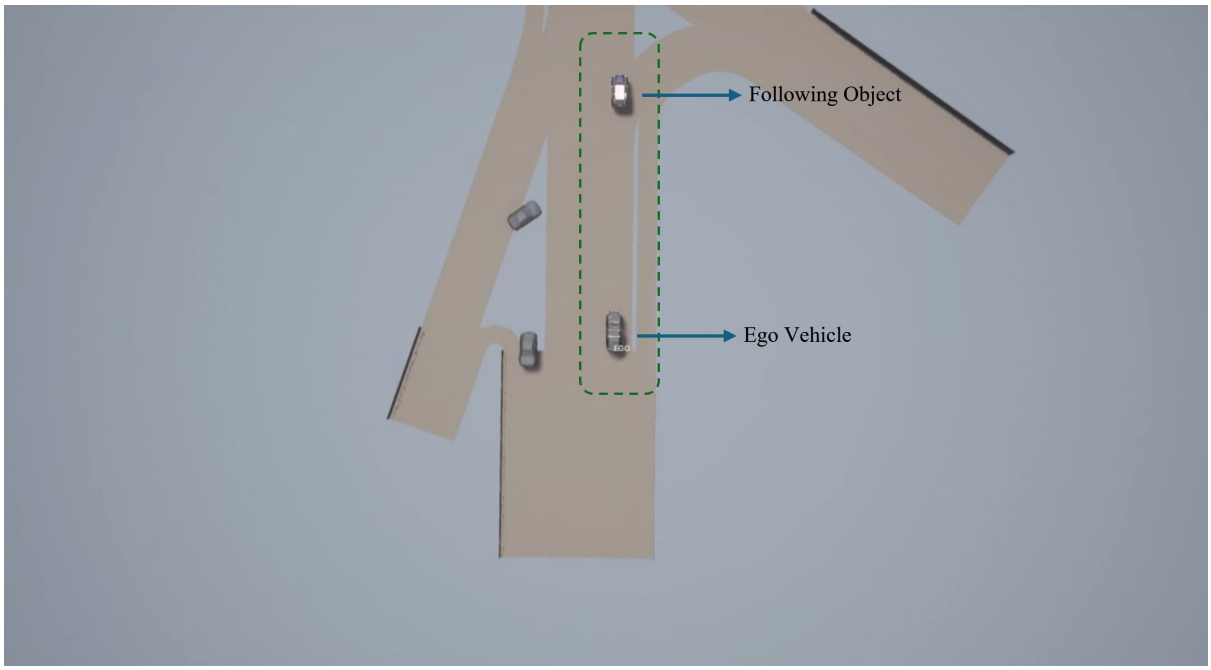


Figure 5.5: Mode 2: Snapshot 1

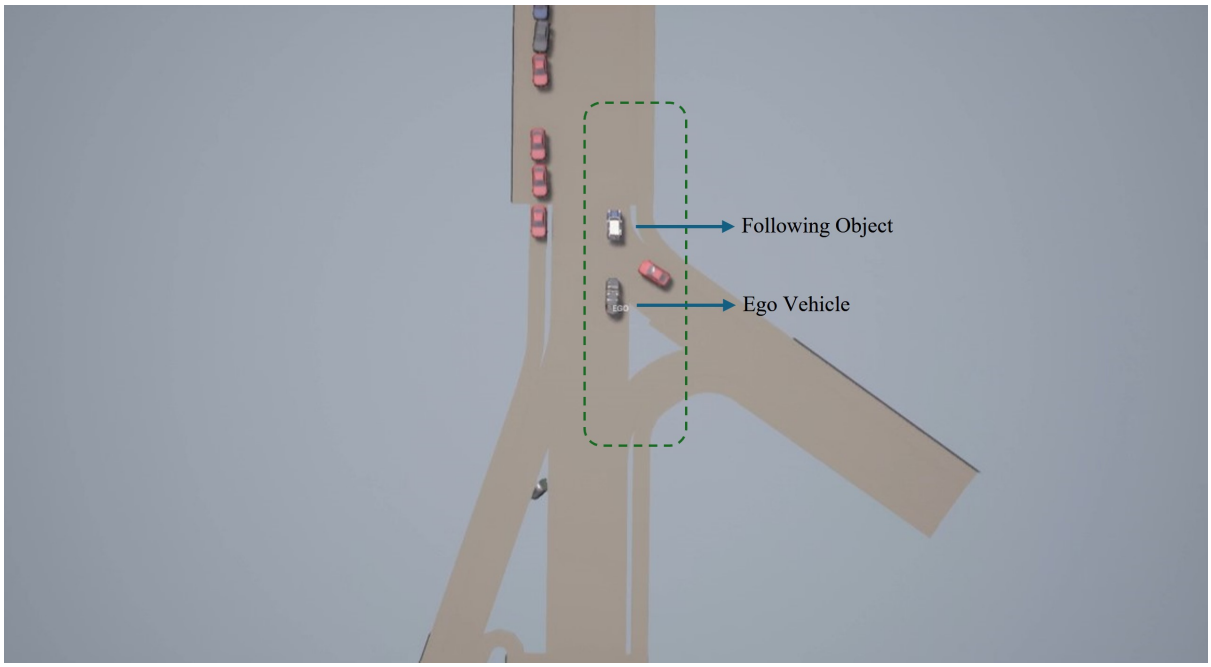


Figure 5.6: Mode 2: Snapshot 2

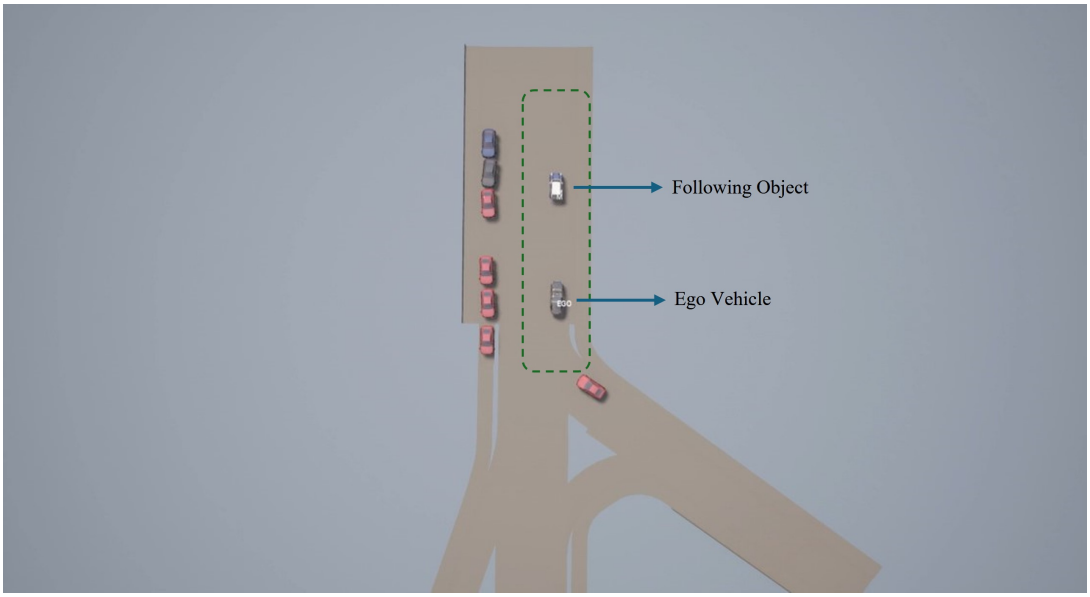


Figure 5.7: Mode 2: Snapshot 3

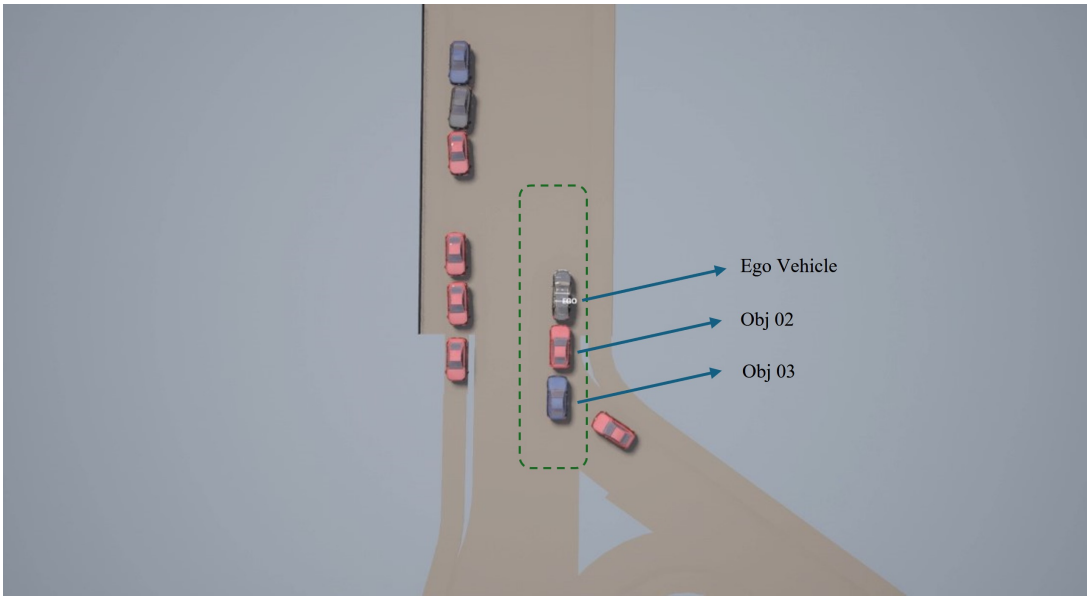


Figure 5.8: Snapshot 4

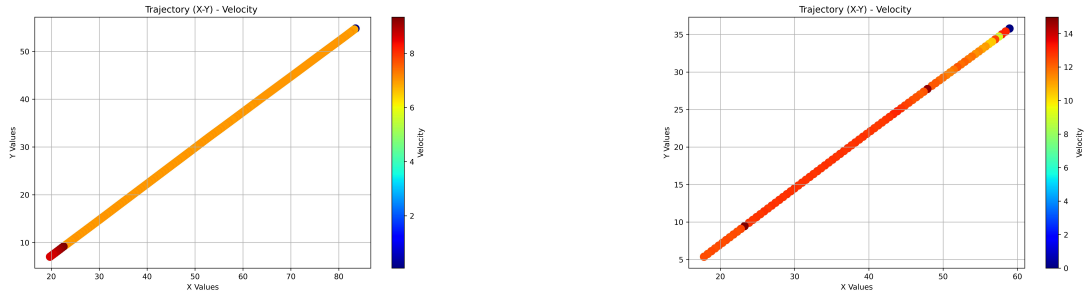
Snapshot 3, Figure 5.7 shows that the AEB is activated in the Ego vehicle, and it stops while the following object continues its trajectory. As the Ego vehicle is stopped, other RUs, obj 02 and obj 03, reach the Ego vehicle at snapshot 4, Figure 5.8, and Accordingly, they behave and stop until the next reaction of the Ego vehicle. Moreover, **Mode 3 (High Speed, AEB activated)** shows the same effect with the difference that no AEB activated and Ego and Object close to each other follow their trajectory without any crash. In conclusion, considering Modes 2 and 3, they show that challenging the scenario by changing the parameters of Ego controllers successfully showed that other RUs were robust against the Ego vehicle and validated the scenario eligibility for scenario-based testing.

5.3 Trajectory-based Analysis

In this section, the analysis of the scenarios is done in terms of affecting parameters in the scenario, like the velocity of the vehicle over its trajectory. This is important to evaluate if the scenario is complex and how robust it is, and it can be focused on the objects where they are in close relation with the ego vehicle and see how the focused object changes its parameters according to the Ego vehicle. Accordingly, apart from visual inspection, trajectory-based analysis provides data to prove the existence of visual inspection.

Trajectory-Velocity Plots: Here are the trajectory-velocity profiles of the three mentioned modes which are based on their visual outputs 5.18, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, where these results validate the robustness of scenario-based testing.

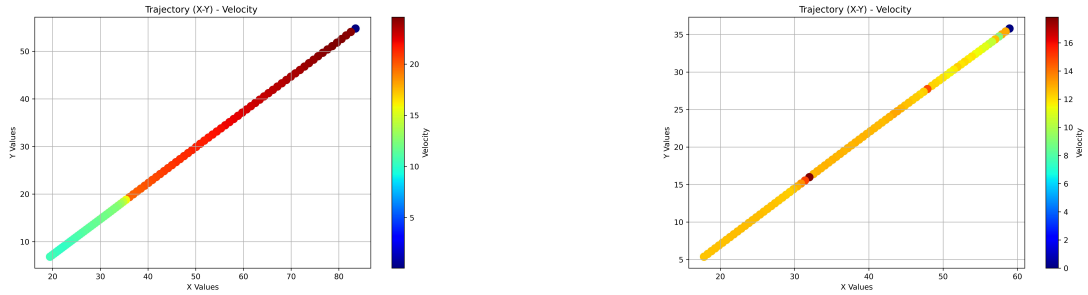
1. Mode 1



(a) Ego Profile (b) Object Profile

Figure 5.9: Trajectory-Speed Profile: Deactivated AEB, Default Ego Speed

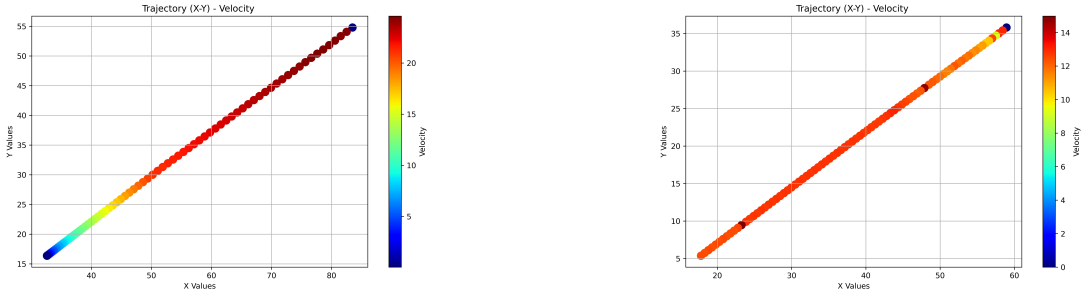
2. Mode 3



(a) Ego Profile (b) Object Profile

Figure 5.10: Trajectory-Speed Profile: Deactivated AEB, High-Speed Ego

3. Mode 2



(a) Ego Profile

(b) Object Profile

Figure 5.11: Trajectory-Speed Profile: Activated AEB, High-Speed Ego

Figures 5.9 present the speed profiles during the trajectory for both the ego vehicle and the object vehicle in the scenario Rec32/ARtS/IAC/env55 under three modes: mode 1, mode 3 and mode 2, respectively. The analysis of these figures provides insights into how variations in speed and braking systems influence the interaction between the ego and object vehicles. As shown in Figure 5.9, the first set of plots corresponds to mode 1. Panel (a) depicts the ego vehicle's trajectory, maintaining a nearly constant speed of approximately seven m/s at the beginning of the trajectory and slightly increasing to around eight m/s by its end. In this situation, the ego vehicle fails to reach the object, which continues on its path unaffected by the ego vehicle's presence. Panel (b) depicts the trajectory of the object vehicle, beginning at a speed close to zero and gradually accelerating to about 12 m/s by the end of its path. This outcome underscores that the object maintains a velocity greater than that of the ego vehicle, preventing the ego vehicle from closing the distance. This indicates that the ego and object are either not competing effectively or aren't fully engaged, regardless of whether the scenario withstands alternative Ego controller parameters.

The second set of plots, illustrated in Figure 5.10, corresponds to the mode 3. Panel (a) shows that the ego vehicle begins with a significantly higher velocity compared to the object. The ego vehicle's speed remains elevated throughout the trajectory, and Panel (b) shows that the object's velocity sharply increases from 12 m/s to 16 m/s as a result of the ego vehicle's speed, after which both vehicles continue at their respective velocities.

The third set of plots, shown in Figure 5.11, illustrates Mode 2. In Panel (a), similar to the earlier example, the ego vehicle starts with a high initial speed. However, the impact of the Automated Emergency Braking (AEB) system becomes evident in the ego vehicle's trajectory. Near the end of the trajectory, the ego vehicle gradually decelerates to a complete stop, as indicated by the blue section in the plot. Panel (b) highlights the interaction point where the ego vehicle approaches the object. The vehicles do not come alarmingly close, likely due to the Time-to-X (TTX) parameters, which help maintain a safe distance. After this interaction, the object vehicle continues at its previous speed without any significant changes. In conclusion, these graphs validate the findings presented in Figures 5.18, 5.3, 5.4, 5.5, 5.6, and 5.7, confirming that the scenario elements behave as expected in response to the ego vehicle's actions. This consistency reinforces the robustness of the scenario for scenario-based testing, demonstrating its reliability in evaluating autonomous vehicle behavior.

5.4 Synchronous and Asynchronous Effects on Crash Occurrence

As discussed in the chapter Methodology 4 regarding the concept of asynchrony, scenarios simulated in asynchronous mode resulted in significantly more crashes, approximately two to three times the number observed in synchronous mode. This increase is attributed to a delay in getting the correct trajectory parameter of the ego vehicle in CARLA, where the actual velocity and position of the ego updates suddenly and cause abnormal behavior, like jumping in trajectory, for the ego vehicle, which is due to the update timing in CARLA, where the vehicle may end up in positions that do not align with its intended trajectory, leading to collisions. Below is an illustration of a scenario where a crash occurred as a result of the asynchronous mode.

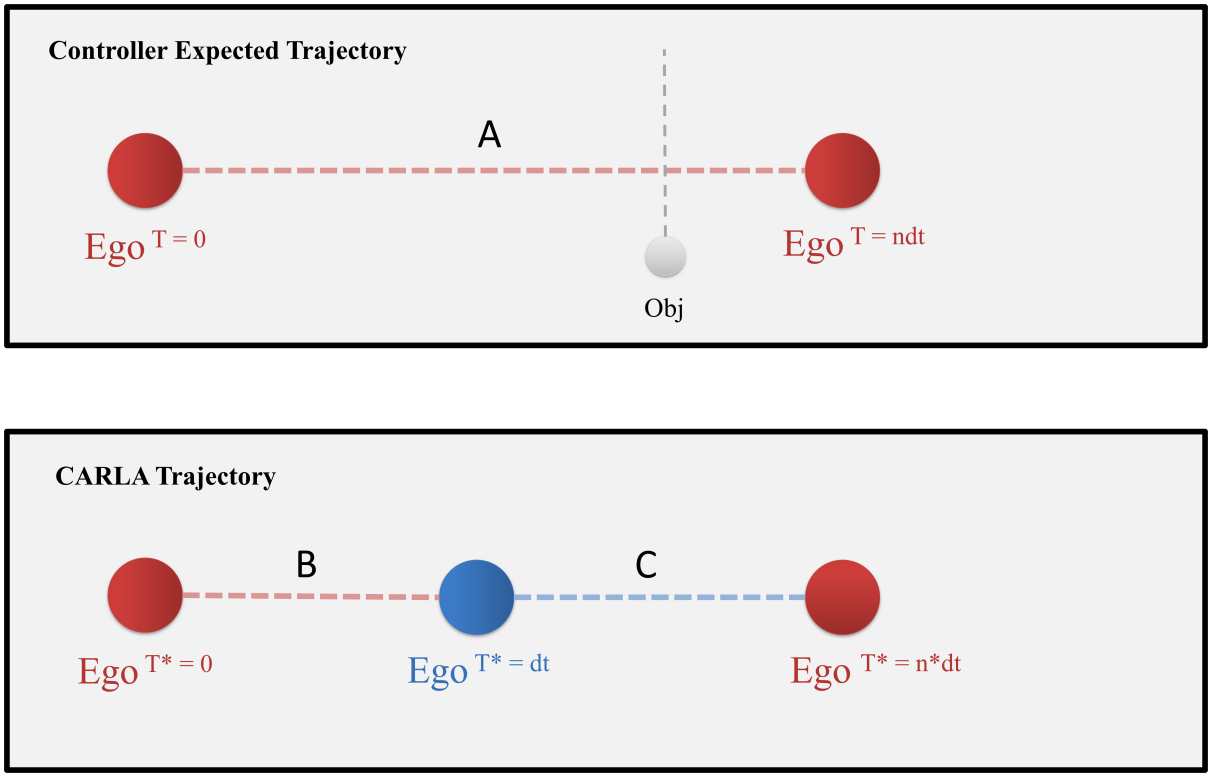


Figure 5.12: CARLA vehicle trajectory (Synchronous and Asynchronous Mode Comparison)

Figure 5.12 illustrates the scenario of using the Asynchronous mode based on the result of the crashes of one scenario. Here we have two cases; one is the Controller expected trajectory where, based on the speed of the ego vehicle, it follows trajectory A in time ndt, but in CARLA, due to the time difference, the ego continues at the same initial speed as the controller and based on the updating time of the CARLA it followed the trajectory B while the in the controller the trajectory A is already finished, after the second update of the CARLA, the vehicle jumps suddenly to the same point as in expected by controller. This causes instability, and if the other vehicle passes the same area intersecting the path of the Ego as indicated by the object below Dashed line A, due to the late update of CARLA mode, it causes the crash with the ego. This scenario non-logical increased the number of crashes in total scenarios. In Conclusion, For scenario-based testing, even if the scenario is robust, the setup between the ego controller and CARLA must be synchronized to prevent unrealistic results.

5.5 Cycling Time Analysis

In this section, another important factor is the effect of how many vehicles are present in the scenario and how these numbers can affect the cycling time of the ego controller. In other words, how can these numbers of vehicles add to the complexity of the scenario? These numbers have value when it comes to real-time testing of the system where if the cycling time of the vehicle is higher than a value, for example, for a good performance of a sensor is 0.1 second, so For assessing the scenario in scenario-based testing, also the controller must perform correctly, that the assessment is done correctly. Figure 5.13 gives a general trend of cycling time based on the number of vehicles where later, these number gives an overview of how the captured data from the scenario can affect the performance of the controller and how it can contribute to correct behavior of the ego vehicle in assessing the scenario in scenario-based testing.

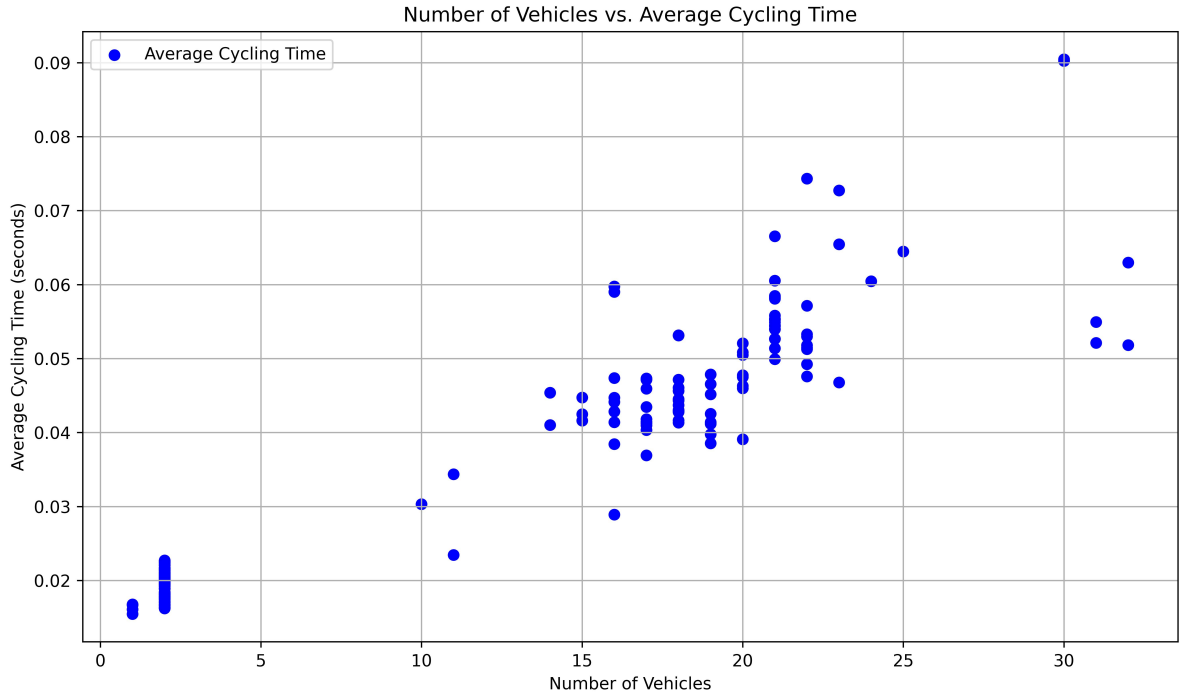


Figure 5.13: Cycling Time (S) - Number of Vehicles

Figure 5.13 illustrates the cycling time values for the ego vehicle's controller based on the number of vehicles. The plots demonstrate that the cycling time follows a polynomial curve, as shown in figure 5.13. Most data points cluster closely together, indicating consistent behavior under standard conditions. Nevertheless, some outliers exist where the cycling times diverge from this concentration. Such deviations may result from interactions with various road users that affect the controller's cycling time. The decision-making process of the controller relies on particular metrics and rules, which can influence the ego vehicle's response time in specific situations. Furthermore, in certain cases involving parked vehicles, these cars may not significantly impact the ego vehicle's cycling time, possibly leading to the observed variations in data.

Furthermore, in more detail, the cycling time plots of the three different modes mentioned in Paragraph 5.2. These values show the cycling time of the controller in limited interaction with other RUs where RUs are far from the Ego vehicle, which does not affect the controller, later in mode two and mode three where the parameters of the controllers are set to a value where there is interaction with other vehicles, which can result in values of the cycling time. These values can show how the presence of other RUs can affect the cycling time and show that the scenario is complex in terms of interaction with the ego and how they behave to represent the robustness of the scenario. Moreover, the trend of changes in cycling time can help to understand Figure 5.13.

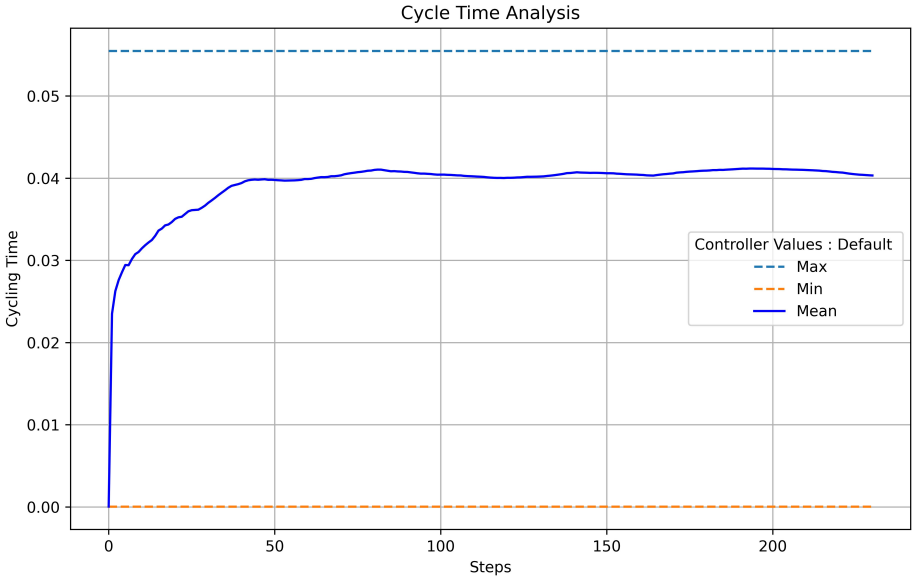


Figure 5.14: Controller Parameter Mode 1 Cycling Time

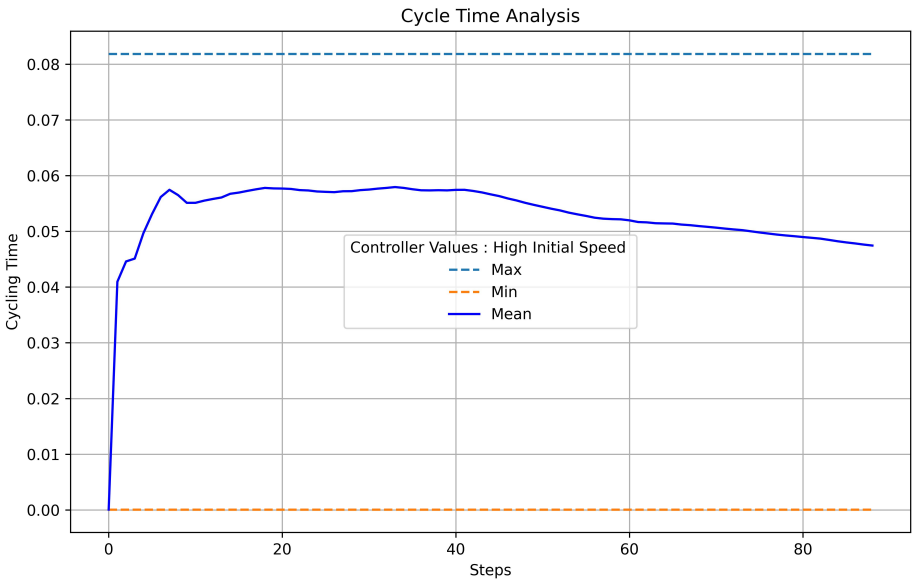


Figure 5.15: Controller Parameter Mode 3 Cycling Time

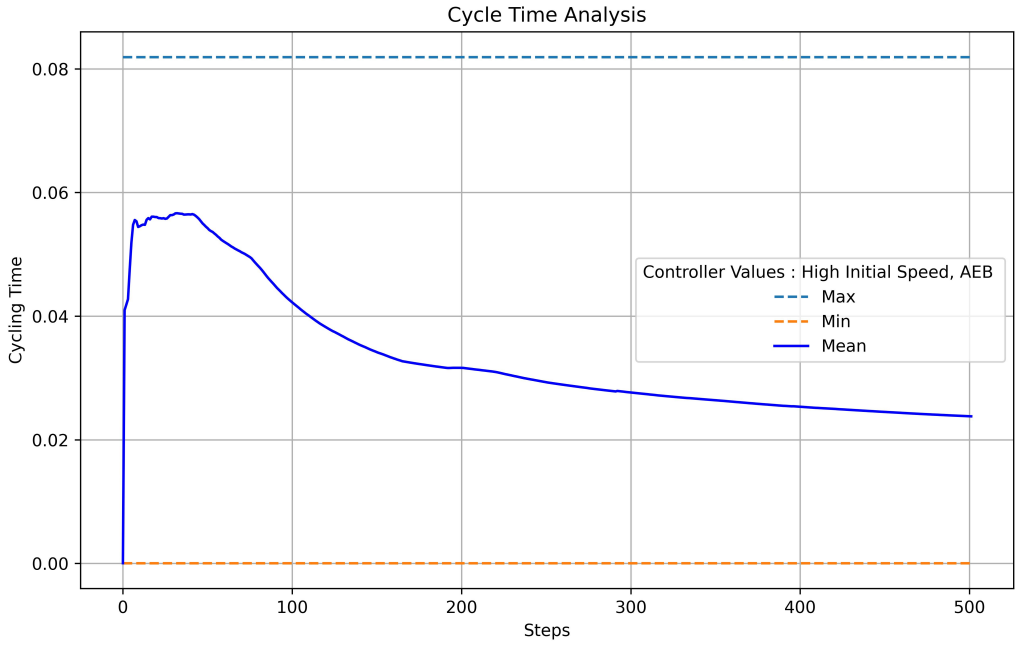


Figure 5.16: Controller Parameter Mode 2 Cycling Time

Figure 5.14, 5.16, 5.15 illustrate the cycling time values for the controller, highlighting how these values vary based on interactions with other vehicles. The cycling time analysis focuses on the Rec32/Combined/ARtS/IAC scenario, assessing three controller configurations: Mode 1, Mode 2, and Mode 3. Plot (a) illustrates the situation with minimal interaction between the ego vehicle and others, where cycling time remains constant at specific points due to the lack of significant interactions. In contrast, plot (b) shows a high-speed scenario, revealing a noticeable spike in cycling time around step 8. This increase occurs as the ego vehicle approaches the object vehicle, resulting in a higher cycling time. Plot (c) shows a similar jump in cycling time; however, due to the activation of AEB, the ego vehicle stops, resulting in a subsequent decrease in cycling time. Both plots (b) and (c), which involve higher interaction levels, demonstrate an overall increase in cycling time values compared to plot (a), and plot (c) represents that if there is no movement, the value decreases where it shows that in Figure 5.13 it might be the reason why in some values for the vehicle numbers the value of cycling time decreased with the fact that there were more RUs. As shown in Figure 5.13, the cycling time can reach up to 0.09 seconds depending on the type of scenario and the number of interacting vehicles. These findings emphasize the sensitivity of cycling time to both the level of vehicle interaction and the specific controller configurations employed. The max and min values indicated in each plot show the maximum or minimum value that the controller reached during the simulation. The trend of cycling time represents how the amount of complexity the ego vehicle is, and the maximum value shows the maximum value it took to simulate a step. In conclusion, the number of vehicles and how and for how long the Ego has interacted with other RUS lead to recognizing the correct controller parameters set for preventing any issues in scenario-based testing.

5.6 TTX Metrics

The analysis of Time to Collision (TTC) and Time Headway (THW) metrics is helpful in evaluating the robustness of scenario-based testing by quantifying criticality and interaction intensity across diverse traffic scenarios. These metrics provide a clear, measurable way to assess how dynamic and challenging a given scenario is, helping to identify edge cases and validate the performance of autonomous systems. By analyzing TTC, we capture the system's ability to respond to time-critical situations, while THW reveals the density and complexity of interactions between RUs. These metrics enable a deeper understanding of how well scenario-based testing covers the spectrum of real-world complexities by identifying which scenarios are robust enough to test the safety reliably.

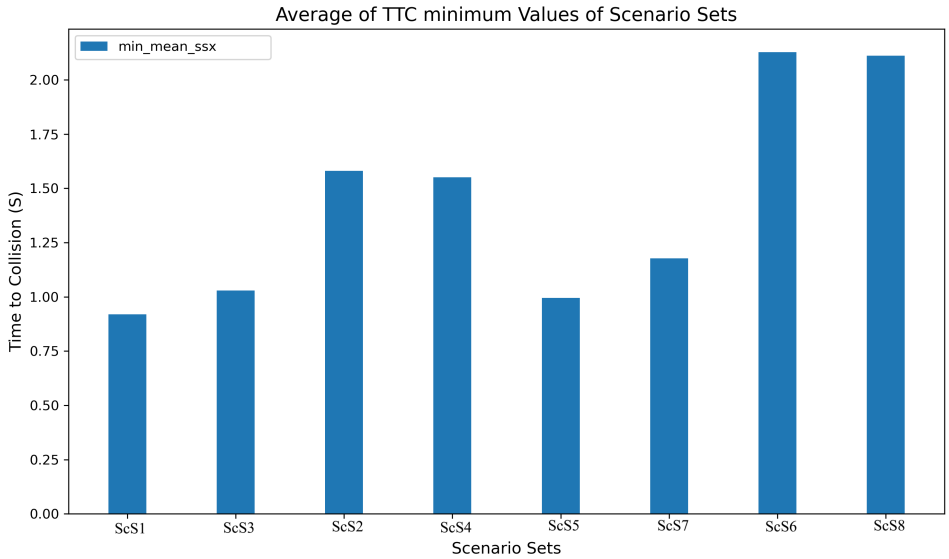


Figure 5.17: Average Min of TTC Values of Scenario Sets

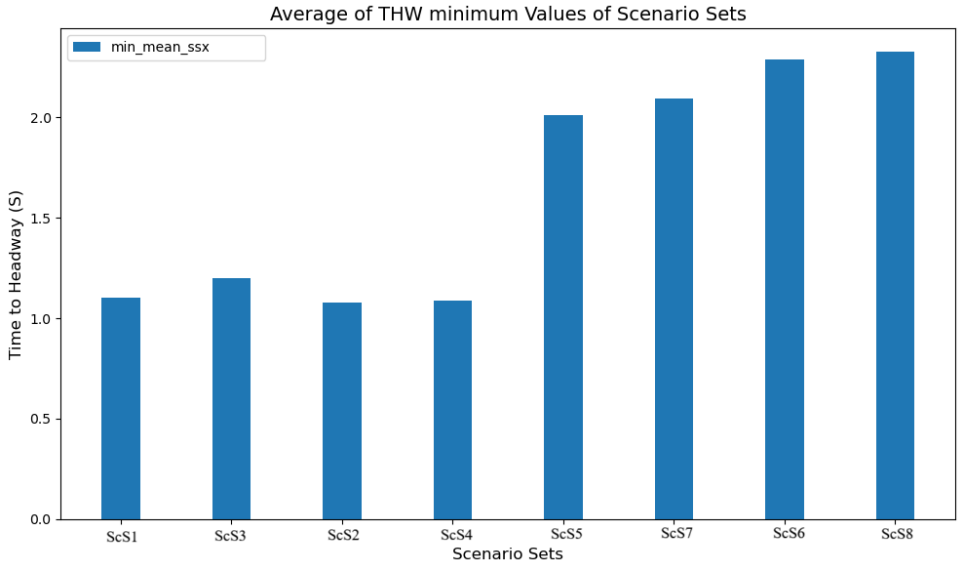


Figure 5.18: Average Min of THW Values of Scenario Sets

The ScS stands for Scenario Set, as described in Table 5.1. Figure 5.17 displays the average minimum TTC metric values for each scenario set. It is evident from Figure 5.17 that scenarios featuring the Intersection Assistance Controller (IAC) (ScS1, ScS3, ScS5, ScS7) have lower TTC values than those without the controller, indicating that the controller's presence leads to more complex and critical situations. Conversely, scenarios lacking the controller (ScS2, ScS4, ScS6, ScS8) tend to have TTC values exceeding 1.5 seconds, implying these situations are less urgent. Notably, Scenarios ScS1 and ScS3 display the lowest THW values, which indicates vehicle proximity and increased interaction density. These align with their low TTC values, reinforcing that these are high-complexity scenarios where vehicles frequently interact dynamically. Conversely, scenarios like ScS6 and ScS8 (Reduced-ARtS-NO-Controller and Reduced-RtS-NO-Controller) exhibit high THW values, indicating larger headway distances between vehicles. Combined with their high TTC values, these scenarios are simpler, involving fewer interactions and less criticality, which reflects lower complexity.

In conclusion, scenarios with low TTC and low THW values (ScS1, ScS3) are more complex because they feature high interaction density and critical conditions. On the other hand, scenarios with high TTC and high THW values (ScS6, ScS8) are less complex, as they represent simpler, less stressful situations. The lower TTC values (below 1.5 seconds) observed in some of the controlled scenarios can be attributed to the limited functionality of the controller, which occasionally failed, leading to critical situations. The most critical scenarios are ScS1 and ScS3, both of which are combined scenarios with the IAC controller. Additionally, Figure 5.18 illustrates the average THW metric values, showing that, in general, combined scenarios tend to have lower THW values compared to reduced scenarios. Interestingly, the presence or absence of the controller has approximately the same effect on the THW metric.

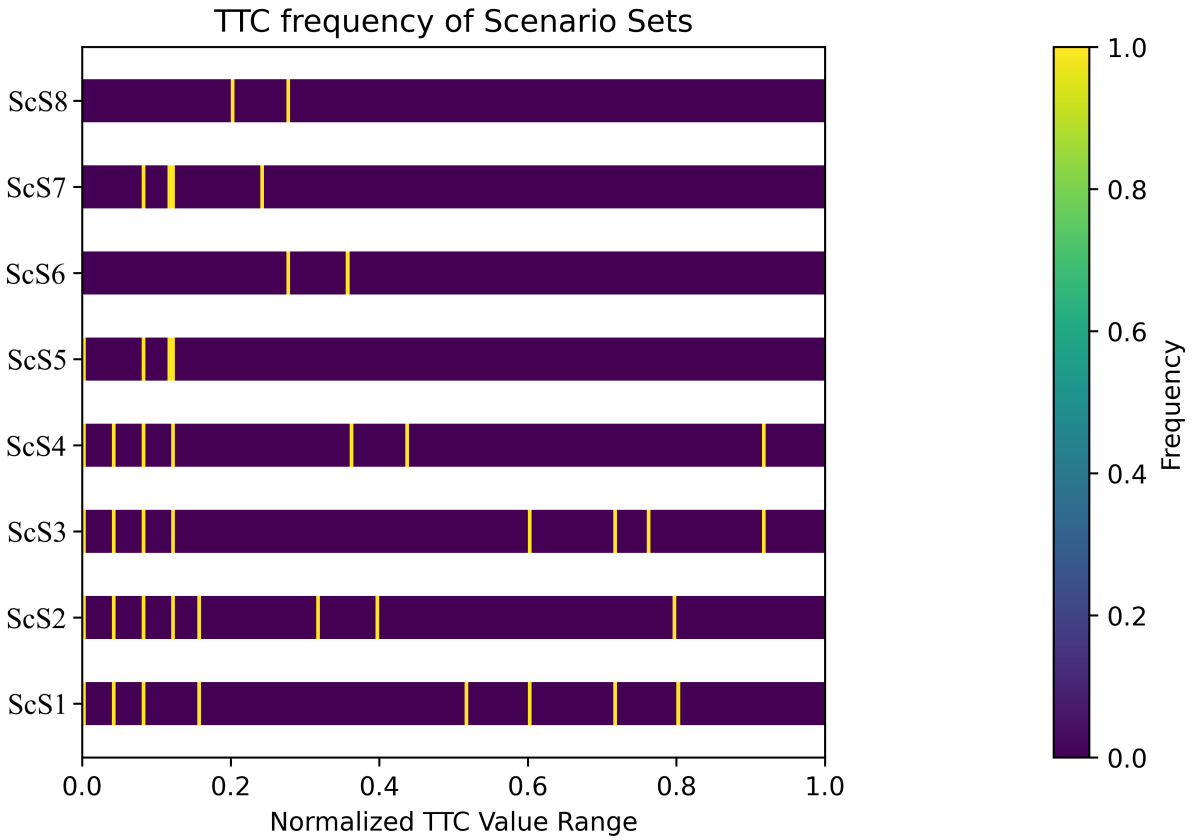


Figure 5.19: Frequency of Min TTC Values of Scenario Sets

Figure 5.19 presents the normalized minimum Time-to-Collision (TTC) values for each scenario set, ranging between 0 and 1. Lower TTC values indicate more critical conditions (shorter time to collision), while higher values correspond to safer scenarios. A high frequency of lower TTC values (closer to 0) suggests greater collision risks, whereas frequencies in the higher TTC range (closer to 1) represent less critical situations. Additionally, Figure 5.19 illustrates the frequency distribution of normalized minimum TTC values across different scenario sets. The combined scenarios (ScS1, ScS2, ScS3, ScS4) exhibit a higher frequency of low TTC values compared to the reduced scenarios, indicating greater risks due to shorter times to collision and increased interaction density. In contrast, the reduced scenarios display significantly lower frequencies in the critical TTC range. Most of these scenarios do not meet the critical TTC threshold (a normalized value of 0.2, equivalent to 1.5 seconds), highlighting their simpler nature with fewer complex interactions. Furthermore, the consistent trend across the combined scenarios suggests that critical factors—such as ARtS constraints and controller response times—are similarly propagated across these scenarios, contributing to lower TTC values and increased criticality. Notably, the vertical yellow line in Figure 5.19 represents only the minimum TTC values, not the higher values. As observed, there are more vertical lines in ScS1 to ScS4 (complex scenarios) compared to ScS5 to ScS8 (simpler scenarios), further illustrating the higher complexity and interaction density in the combined scenarios.

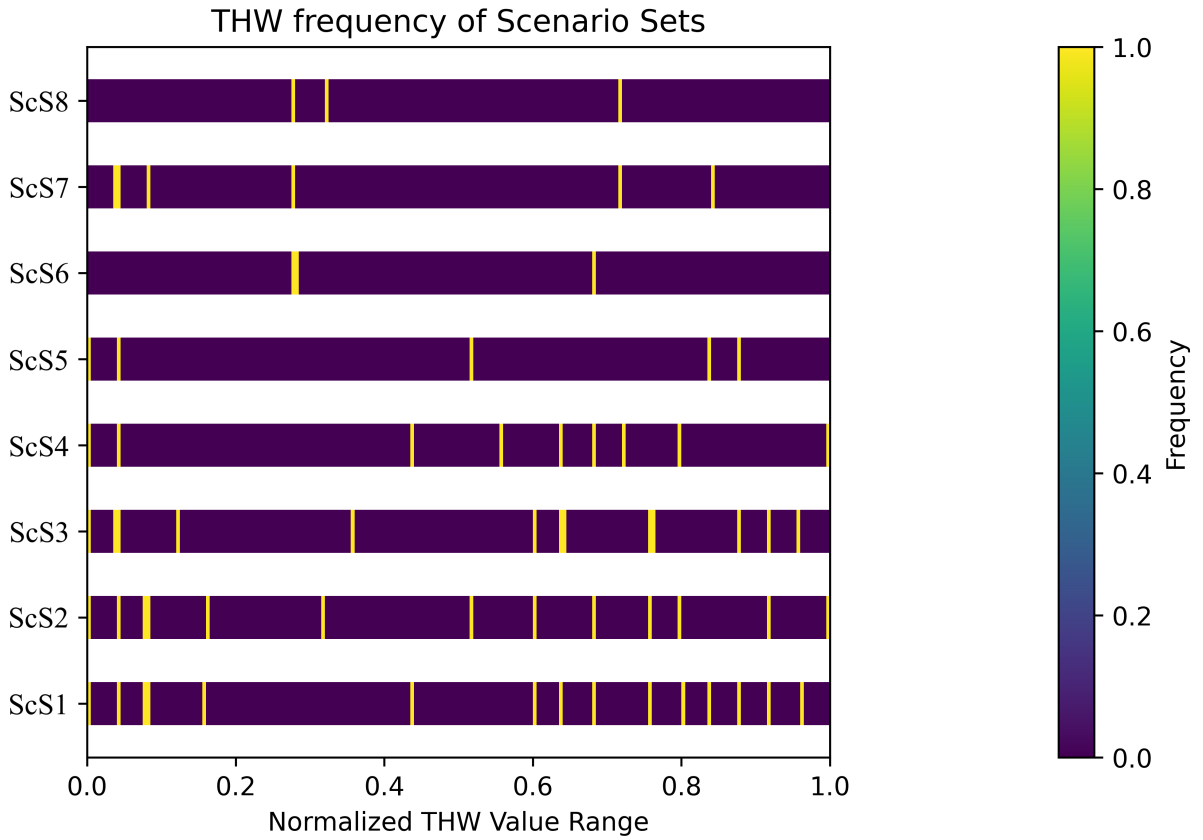


Figure 5.20: Frequency of Min THW Values of Scenario Sets

Figure 5.20 shows Scenarios with higher frequency concentrated in the lower THW range (closer to 0) reflect closely spaced vehicles with high interaction density. Scenarios with high frequency in the upper THW range (closer to 1) represent vehicles maintaining greater headway distances, indicating less interaction. Figure 5.20 shows the frequency distribution for normalized THW values, which appear shifted to the right compared to the TTC distribution. This reflects larger headway distances and less frequent critical interactions in reduced scenarios. However, for ARtS, a noticeable portion of THW values shifted from 0.05 to 0.1, demonstrating that the ARtS was effective in increasing headway and, by extension, reducing the likelihood of critical interactions. Despite this, combined scenarios still exhibit a higher frequency of low THW values, indicating dense interactions and higher complexity. These distributions reinforce the understanding that combined scenarios are more robust for testing, as they stress the system with both high interaction density and critical conditions. In contrast, reduced scenarios, with their simpler interaction patterns and lower criticality, are better suited for baseline system validation.

Chapter 6

Discussion

In this chapter, I evaluate how the implemented methodology addresses the existing research gap in generating complex traffic scenarios that are both robust and closely resemble real-world conditions. By analyzing the extent to which the derived results fulfill this gap, we aimed to demonstrate the effectiveness of our approach in producing robust scenarios for testing and validation purposes in scenario-based testing. The results presented in Chapter 5 demonstrate that the implemented methodology effectively generates complex scenarios. When excluding limitations related to input data parameters, such as the ARtS method and the handling of recordings within the envelope, the failure rate of scenarios drops below 5% or even approaches zero. This indicates that while the methodology produces complex scenarios, the robustness of these scenarios is influenced by the quality and robustness of the input data, including the ARtS. Our analysis began by calculating the percentage of crash cases in each scenario set, followed by a detailed failure case analysis. This analysis aimed to determine whether crashes were attributable to the methodology or input data limitations.

The failures were linked to the backward-forward behavior of the ego or object vehicles, which was a consequence of issues within the envelope data. In some instances, although the scenario elements reacted correctly, crashes occurred due to the limited flexibility of the ego vehicle controller for scenario-based testing. Still, failure in the controller can be neglected because it is the system under the test not the scenario under the test. This suggests that enhancements in the ego controller are necessary. Another significant finding was related to the sudden appearance of vehicles near the ego or object vehicle, which did not allow sufficient time for reaction by the ego or other ARtS objects. After identifying the logical reasons for these crashes, which were primarily based on input data limitations, I selected sample working scenarios to chal-

lenge them further. By modifying the control parameters of the ego controllers to make the scenarios more challenging, It was observed that the selected scenarios demonstrated robustness against these changes, indicating the methodology's potential for adaptability, but this still does not ensure as there is a need to test the methodology to challenging geographical and more city intersection to find out possible issues in scenario generation. Another critical discovery was the cycling time's impact on the scenarios' performance and robustness. It was found that setting the synchronization time to an asynchronous mode significantly affected the scenario's ability to facilitate reactions between the ego and object vehicles properly. The unrealistic vehicle movements caused by the cycling time discrepancy between the ego controller and CARLA environment resulted in artificial crashes. This highlights the significance of synchronization as a means of guaranteeing genuine interactions within the simulation. The results have significant implications for the creation and verification of autonomous driving systems. The ability of the methodology to produce intricate and effective scenarios indicates promise for use in scenario-based testing and validation. Improved quality of input data and simulation parameters of the generated scenarios will enable them to act as useful tools for verifying autonomous vehicles' functionality and reliability in simulated real-world environments. It can also help to enhance the development of autonomous driving technologies and safety features because it generates complex scenarios as well as edge or rare scenarios based on limited and real-world data. For the future work, it is suggested that the quality of the input data should be enhanced in order to address the limitations that were observed in the scenario generation process. Some of the recommendations for future work include improving the ARtS method, optimizing the envelope cutting procedure and the teleportation mechanisms within CARLA.

Additionally, setting up synchronization techniques between the ego controller and the simulation environment could mitigate issues caused by asynchronous cycling times. These improvements will strengthen the methodology and expand its applicability for scenario-based testing and validation. In conclusion, the implemented methodology shows significant promise in generating robust and complex traffic scenarios that closely resemble real-world conditions. By addressing the identified limitations and implementing the suggested improvements in future work, the methodology can become an even more valuable tool for testing and validating autonomous driving systems.

Chapter 7

Conclusion

This study aimed to enhance scenario-based testing methodologies by combining scenario elements to generate more complex and challenging environments for the ego vehicle. The results show that achieving intricate scenarios is closely tied to the robustness and quality of input datasets and parameters. Through the creation of these complicated scenarios I made the testing environment more demanding which contributed to the growth of scenario-based testing practices with the benefits of coming up with complex scenarios from a smaller amount of data as well as rare scenarios. This work highlights key findings, such as the need for an adaptive ARtS mode within the scenario-based testing frameworks. These findings are important for enhancing the accuracy and efficiency of scenario-based testing. The findings of this study are the method of producing comprehensive scenarios and determining important parameters such as CARLA simulator configurations, the envelope cutting method, and ARtS settings that affect the scenario-based testing performance. By understanding and optimizing these parameters, simulations can better reflect real-world complexities and ensure that testing scenarios are both challenging and representative.

Looking to the future, there are several promising directions for further improving scenario-based testing. The methodology is based on finding the similarity of the Ego Object base scenario and the following path of the Ego vehicle inherited from a similar scenario, and the new complex scenario is based on a new complex scenario where the Ego vehicle goes the same path but reacts differently in terms of vehicle reaction like braking, speed, and other parameters, then the Scenario under the test in scenario-based testing face challenges of the Ego vehicle behavior. In addition to the above-mentioned, scenario environment setting up plays an important role in the validity of the scenario-based testing, as the syncing between the con-

troller and the scenario under test would not cause an illogical crash which is due to the setup of the scenario, this is an important finding as not only the methodology but also they way it runs must be correctly defined to prevent illogical or confusion in the methodology and to prevent unreal simulations

Furthermore, to address the challenge of the path, for future works, I propose the Adaptive path with respect to the other RUs and the Ego vehicle and also the intersection geometry; this proposed work can bring flexibility in the combination process of different intersections and would be useful if there are high challenging scenarios like when the driver immediately turned over, and the reaction of the RUs are supposed to be tested in changing of their path for preventing the crashes. Moreover, Refining the envelope-cutting method to prevent undesirable object behaviors, enhancing ARtS flexibility, and preventing object spawning could improve scenario performance and realism. Adaptive methods allowing vehicles to adjust their trajectories to various road geometries, such as narrower roads or different intersection scales, would introduce further diversity and complexity into scenario-based testing.

In summary, this study makes a contribution to the field by presenting effective methods for creating complex testing scenarios, pinpointing critical performance factors, and proposing future improvements for scenario-based testing. These findings are important for developing safer and more efficient automated driving systems, as they support the establishment of robust and reliable scenario-based testing. Continued research and optimization in this area will be crucial for the advancement of autonomous driving technologies. This work highlights key findings, including the negative impact of using asynchronous mode in CARLA when employing controllers, in addition to the

Chapter 8

Appendix

8.1 Failure Case Analysis Description Table

Summary of crash reasons: Below, the summary of crash reasons is shown in letters A, B, C, D, and E. Moreover, in the scenario naming, inD stands for inD map, com for Combined or Complex Scenario, Red for Reduced or simple scenario, IAC for Intersection Assistant Controller, and NC for No Controller.

A: ARtS Limitation
B: Backward-Forward Behavior (Ego and/or object)
C: Sudden Appearance, Teleportation
D: Controller Limitation
E: RtS failure (if there is a failure with ARtS, that also happens in RtS)

Scenarios	Crash Reason
inD24/com/ARtS/IAC/env02	A
inD24/com/ARtS/IAC/env24	A
inD24/com/ARtS/NC/env02	C
inD24/com/ARtS/NC/env24	A
inD24/com/RtS/IAC/env02	E
inD24/com/RtS/IAC/env24	E
inD24/com/RtS/NC/env02	E
inD24/com/RtS/NC/env24	E

Scenarios	Crash Reason
inD32/com/ARtS/IAC/env04	B
inD32/com/ARtS/IAC/env21	A
inD32/com/ARtS/IAC/env33	D
inD32/com/ARtS/IAC/env38	C
inD32/com/ARtS/IAC/env49	A
inD32/com/ARtS/IAC/env62	C
inD32/com/ARt/NC/env04	B
inD32/com/ARtS/NC/env14	B
inD32/com/ARtS/NC/env21	E
inD32/com/ARtS/NC/env38	C
inD32/com/ARtS/NC/env54	C
inD32/com/ARtS/NC/env60	A
inD32/com/RtS/IAC/env04	B
inD32/com/RtS/IAC/env21	A
inD32/com/RtS/IAC/env33	D
inD32/com/RtS/IAC/env38	C
inD32/com/RtS/IAC/env49	A
inD32/com/RtS/IAC/env62	C
inD32/com/RtS/NC/env04	B
inD32/com/RtS/NC/env14	B
inD32/com/RtS/NC/env21	E
inD32/com/RtS/NC/env38	C
inD32/com/RtS/NC/env54	C
inD32/com/RtS/NC/env60	E
inD32/red/ARtS/IAC/env04	A
inD32/red/RtS/IAC/env04	E

Table 8.1: Table of Scenarios with Crashes

8.2 Controller Properties

Listing 8.1: Controller Properties, Mode 1

```
<Properties>
            <Property name="module" value="intersection_assistent_control.
                py" />
            <Property name="config_initial_speed" value="7.0" />
            <Property name="config_max_speed" value="20.0" />
            <Property name="config_target_speed" value="7.0" />
            <Property name="config_enable_aeb" value="False" />
</Properties>
```

Listing 8.2: Controller Properties, Mode 2

```
<Properties>
            <Property name="module" value="intersection_assistent_control.
                py" />
            <Property name="config_initial_speed" value="25.0" />
            <Property name="config_max_speed" value="20.0" />
            <Property name="config_target_speed" value="7.0" />
            <Property name="config_enable_aeb" value="True" />
</Properties>
```

Listing 8.3: Controller Properties, Mode 3

```
<Properties>
            <Property name="module" value="intersection_assistent_control.
                py" />
            <Property name="config_initial_speed" value="25.0" />
            <Property name="config_max_speed" value="20.0" />
            <Property name="config_target_speed" value="7.0" />
            <Property name="config_enable_aeb" value="False" />
</Properties>
```

Bibliography

[1] Synchrony and time-step. https://carla.readthedocs.io/en/latest/adv_synchrony_timestep/.

[2] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections, 2019.

[3] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934, 2020.

[4] Michael Schuldes ika Christoph Glasmacher, ika – RWTH Aachen. *Scenario-based Model of the ODD through Scenario Databases*. vvm-projekt, 2023.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[6] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, November 2019.

[7] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955, 2019.

[8] Fin Malte Heuer. Scenario generation for testing of automated driving functions based on real data. Master's thesis, Technical University of Braunschweig, Institut für Softwaretechnik, 2022.

[9] Nidhi Kalra and Susan Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 12 2016.

[10] Anshul Khandelwal. Development of a framework for similarity analysis of urban driving scenarios. Master's thesis, RWTH Aachen University, Institute for Automotive Engineering(IKA), Aachen, Germany, 2022.

[11] Philip Koopman and Michael Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[12] Tjark Koopmann Lukas Westhofen, Christian Neurohr. Criticality metrics for automated driving: A review and suitability analysis of the state of the art. *arXiv https://arxiv.org/abs/2108.02403v2*, 2021.

[13] Michael Schuldes Maike Scholtes. Omegaformat: A comprehensive format of traffic recordings for scenario extraction. 12, 2022.

[14] SAE International. SAE Updates J3016 Automated Driving Graphic. https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic, 2019. Accessed: 2024-10-08.

[15] Maike Scholtes, Lukas Westhofen, Lara Ruth Turner, Katrin Lotto, Michael Schuldes, Hendrik Weber, Nicolas Wagener, Christian Neurohr, Martin Bollmann, Franziska Körtke, Johannes Hiller, Michael Hoss, Julian Bock, and Lutz Eckstein. 6-layer model for a structured description and categorization of urban traffic and environment, 2021.

[16] Michael Schuldes, Christoph Glasmacher, and Lutz Eckstein. scenario.center: Methods from real-world data to a scenario database, 2024.

[17] Zhi Tu Xi Zheng Mengshi Zhang Tianyi Zhang Yao Deng, Jiaohong Yao. Automated scenario generation from traffic rules for testing autonomous vehicles. *arXiv https://arxiv.org/abs/2305.06018*, 2023.