Exploration and Cluster-Based Analysis of Simulated Logical Scenarios for Testing Automated Driving Systems

Zur Erlangung des akademischen Grades einer

DOKTORIN DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für
Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Barbara Schütt geb. Konz
geboren am 20. September 1986 in Regensburg

Tag der mündlichen Prüfung:	07.11.2024
Hauptreferent:	Prof. Dr.-Ing. Eric Sax
Korreferent:	PD Dr. Hardi Hungar

Kurzfassung

Mobilitätssysteme der Zukunft werden aufgrund der zunehmenden Verstädterung mit immer zahlreicheren Herausforderungen konfrontiert werden, die Verkehrssysteme an ihre Grenzen bringen können. Infolgedessen besteht die Gefahr, dass Staus und Unfälle im Verkehr in Zukunft zunehmen. Assistierte und automatisierte Fahrsysteme haben das Potenzial, die Mobilität und die Effizienz zu steigern, die Umweltfreundlichkeit zu erhöhen und die Sicherheit zu verbessern. Die Einführung solcher Systeme führt jedoch zu einem höheren Maß an Komplexität und Vernetzung der Funktionen im Fahrzeug. Vor allem die Gewährleistung der Sicherheit wird zu einem entscheidenden Faktor für die erfolgreiche Integration und Akzeptanz dieser Systeme in der Gesellschaft.

Um die Sicherheitsanforderungen zu erfüllen, die erforderlichen Fähigkeiten für verschiedene Anwendungsfälle nachzuweisen und Risiken zu minimieren, sind gründliche Tests auf verschiedenen Ebenen unabdingbar. Während gegenwärtig Testfahrten unter realen Bedingungen durchgeführt werden, um das sichere Verhalten von Fahrzeugen mit Fahrassistenzsystemen zu validieren, werden ab einem höheren Automatisierungsgrad abstandsbasierte Testansätze (Testfahrten) zunehmend unpraktikabel. In diesem Kontext gewinnt der szenario-basierte Testansatz an Bedeutung, da er effizient das Verhalten der Fahrzeuge in vielfältigen Situationen untersuchen und bewerten kann. Die Identifizierung und Zusammenstellung geeigneter und relevanter Szenarien für das Testen ist dabei von entscheidender Bedeutung.

Die vorliegende Arbeit untersucht zwei logische Szenarien mittels Szenarienexploration und analysiert die Ergebnisse. Dabei werden zunächst die Einflussfaktoren untersucht, die die Qualität und Kritikalität eines Szenarios bestimmen. Es wird eine Kritikalitätsmetrik eingeführt, um Szenen entsprechend ihrer Kritikalität zu bewerten. Bayes Optimierung wird verwendet, um logische Szenarien effektiv nach kritischen Bereichen zu durchsuchen, ohne eine gitter-basierte Simulation durchführen zu müssen. Die Ergebnisse können daraufhin aufgrund der Ähnlichkeit verschiedener Metriken geclustert und inhaltlich zu einer reduzierten Anzahl von proto- und archetypischen Szenarien zusammengefasst werden. Schließlich kann das während der Explorationsphase trainierte Vorhersagemodell genutzt werden, um Aussagen über die Konfidenz und Sicherheit der Ergebnisse bezüglich der Kritikalität zu treffen. Diese Methodik zielt darauf ab, das Verständnis für Vorgänge in logische Szenarien durch Exploration und Auswertung zu verbessern

und Szenarienkataloge für umfassende Test erstellen zu können. Weiterhin stellt die Auswertung der Ergebnisse eines logischen Szenarios einen wichtigen Schritt für die Coverageaussage eines solchen Szenarios dar.

Abstract

The future of mobility systems will face many challenges as urbanization and the number of registered vehicles increase, potentially pushing transportation systems to their limits. Assisted and automated driving systems have the potential to provide greater mobility for all, driving more efficiently, environmentally friendly, and safely. However, the introduction of such systems leads to increased complexity and interconnectivity of vehicle functions. Safety is a critical factor for the successful integration and acceptance of these systems in society. Thorough testing of vehicles at different stages of development is essential to meet requirements, demonstrate the necessary capabilities for different use cases, and minimize risks. Currently, test drives under real-world conditions are used to validate the safe behavior of vehicles with driver assistance systems. However, at higher levels of automation (e.g. SAE Level 4 and Level 5), the distance-based testing approach becomes less feasible due to the excessive number of test kilometers required to ensure safety. Scenario-based testing is gaining recognition as an efficient way to investigate and evaluate vehicle behavior in different situations. It is necessary to identify and group suitable and relevant scenarios for testing. Scenarios can be obtained in a variety of ways: from real data, for example, or created by experts. However, especially with data-driven approaches, there is a risk that many scenarios are very similar and show little difference, especially if they are assigned to the same logical scenario. This thesis investigates the results of scenario exploration of a logical scenario in the context of scenario-based testing.

This thesis examines two logical scenarios using scenario exploration and analyzes the results. To achieve this, the first step is to clarify what defines quality and criticality in logical or concrete scenarios. Then, a criticality metric is introduced that allows a comprehensive evaluation of scenes, inspired by the traffic quality assessment used for highways. Bayesian optimization is used to effectively search the example scenarios for critical areas without having to run a grid-based simulation. The results can then be clustered based on the similarity of different metrics and summarized into a reduced number of proto- and archetypal scenarios. Finally, the predictive model trained during the exploration phase can be used to make statements about the confidence and certainty of the results with respect to criticality. This methodology aims to improve the understanding of processes in logical scenarios through exploration and evaluation, and to be able to create scenario catalogs for comprehensive testing. In addition, evaluating the results of a logical scenario is an important step in determining the coverage of such a scenario.

Contents

Acronyms and symbols

Acronyms

CoK	Cohen's Kappa
CSI	Critical Scenario Identification
DSL	Domain Specific Language
ET	Encroachment Time
F1S	F_1-Score
FN	False Negatives
FNR	False Negative Rate
FP	False Positives
FPR	False Positive Rate
KPI	Key Performance Indicator
MCC	Matthews Correlation Coefficient
MR	Miss-Classification Rate
NPC	Non Player Character
RSS	Residual Sum of Squares
SSM	Surrogate Safety Measure
SiL	Software in Loop
TCT	Traffic Conflict Technique
TN	True Negatives
TNR	True Negative Rate

$\mathbf{TP}$	True Positives
$\mathbf{TPR}$	True Positive Rate
$\mathbf{TTC}$	Time-To-Collision
$\mathbf{TQ}$	Traffic Quality

ADAS Advanced Driver-Assistance Systems

ADS Automated Driving System

BO Bayesian optimization

DBSCAN Density Based Spatial Clustering of Applications with Noise

DTW Dynamic Time Warping

EA Evolutionary Algorithm

GP Gaussian process

HAD Highly Automated Driving

IDM Intelligent Driver Model

ODD Operational Design Domain

OSI Open Simulation Interface

PCA Principal Component Analysis

PET Post-Encroachment Time

TTC Potential Time-To-Collision

RBF Radial Basis Function

ROS Robot Operating System

SUT System Under Test

TTC Time-To-Collision

TDP Traffic Density Potential

WTTC Worst Time-To-Collision

CCP Coupon Collector's Problem

Constants

Latin symbols and variables

Greek symbols and variables

Operators and math symbols

General deep indexes

1 Introduction

Assisted and automated driving is one of today's key technologies for a future transport and mobility transition. The advantages and disadvantages of these technologies are intensely debated. They promise mobility for everyone and a reduction in accidents, particularly those caused by human error. However, there is potential for new types of accidents, such as individuals attempting to exploit or bypass safety boundaries. In 2019, Tesla announced plans to launch one million robot-taxis by 2020 [111]. However, they have since revised this statement multiple times. The roboticist Rodney Brooks [127] predicted a more conservative outcome, estimating that in 2022, there would be limited taxi services with dedicated pick-up and drop-off points and severe environmental restrictions. In contrast, there might even be a ban on parking and human-driven cars in downtown areas of major US cities in 2045 [127]. With that in mind, autonomous driving has the potential to significantly alter urban life and the cityscape in the long term. However, it is important to address the issue of fulfilling all necessary safety requirements for autonomous cars to drive among other traffic participants. It is crucial to ensure that safety requirements are met before autonomous cars are on the road.

1.1 Motivation

In general, is essential to ensure and validate the safe behavior of vehicles equipped with Advanced Driver-Assistance Systems (ADAS) or Automated Driving System (ADS) before their release. To achieve reliability, the automated driving system and the vehicle must undergo thorough testing, verification, and validation at every step of the development process. In current research, scenario generation (cf. Section 2.3) is used to collect different scenario sets and to create scenario catalogs for this purpose. One example is scenario exploration [34], where scenario parameters are varied to find critical situations. However, the results of the exploration are typically not assessed for their usefulness in the test set. Frequently, generated scenarios are related or identical and do not provide any additional value for testing. Additionally, the required kilometers for test drives are no longer feasible [148]. Therefore, scenario-based verification and validation play a crucial role in the future as they constitute an integral part to assure requirements are met, the system has the necessary capabilities in all intended use cases, and unreasonable risk is avoided [157, p. 6ff.].

1.1.1 Legal Aspects of Assisted and Automated Driving Systems

On 6 July 2019, EU Regulation No. 2019/2144 [41] came into effect and is now mandatory to be followed in all EU member states. This regulation amends the previously passed Regulation 2018/858 [40], which was released a year earlier. Regulation 2019/2144 affects the type-approval of automated driving systems, their additional components, and systems related to the general safety of vehicles, as well as the protection of vehicle occupants and vulnerable road users. Therefore, automated driving systems must undergo testing and certification before being sold or used on public roads.

A key objective of this regulation is to decrease the number of fatalities and injuries through the obligatory introduction of vehicle safety systems. The regulation requires the introduction of several new technologies and safety measures on a mandatory basis, according to a defined schedule. The detailed technical requirements for each safety measure are currently being developed at the EU and UNECE levels, or are already implemented in some cases. For instance, it states that newly released motor vehicles shall be equipped with intelligent speed assistance, advanced driver distraction warnings, or event data recording. Furthermore, this regulation references technical specifications from implementing acts that automated vehicles and fully automated vehicles must comply with. However, this part of the regulation is currently under work and needs further research and refinement.

In August 2022, a new implementation regulation [42] regarding EU Regulation No. 2019/2144 was introduced. This regulation lists a set of methods in Annex 3 for the overall compliance assessment of an automated driving system. Part 1 of this annex specifies that a minimum set of traffic scenarios should be used in case these scenarios are relevant to the automated driving system's operational design domain (ODD). The regulation also defines parameters and thresholds for the Time-To-Collision (TTC), which is used as a safety performance metric. It determines the threshold for collision avoidance with pedestrians and cyclists and defines rule sets for different driving environments, such as urban and rural traffic or motorway entry.

The ISO/PAS 21448 "Road vehicles - Safety of the intended functionality" (SOTIF) [141] provides a standardized safety assessment framework. It recommends evaluating the driving system and its components for unknown hazardous scenarios during the late testing phase, located in the upper right arm of the V-model. Another norm, the ISO 26262 "Road vehicles - Functional Safety" [79], is a functional safety standard for electrical and electronic systems installed in mass-produced road vehicles. However, there is currently no legal obligation to apply these two norms.

1.1.2 Amount of Necessary Testing

First, it is necessary to determine the level of automation of the vehicle or driving system to determine the required testing. The SAE Standard J3016 (2021) [129] outlines six levels of automation. Wood et al. [157] state that each layer of a driving system requires specific verification and validation:

L0 **No Driving Automation:** Human driver is responsible for the entire driving task.

L1 **Driver Assistance:** Lateral or longitudinal driving support is provided to a human driver, who remains responsible for the primary driving task. Typical example functions for L1 include lane-keeping assistance (LKA) or active cruise control (ACC).

L2 **Partial Driving Automation:** A combination of lateral and longitudinal driving support, where the human driver either completes a driving task or supervises the driving system. An example is Tesla's Autopilot.

L3 **Conditional Driving Automation:** Defined tasks can be achieved by an automated driving system (ADS) without human support. However, a human driver is always available to intervene or take over. Examples for L3 are the traffic jam pilot (TJP) or highway pilot (HWP) [157].

L4 **High Driving Automation:** The ADS can perform predefined driving tasks without human support or fallback. It can reach a minimal risk condition if the driver is not available. Examples for L4 are the urban pilot (UP) for fleet operations in urban areas or the car park pilot (CPP) [157].

L5 **Full Driving Automation:** The ADS performs the entire driving task without the need for human supervision.

This work focuses on level 4 and 5 autonomous driving systems in urban scenarios. These scenarios present a wide range of possibilities due to the presence of different types of traffic participants, such as pedestrians, bicycles, and objects like baby strollers, as well as a variety of street layouts that may vary from town to town and between countries and continents.

Currently, real-world test drives are used in the distance-based test approach to ensure the safe behavior of assisted driving systems [148]. However, this approach is not feasible anymore at a higher degree of automation. For example, for L4 or L5 functions, almost 8 billion kilometers are necessary to make a statistically significant comparison to show that the failure rate of an automated vehicle is lower than the human driver failure rate. Additionally, any changes or variations in the

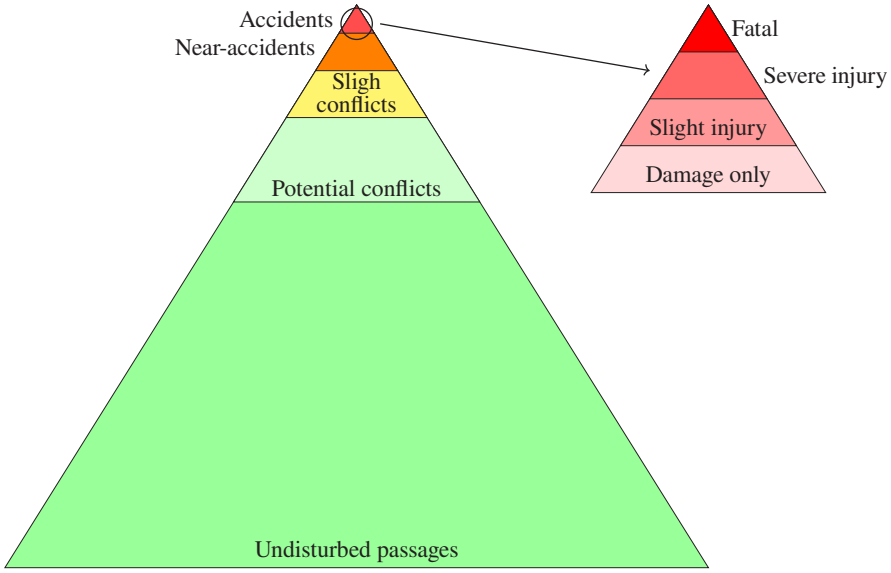


Figure 1.1: Safety Pyramid: representation of the severity of all interactions between two traffic participants [77].

automated driving system necessitate repeating all testing [80, 148]. Furthermore, reproducing test results generated from real-world test drives is both difficult and expensive.

An intuitive idea of why so many kilometers are needed is given by Hydén [77]. He developed the Swedish Traffic Conflicts Technique, a method for observing traffic conflicts. This technique classifies interactions between road users into different events, depicted in the safety pyramid (also known as the conflict pyramid) shown in Figure 1.1. It is important to note that this information is based on objective observations and classifications of traffic interactions. The sizes of the different pyramid layers do not accurately represent the relationships between the number of different conflict types. However, they clearly show that the more severe an encounter type is, the less frequently it occurs. The aforementioned vast number of kilometers is required to cover all possible encounters statistically.

1.1.3 Challenges and Scenario Coverage During Testing

Scenario-based testing offers an alternative or supplementary method to random test cases that arise during real-world test drives. This approach involves systematically deriving and testing new and relevant scenarios at different stages of development and testing [119, 102]. The objective is to set up a collection of critical or relevant scenarios, depending on the test object, its objectives, and its requirements. These scenarios should be abstract during the concept phase and

become more detailed and concrete throughout the development and testing process. As shown in Section 1.1.1 and Section 1.1.2, scenario-based testing is now mandatory for new ADS and ADAS and the necessity for getting test scenarios is growing with each driving function. However, the question arises of where these test scenarios come from. Scenarios must be created or collected meticulously, but only some scenarios might provide new insights. Additionally, a large set of scenarios does not necessarily represent a valid basis for testing, as the set size alone does not indicate quality. Promising methods for finding new and relevant scenarios or building scenario databases include scenario extraction, generation, or expert-based creation. Scenario coverage metrics and criticality metrics are crucial for selecting or generating desired scenario types for these methods. These metrics must be identified for each considered use case during the scenario acquisition process and depend on the System Under Test (SUT).

This work focuses on scenario exploration as one of several methods for scenario acquisition. Scenario exploration identifies critical scenario parameters within a defined parameter space. However, current research has not yet evaluated the similarity of the scenarios found within this space. Scenario exploration can result in critical and non-critical scenario parameter sets. Nonetheless, these sets are only useful for the testing process if they provide new insights and do not lead to the same scenario multiple times. For instance, if the ego vehicle approaches an intersection from a distance and a pedestrian is waiting before crossing the street, it is equivalent to the ego vehicle starting near the intersection and the pedestrian not waiting to cross. The objective of this thesis is to identify not only critical, relevant, or interesting scenarios but also to make assumptions about the found scenarios and their usefulness.

This leads to three main challenges:

(1) All relevant scenarios and, thus, all potentially important situations for a driving function from its Operational Design Domain (ODD) have to be derived and addressed.

(2) The coverage of the scenario space must identify all critical situations regarding the system under test for sufficient test case generation.

(3) The chosen quality or criticality metric must be appropriate to address the question at hand. In this thesis: Does the metric adequately measure the criticality of traffic scenes?

Throughout this thesis, these challenges will be addressed and refined. To this end, a list of research questions (Section 1.2) is formulated to identify the issues that need to be addressed. From these questions, a list of requirements (Section 4.2) for a more formal goal definition is derived.

1.2 Research Questions and Objectives

The first set of research questions focuses on scenario evaluation. This is a crucial step in determining the quality and informational value of a scenario and, therefore, must be established before scenarios can be explored or generated.

RQ1.1: What is the definition of scenario quality or criticality, and how does quality or criticality differ and resemble known criticality metrics?

RQ1.2: Are different levels of abstraction of criticality required during the development and testing process?

RQ1.3: What metrics are appropriate for providing a comprehensive assessment of the criticality of a scene or scenario?

RQ1.1 and RQ1.2 are addressed in Section 4.3 and RQ1.3 in Section 4.8. Both parts serve as supporting aspects for the following research questions and definition of the field where the research of this thesis is done. A second part of this work focuses on scenario exploration, which leads to the following set of research questions:

RQ2.1: What algorithms can be found and used for a fast scenario exploration to find critical parameter sets for logical scenarios?

RQ2.2: What influence does the criticality metric have on the results of the scenario exploration and evaluation?

RQ2.3: How can further analysis help to reduce and summarize the set of found concrete scenarios?

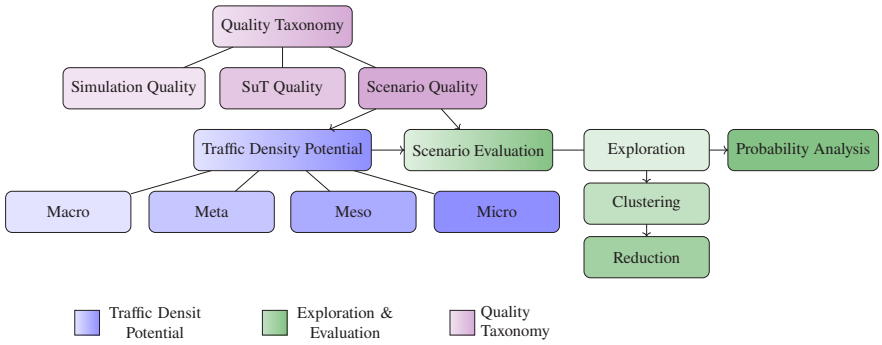


Figure 1.2: Areas of interest of this work.

The research questions and areas of interest presented in this thesis are divided into three main categories (see Figure 1.2):

1. Scenario quality and criticality (purple)

2. Scenario exploration and result evaluation (green)
3. Criticality metric for urban traffic scenes

In Figure 1.2, all categories with related sub-categories are shown with their relationships among each other. The gradients are used for the main categories and include all the shades of the subcategories. All three areas are addressed in this work.

1.3 Structure of this Work

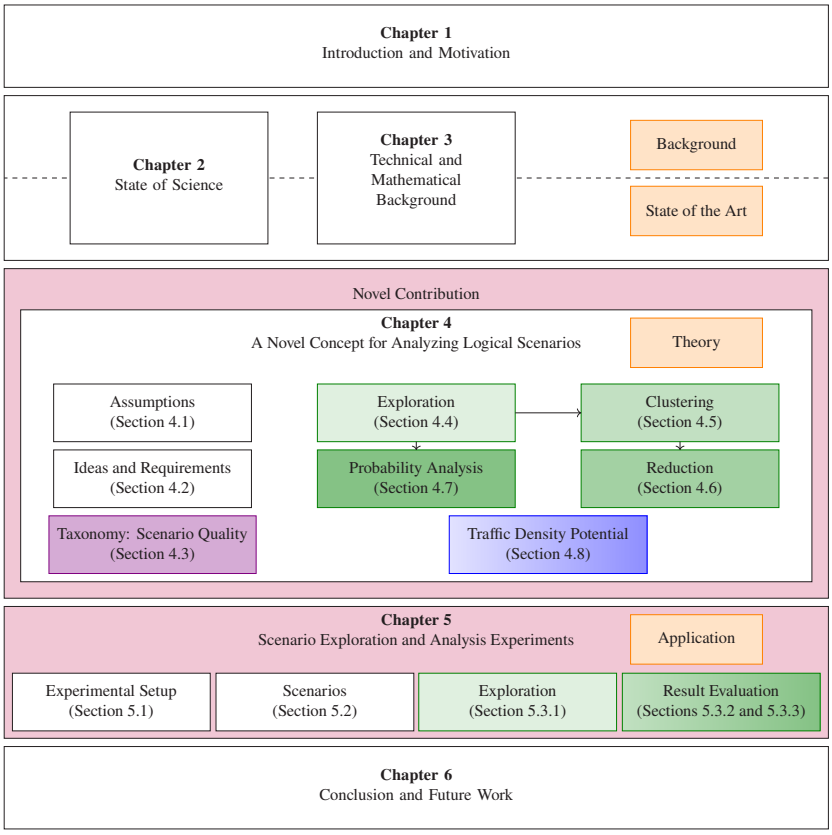


Figure 1.3: Structure of this work.

The structure of this work is shown in Figure 1.3. It shows the same areas of interest located within the chapters of this work and uses the same color coding as Figure 1.2 for the different

steps and topics. After an introduction, the requirements and research questions leading to this work are given.

Chapter 2 gives a brief insight into relevant current research. It provides a concise overview of current research on scenario-based testing and its development process. It begins with an introduction to the topic and a brief explanation of important terms used in scenario-based testing. This includes definitions of established terms, scenario abstraction levels, scenario description layers and languages, simulation environments, and criticality analysis. After that, criticality metrics and existing metric systematics are summarized. It distinguishes between constellation-based and environmental criticality metrics and gives examples for all categories.

The final section of this chapter provides a brief overview of current methods for scenario acquisition, which can be used to expand scenario sets for development and testing. The chapter classifies acquisition methods into six categories and provides examples of representative methods for each category. It also demonstrates how these methods can be combined.

Chapter 3 provides the technical background relevant to this work. It covers several scenario description languages, simulation environments, and resolution levels of traffic simulation. The second part of this chapter introduces the mathematical background used for scenario exploration and analysis.

The core of this work starts in Chapter 4, where the concept of this work is presented. Section 4.1 outlines the assumptions for the subsequent experiments, including those related to the ego vehicle and scenario exploration conditions. Following this, the conceptual idea and requirements are presented.

Section 4.3 presents a new taxonomy for quality and criticality metrics in various stages of the simulation process. This taxonomy bridges the gap between the importance of a driving function and the assessment of scenarios. Prior to this work, scenario quality was seldom considered. The objective of scenario quality is to enhance the understanding of quality in the development and testing process and to provide clear separation and insight into where further testing is required. Furthermore, it helps to understand quality in the development and testing process and to have a clear separation and insight into where further testing is needed. In particular, a focus is set on scenario metrics, as many commonly used metrics, e.g., time-to-collision, can only evaluate small parts of a scenario or scene and tend to neglect several aspects of a scenario, e.g., actors with unpredictable movement like children or animals. Sections 4.4 to 4.7 show the concept of scenario exploration, the optimization algorithm, and its utilization. This is followed by an illustration of scenario clustering using similarity and dimensionality reduction. An archetypal analysis for scenario set reduction is proposed, and the uncertainty calculation of the results is discussed. To address the issue of evaluating the criticality of scenarios involving more than two

cars, including the ego vehicle, a new metric based on the traffic quality proposed by Hallerbach et al. [70] is presented in Section 2.2.5. This metric can assess an entire scene surrounding the ego vehicle and applies to urban traffic. On this basis, a novel method is employed to calculate a comprehensive score for all submetrics without requiring the learning of weights for each new scenario or map. Chapter 5 introduces the experimental setup. Section 5.1 presents the scenario exploration experiment setup, which serves as the structure between the simulation environment and optimization algorithm. This setup uses optimization algorithms to find critical concrete scenarios within a logical scenario without simulating all possible parameter combinations. In Section 5.2, the layouts of the two experimental logical scenarios are shown. The first scenario includes two intersection situations with different structural setups divided into scenarios 1A and 1B. Scenario 1 contains various trajectory parameterizations, whereas scenario 2 parameterizes synchronization points. The simulation environment differs between both scenarios, allowing different ego and actor behaviors. Finally, Section 5.3 discusses the clustering, set reduction, and probability results. Chapter 6, concludes this work addressing the given research questions and gives an outlook on future work.

2 State of Science

2.1 Scenario-based Testing

Quality assurance is a crucial aspect of the development process. Simulation and scenario-based testing are effective methods for ensuring quality and should be considered from the beginning. This work proposes scenario exploration and result evaluation as a step in scenario-based testing during the phase where scenarios are collected or generated. However, scenario-based testing must still be integrated into the current development process for automotive systems. The V-model, as shown in Figure 2.2, represents a well-established development process [130] that includes scenario abstraction levels (Section 2.1.3) at various stages [30]. The left part of the V describes the top-down design process that follows the concept phase in contrast to the right part, which describes a bottom-up verification and validation process. During this process, scenarios are derived at the beginning and get substantiated in later stages. Simulation can be used at any stage of function development, before building and testing vehicle prototypes. According to Figure 2.2, model- and software-in-the-loop tests can be used to verify and validate initial ideas, design, and concepts of the ADS. In this phase of testing, every part is described by a simulation model, including sensors and vehicle parts. Later on, simulation models can be replaced by hardware or even a test vehicle, leading to hardware- and vehicle-in-the-loop tests [148].

2.1.1 X-in-the-Loop Methods

As illustrated in Figure 2.2, the V-model process comprises several in-the-loop steps, also refereed to as X-in-the-loop (XiL). These methods are coupled with the development process, starting with a more abstract procedure called model-in-the-loop (MiL). As more software and hardware components become available during the development process, the abstraction is replaced by concrete models in software-in-the-loop (SiL), hardware-in-the-loop (HiL), and vehicle-in-the-loop (ViL) [156, 130].

The model-in-the-loop method allowes for the evaluation of requirement analysis and software design. It involves designing a software model of the desired behavior or algorithm based on its

functional goals and testing it interactively in a simulation environment that provides all necessary components. Integrating it into a virtual prototype provides insight into specific requirement specifications and test guidelines. A simulation environment coupled with a driving simulator enables the evaluation of new ADS functions to reduce risk compared to a function that goes through all steps of the V-model before evaluation [156].

The following step in the simulation process is the software-in-the-loop simulation. This allows for testing of modules for serial development within the architecture of the final system. The test environment must provide the interfaces and runtime environment of the test object. Standardized architectures, such as AUTOSAR, can simplify this process by only requiring modeling of the lowest layers interfaces in the layer model. This approach enables the seamless integration of the function under development into the final system virtually. It can be evaluated by humans as a virtual prototype [156, 130].

In the next step, the hardware-in-the-loop method transfers the models stepwise to hardware components. After this system integration step, the iteration cycles become longer, and the focus shifts to validation with a greater emphasis on automated rather than interactive testing. The first HiL-Tests verify the components according to the specifications. Once these tests are complete and all components are verified, integration begins, and the modules are verified for their interaction. At the end of this phase, the systems hardware components are finalized and optimized in accordance with their specifications. [156, 130].

The final method is vehicle-in-the-loop, which enables real vehicle testing in a virtual environment. One potential use case involves feeding sensor data from a simulation environment through the vehicles environmental sensor interfaces. Another option is to stimulate the sensors with artificial data, such as ultrasonic sensors. Both methods allow for reproducible test cases for validation purposes [156]. The methods of this thesis primarily fall under the categories of MiL and SiL testing.

2.1.2 EU Regulations for Scenario-based Testing

As stated in Section 1.1.1, the EU Regulation 2019/2144 [41] and its implementation regulation [42] were recently introduced. They describe scenario-based testing as an essential testing approach during the development and the type approval process.

Figure 2.1 shows the recommended process outlined in EU implementation regulation 2022/1426 for deriving relevant scenarios from the ODD of an ADS. It describes a scenario-specific approach based on simulations, test track maneuvers, and real road driving tests.

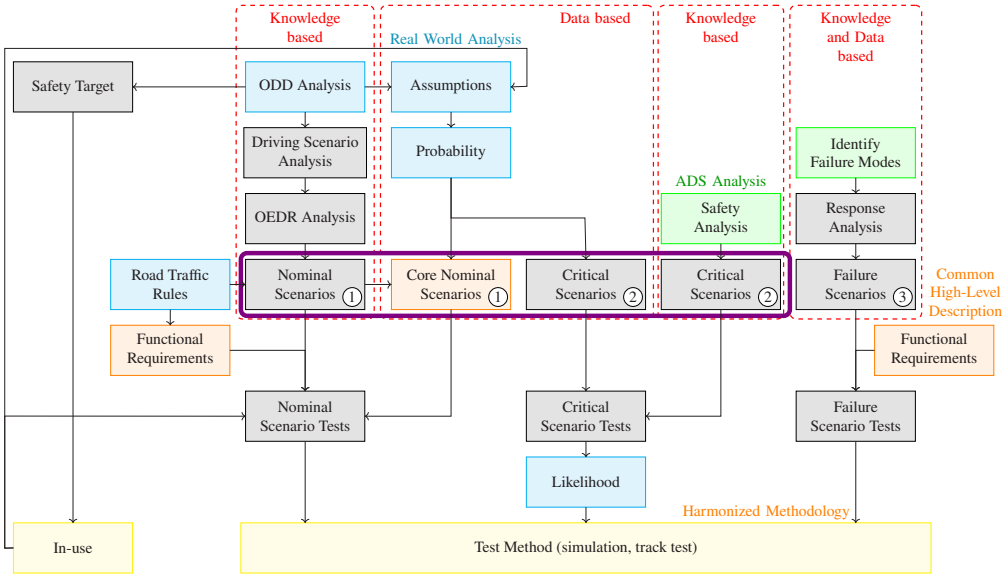


Figure 2.1: The principles to be followed to derive scenarios relevant for the ODD of the ADS [42]. The purple frame indicates where the scenario exploration results (the core of this work) is located.

In general, there are two approaches to the generation or retrieval of traffic scenarios: the knowledge-based approach and the data-based approach. The knowledge-based method involves experts to define critical and hazardous situations and to generate scenarios (see Section 2.3.5 and Section 2.3.2). The data-based approach, on the other hand, uses existing data and evaluation methods to find and classify relevant scenarios for testing. Such methods are scenario extraction (Section 2.3.4), scenario exploration (Section 2.3.3), or scenario generation (Section 2.3.1). In all cases, the scenarios must be derived from the ADSs ODD.

In the regulation, scenarios are classified into three types: nominal (1), critical (2), and failure (3). The numbers correspond with the numbered circles in Figure 2.1. In the following text, the expressions in brackets mark the according names in Figure 2.1. The regulation [42] distinguishes between the derivation of all three types of scenarios. At the beginning, the nominal scenarios can be derived from the *ODD Analysis* of the automated driving system or the fully automated vehicle. The goal is to collect all possible interactions between the ego vehicle and objects within the ODD. Table 2.1 shows a list of possible interactions with other vehicles. This helps to determine how the ODD affects the behavior competency of the ADS (Driving Scenario Analysis).

Following this, an analysis of the *object and event detection and response (OEDR)* is conducted, which results in a set of behavioral competencies for the driving system and a corresponding set of scenarios. The goal is to map all potential events to an appropriate response from the ADS.

Table 2.1: Example object and events/interactions relation from [42]. According to an ADS' ODD, the object on the left side can participate in the events listed on the right.

Objects - Vehicles	**Events/Interactions**
Cars, Light Trucks, Heavy Trucks,	Lead vehicle decelerating (frontal)
Buses, Motorcycles	Lead vehicle stopped (frontal)
	Lead vehicle accelerating (frontal)
	Changing lanes (frontal/side)
	Cutting in (adjacent)
	Turning (frontal)
	Encroaching opposing vehicle (frontal/side)
	Encroaching adjacent vehicle (frontal/side)
	Entering roadway (frontal/side)
	Cutting out (frontal)

The regulation [42] provides a list of examples, such as *Leading vehicle stopped*, to which the ADS responds with *Decelerate, stop*. This list of events and responses is dependent on the objects and events identified in the previous step (ODD Analysis). In addition, the real world is analyzed with respect to the ADS. This analysis leads to *Assumptions* that must be evaluated for their *Probability* of occurrence in order to obtain a set of defined *Core Nominal Scenarios*. These core nominal scenarios must be included in the general set of *Nominal Scenarios*.

Critical scenarios may be edge cases of nominal scenarios or standardized methods for evaluating operational inefficiencies. Edge cases can be derived from real-world analysis or a safety assessment conducted by experts. This assessment should include critical scenarios where the ADS must respond with emergency reactions to environmental conditions. These cases may be supported by an enhanced combination of scenario descriptors from different ODD parts, e.g., infrastructure, dynamic and static objects, or environmental conditions.

The last category, the *Failure Scenarios* (ADS internal failures), include failure situations to assess the driving systems response. During the development phase, the manufacturer must prepare strategies for addressing each identified failure (*Identify Failure Modes + Response Analysis*). These scenarios can be utilized in both simulation and physical testing.

Furthermore, the regulation defines criticality metrics, e.g., TTC, including its use cases, formulas, and thresholds. Regarding the safety concept, it requires a reasonable scenario coverage of selected test scenarios that represent the ODD.

The purple frame around the boxes with nominal and critical scenarios in Figure 2.1 highlights the part of the diagram where scenario exploration results and the core of this work are located.

2.1.3 Foundations of Scenario-Based Testing

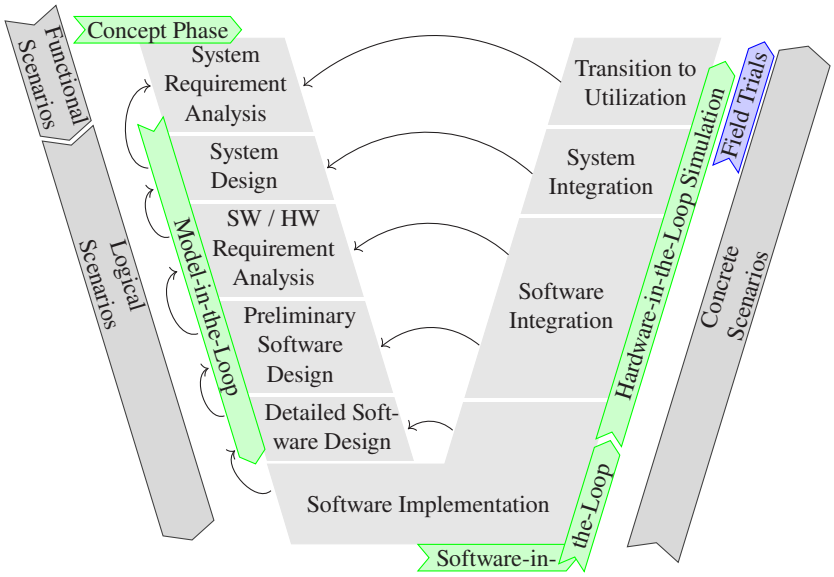


Figure 2.2: V-Model showing different X-in-the-loop phases [130, 83] combined with derivation of scenarios at different development stages [30]

The ISO/PAS 21449 [141] and SAE J3016 [129] standards provide standardized definitions for terms related to automated vehicles as well as the verification and validation context. This section briefly introduces the most important terms used throughout this work to ensure a shared understanding and avoid ambiguity. The definitions of the following terms are narrowed down from their standardized form to fit the scope of this work.

Definitions of Important Terms

- *Action [141]*: Atomic behavior that is executed by any actor in a scene. Also, a definition of what an actor is supposed to do during a specified scene in a scenario.
- *Actor*: Based on the previous definition, an actor is a person or vehicle that serves as a dynamic object in a scenario. Other possible terms include traffic participant, vehicle, or pedestrian.
- *Automated Driving System (ADS) [129]*: Software that is can perform a driving task within a given scenario.

- *Intended functionality [141]*: The specified behavior that a system should exhibit, particularly prior to executing a scenario.
- *Operational Design Domain (ODD) [129]*: The Operational Design Domain specifies the conditions under which an ADS is intended to function. The ODD defines *where*, e.g., environmental or geographical characteristics and *when*, e.g., time-of-day restrictions.
- *Scenario [141]*: A chronological sequence of individual scenes and their transitions.
- *Scene [141]*: Snapshot of all dynamic objects, actors, and their observable states, as well as the scenery, such as weather and lane network.
- *Test Case [141]*: A set of conditions that determine whether an ADS is functioning as intended.
- *Validation [85]:* Evaluation whether the model or software is valid for its intended functionality, specifically for building the correct simulation model.
- *Verification [85]:* Evaluation of the accuracy of transforming a problem formulation into a model or software, i.e., building it right.

Another set of definitions that are not defined by standards or other documents, but play an important role in understanding this work are:

- *Parameter range*: Parameter values bounded by the lowest and highest possible number, e.g., a parameter p can be defined as follows $p \in [0, 1]$, where p takes a value between 0 and 1.
- *Parameter set*: A set of n concrete values for all n parameter ranges of a scenario. The size of the set depends on the scenario description and holds a concrete value for each previously defined parameter range. In the previous 1-dimensional example of $p \in [0, 1]$, an example of a parameter set is $(p = 0.5)$.
- *Parameter space*: n-dimensional space, defined by all n parameter ranges used within one scenario. Each point within the parameter space depicts a parameter set.
- *Simulation model*: All parts of a simulation that represent a modeled version of real-world objects, such as sensor models and road users. These objects are often simplified to varying degrees compared to their real-life counterparts.

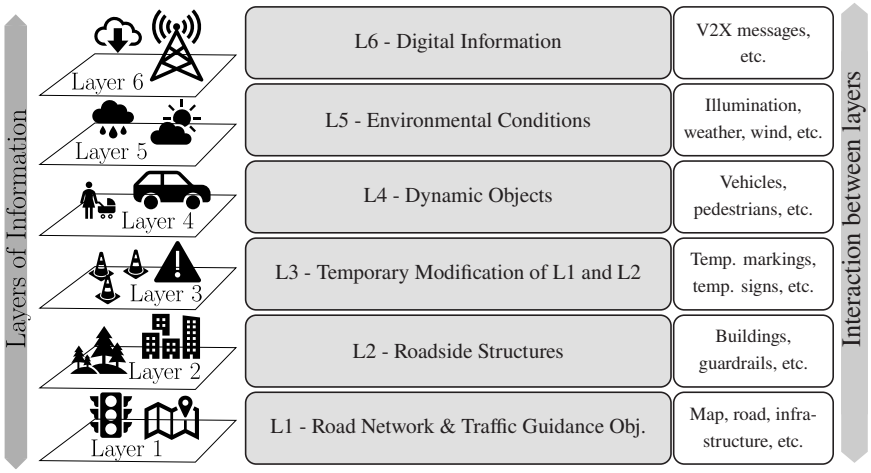


Figure 2.3: PEGASUS **6-Layer model** with examples for information defined on each layer [132]

Scenario Description Layers

The scenarios and their description languages simulated during the exploration are typically designed in different layers. Scenario descriptions can be divided into information layers with equally grouped information. The PEGASUS method [119] includes a 5-layer model, which was further developed in the VVM project [147]. Scholtes et al. [132] suggest extending the 5-layer model by Bagschik et al. [25] to a 6-layer model. Figure 2.3 displays all six layers with examples of information belonging to each layer in the right column. The layers interact with each other and define a complete scenario. Typically, maps describe Layer 1 (L1) and Layer 2 (L2), including related formats such as ASAM OpenDRIVE and OpenCGR [21], which provide information about the unchanging infrastructure and road surface. Layer 3 (L3) through Layer 5 (L5) are described by scenario description languages, such as OpenSCENARIO [21]. They describe all participating actors (L4), their maneuvers (L4), additional traffic-relevant objects (L3 and L4), temporary changes of L1 and L2 (L3), and environmental settings (L5). However, there is currently no established description standard for Layer 6 (L6), which defines digital infrastructure. The primary focus of this work is at Layer 4 since mainly scenario objects and their parameters are changed and explored.

Scenario Abstraction Levels

This work explores scenario parameter spaces to find critical or interesting scenario parameter sets. According to Menzel et al. [102] and the PEGASUS method [119], a scenario parameter

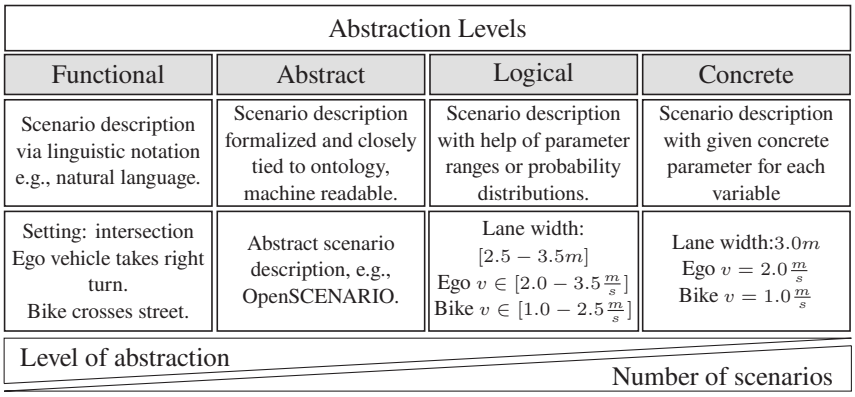


Figure 2.4: Four abstraction levels: functional, abstract, logical, and concrete scenario [102, 109] with urban intersection example: right turning ego vehicle and bike crossing the street

space and a parameter set from within this space are different levels of scenario abstraction. Thus, scenarios can be categorized into different levels of abstraction (functional, logical, and concrete scenarios). This categorization depends on the stage of development and testing. During the development and testing process, the scenarios become increasingly refined.

The most abstract level of scenario representation is called *functional* and describes scenarios using natural, non-structured language terminology. The main goal of this level is to create scenarios that are easily understandable for experts. The next abstraction level is *logical* scenarios, which refine the representation of functional scenarios with the help of parameters. These parameters may include ranges for road width, vehicle positions and velocity, or time and weather conditions. They can be described as either parameter ranges or probability distributions. The last and most detailed level consists of *concrete* scenarios that depict operating scenarios with concrete values for each parameter in the parameter space. One logical scenario can result in multiple concrete scenarios, depending on the number of variables, their range, and step size.

In VVM, the three abstraction levels were refined: Neurohr et al. [109] propose a fourth level, the abstract level, which is located between the functional level and the logical level. The abstract scenario is a formal, machine-readable, and declarative description of a traffic scenario that focuses on complex causal relations. It links the scenario description closer to an ontology and a related ODD from which the scenario was derived. An abstract scenario description can be written in description languages like TSC [44] or OpenSCENARIO [21], which are further explained in Section 3.1.1.

Figure 2.4 illustrates the four levels of abstraction at an example: The functional scenario describes the setting and actions of all participants. The abstract scenario is represented by an executable scenario file, e.g., OpenSCENARIO. In the logical scenario, all relevant parameters are assigned

parameter ranges: lane width, ego speed, and bike speed. In the last level, the concrete scenario, all parameters are assigned concrete values.

2.1.4 Scenario-based Testing Concepts

As stated by Wood et al.[157, p. 83f.], scenario-based testing is a suitable method to supplement the statistical approach of real-world driving, thus reducing the necessary mileage. This method includes the techniques and strategies listed below during the test process to gain information and make statements about the quality of a SUT:

Scenario acquisition: Scenarios have to be generated, extracted, or designed by experts. Section 2.3 provides an overview of various methods for acquiring scenarios, their characteristics, and their interconnections. A focus is put on scenario exploration since it is part of this work.

Scenario classes: Scenarios can be categorized into equivalence classes to determine the necessary amount and conditions of interesting test cases. Expert knowledge, automatic traffic data extraction, and accident databases can be utilized and included in the desired test strategy. This approach considers known safety features, e.g., exposure, severity, and controllability (ISO 26262) (see Section 1.1.1), and can be used to assess test coverage and control influencing factors. Additionally, scenarios equivalence classes may also be used in simulation and real-world test driving.

Surrogate metrics: Surrogate or criticality metrics can be used to evaluate the behavior of an ADS or human driver in different scenarios. In general, metrics can vary in the amount of scenic knowledge they consider. Simple metrics might only consider the position, orientation, and speed of two traffic participants (e.g., TTC [73]) to make a statement about the danger in a particular situation. In contrast, more complex metrics analyze the current situation in greater depth (see Section 2.2). Surrogate metrics are also known as criticality metrics [2] or key performance indicators (KPIs)

System decomposition: The ADAS or ADS can be decomposed into various (sub-)systems, which enable individual testing of all components, e.g., individual tests and test strategies for sensors, motion control, drive planning, and localization.

Test variation: Scenario-based testing allows to combine different platforms and design techniques, e.g., variable variation or stochastic variation, to increase test coverage in the SiL environment.

According to Steimle et al. [138], a test case in scenario-based test approaches comprises a scenario and at least one evaluation criterion. Ulbrich et al. [146] define the temporal development of an initial scene and its participating static and dynamic actors, similar to a storyline described by a domain model specification [24]. Therefore, scenario-based testing refers to the practice of using predefined scenarios to execute test cases [51, 144].

2.2 Quality and Criticality Metrics

Criticality metrics play an essential part in finding critical parameter sets for testing and evaluate the performance of a SUT. Various names are used for the quality and criticality metrics that quantify a current situation or scenario regarding the ego vehicles safety for verification and validation purposes or traffic observations to make road and traffic infrastructure safer. The terms *traffic conflict technique (TCT)* or *surrogate safety measure (SSM)* describe metrics to analyze safety from observable and potentially dangerous traffic events that occur more often than crashes. Figure 1.1 illustrates a pyramid of traffic conflict severity, ranging from potential conflicts to severe injuries or fatalities. Hydén [77] assumes that conflicts precede collisions. The TCT or SSM metrics are referred to as *(traffic) conflict indicators*, *surrogate indicators*, or *criticality metrics* [97, 2], although criticality only encompasses a portion of all relevant metrics in the field of automated driving. In some contexts, e.g., the PEGASUS project [119], the term *key performance indicator (KPI)* is used interchangeably with criticality metric. This chapter does not provide a list or review of available criticality metrics. Instead, the aim is to demonstrate different ways in which these metrics have been categorized or classified. There are numerous reviews and collections of criticality metrics available that define and explain standard metrics, e.g., time-to-collision (TTC) [2, 108, 100].

2.2.1 Proximal Surrogate Indicators

Mahmud et al. [97] propose various categories of proximal surrogate indicators: temporal-based metrics, distance-based metrics, deceleration-based metrics, and other metrics. Most popular criticality metrics are based on temporal or spatial proximity, assuming that the distance between two traffic participants decreases while the probability of a collision increases. These metrics are inexpensive to calculate since velocity and positional data can be recorded by simulated and real-world systems. Metrics of this type typically indicate a conflict if the resulting value is below a predetermined threshold.

- **Temporal proximal indicators example**:

  - Time-To-Collision (TTC) measures the time until a collision between two road users if they continue on their current path and speed.
  - Headway describes the elapsed time between the leading vehicle and the following vehicle passing a given point in space.
  - Post-Encroachment Time (PET) is the time difference between the departure of one traffic participant from an area of interest and the arrival of another traffic participant at the same area.

- **Distance-based metric example**:
  - Proportion of stopping distance (PSD) determines the ratio between the remaining distance to a potential point of collision and the minimum acceptable stopping distance.

- **Deceleration-based metric example**:
  - Deceleration rate to avoid a crash (DRAC) measures the difference in speed between a following vehicle and its lead vehicle, divided by their closing time.

There are additional metrics, such as jerk or standard deviation of the lateral position, that cannot be classified into the three previously mentioned groups. However, these are less commonly known than the aforementioned examples. A potential limitation of using metrics such as TTC is that they may only be applicable in certain scenarios. For instance, although TTC can be helpful in assessing rear-end collisions, it may not offer usable information for scenarios where traffic participants have intersecting trajectories at an intersection.

2.2.2 Motion-Prediction-based Indicators

Another classification was suggested by Lefèvre et al. [92], where each metric is classified regarding the underlying motion model. The authors distinguish between three different motion models with increasing abstraction levels: physics-based models, maneuver-based models, and interaction-aware models. Figure 2.5 shows all three motion models, their variables, typical challenges, and potential applications.

Motion prediction utilizes dynamic and kinematic models for physics-based models, where control inputs (e.g., acceleration) are combined with vehicle properties (e.g., size) and external conditions (e.g., road friction) to get information about future vehicle states (e.g., position or speed). Dynamic and kinematic models are called evolution models and are used for trajectory prediction. Trajectory

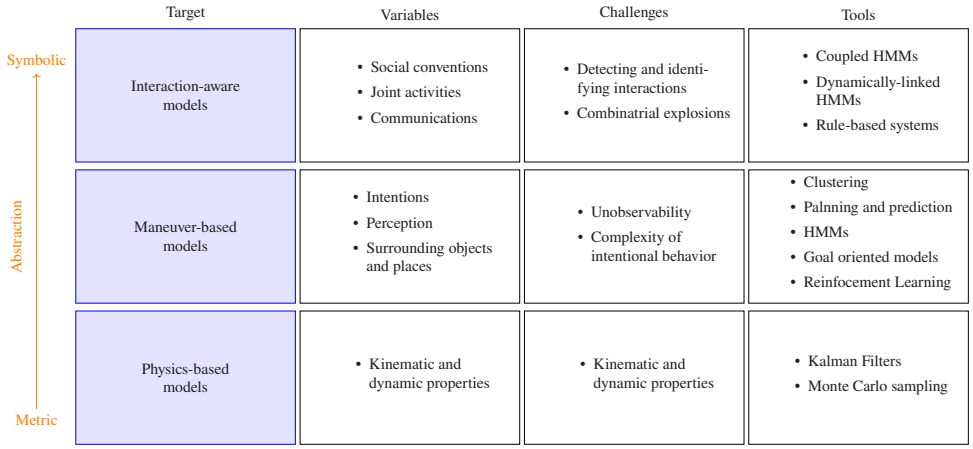


Figure 2.5: Motion modeling overview [92]. All three model types use different information abstraction to describe the motion model.

prediction methods differ in the way they handle uncertainties. These methods include single trajectory simulation, Gaussian noise simulation, or Monte Carlo simulation. An example of single trajectory simulation is TTC, which usually employs a constant velocity motion model. A limitation of physics-based motion models is their inability to predict motion in the long term since they typically cannot anticipate changes in a vehicles motion, e.g., slow down to turn right or avoid a crash.

Maneuver-based motion models are based on the assumption that the motion of a vehicle is based on the road network and consists of a series of maneuvers that are independent of other traffic participants. The objective is to recognize the intended maneuvers as fast as possible to have a better and more reliable knowledge of the trajectory in the long term. However, this approach has a downside in that it assumes independent vehicles, which may not always be the case. In situations with high traffic density, traffic participants regularly react to each other, which can lead to erroneous interpretations.

The final category comprises interaction-aware motion models. This approach addresses the problematic assumption of maneuver-based motion models that traffic participants do not interact with each other. It combines a better long-term prediction than physics-based models with better reliability than maneuver-based models. However, interaction-aware motion models often require computation of all possible trajectories, making them too expensive for real-time assessment.

2.2.3 Constellation-based Criticality Metrics

Many criticality metrics are based on certain car or actor constellations, such as car-following situations or intersections. These metrics are only applicable in these situations and may produce unusable results when applied elsewhere. This section primarily describes the metrics used in the experiments conducted in this work.

Criticality Metrics for Car-following Situations

The best-known metric for car-following situations is Time-To-Collision (TTC). TTC was developed by Hayward [73] in 1972 to measure the danger of a traffic situation. It describes the time a following vehicle in a rear-end situation needs to collide with the rear of the leading vehicle. In case the leading vehicle is faster than the following vehicle, TTC approaches infinity. TTC can be calculated by the following equation [74]:

$$TTC = \frac{d(A_1, A_2)}{v_{rel}}, \tag{2.1}$$

where A_1 and A_2 are the leading and following vehicles, d describes the distance or headway between both, and v_{rel} is the current relative velocity between both vehicles. Since TTC was developed in 1972, many derivatives, alternatives, and related metrics have been developed. A short list and summary of the most common and recognized variations can be found in Westhofen et al. [2].

Criticality Metrics for Intersections

Post-encroachment time (PET), gap time (GT), and encroachment time (ET) were developed by Allen, Brian, L. et al. [16] in 1978. They videotaped collisions and defined certain situations: T_1, where the first (ego) vehicle enters the intersection area or point, T_2, where the first vehicle leaves this area again, T_{3act} is the actual points in time where the second vehicle enters the intersection. T_{3act} is complemented by T_{3ant}, which is the anticipated time when the second vehicle arrives at the intersection. T_{3act} varies from T_{3ant} insofar that a braking or acceleration maneuver can influence T_{3act} and might not be known beforehand. It can be concluded that metrics based on the actual time T_{3act} can only be calculated after passing the intersection. In contrast, the anticipated time T_{3ant} can be used and adjusted during the crossing of the intersection.

Encroachment time is calculated by

$$ET = T_2 - T_1, \tag{2.2}$$

and defines the time a vehicle is occupying the intersecting area and with that being in danger of a possible collision.

The equation for PET is:

$$PET = T_{3act} - T_2, \tag{2.3}$$

and measures the elapsed time between two traffic participants crossing a given point, while GT predicts the elapsed time for each time step and therefore approximates PET:

$$GT = T_{3ant} - T_2. \tag{2.4}$$

2.2.4 Criticality Metrics Independent of Constellations

Worst-Time-To-Collision

Worst Time-To-Collision (WTTC) was developed by Wachenfeld et al. [149] and describes a constellation-independent metric that extends TTC. It can be used for every situation between two traffic participants, e.g., following, overtaking, oncoming, and intersection. The over-approximating driver motion model considers every possible trajectory for both actors and anticipates the worst possible situation as the outcome. The worst possible situation, in this case, is the fastest possible collision. Calculations are performed using a system of five equations.

Distance

Euclidean distance is a common metric used for criticality [15]. Distance is a simple and cost-effective metric, requiring only position and orientation in space. However, it has limitations. For example, two cars driving parallel to each other on different lanes may have a small distance between them, but would not be considered critical by a human observer. While distance can serve as a starting point for assessing criticality, it should not be the sole determining factor. It may be beneficial to combine distance with other criticality metrics or impose penalties for traffic participants passing each other on neighboring lanes, as these are common traffic situations that occur regularly [15]. In car-following scenarios, the distance between vehicles is also referred to as headway.

2.2.5 Environmental Criticality Metrics

Often used metrics like TTC [73] or PET [16] only need speed and position as input variables, and thus, are cheap to calculate. They are computed concerning two traffic participants: the ego vehicle and its current adversary. These metrics are based on a pairwise criticality evaluation and only consider two traffic participants from a possibly much larger list. Critical situations involving more than two parties might not be adequately recognized. Therefore, metrics were developed to consider more aspects for criticality evaluation, e.g., traffic density or the underlying map.

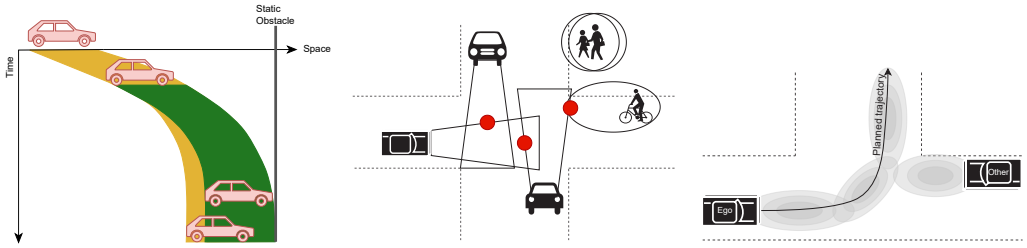
Traffic Quality

Hallerbach et al. [70] propose a method for evaluating the criticality of the traffic on a highway section: the focus is not solely on the ego vehicle and one other traffic participant, but on a combination of different domains of interest, including larger road sections, traffic around the ego vehicle, and the ego vehicle itself. However, this approach is only suitable for highway traffic. The weights used in the proposed formula are dependent on the situation and the road segment and require analysis and optimization for new scenarios. The calculation of traffic quality is divided into four parts with coefficients of varying weights, which sum up to a total criticality score. A circle around the ego vehicle designates the area of a highway section which is considered for the metric calculation.

The first part, the macroscopic metric, calculates the traffic density calculated from the traffic flow rate and the average travel velocity. Additionally, the microscopic metric considers the velocity deviation and the average velocity on a highway section around the ego vehicle. The third part of the traffic density term, the nanoscopic metric, focuses on close-range interactions within a smaller radius around the ego vehicle. Its calculation is based on velocity deviation and mean value within it. The last part is the individual metric and considers the ego vehicles mean velocity and standard deviation of the acceleration. Finally, a supervised training algorithm determines optimal weights as coefficients for all four terms.

Safety Force Field and Reachable Sets

Other metrics, such as Safety Force Field (SFF) [113, 114] and reachable sets [18], compare the current situation with a given time horizon and the area that is most likely occupied by the ego vehicle and other traffic participants. The idea behind the safety force field is to avoid collisions by not contributing to an unsafe environment since an ego vehicle cannot guarantee a safe journey when other actors share the roads [113]. Figure 2.6a illustrates a one-dimensional example of



(a) One-dimensional example of occupied space over time as proposed by [113]. The occupied space is dependent on the breaking schedule of a vehicle. The dark green space indicates the required deceleration, while the yellow space indicates the maximum possible deceleration. Together, these spaces are referred to as the claimed set.

(b) The safety force field is available for all actor types. However, each actor type has its own behavior and safety procedure in case a collision has to be avoided. The conflicts between the claimed sets of the shown actors are depicted in red [113].

(c) Stochastic reachable sets for ego vehicle and another car which can be used to compute the crash probability in a second step [18]. The planned trajectory of the ego vehicle serves as the reference line for further computation of possible road occupancy.

Figure 2.6: Examples for safety force field and reachable sets.

the space required for a car to decelerate and avoid hitting a static obstacle. The combination of the space occupied by the minimum required deceleration and maximum possible deceleration is known as the claimed set. It is represented by the green and yellow area over time in said one-dimensional space. All actors must apply their safety procedure in case their claimed sets intersect, where the conflict between two actors claiming the same time-space is called safety force field. This approach can also be applied in a 2-dimensional space, taking into account longitudinal and lateral movement. This method is applicable to any actor, as demonstrated in Figure 2.6b. Strategies such as the safety potential for safe behavior can be applied based on the concept of claimed sets and safety force fields. The safety potential is based on the idea that areas with high overlapping of claimed sets also have a high potential, and the goal for all actors is to minimize the potential of the area they are located in. All actors are required to adjust their trajectories accordingly to create a safe environment.

Althoff et al. [18] propose a related idea with reachable sets and stochastic road occupancy, which is shown in Figure 2.6c. The reachable sets describe road occupancy similar to SFF, taking into account deviation from the reference trajectory and map layout. Again, this approach is based on a probability distribution that determines the most likely location of a road user. The resulting crash probability can then be used to adjust the planned trajectories.

Responsibility Sensitive Safety: Dangerous Situations

The Responsibility Sensitive Safety framework offers a set of mathematical formulas and logical rules based on how humans characterize safe driving behavior to guarantee safety during an automated vehicles drive [137]. RSS is based on five rules:

- Do not hit someone from behind (longitudinal distance).
- Do not cut in recklessly (lateral distance).
- Right of way is given, not taken.
- Be cautious in areas with limited visibility.
- If the vehicle can avoid a crash without causing another one, it must.

To consider these rules, Responsibility Sensitive Safety distinguishes between functional safety, which refers to hardware failure or software bugs, and nominal safety, which considers the behavior and decision-making of an automated driving system based on the assumption that software and hardware are working error-free. The indicator of a dangerous situation is a combination of the safe longitudinal and lateral distances with their thresholds, depending on the current situation and utilized prediction models, e.g., intersection, highway, or country road. According to Koopman et al. [87], a significant problem with this approach is that Responsibility Sensitive Safety does not consider certain edge cases, e.g., perception uncertainties or slopes. To avoid confusion with the term *residual sum of squares* later, the abbreviation RSS is not used for Responsibility Sensitive Safety.

2.3 Scenario Acquisition

Finding new scenarios is one of the first steps in defining test cases to assess the safety of an automated driving system.The following chapter presents recent research approaches, commonly referred to as *scenario generation*. While most of these methods return scenarios for testing, the underlying concepts and types of scenarios produced can vary from approach to approach. For example, one generation method may involve creating new trajectories [15], while another generates scenarios with new sequences of scenario building blocks and actor behavior [67].

This work uses the term *scenario acquisition* as a more general expression for obtaining new scenarios.[1]

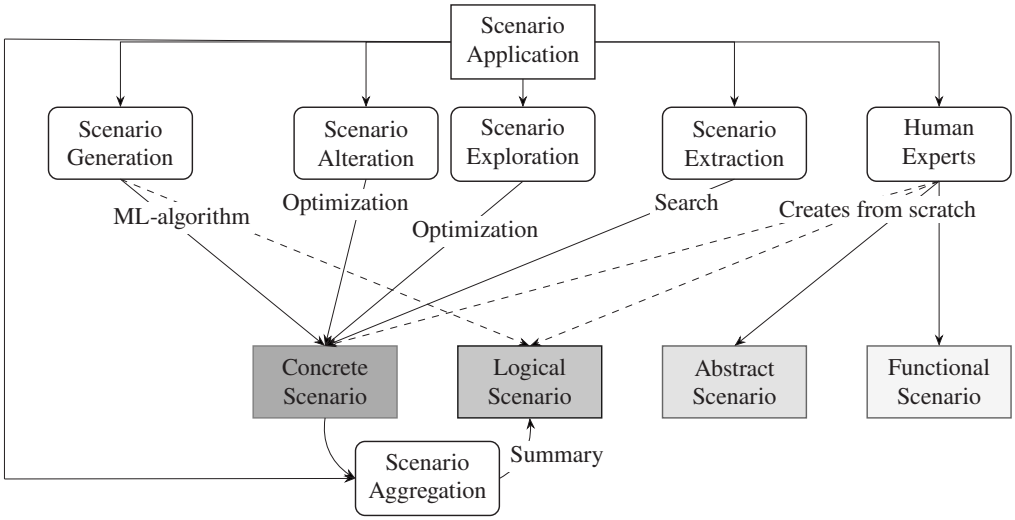


Figure 2.7: This context diagram illustrates the six main categories of scenario acquisition methods and their major results. Solid arrows indicate the primary output scenario type, while dashed arrows indicate less common output types.

Different concepts, methods, and algorithms are proposed in the literature. In order to get a structured overview of the field, the following taxonomy was used to categorize different acquisition methods. This taxonomy highlights major conceptual differences between individual approaches and groups them accordingly. Scenario acquisition can be grouped into six major categories and is shown in Figure 2.7 as a schematic overview. The initial step involves defining **what** needs to be tested (Scenario Application) since the goal is to help perform a targeted search and to define sensible criticality metrics. Various methods for discovering new scenarios are available from this point (white boxes). The primary output scenario type is indicated by the solid arrows connecting the categories and scenario types. Dashed arrows indicate rare but possible outputs. The absence of an arrow does not exclude a scenario type as a format. However, recent literature has yet to identify such cases. Scenario acquisition methods fall into six categories:

1 Scenario Generation: New scenarios or sub-sequences of existing scenarios are (automatically) generated for fulfilling a given set of criteria. The primary characteristic of this

[1] Parts of this chapter have been published in [10]. Additional and more detailed information can be found in Appendix A.1.

category is the creation of new scenarios with novel content that do not necessarily belong to the same logical scenario.

2 Scenario Alteration: In this process, a parameter value is modified to explore other related scenarios. The initial scenario does not necessarily begin with a predefined parameter range, but with an initial value that is adjusted during the search.

3 Scenario Exploration: The parameter ranges or distributions of a given logical scenario are explored while fulfilling a given set of criteria, e.g., explored by optimization algorithms. The new parameter values are determined by a step size within the defined range or drawn from a given distribution.

4 Scenario Extraction: Recorded data is analyzed to identify specific features that meet the desired criteria, using machine learning algorithms or rule-based methods.

5 Experts: Human experts define a scenario. This can be done by drawing the scenario sequence as images, as well as modeling concrete scenarios directly in a simulation software.

6 Scenario Aggregation: Concrete scenarios discovered by any acquisition method can be grouped into logical or related scenarios. These are scenarios that share features and content, making them suitable for clustering or grouping.

The identified scenarios can be used not only for testing purposes but also for other applications, such as training automated driving functions with reinforcement learning [15].

2.3.1 Scenario Generation

The first concept, scenario generation, involves creating new scenarios, e.g., Goss et al. [67] use atomic scenario building blocks, such as Accelerate or Keep Velocity, to build a sequence and thus a new scenario. It is important to note that the term **scenario generation** is not used in the usual way, which typically refers to the creation of new scenarios even if they only have minor changes, such as a different velocity of one actor. This category summarizes methods that can identify specific scenarios from various logical scenarios, as well as methods that are not dependent on a single base scenario, which is neither concrete nor logical. Scenario generation involves modifying the scene sequence, actor behavior and number, or actor tasks within a scenario. This means that more than just parameters are changed when a new scenario is generated. Other than scenario alteration and exploration, scenario generation can identify logical scenarios or concrete scenarios that do not belong to the same logical scenario.

Furthermore, scenario generation can be divided into two main sub-categories. The first sub-category generates scenarios from scratch, where changes on all six scenario layers are possible. The second category, data-driven scenario generation, works with existing actor behaviors and road networks gathered from real-world data. These methods mainly consider modifications on layer 2 and above. Scenario generation from scratch can start by creating a road network (layer 1) and adding objects in the following layers, most commonly layer 4. These two sub-categories are not displayed in Figure 2.7 because they do not differ in terms of generated or used scenario types.

Generating scenarios from scratch usually starts with a set of scenario building blocks defined by the framework or task at hand. These blocks are combined and changed with a generation algorithm to generate new (concrete) scenarios. Additionally to layer 4, it often utilizes map or road generation to add changes on layer 1 [93, 125, 118], which is a major difference to most other scenario acquisition types. Other approaches are keyword-based [103] or ontology-based [25].

The utilization of real-world data for scenario generation typically involves starting with a base of scenario blocks from recorded concrete or logical scenarios. After processing, existing objects can be modified, new trajectories can be assigned, or additional static and dynamic obstacles can be added. Data-driven approaches for scenario generation primarily focus on layer 4 and employ various generative algorithms.[2]

2.3.2 Scenario Alteration

Scenario alteration involves modifying a concrete scenario by changing specific values until achieving a desired outcome. This results in a set of new concrete scenarios derived from the initial concrete scenario. Most alteration methods are algorithm- or optimization-based, and the algorithm used plays a vital role in determining which scenarios are found. Scenario alteration begins with a modified concrete scenario. It should be noted that the scenarios resulting from scenario alteration may not necessarily fall within a single logical scenario, unlike those resulting from scenario exploration [152]. Both categories are grouped in Figure 2.7.

Scenario alteration refers to methods that modify one or more concrete starting scenarios by changing parameters and scenario parts across different scenario description layers. The most common changes occur on layer 4, which contains dynamic objects. These variations mainly involve altering the trajectories of different types of actors, such as pedestrians or adversary vehicles.[3]

[2] More detailed examples for both scenario generation categories can be found in Appendix A.1.1.
[3] More detailed examples can be found in Appendix A.1.2.

2.3.3 Scenario Exploration

Scenario exploration is a sub-category of scenario acquisition that can be challenging to differentiate from scenario alteration without the use of clear nomenclature, such as logical and concrete scenarios. In the case of scenario exploration, the goal is to explore a predefined range of parameters or distributions that define the behavior within a scenario. It starts with a logical scenario and optimizes the values of various parameters within defined ranges and distributions towards a specific goal. This process usually aims to find the most critical scenarios within the defined parameter ranges. Scenario exploration results in a set of concrete scenarios derived from the initial logical scenario. Although both methods typically produce concrete scenarios, they differ in their starting point and level of abstraction. The method classifies as scenario exploration category only if the parameters are constrained within this predefined range. If not, the method is categorized as scenario alteration. Typically, most exploration approaches are optimization-based and employ optimization algorithms. The primary objective in this category is to discover concrete scenarios that meet specific criteria, such as scenario criticality metrics [2].[4]

The simplest way of scenario exploration is to choose a uniform distribution for all given parameter ranges of a logical scenario. However, Mori et al. [106] showed in their experiments that a discrete set of predefined parameters within a given range can yield problems. They observed that for a common deep neural network perception algorithm, outliers and fluctuations can be overlooked by the grid sampling, resulting in error rates depending on the grid size. As expected, this error rate grows with a higher grid step size but does not decrease after a certain small step size threshold. They hypothesize, that these outliers and fluctuations might be overlooked during scenario-based testing. They conclude, distribution-based sampling or optimization over parameter ranges might solve this problem.

Furthermore, simulating scenarios may not always be necessary during exploration: Scenario evaluation using surrogate models is an alternative to data-driven scenario derivation and evaluating scenarios after simulation execution [29]. Instead of possible expensive physics-based simulations, Ben Abdessalem et al. [29] proposed a surrogate model for each fitness function, i.e., the minimal distance between ego and pedestrian, the minimal distance between pedestrian and a warning area of the ego vehicles sensor, and minimum time-to-collision, to predict the criticality metric results without running the actual simulations. The example scenario for this algorithm was the same as used by Abdessalem et al. [14], a pedestrian crossing the street in front of an approaching ego vehicle. These surrogate models can be developed and trained using a combination of multi-objective search and neural networks and are then used to train and evaluate different evolutionary algorithms.

[4] Detailed examples can be found in Appendix A.1.3.

Furthermore, optimization-based methods cannot guarantee to identify the best possible scenario, such as the most critical, within a parameter space. In the PEGASUS and SET Level project, Hungar [76] delineates an analytical approach how this deficiency might be overcome by the combination of Lipschitz-based extrapolations and systematic, semantics guided search for extrapolation boundaries.

2.3.4 Scenario Extraction

In scenario extraction, scenarios are extracted from recorded driving datasets or other available data, resulting in concrete scenarios as shown in Figure 2.7. Scenario extraction can be further classified into methods that extract scenarios from naturalistic driving data and methods that derive scenarios from crash data.[5]

One approach for scenario extraction is the use of rule-based methods, which utilize predefined rules to extract features from naturalistic driving datasets, e.g., the INTERACTION dataset [161] or the inD dataset [31]. Examples of these features can be maneuvers, as demonstrated by Braun et al. [32], who used recorded data to find concrete scenarios for re-simulation. In addition, experts have the ability to manually modify the scenarios to better suit the specific task at hand.

Catalogues of accident data provide another source for scenario extraction. Unlike naturalistic driving data, accident databases contain only preselected critical scenarios. For example, Cao et al. [36] extracted typical pre-crash scenarios between two-wheelers and passenger vehicles from a catalogue of 216 accident scenarios.

2.3.5 Human Experts

Human experts can create scenarios at all four levels of abstraction (see Section 2.1.3), including functional scenarios that are more human-readable and less oriented towards machine interpretation [102]. The abstract level involves scenarios that can be interpreted by simulation software. On this level, scenarios often define constraints and dependencies rather than concrete values. Examples of abstract-level graphical description languages include Traffic Sequence Charts (TSC) [44] depicted in Figure 3.2, which is similar to a theater / movie storyboard. The CARLA simulator provides a scenario description in the form of behavior trees [49], with a graphical interface that is not part of the main project. Most simulation software vendors offer different scenario modeling interfaces, such as dSpace [50], CarMaker [78], and VDT [107]. Some of these use a

[5] More detailed examples for both categories can be found in AppendixA.1.4.

version of the new standard OpenSCENARIO [21], which is described in Section 3.1.1. However, modeling in these simulators often occurs with the help of a graphical interface, which places the scenarios on a more abstract level than the underlying description languages.

2.3.6 Scenario Aggregation

Scenario aggregation can be used to group similar concrete scenarios into one logical scenario and is often a second step after a set of concrete scenarios has been acquired. King et al. [84] showed that concrete scenarios in urban environments can be extracted from datasets and aggregated into logical scenarios based on maneuver recognition. In the first step of scenario extraction, the actors are categorized based on their role within a scene, i.e., whether they interact with the ego vehicle or pass by. This leads to three different types of interaction: merging, diverging, and crossing. After this, maneuver recognition is used to classify and extract maneuvers from the dataset. Before the extracted concrete scenarios can be aggregated into logical scenarios, they have to be identified. It is possible that a single driving scene contains more than one scenario, and all of them are extracted and identified. In case the maneuver sequences are identical, they should be categorized as derived from the same logical scenario. The resulting logical scenarios consist of a maneuver sequence, the assigned concrete scenarios, and parameter ranges and distributions.

Other aggregations are proposed by Langner et al. [91] (extraction of concrete scenarios from real-world driving data before being clustered into sets of logical scenarios) and by Weber et al. [155] (feature-based clustering). A table with all mentioned scenario acquisition methods and their main features and sub-categories can be found in Appendix A.1.5.

2.3.7 Combination of Methods

Several approaches involve a combination of methods, such as aggregating concrete scenarios after the initial extraction step [155, 84, 91]. Another combination is scenario extraction and alteration [33, 163] as depicted in Figure 2.8. Figure 2.8 provides an overview of possible combinations of automated methods and how they can be used to expand the set of existing scenarios. Scenario extraction from datasets or other recordings of traffic situations can result in new concrete scenarios. Scenario alteration can produce new but related scenarios based on an initial scenario. The resulting concrete scenarios can then be aggregated into logical scenarios. Scenario exploration can also be used to identify interesting or critical areas within a scenario space.

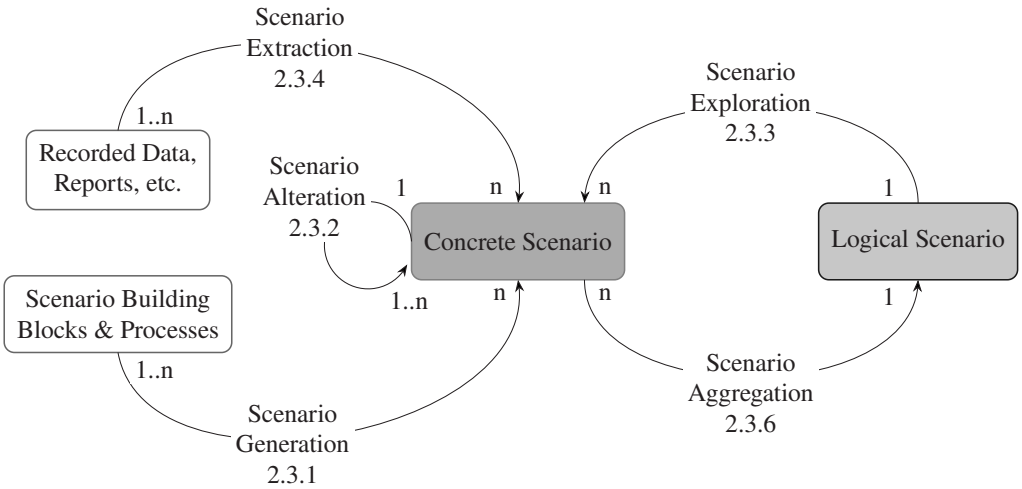


Figure 2.8: Context diagram of the input and output relationships of the proposed categories, their relations, and multiplicities.

3 Technical and Mathematical Background

3.1 Technical Background for Simulation-based Techniques

This work focuses mainly on the software-in-the-loop part of the V-Model (Figure 2.2). Therefore, simulation-related concepts are vital for scenario-based testing during this step. In particular, there are several simulation environments and scenario description languages that provide different features and functionalities.

3.1.1 Scenario Description Methods and Languages

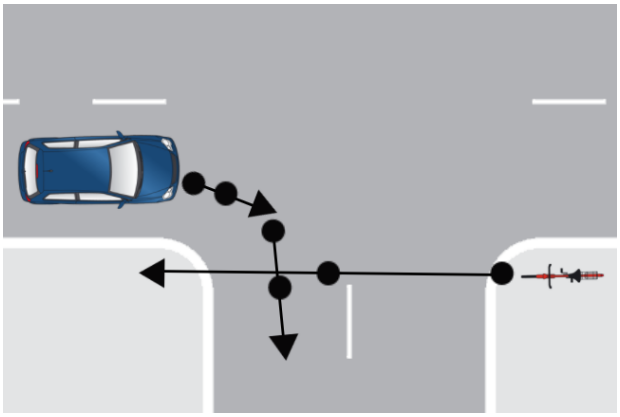


Figure 3.1: Simple scenario representation: the blue car turns right and the bicycle crosses the bottom arm of the intersection

Scenarios can be described with different description languages or methods, motivated by specific use cases. Graphical description and formalized scenario description languages are the two

main approaches in describing scenarios. One crucial trade-off is between intuitive usage of a human-readable description and a well-defined machine-readable and executable language that simulation tools can understand. A common way of describing a scenario for or between humans is an image, as shown in Figure 3.1, or a sequence of pictures depicting all actors' starting points and movements throughout the scenario timeline [128, 149, 160]. A picture can provide a general idea of what is intended, but it may lack essential details such as the order of actions, synchronization points, or conditional behavior. As a further downside, pictures are usually not machine-readable.

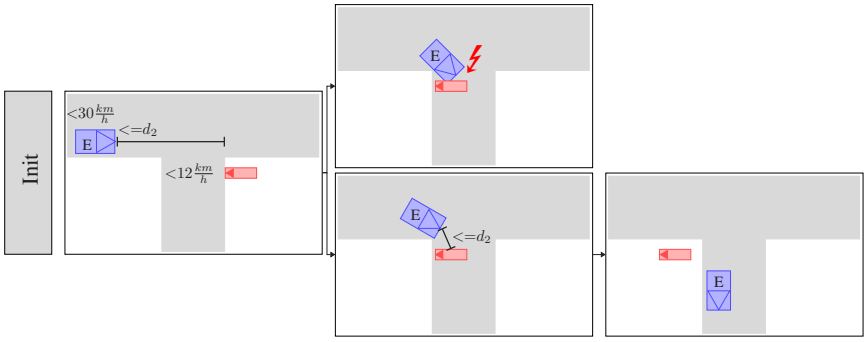


Figure 3.2: Traffic Sequence Charts: Two possible outcomes for a scenario with a right turning car (blue) and a crossing bicycle (red).

A progression of simple images and an example for a graphical scenario description language are Traffic Sequence Charts (TSC) [44], shown in Figure 3.2. In TSC, a scenario consists of several snapshots with information about each important event ordered as a sequence. This method solves the missing information problem and defines a powerful graphical notation in describing a scenario with an underlying first-order logic. However, it defines the scenario outcome, which might not have been known before executing the simulation of a scenario.

A sequence of images leads to a storyboard as it is used in movies or comics and shows an analogy to movie- or theater-related description of scenarios. Based on the definition of *scene* by Ulbrich et al. [146], a scene consists of a subset of information as described by the 6-layer model (Section 2.1.3). It comprises dynamic objects, a description of the scenery, and all actors. A *situation* is derived from a scene by information selection based on given goals and values. It describes all circumstances that have to be considered for behavior planning. A situation consists of all relevant parts of a scene regarding the driving function under consideration. Additionally, it gets extended by the systems goals, e.g., the mission of a driving function, and values, as traffic rules, are included to evaluate the outcome of a scenario and the actions and events of the actors involved. To complete this approach, a scenario consists of at least one scene and a

derived situation with goals, values, and actions. A related approach is introduced by Bach et al. [24]. There, a scenario consists of a sequence of different *acts*, where subsequent acts differ in precisely one actor's maneuver to guarantee a consistent story-line. Therefore, certain events trigger a behavior change and initiate a transition between acts. Each act contains a maneuver description or action plan for all actors.

Other approaches define textual or formalized ways of describing a scenario. A tool-independent scenario description standard currently under development is ASAM OpenSCENARIO. Since 2023, ASAM e.V. distinguishes between OpenSCENARIO XML [56] and OpenSCENARIO DSL [57]. In May 2022, OpenSCENARIO XML version 1.2.0 was released. The OpenSCENARIO DSL was derived from the OpenSCENARIO 2.0 standard, published in 2022. Both standards serve different purposes in the simulation toolchain. OpenSCENARIO XML defines logical and concrete scenarios, while OpenSCENARIO DSL operates at a higher abstraction level and includes KPIs, checks, and coverage metrics. OpenSCENARIO works together with ASAM OpenDRIVE and OpenCGR, which complement each other to describe all aspects of a traffic scenario: from the road topology to all participating vehicles and their maneuvers. All of those OpenX standards, except OpenSCENARIO DSL, utilize an XML-notation and aim to machine-readability, as well as being understandable by human experts. OpenSCENARIO XML only supports concrete scenarios but allows parametrization to load different parameter sets to realize logical scenarios.

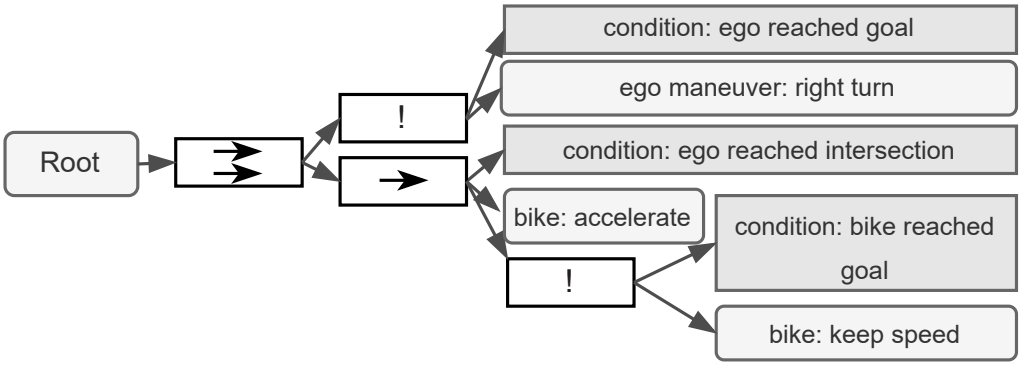


Figure 3.3: Behavior tree scenario representation: same scenario as shown in Figure 3.1 and Figure 3.2 but represented as behavior tree

Another type of scenario description is a behavior tree [39, 99], an established way of expressing and structuring artificial intelligence in video games. The open-source simulator Carla uses behavior trees as a modeling language for scenarios [49]. Figure 3.3 shows the previous scenario represented as a behavior tree notation [39, 99]: An ego vehicle takes a right turn at an intersection

and continues until it reaches a defined goal position. At the same time, a bike accelerates once the vehicle reaches the intersection. Once the bike reaches a given speed, it maintains its velocity until it reaches its goal position. The scenario is completed once both actors have finished their maneuvers.

The SET Level project developed a holistic open-source scenario engine (YASE) with concrete scenarios, implicit traffic, scenery behavior, and test criteria evaluation [26]. YASE's scenario description is also based on behavior trees and has an architecture similar to that of programming language compilers. It provides a middle end that, on one side, can be filled with different scenario formats and, on the other side, can be connected to different simulators, e.g., OpenPASS [117] or Esmini [54].

Most scenario description languages cover abstraction layers 4 and 5 since they describe the order of dynamic objects and the behavior of traffic participants. Additionally, environmental features, such as lighting or weather conditions, can be described; e.g., OpenSCENARIO offers its own environmental keywords and tags. Layers 1 to 3 are rarely described within scenario description languages and are usually represented in map formats, e.g., OpenDRIVE.

3.1.2 Simulation Environments

The automotive domain has brought forth a vast number of simulation software tools which are changing and increasing constantly. Table 3.1 gives a brief overview of eight commercial and open source tools with a selection of their supported standards and relevant functionalities for this work. However, it is not a complete list. Most traffic simulation tools can represent scenario description layers 1 and 2 using map formats, while layers 4 and 5 are typically handled through scenario description formats. In the case of most simulators like Carla, layer 3 is integrated into the infrastructure or the scenario description and does not exist independently. On the other hand, layer 6, which was introduced at a later stage, is not considered part of the Carla environment.

Commercial tools for automotive simulation are available from Vires VTD [107], dSPACE [50], and IPG [78]. All three simulation tools provide modules for map and scenario creation, sensor and dynamic models, and different types of X-in-the-Loop testing, to name some examples. Another tool that was already mentioned is Carla, an open-source simulator with a growing community and based on the game engine Unreal [49]. It offers several additional modules, e.g., a scenario tool that includes its own scenario format with behavior trees and support for OpenSCENARIO, a graphical tool for creating scenarios, a ROS-bridge, and SUMO support. SUMO is an open-source simulation package for modeling microscopic traffic simulation and was developed by the German Aerospace Center [96]. It specializes in big scale traffic simulation and can, for instance, be utilized for evaluating traffic light cycles, evaluation of emissions (noise,

pollutants), traffic forecast, and many others. Other tools worth mentioning are openPASS [117], a simulation tool mainly developed by the Eclipse Foundation and BMW and was further developed during the SET Level project, PTV Vissim [120] for microscopic traffic simulation, or esmini [54], an OpenSCENARIO player. Other tools are NVIDIA Drive [116] which is still in its beta phase, or AirSim from Microsoft [104] which was developed for drone simulation but can also be utilized for road vehicle simulation.

On October 19, 2023, Google Waymo released Waymax [66], a large-scale traffic simulator for simulating scenarios with the Waymo Open Motion Dataset. It enables the evaluation of traffic agents and offers several metrics, e.g., log divergence and collision detection. Waymax also offers dynamic simulation models and supports two modes for simulating uncontrolled objects in scenes and aims to be used as a training tool for reinforcement learning with adaptable interfaces. Waymo's tool is not mentioned in Table 3.1 since it does not support the listed features.

Table 3.1: Overview of simulation environments and their features. Abbreviations: exp.: experimental, part.: partial, y: yes, n: no, u: unknown

Feature		**Simulation Environment**			
Tool		**CarMaker** [78]	**dSPACE Semaphora** [50]	**Carla** [37, 98]	**VTD** [107]
Developer		IPG GmbH	dSPACE GmbH	Open Source	Vires
OpenSCENARIO	0.9.x/1.x	exp.	exp.	part.	y
	2.x	n	n	n	n
Behavior Tree		n	n	y	n
OpenDRIVE		y	y	y	y
OpenCRG		y	y	y	y
SUMO		y	y	y	y
GUI	Map Creator	y	y	n	y
	Scenario Creator	y	y	exp.	y
Sensor Model	Camera	y	y	y	y
	LiDAR	y	y	y	y
	RADAR	y	y	y	y
	Ultrasonic	y	y	n	y

Feature		**Simulation Environment**			
Tool		**openPASS** [117]	**Vissim** [120]	**NVIDIA Drive** [116, 98]	**AirSim** [104, 98]
Developer		Open Source	PTV AG	NVIDIA	Open Src
OpenSCENARIO	0.9.x/1.x	y	n	n	n
	2.x	n	n	n	n
Behavior Tree		n	n	y	n
OpenDRIVE		y	y	y	n
OpenCRG		n	u	n	u
SUMO		u	exp.	n	n
GUI	Map Creator	n	n	u	u
	Scenario Creator	u	y	u	part.
Sensor Model	Camera	u	n	y	y
	LiDAR	u	n	y	y
	RADAR	u	n	y	n
	Ultrasonic	u	n	n	n

3.1.3 Established Resolution Levels of Traffic Simulation

In general, traffic simulation can be divided into different stages of information resolution: nanoscopic, microscopic, mesoscopic, and macroscopic [110, 131, 52]. Figure 3.4 shows a summary of these resolution levels. In macroscopic traffic simulation, the traffic is modeled as fluid. This type of simulation can evaluate traffic flows or congestion in high traffic situations. The following resolution level is mesoscopic traffic simulation. Here, every participant is modeled as a discrete particle whose position lacks personality, such as mass or size. This lacking personal information is added at the microscopic level. At this level of resolution, each participant has its own modeled behavior with an individual state and variables, such as mass, speed, and

acceleration. Additionally, individual maneuvers relevant for specific scenarios are modeled. The highest resolution in traffic simulation is nanoscopic (sometimes called sub-microscopic) and views each vehicle as a composition of different sub-units that need to be coupled to achieve a higher level of detail. Scenario-based testing often occurs in micro- and nanoscopic simulation, since the main goal is to evaluate (sub-)units and their individual behavior in given scenarios.

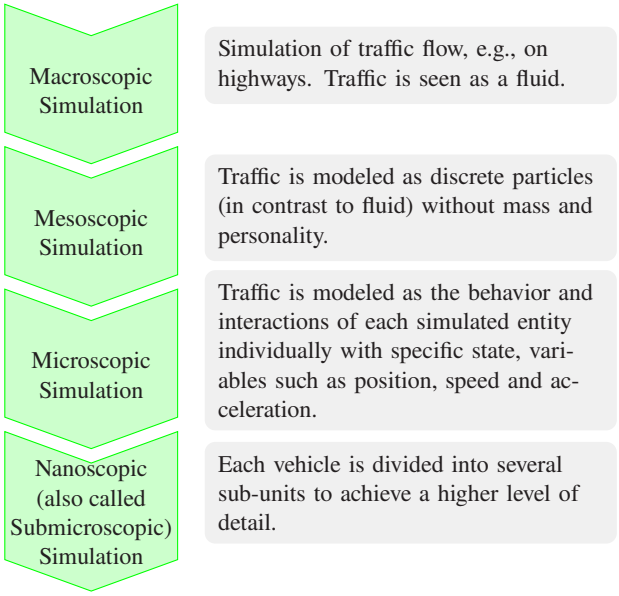


Figure 3.4: Simulation resolution levels: 4 established levels of resolution in traffic simulation [110, 131]

3.2 Mathematical Background

This section describes the main mathematical approaches underlying the concept (see Chapter 4) and experiments (see Chapter 5) in this work.

3.2.1 Gaussian Processes

In this work, Bayesian optimization with Gaussian processs (GPs) is employed as an optimization approach for identifying the most critical concrete scenario with respect to the utilized criticality metrics. Prior to the execution of any simulations of concrete scenarios, there is no information available regarding the criticality within the parameter space of a logical scenario. By simulating

concrete scenarios, a set of observations can be collected. These observations $\mathcal{D}$ representing the simulated concrete scenarios, can be written as

$$\mathcal{D}_{1:i} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ...(\mathbf{x}_i, y_i)\}, \tag{3.1}$$

where $\mathbf{x}$ describes the input vector and y the output. The goal is to find a surrogate model that approximates an objective function f with the given set of observations [122]:

$$y = f(\mathbf{x}) + \epsilon, \tag{3.2}$$

where ϵ is the noise. This is not present if the simulation of a concrete scenarios is deterministic. Equation (3.2) can be described as a standard linear regression model

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}, \tag{3.3}$$

where $\mathbf{w}$ is the vector of weights of the linear model. Finally, the conditional distribution of y given $\mathbf{x}$ can then be written as:

$$\begin{aligned} p(y|\mathbf{X}, \mathbf{w}) &= \prod_{i=0}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= \prod_{i=0}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} exp(-\frac{(y_i - \mathbf{x}_i^T \mathbf{w})^2}{2\sigma_n^2}) \\ &= \mathcal{N}(\mathbf{X}^T \mathbf{w}, \sigma_n^2 \mathbf{I}), \end{aligned} \tag{3.4}$$

where $\mathbf{X}$ is a $D \times n$ matrix of the input observations with dimension D. Nevertheless, in specific contexts, linear models may prove inadequate, necessitating the use of feature mapping to effectively address non-linear dependencies. For this, the inputs can be projected into a high-dimensional feature space $\mathbb{R}^N$ of dimensionality N with

$$\Phi : \mathbb{R}^D \to \mathbb{R}^N. \tag{3.5}$$

After the mapping, the parameter vector is now of length N. With that, the linear model in Equation (3.3) can be adjusted:

$$f(\mathbf{x}) = \Phi(\mathbf{x})^T \mathbf{w}. \tag{3.6}$$

As a linear combination of n kernel functions it is written as [122]:

$$\overline{f}(\mathbf{x}_*) + \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*), \tag{3.7}$$

where $\alpha = (K + \sigma_n^2 I)^{-1}\mathbf{y}$. The optimal solution in this case can be represented as a linear combination of the training data points (representer theorem). The representer theorem enables the "kernel trick" [151]. Instead of projecting into a high-dimensional space, it computes the scalar product directly in the original space, capturing the relationships between data points without the need for transformation. This scalar product is defined as $k(\mathbf{x}, \mathbf{x})$ or covariance function. In this work, the Radial Basis Function (RBF) kernel from Equation (3.10) is used. From that, the predictive equations of the mean $\overline{f}_*$ and the variance $\mathbb{V}[f_*]$ are as follows [122]:

$$\overline{f}_* = K_*^T (K + \sigma_n^2 I)^{-1}\mathbf{y}, \tag{3.8}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1}\mathbf{k}_*, \tag{3.9}$$

with the kernel function $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ where $K = K(\mathbf{X}, \mathbf{X})$ and $K = K(\mathbf{X}, \mathbf{X}_*)$, both the $n \times n$ matrix of the covariances of all pairs of training points or for K_* the pairs of training and test points.

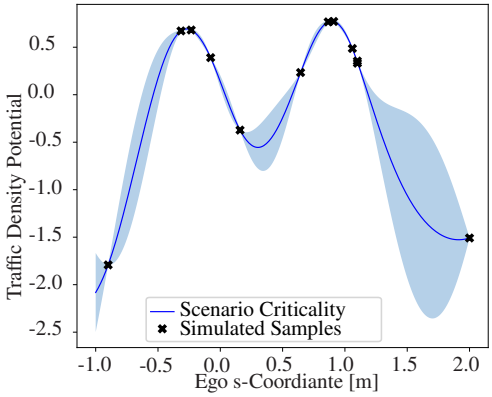


Figure 3.5: One-dimensional example of a Gaussian process with a 95% confidence interval and sampled by Bayesian optimization.

Figure 3.5 illustrates an example of a GP in a one-dimensional parameter space for Traffic Density Potential (TDP). The blue plot (scenario criticality) represents the mean of all functions that fit the observations (black x's). The light blue area around the mean indicates the confidence interval according to the model. New points $\mathbf{x}_*$ are predicted along the mean with the variance of the model.

A typical kernel for Gaussian processes is the RBF kernel [68]. It describes that the correlation of the outputs decreases with a growing distance between two points, where θ defines the length scale at which this happens. As stated by Schulz et al. [133], the RBF kernel used by the Gaussian process inherently makes assumptions about the underlying objective function, such

as smoothness. Moreover, the Gaussian process makes two assumptions about the noise in the used dataset [101]. This work employs kernel methods for Gaussian process (GP) and Principal Component Analysis (PCA) (cf. Section 3.2.3). Hofmann et al. [75] describe kernels as a non-linear approach used in machine learning when linear methods are insufficient to detect dependencies for prediction and classification of properties. In this work, the RBF kernel is used as defined by Schulz et al. [133]:

$$RBF : k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{1}{2\lambda^2}\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{3.10}$$

Usually, it assumes that the correlation between two data points (e.g., criticality of two distinct concrete scenarios) decreases with the distance in the parameter space between these points. In conclusion, the closer the two points are, the more likely they are expected to behave in the same manner (e.g., have similar criticality results). This assumption is called stationarity and can be tuned by the hyperparameter length-scale λ and signal variance σ_f^2. This work utilizes the RBF kernel in the optimization step and later in the preprocessing step for clustering. For the optimization, different kernels, such as RBF and polynomial, are proposed by Mika et al. [105]. However, due to limited data availability and a lack of comprehensive knowledge about the scenario space (cf. 4.5.1.3), only the Gaussian radial basis function RBF kernel can be used and is, thus, explained further. In general, Gaussian RBF kernels are a popular choice since they only have one parameter (kernel width θ) for hyperparameter tuning and are flexible enough to capture non-linear relationships between data points. It allows for the use of linear methods on the kernel.

3.2.2 Bayesian Optimization

The GP methodology generates a surrogate model for the criticality within a logical scenario parameter space. At this point, a strategy for selecting a new potential candidate for simulation is required. Bayesian optimization (BO) is a strategy for optimizing black-box functions [68], aiming to find the optimal set of input parameters that minimize or maximize a given objective function $f(\mathbf{x})$. It is commonly employed for hyperparameter tuning in machine learning algorithms [61] and iteratively minimizes or maximises an objective function f. Concerning scenario exploration, the objective function describes the relationship between input parameters (concrete scenario) and the criticality of a scenario. Each point $\mathbf{x}_i$ in the input space corresponds to scenario parameter set defining the i-th concrete scenario. Observations y_i describe the associated outcome of simulating a concrete scenario, i.e., criticality values. The GP can be used as a surrogate model for result prediction and further analysis of the scenario space (Section 4.7). The need for a surrogate model excludes algorithms like evolutionary learning, which can find a reduced parameter set but cannot

make predictions for non-simulated sets within the parameter space. Due to its fast computation and an exhaustive number of existing implementations, Bayes optimization was used for the rest of this work. Note that Bayes optimization only works for parameter spaces with up to 15-20 parameters [68]. In case this number is exceeded, other methods have to be explored.

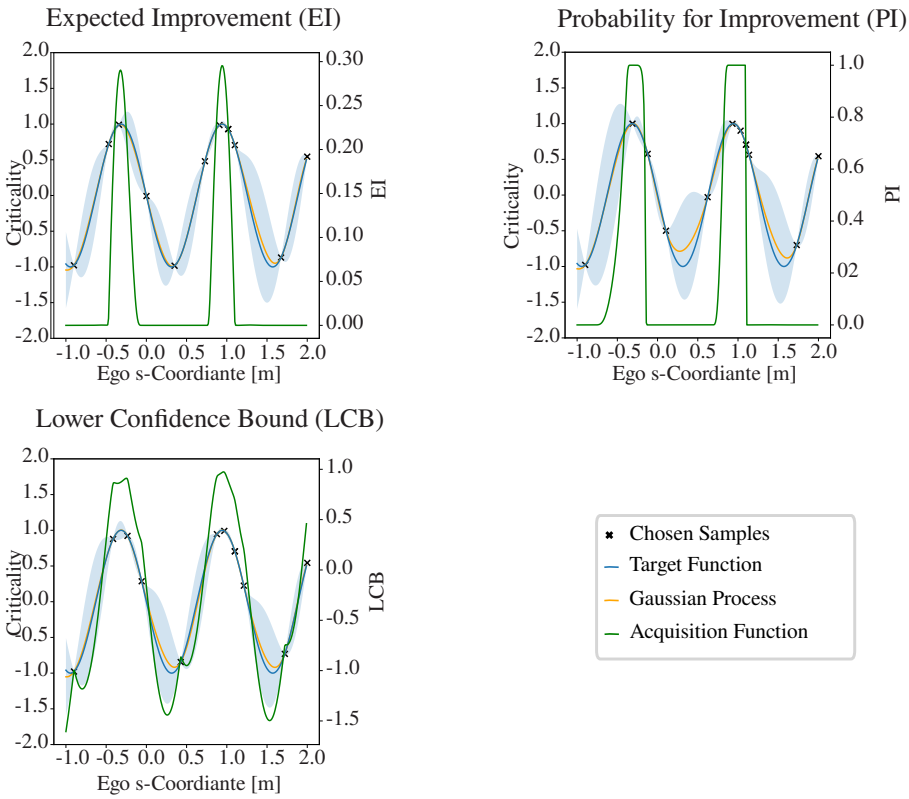


Figure 3.6: One-dimensional example acquisitions functions EI, PI, and LCB. The green curve represents the acquisition function for each point in the space of the target function (blue). The orange curve represents the predicted curve of the Gaussian process. The black x symbols describe the already chosen and simulated concrete scenarios. All three acquisition functions (green) were applied to the same underlying logical scenario.

Acquisition functions are used to evaluate the beliefs about the objective function regarding the input space, based on the predicted mean $\mu_t(\mathbf{x})$ and uncertainty $\sigma_t(\mathbf{x})$ and chooses the most promising parameter set [68]. It guides the selection of the next set of parameters or concrete scenarios to be evaluated based on a trade-off between exploration (sampling in unexplored regions) and exploitation (sampling in promising regions).

The sampling takes place iteratively, where parameters are selected sequentially based on the current state of the surrogate model and the acquisition function. Uncertainty estimation is

typically in the form of confidence intervals or posterior distributions, guiding exploration by balancing exploitation and exploration trade-offs.

Typical acquisition functions are [68]:

- **Probability of Improvement (PI)**: PI chooses the point (concrete scenario) where improvement of the objective function (criticality) is most likely [68].
- **Expected Improvement (EI)**: EI is the most commonly used acquisition function [61] and assumes noiseless observations. It computes the expected value for a point (concrete scenario) by considering the best value to optimize its choice. This approach works similar to PI but additionally considers the magnitude of improvement.
- **Lower Confidence Bound (LCB)**: It is designed to find a balance between exploration and exploitation. This is done by a given learning rate that controls the trade-off between both goals, e.g., a high learning rate favors exploration [150].

Figure 3.6 shows the three acquisition functions for the same one-dimensional parameter space of a logical scenario. The orange curves represent the Gaussian process after ten sampling steps with BO. The blue curves describe the target or objective function or criticality with observed (x) concrete scenarios regarding the scenario parameter set (ego s-coordinate). The green curves depict the three different acquisition functions.

Bayes Optimization Examples

Figure 3.5 provides an example for a one-dimensional Bayes optimization with GP problem of a logical scenario with one parameter. The x-axis describes the criticality value (traffic density potential) where higher values mean more critical concrete scenarios. The y-axis displays the parameter space, or since it is one-dimensional, also the parameter range $egos - coordinate \in [-1, 2.0]$. The example shows 13 data points (x), an estimated scenario criticality for the complete parameter space (dark blue line), and a confidence interval for the parameter space (light blue surface). Typically, in simulating concrete scenarios, the experiment involves executing more scenarios in regions where high criticality is expected. However, regions of uncertainty, such as the rightmost part of the graph shown in Figure 3.5 with a diverging confidence interval, are more likely to be selected for the next simulation step, depending on the acquisition function.

Figure 3.7 gives a 2-dimensional example for Bayes optimization. In this example, input variables 1 and 2 stand for two scenario parameters as parameter ranges. Both parameter ranges are defined as $p_1, p_2 \in [0, 5]$. Examples of input parameters are positional data or vehicle velocity. Figure 3.7 (a) displays the predicted mean for the parameter space, with the yellow end of the

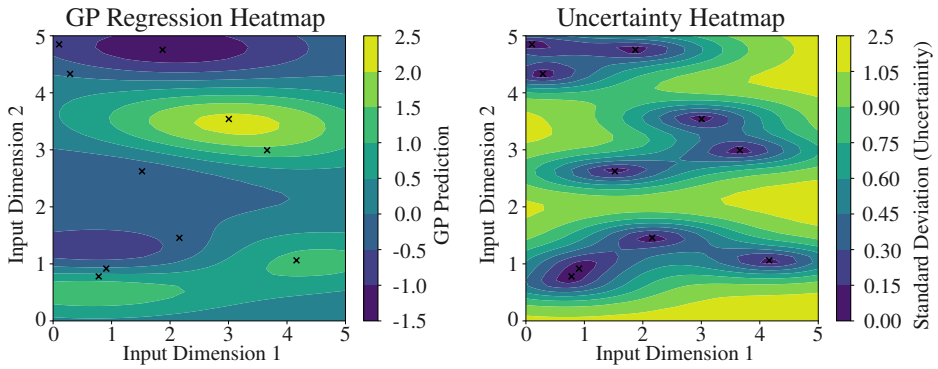


Figure 3.7: 2-dimensional example of Bayes optimization with uncertainty.

color bar indicating high criticality and the blue end indicating low criticality. The parameter space contains both critical and low-critical areas. Therefore, the parameter space exhibits areas where critical scenarios can be found, as well as low-critical areas. Figure 3.7 (b) shows the standard deviation, representing the uncertainty within the parameter space. In Figure 3.7 (b), one can observe that areas near a simulated point have a higher certainty than areas further away.

Comparison to Other Methods

The naive approach to scenario exploration is to employ a uniform distribution for all given parameter ranges in a logical scenario, implemented through techniques such as grid search with a predefined step size where the space is uniformly sampled and simulated. However, Mori et al. [106] conducted experiments that revealed potential issues with using a discrete set of predefined parameters within a given range. The authors note that when using a perception algorithm, outliers and fluctuations may be missed by grid sampling, leading to error rates that depend on the grid size. The error rate increases with larger grid step sizes, but does not decrease after a certain small step size threshold. The authors suggest that the main issue is the potential for outliers and fluctuations to be overlooked during scenario-based testing. Another problem of grid search is, that it is computationally intensive and becomes exponentially more complex as the number of parameters increases. To address this issue, optimization algorithms and approaches are proposed as alternatives.

One alternative approach for discovering critical scenarios involves utilizing Evolutionary Algorithms (EAs). EAs are inspired by biological principles of fitness and use the same principle for optimization [35, 14]. The exploration within the parameter space occurs through random searches across multiple experiment runs. An EA starts with a randomized initial population of

individuals, each representing a concrete scenario. Each individual is assessed for its fitness, i.e., the concrete scenario is simulated and evaluated regarding its criticality.

If the goal is to find more critical scenarios, a part of all individuals, e.g., half of them, is discarded and replaced with a new set of individuals after evaluating the whole generation. These new individuals can either be randomly generated or produced using the remaining individuals as a parent generation. This generational approach incorporates three fundamental components: a selection mechanism, a mutation mechanism, and a fitness function. The **selection mechanism** determines the parents selected for producing offspring in the subsequent generation. It significantly impacts the convergence rate, steering the exploration towards potentially better solutions. However, maintaining population diversity is crucial to prevent the dominance of a single highly fit solution, thereby avoiding premature convergence. The **mutation mechanism** introduces minor random variations to individuals, fostering the creation of new solutions while preserving genetic diversity. Mutation plays a pivotal role in exploring the search space and discovering potentially optimal solutions, typically applied with a low probability to prevent exhaustive random searching. The **fitness function** which is central to the EA process, evaluates the quality or fitness of a candidate solution in terms of criticality with respect to the problem at hand. This function determines an individual's survival chance for the next generation or elimination if the individual is unfit for progression. [1]

A second alternative is reinforcement learning [27]. In this approach, a reinforcement learning algorithm explores the parameter space of a logical scenario to find concrete scenarios. Again, the concrete scenarios are generated and executed in a simulation environment. The results are evaluated using criticality metrics, which can be used as a reward function for generating new test cases. The higher the criticality, the higher the reward given to the model. Like for EAs, this iterative cycle employs the evaluation outcomes as a feedback loop to guide the generation of new scenarios. By leveraging these criticality-driven rewards, the reinforcement learning algorithm iterates through the parameter space, concentrating on scenarios that exhibit increased levels of criticality.

Compared to the methods mentioned above, Bayesian optimization (BO) with Gaussian processs (GPs) (see Section 3.2.2) is the algorithm of choice in this work since it provides information about the confidence of the criticality at the areas around simulated data points (concrete scenarios). This information can later be used to make statements about the quality of the simulation results. However, this information is based on assumptions made in the model design step, such as the choice of a Gaussian RBF kernel. An analysis of the resulting surrogate model obtained from the GP of the optimization algorithm can then be performed in subsequent steps.

[1] Parts of this chapter have already been published in [11]. Additionally, a Master's thesis was supervised [166].

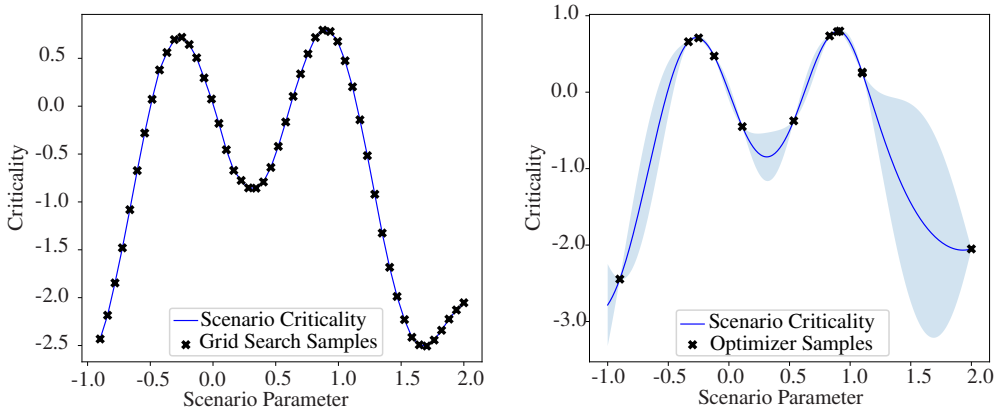


Figure 3.8: Comparison of grid search (a) and optimization (b) on a one-dimensional parameter space.

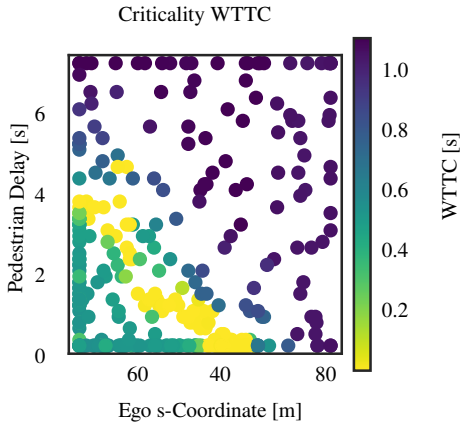


Figure 3.9: Example of a 2-dimensional parameter space after Bayes optimization took place.

Figure 3.8 provides a visual comparison between grid search and BO with GP in a one-dimensional scenario space. The left-hand side illustrates the grid search approach, where parameter sampling is performed with a fixed step size resulting in 50 samples. In contrast, the right-hand side demonstrates the BO method, where all 10 samples are chosen by the optimization algorithm, with the light blue region indicating the uncertainty or confidence interval associated with that specific part of the parameter space. This uncertainty reflects the level of confidence in the given scenario results, considering the limited number of samples and the inherent variability in the scenario exploration process. Moreover, Figure 3.9 shows a 2-dimensional parameter space after a successful BO with GP was used. The criticality depicted is WTTC and shows clear critical (low values, yellow) and non-critical (high-values, purple) areas. Results shown in Figure 3.9 are further explained and evaluated in Section 5.3.1. Uncertainty information is the reason why

BO and GP were chosen for this work. EAs and reinforcement learning usually do not provide information about unknown regions.

3.2.3 Kernel Principal Component Analysis

Principal Component Analysis (PCA) is a method for reducing data dimensionality while retaining most of the data variation in the dataset. It identifies the principal components, which are the directions in which the variation of the data is highest [126]. Kernel PCA is a feature extraction approach that is a useful preprocessing step for classification and clustering tasks. It projects the original data into a higher-dimensional feature space so that a linear PCA can be applied [105]. Therefore, this technique can achieve a dimensionality reduction for a non-linear feature space.

Principal Component Analysis

Linear PCA diagonalizes the covariance matrix with a set of M centered observations $\mathbf{x_k}, k = 1, ..., M, \mathbf{x_k} \in \mathbb{R}^N, \sum_{k=1}^{M} = 0$:

$$C = \frac{1}{M} \sum_{j=1}^{M} \mathbf{x_j} \mathbf{x_j}^T. \tag{3.11}$$

For this, the eigenvalue equation has to be solved:

$$\lambda \mathbf{v} = C\mathbf{v} \tag{3.12}$$

where $\lambda \geq 0$ and $\mathbf{v} \in \mathbb{R}^N \setminus \{0\}$. From Equation (3.11) and Equation (3.12)

$$C\mathbf{v} = \frac{1}{M} \sum_{j=1}^{M} (\mathbf{x_j} \mathbf{v}) \mathbf{x_j}^T. \tag{3.13}$$

can be derived and Equation (3.12) can be seen equivalent to

$$\lambda(\mathbf{x_k} \cdot \mathbf{v}) = (\mathbf{x_k} \cdot C\mathbf{v}) \text{ for all } k = 1, ..., M. \tag{3.14}$$

The resulting eigenvectors represent the principal components after this projection.

Kernel PCA is based on linear PCA and utilizes the following projection of observations into the feature space:

$$\Phi : \mathbb{R}^N \to F \tag{3.15}$$

Instead of the observations from the feature space the mapping above is used in Equation (3.11). Now, the covariance matrix C in a feature space F can be defined as follows under assumption that centered data is used [134] :

$$C = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x_j}) \Phi(\mathbf{x_j})^T \tag{3.16}$$

Kernel PCA Example

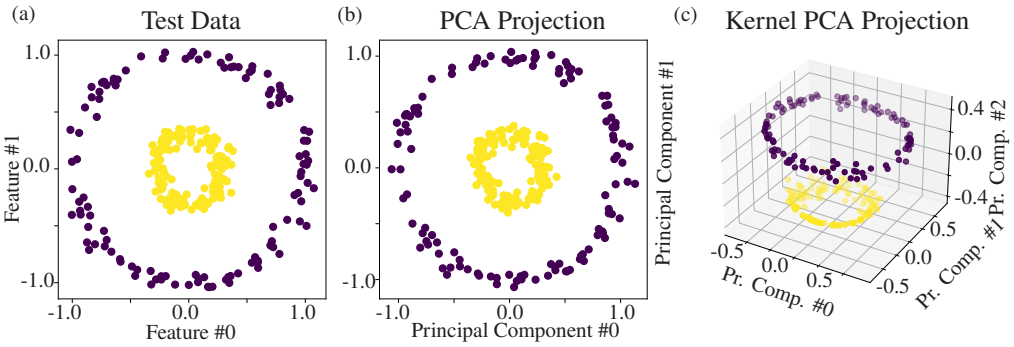


Figure 3.10: (a) Plot of 2-dimensional artificial test data for an easier understanding; (b) Projection of test data after a PCA; (c) Projection of test data in 3-dimensional kernel space.

Figure 3.10 depicts an artificial test dataset (a) with two classes located in circles that cannot be separated by a standard PCA. In Figure 3.10 (b), this dataset is projected onto two principal components using a standard PCA (b) and onto a 3-dimensional space using kernel PCA Figure 3.10 (c). Notably, Figure 3.10 (c) exhibits a possible spatial separation compared to the data in Figure 3.10 (b) since after PCA the artificial dataset is not linearly separable. It is important to note that PCA only enables projection into the dimensions of the original feature space, whereas kernel PCA supports an infinitely high feature space.

In scenario exploration, each feature in Figure 3.10 (a) stands for a parameter range, and both features span the parameter space. Therefore, in this case, there is no continuous criticality value. However, the criticality classification can be "collision" (yellow) and "no collision" (blue). The kernel PCA is then used as a preprocessing step to make clustering easier than in the original parameter space. The projection in Figure 3.10 (b) does not provide any additional value in contrast to Figure 3.10 (c), where the data can be divided into a third dimension after projecting onto a 3-dimensional space.

In Section 4.5.1.3, a kernel PCA with RBF kernel is used for the projection into a higher dimensional space. However, the spacial distance between the data points (concrete scenarios) cannot be used. Therefore, $\|\mathbf{x}-\mathbf{x}'\|$ is replaced by the DTW distance matrix D (see Section 4.5.1.3) with behavior-based (see Section 4.5.1.2) or criticality-based (see Section 4.5.1.1) distance leading to

$$k(D) = exp\left(-\frac{1}{2\theta^2}D^2\right) = exp\left(-\gamma D^2\right) \tag{3.17}$$

3.2.4 Dynamic Time Warping

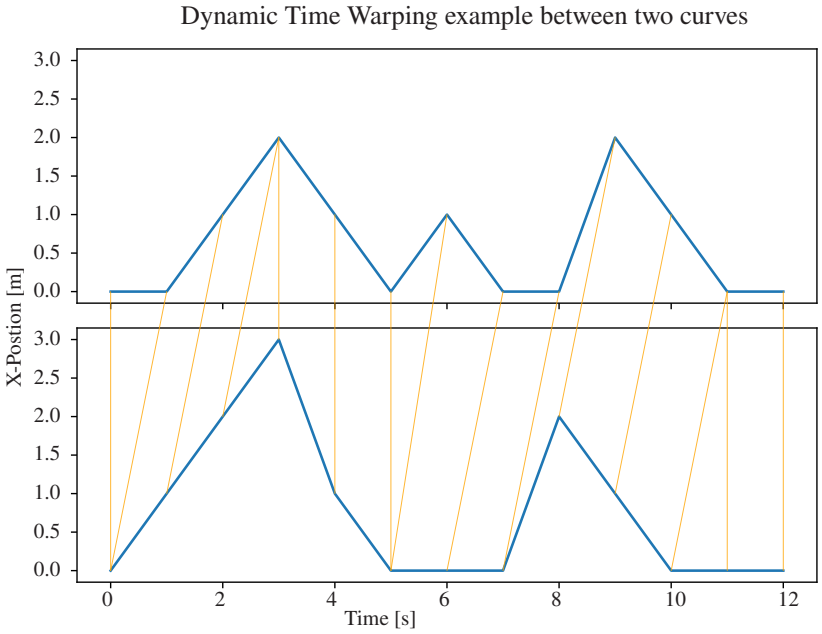


Figure 3.11: Example of dynamic time warping between two one-dimensional trajectories.

According to Senin [136], Dynamic Time Warping (DTW) is a measure to compare time-series that assesses the distance between two trajectories with related shapes but different phases, taken from equidistant points in time. Similarity via distance metrics is described by small values, which increase with more differences and cannot be less than zero. In the following equation, d denotes the distance measured in a feature space Φ (e.g., x- and y-coordinates of the trajectory), and both trajectories $X, Y \in \Phi$ [136]:

$$d : \Phi \times \Phi \rightarrow \mathbb{R} \geq 0 \tag{3.18}$$

DTW aims to minimize the distance as a cost function. It starts with calculating a pairwise distance matrix between all points of both time-series, called the local cost matrix. The goal of the algorithm is to find the alignment path through the lowest possible points (valleys) on this cost matrix. This path is called warping path or warping function and assigns each point $x_i \in X$ to a point $y_j \in Y$, and therefore results in a sequence of tuples of points [136]:

$$p = (p_1, p_2, ..., p_L) \text{ with } p_l = (n_i, m_j) \in [1 : N] \times [1 : M] \text{ for } l \in [1 : L] \tag{3.19}$$

An example is shown in Figure 3.11, where each point in both time-series is assigned to at least one point in the other. These points have to fulfill a list of criteria [136]:

- The start and end points must be the first and last points in each time series.
- The pairing has to be monotonous: $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$. In Figure 3.11, this criterion is fulfilled since there are no crossing orange lines.
- The path is not allowed to make jumps (leave out points in between). Therefore, the step size between two points is defined as follows: $p_{l+1} - p_l \in \{(1, 1), (1, 0), (0, 1)\}$.

This work utilizes Dynamic Time Warping (DTW) [136] as a distance measure for vehicle trajectories and the criticality curves obtained from simulating a concrete scenario. Furthermore, the coordinates of a two-dimensional trajectory can be extended by the speed as an additional dimension of the ego vehicle when braking and accelerating need to be considered. DTW is chosen due to its ability to handle temporal distortions, variation in the duration and the speed of trajectories. In many simulated scenarios, the trajectory of the ego vehicle remains mostly the same. However, it must react and brake at various times during the simulation in response to different adversary behaviors. Thus, DTW can quantify the difference in ego behavior in concrete scenarios.

3.2.5 Clustering

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [55] was chosen for clustering scenarios as an unsupervised clustering method that does not require prior knowledge of the number of clusters. An advantage of DBSCAN is that it can find clusters of any shape, and unlike k-means, it does not assume clusters to be convexly shaped. It only has two input hyperparameter that require tuning: the minimum number of points of a cluster and the size of the neighborhood of this cluster.

According to Ester et al. [55], the algorithm starts with an arbitrary point p and searches for density-reachable points from this starting point based on the hyperparameters, the minimum number of points ($MinPts$) and neighborhood size (Eps). In case p is a core point, a cluster can be discovered. On the other hand, if p is a border point, no density-reachable points can be located, and the algorithm moves on to the next point. A point is direct density-reachable if

$$p \in N_{Eps}(q) \tag{3.20}$$

$$|N_{Eps}(q)| \geq MinPts \tag{3.21}$$

where $N_{Eps}(q)$ describes the neighborhood of point q. Following this definition, a point p is density-reachable from a point q with respect to Eps and $MinPts$ if there is a chain of points $p_1 ... p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

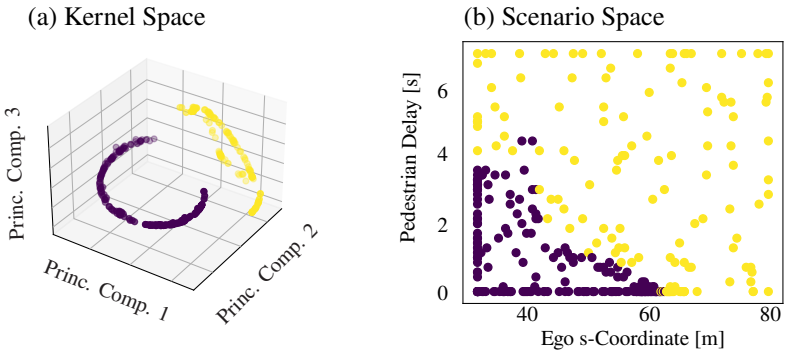


Figure 3.12: (a) Example for behavior-based clustering with DBSCAN in the kernel space. The axes are the three most influential principal components; (b) Cluster assignments from the kernel space taken back to the original scenario space.

Figure 3.12 shows a 2-dimensional parameter space, first projected into a 3-dimensional space (a), where DBSCAN was used to find two major clusters. (b) depicts the same cluster assignment in the original space. In both (a) and (b), each point represents a parameter set defining a concrete scenario (cf. Section 4.5.3 and Section 5.3.2).

Alternatives

The clustering algorithm used must determine the appropriate number of clusters. One possible alternative is hierarchical clustering [159]. This unsupervised technique is based on the distance between data points and builds a binary merge tree. Agglomerative hierarchical clustering begins with all data points as leaves and merges the closest points pairwise. In the next step, points

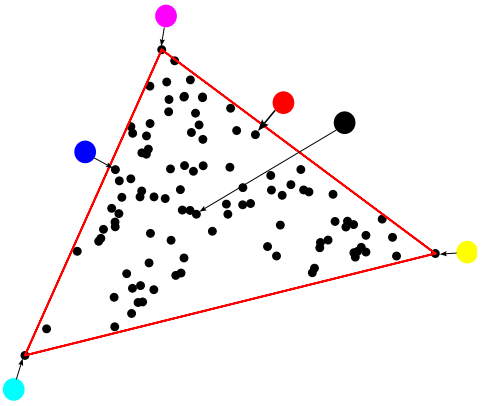
that already have the closest neighbors are merged with the previous pair until a given distance threshold is reached. Divisive hierarchical clustering begins at the root, which includes all data points, and divides them into two sets. Each point must be assigned to only one cluster, making it computationally expensive ($2^{N-1} - 1$ possible combinations for N data points) [159].

Ordering points to identify the clustering structure (OPTICS) is an alternative algorithm using a density-based approach [19]. However, unlike DBSCAN, it can handle clusters of varying density and consider points assigned to a denser cluster. Like DBSCAN, it uses the parameter ϵ, which describes the maximum distance allowed for a cluster assignment. First, the core-distance of point p is calculated for each point. In a 2-dimensional space, it is like drawing a circle of radius r around a point. r defines the minimum distance necessary to hold the minimum number of points for a cluster within a circle. When this is fulfilled, r is stored as the core-distance. The second distance that has to be computed is the reachability-distance. It defines the distance between two points while ensuring they are within a defined neighborhood density. The reachability-distance of p is the maximum value of the core-distance of p and the distance between p and another point o. When both distances are computed, a reachability graph with cluster assignments regarding the reachability-distance is made.

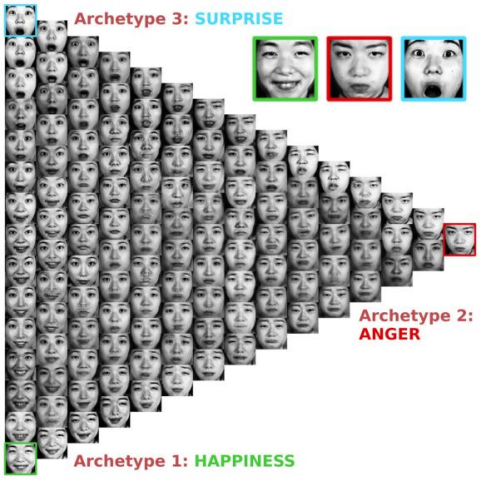
DBSCAN was chosen since it can handle clusters of non-convex shapes and has a comparatively low complexity of ($O(nlogn)$) [159]. It does not need an a priori number of clusters, and compared to OPTICS it tends to have a shorter run time [135]. Additionally, OPTICS can mark peripheral points as noise instead of belonging to a cluster. Therefore, DBSCAN produced better results in the following experiments.

3.2.6 Archetype Analysis

Archetypes are uniquely distinguishable data points or extreme examples within a cluster representing distinct patterns or characteristics. Figure 3.13 shows two examples of such extremes: color (a) and emotions on female faces (b). Therefore, archetypes can be visualized as the *corners* or outer points of a cluster. In theory, the proximity of a data point to the center of the cluster indicates its degree of similarity to the archetypes. Data points that are closer to the center are a combination or mixture of archetypal patterns, while those closer to a single archetype resemble it more. The archetypes can be found at the vertices of the outer boundary of a cluster or hull. Figure 3.13 shows two examples for archetypes: Figure 3.13 (a) depicts the CMYK color space with magenta, cyan, and yellow as the archetypes. The black individual in the middle represents a prototype for this cluster, as it combines all three archetypes to the same degree. The red and blue dots are other illustrations of mixed individuals. Figure 3.13 (b) shows an example from Keller et al. [82] of emotions in Japanese female faces with three archetypes: happiness, anger,



(a) Three archetypes of the CMYK color space. The three corners represent the archetypes of this space and the other points represent individuals with varying degrees of feature manifestation. The black individual in the middle represents a prototype (average).



(b) Three archetypes for emotions in female faces as established by Keller et al. [82] from the JAFFE dataset.

Figure 3.13

and surprise that were analyzed with the help of a neural network. The faces in the middle usually have a more neutral expression. In the case of a logical scenario, it is assumed that these archetypes are scenarios with the highest variability in a nactor's behavior. Unfortunately, the archetype analysis can only be done for convex clusters.

Principal Convex Hull

The analysis is done via a principal hull analysis [43], which estimates the principal convex hull of a dataset. In this case, archetypes are not actual individuals from the exploration step, but rather mixtures of the individuals present in the dataset around the assumed archetype. The selection of archetypes involves minimizing the squared error to achieve an optimal representation of each individual as a mixture of these archetypes. An alternating minimizing algorithm is used to compute the archetypes for addressing a nonlinear least squares problem.

In general, the archetypes are a variant of principal components. As described by Cutler and Breiman [43], for a given feature space $x_1, ..., x_n$ (e.g., scenario parameters), the archetypes $z_1, ..., z_p$ shall be found, where

$$z_k = \sum_{j=1}^{k} \beta_{kj} x_j \tag{3.22}$$

with $\beta_{kj} \geq 0, \sum_j \beta_{kj} = 1, k = 1, ..., p$. Additionally, $z_1, ..., z_p$ minimize the residual sum of squares (RSS):

$$RSS = min_{\{\alpha_{ik}\}} \sum_{i=1}^{n} ||x_i - \sum_{k=1}^{p} \alpha_{ik} z_k||^2 \tag{3.23}$$

where each x_i is approximated by a mixture of $\sum_k \alpha_{ik} = 1$ with $\alpha_{ik} \geq 0$. This leads to

$$RSS = min_{\{\alpha_{ik}\}} \sum_{i=1}^{n} ||x_i - \sum_{k=1}^{p} \alpha_{ik} \sum_{j=1}^{k} \beta_{kj} x_j||^2 \tag{3.24}$$

Cutler and Breiman [43] propose that finding the α's and β's can be solved by general-purpose constrained nonlinear least squares algorithm. However, it is only applicable for smaller problems, and therefore, they offer an alternating constrained least squares algorithm. This algorithm aims to find the set of archetypes (x-mixtures) and their coefficients (α's) for a given dataset. It alternates between optimizing the coefficients for a fixed set of archetypes and optimizing the archetypes for the current set of coefficients through convex least squares problems, ultimately producing the archetypes that best represent the dataset.

Figure 3.14 shows the comparison of a convex hull (a) and principal convex hull (set of archetypes) for the same data. The red dots in (b) depict the archetypes and are not necessarily part of the original samples (simulated concrete scenarios). Other than the convex hull, the number of possible archetypes depends on the dimensionality and size of the feature space. In a logical scenario, these archetypes are usually parameter sets of concrete scenarios that have not been simulated before. The convex hull itself is not suitable for finding the archetypes.

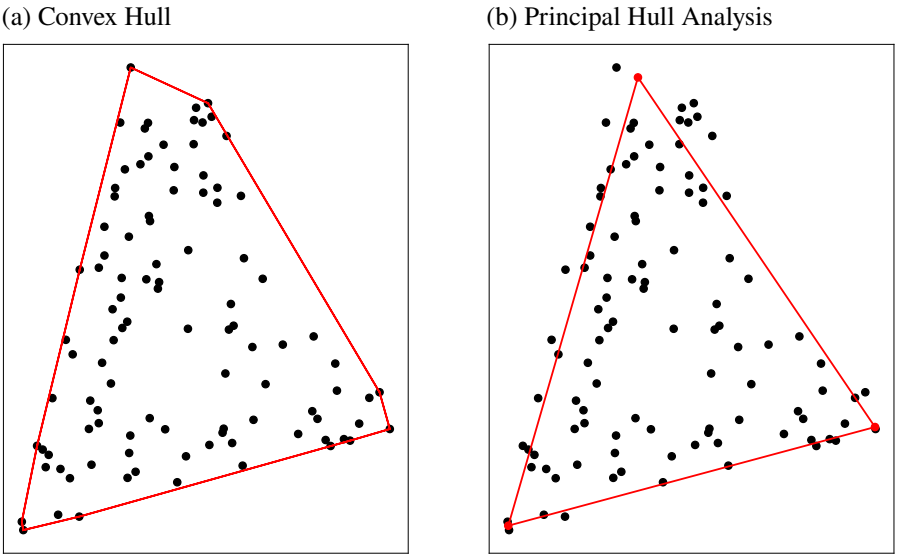


Figure 3.14: (a) Convex hull and (b) principal convex hull with archetypes (consisting of three points) for the same dataset.

4 A Novel Concept for Analyzing Logical Scenarios

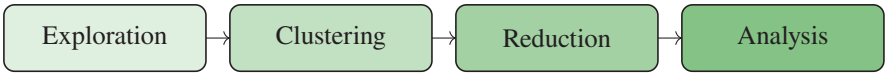


Figure 4.1: Four main steps in the chain of the proposed analysis of logical scenarios.

In the context of this work, scenario exploration (see Section 2.3.3) is extensively investigated, introducing and evaluating a method for scenario exploration (see Sections 4.4 to 4.7). A four-step approach to analyzing logical scenarios is presented in Figure 4.1 and developed further in subsequent chapters. The approach aims to identify significant scenarios, particularly critical ones, and is elaborated upon throughout the chapter. During the exploration step, exemplary scenarios are simulated and evaluated to explore the parameter space. Areas of interest within the scenario parameter space, such as critical situations and collisions, are identified based on their criticality results. Subsequent clustering groups these areas based on similar behavior of the ego vehicle or critical region clustering. These areas are then represented with prototype and archetype scenarios, capturing cluster characteristics. Finally, an analytical step assesses result uncertainty and evaluates confidence in the optimization model.

4.1 Assumptions

A set of assumptions has to be defined for the concepts, experiments, and the used models explained later in this work before the concept is presented. These assumptions serve as the foundation for constructing the concept and determining which simulation tools to use. Additionally, they are important when making decisions about tools, software, or methods.

4.1.1 Logical Scenario Space

A logical scenario (see Section 2.1.3) is an pre-defined scenario that specifies actors, behaviors, or maneuvers. However, some parameters within this scenario are not set and are instead given as parameter ranges or distributions from which concrete scenarios are drawn. A set of concrete parameters then defines a concrete scenario. The parameter space, which is the logical scenario space spanned by the parameter ranges, is referred to as the *scenario space* or *parameter space* throughout this work. Figure 3.9 illustrates an example of a logical scenario defined by a parameter space with two parameter ranges. Each point in this plot represents a parameter combination and, therefore, one concrete scenario. In Figure 3.9, the color represents the criticality measured during simulation for the parameter set.

4.1.2 Automated Driving System

The automated driving system or ego vehicle is assumed to act reasonably, i.e., it tries to avoid dangerous situations, follows traffic rules, and its software components are developed with the best knowledge of the developers and designers. As assumed for the Responsibility Sensitive Safety metric (Section 2.2.5), functional and nominal safety is assumed for the ego vehicle. Functional safety refers to hardware failure or software bugs. The experiments in this work take solely part in SiL environments. Therefore, it is assumed that software bugs are avoided to the best knowledge of the developer. Nominal safety considers the behavior and decision-making of an automated driving system based on the assumption that software and hardware are working error-free.

Additionally, the ego's behavior is deterministic since it should not make different decisions when the same concrete scenario is executed more than once. The driver function used in the following experiments are further explained in Section 5.1.1.

4.2 Idea and Requirements

The objective of this work is to provide a clear and concise analysis of a logical scenario, including its exploration (Section 4.4), clustering (Section 4.5), and a summary of typical and edge cases (Section 4.6). Additionally, a statement regarding the uncertainty and credibility of the results will be included (Section 4.7).

The important parts of this work are illustrated in Figure 4.2, which is an elaborated version of Figure 1.2. While the previous version provided a simplified overview of all fields, Figure 4.2 contextualizes and further details these areas in relation to this thesis. The gradients are used for

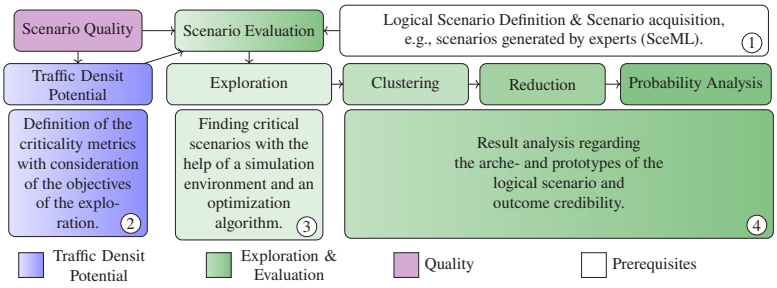


Figure 4.2: Concept of scenario acquisition of this work.

the main categories and include all the shades of the subcategories. The white boxes (1) highlight important areas of recent work relevant to this thesis. First, in a pre-step the logical scenario and criticality metrics for the exploration must be defined (1). The scenario can be found through the different scenario acquisition methods (Figure 2.7), e.g., generated by experts or derived from recorded data. Next, a list of criticality metrics must be compiled to assess scenario quality (purple box). Only scenario quality is mentioned here, as other types of quality are not relevant for this work. The metrics depend on the scenario and must be suitable for the test or research question, e.g., in a car-following scenario, the metrics must adequately assess the criticality of the situation, such as TTC. This work proposes a novel metric suitable for situations involving more than two actors (2) that can be applied in urban environments. Once the appropriate metrics have been selected, the exploration can be started (3), and a simulation environment with an exploration tool is required. An optimization algorithm must be selected apart from the exploration tool setup. Once the exploration is finished and all results are generated, an analysis of the results can be done (4). This analysis includes scenario clustering and reduction to determine the scenario outcome throughout the parameter space of the logical scenario and an edge case identification. From that, the first requirement for this research can be derived:

REQ.1: The used criticality metrics shall be capable of identifying critical situations regardless of the actor constellation. Metrics that are only suitable for car following scenarios are unsuitable for scenarios with more than two actors.

In this context, actors refer to traffic participants, e.g., the ego vehicle, pedestrians, or other vehicles. The *actor constellation* describes the spacial relation between these actors, e.g., car-following situation or intersecting trajectories. The quality of an autonomous driving function or the criticality of a scenario has to be quantified to make a statement about safety of a function. The most fitting metrics for the used case have to be chosen for each use case [2], and the assessment of simulated data varies depending on the stage of testing and development. Generally, metrics can be classified into different levels of abstraction and usage during the process. This classification also determines which metrics are suitable for identifying new scenarios through scenario exploration or generation.

After the question **what** metrics shall be used is established, a second requirement arises:
REQ.2: The scenario exploration tool shall find all critical scenarios within the scenario space using metrics that meet REQ.1
The starting point for scenario exploration is a logical scenario with a set of parameter ranges. Selected values from within these ranges are used to get concrete scenarios. The number of possible concrete scenarios depends on the selection process of values, e.g., a fixed step size of the parameter range or a distribution from which values are drawn and the number of variables within a logical scenario. Since the number of scenarios grows exponentially with each introduced variable or parameter range, the number of concrete scenarios associated with one logical scenario might be too high to execute all available scenarios. Furthermore, not all scenarios provide new insights about an ADS. To reduce simulation time, a guided process can be used to identify critical concrete scenarios.

The next part states what shall be done with the exploration results:
REQ.3: The logical scenario shall be modeled by a reduced set of representative concrete scenarios.
The resulting set of scenarios can be analyzed regarding several aspects. Firstly, different similarity and clustering approaches can be used to make assumptions and statements about the behavior of the ego vehicle and the criticality within the examined logical scenario. Another outcome is the reduction of the resulting set to a smaller subset that does not contain redundant scenarios, which do not provide further insight into the behavior of the ego vehicle and the scenario.

Further, the credibility of the results has to be assessed:
REQ.4: The logical scenario space must be analyzed regarding its credibility and uncertainties.
Reducing simulation effort and the number of scenarios can be beneficial. However, it is crucial to investigate the credibility and uncertainties of the achieved results. It is important to note that making such statements is only possible for some optimization algorithms, which should be considered when selecting an algorithm to use. Additionally, even if it is possible to make statements about this information, it may not be sufficient for evaluating coverage. Furthermore, it is necessary to collect and organize all assumptions related to the model, simulation environment, and algorithm used.

4.3 Taxonomy for Quality in Simulation and Scenario-based Testing

Evaluating a scenario (logical or concrete) is essential to the exploration phase, as described in Section 4.4. Therefore, it has to be established what parts of a simulation and scenario must be evaluated, in particular, so that no information gets ignored. This section aims to fill the gap between scenario quality and criticality in the already existing concepts of simulation and System Under Test (SUT) quality. It describes quality from the point of view of a scenario and locates it among existing concepts by drafting a taxonomy.

Before, during, or after a simulation, various aspects need to be evaluated to ensure the reliability of the simulation. These aspects include the quality of all parts of the simulation: the simulation models employed, such as sensor models or vehicle dynamics models, as well as their coupling mechanisms and the simulation environment itself. These models should meet a predetermined level of quality to ensure their intended purpose is achieved. Once this level of quality is met, the Automated Driving System (ADS) can be tested in a simulation to determine its intended functionality. Typically, the performance of the SUT, the driving function, and its behavioral safety are assessed. In the scope of this work, the SUT is a software model of the driving function (Section 5.1.1). For instance, the adherence of the system to traffic rules and functional safety is evaluated [154]. [1]

4.3.1 Quality-related Terms and their Interactions

Figure 4.3 shows a UML diagram that visualizes the relationships between different quality aspects within a simulation. In scenario-based testing, a **Simulation** contains one or more **Scenarios** (Section 2.1.3), and a SUT. The simulation, scenario, and SUT must meet specific quality requirements and testing standards. However, different types of quality must be ensured at different stages. First, the **Simulation Environment Quality** must be evaluated before it can be used. Additionally, the **Scenario Quality** must be guaranteed. The scenario must be sensible regarding the testing of the SUT, i.e., the **Structural Scenario Quality** must be evaluated concerning the credibility of the scenario. The structural scenario quality describes the required level of realism for testing purposes. This quality is difficult to measure automatically, and there is limited prior research available. The degree of credibility can also vary depending on what needs to be tested. For some test cases, it might be acceptable if actors have no collision detection and pass through each other. After that, the **Functional Scenario Quality** must be ensured. It

[1] Parts of this chapter have already been published in [1].

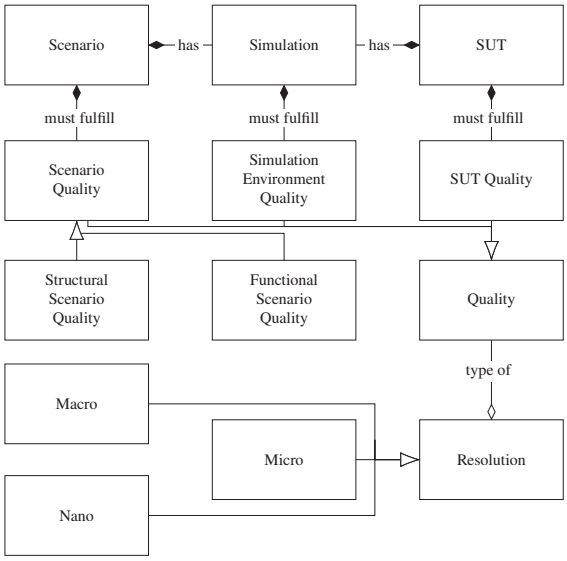


Figure 4.3: UML diagram visualizing the proposed terms for quality and their relations, providing an overview of general terms.

might not matter to the behavior of a SUT if the environmental light is the primary focus in a scenario when a radar sensor is tested. Different levels of resolution are available for each type of quality (Section 4.3.4).

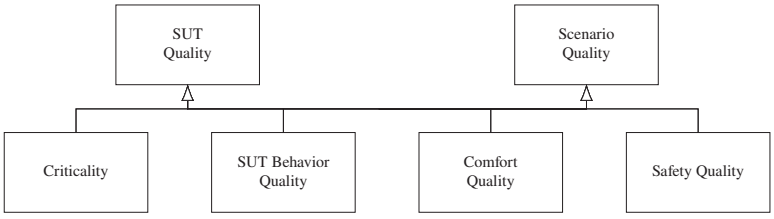


Figure 4.4: UML diagram displaying examples for specific quality types and aspects.

Figure 4.3 shows different aspects (however, not a complete list) of SUT and Scenario Qualities. Both mainly differ in the point of view regarding what category they fall into, e.g., if criticality metrics are used to assess the criticality within a scenario, then it is of type **Scenario Quality** if they are used to evaluate a SUT's performance it is called **SUT Quality**.

This work focuses on scenario quality since the exploration explicitly evaluates the outcome and events in a concrete scenario. Therefore, simulation environment quality and SUT quality are mentioned here. Detailed information and examples can be found in Appendix B.1.

4.3.2 Taxonomy Overview

To facilitate consistent understanding and naming, as well as to establish a clear delineation of different simulation aspects, a taxonomy is proposed. The proposed taxonomy (Figure 4.5) encompasses three domains of interest for ensuring quality during the simulation process: simulation environment quality, system under test (SUT) quality, and scenario quality.

Each domain can be examined at various levels of resolution, similar to the approach used in traffic simulation, which includes nanoscopic, microscopic, and macroscopic levels. The suggested combinations of domains of interest and resolution levels are presented in Figure 4.5 in a tabular format.

Overall, the proposed taxonomy provides a comprehensive framework for evaluating simulation quality that is flexible enough to accommodate a wide range of simulation aspects and can be applied to different levels of detail.

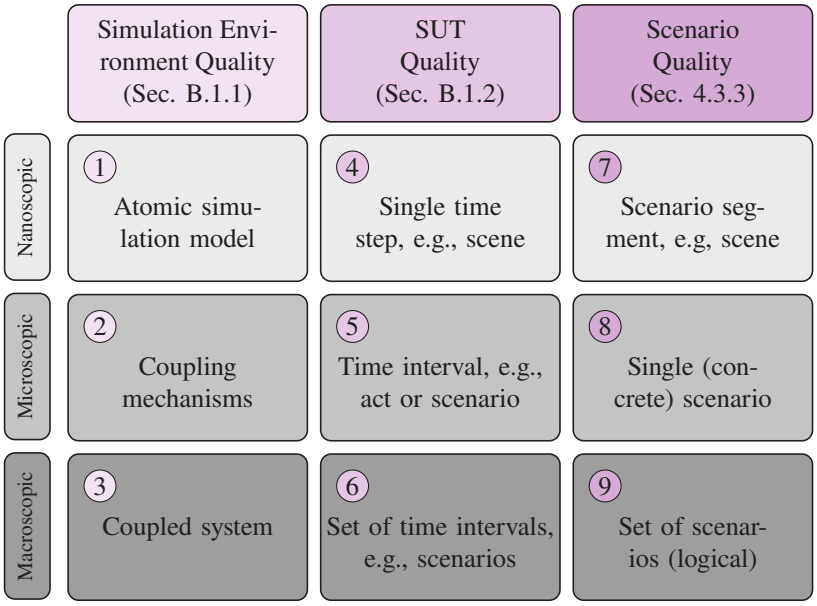


Figure 4.5: Quality matrix with three quality categories and resolution levels: simulation environment quality, system under test (SUT) quality, scenario quality

4.3.3 Domains of Interest

The taxonomy consists of three domains of interest from the simulation field, but it can be extended to include other domains if necessary.

Simulation Environment and System Under Test Quality

Simulation environment quality, the quality of the used simulation tools, pertains to the assessment and assurance of the fidelity of simulation models, their interconnections, and the overall infrastructure of the simulation framework. The simulation tools used in this work (Section 5.1.1) are one example. The quality of a SUT (Section 5.1.1) encompasses evaluating its observable behavior and performance concerning intended functionality based on predefined requirements. Various types of quality, such as safety and performance in critical scenarios, can conflict with each other due to trade-offs in specific metrics. Regarding the scope of this work, an more in-depth explanation can be found in Appendix B.1.

Scenario Quality

The concept of scenario quality aims to provide an unbiased assessment of a scenario, i.e., how *good* a scenario is. This means that the evaluation is based solely on the ego vehicle, its motion model, and a single traffic participant. However, urban traffic scenarios typically involve more than two road users, and some metrics may not fully assess the criticality of certain situations. To address this issue, additional objective metrics are required. For instance, the metric for traffic quality can be divided into domains of interest of varying sizes [70]. Each domain covers a range from the ego vehicle to a large road section with multiple other road users.

Scenarios, or their components, can be evaluated for their quality during the development and testing of a system, e.g., to assess the criticality of a situation with respect to the system under test. To establish scenario-based testing as a testing standard, scenarios must represent realistic situations and traffic participants, which overlaps with simulation model quality.

Scenario evaluation encompasses spatial or temporal scenario segments, a complete concrete scenario, or a set of scenarios that correspond to logical or functional scenarios. The field of scenario quality is closely related to system under test quality [149, 86], but differs in terms of the perspective from which a simulation is evaluated. In particular, the goal is to assess either the system under test performance (SUT quality) or the scenario quality or criticality itself, in order to identify appropriate scenarios for testing.

The evaluation of scenario quality is based on the ego vehicle, its motion model, and a single traffic participant, using an objective point of view. However, urban traffic scenarios typically involve more than two road users, and some metrics may not fully evaluate the criticality of certain situations. Standard criticality metrics, e.g., TTC, PET, or distance, can only consider one other road user. However, urban traffic typically involves more than two road users, and many metrics may not fully evaluate the criticality of these situations. To resolve this issue, it is necessary to employ more objective metrics. For example, the traffic quality metric can be divided into differently sized domains of interest [70] (see Section 2.2.5), each covering a range from the ego vehicle itself to an extensive road section with multiple other road users.

4.3.4 Levels of Resolution

The levels of resolution describe the granularity at which a domain of interest is investigated. The level with the highest resolution and, in most cases, the atomic level with the smallest units possible is the nanoscopic level. The nanoscopic level includes the observation of single time steps or sub-sequences of scenarios, such as atomic modules or units, or spatial sub-sections from a scene, such as only one or two traffic participants from a map with more than the observed objects.

Microscopic resolution is the next level and describes coupled entities from the nanoscopic level or their coupling mechanism. For instance, this can include the coupling between simulation model units or designated time intervals, such as a concrete scenario.

The final level is the macroscopic level, which provides a more holistic view. This level encompasses the quality of coupled systems, the quality of the system under test over a set of tests or scenarios, and the quality or criticality of a set of scenarios.

The mentioned domains of interest and levels of resolution lead to the following list. The numbers correspond to the numbers of the table entries in Figure 4.5:

1: Nanoscopic simulation (environment) quality: quality of single atomic simulation models, e.g., pedestrian model or subunits of opponent vehicles,

2: Microscopic simulation (environment) quality: quality of coupling, e.g., synchronization, message format, or actor availability,

3: Macroscopic simulation (environment) quality: quality of coupled system, e.g., opponent vehicle or simulation environment module.

4: Nanoscopic SUT quality: quality of system under test for one time step, e.g., headway or gap time,

5: Microscopic SUT quality: quality of system under test for a time interval, e.g., post-encroachment time or encroachment time,

6: Macroscopic SUT quality: quality of system under test for a set of time intervals, e.g., overall collision probability for a functional scenario or ODD.

7: Nanoscopic scenario quality: quality of scenario segment, e.g., time-to-collision (time step), headway (time step), or traffic density (time step or time interval),

8: Microscopic scenario quality: quality of single scenario, e.g., min(time-to-collision) or min(headway),

9: Macroscopic scenario quality: quality of set of scenarios, e.g., variable coverage within this set.

4.3.5 Interaction Between Different Types of Quality

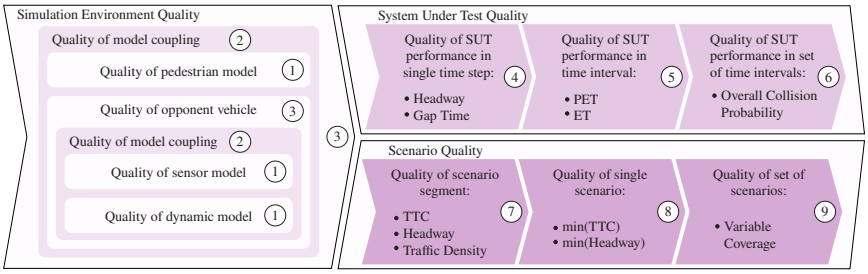


Figure 4.6: Basic dependencies of quality interaction described by Section 4.3.2 with examples. Abbreviated metrics are post-encroachment time (PET), encroachment time (ET), Time-To-Collision (TTC).

On a closer look, the different levels of resolution and the different domains of interest interact with each other or serve as a base for each other, e.g., simulation environment quality should be assured before SUT quality or scenario quality. The basic structure of quality interaction is illustrated in Figure 4.6. The simulation environment quality, which includes different parts and areas of the simulation environment, has to be ensured before the system under test quality and scenario quality since both need a working simulation environment to be measured. Several metrics can be used in more than one category, e.g., time-to-collision can measure a SUT's performance. Nevertheless, also to determine the criticality of a scenario regarding the chosen ego vehicle.

In addition, nanoscopic metrics can be evaluated using functions such as *min* or *max* at the microscopic level. Using these functions makes the microscopic level dependent on the nanoscopic metrics, unless metrics are computed on the microscopic scale, as is the case with post-encroachment time or encroachment time.

4.3.6 Scenario Quality in the Scope of This Work

This work assesses the quality of a scenario based on its criticality. There are two types of scenario quality: nanoscopic resolution, which involves calculating metrics such as distance or WTTC, and microscopic scenario quality, which includes criticality metrics such as PET or summarizing nanoscopic metrics to derive microscopic metrics like the minimum encountered distance in a specific scenario. These microscopic quality measures are then aggregated to demonstrate the macroscopic quality of logical scenarios, as illustrated in Figure 5.9 and Figure 5.8. This work assumes that all scenarios meet the previously defined criteria for structural and functional scenario quality unless explicitly stated otherwise.

4.4 Scenario Exploration

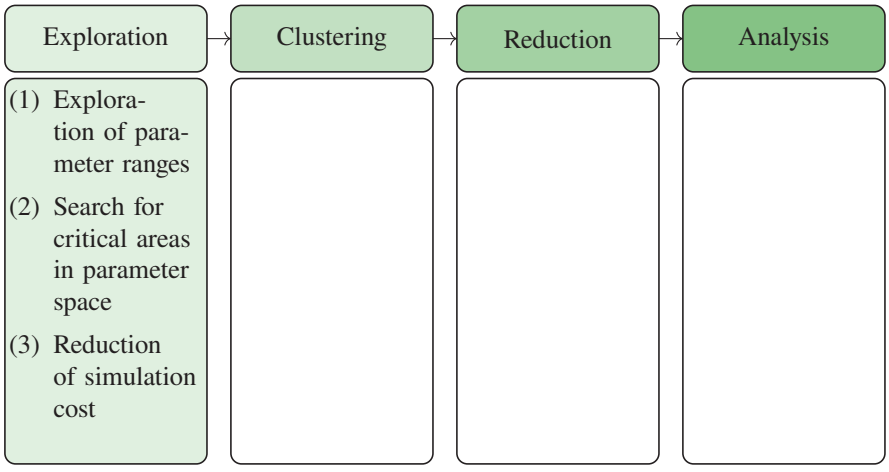


Figure 4.7: Scenario exploration step: search for critical parameters within parameter ranges and reduction of costs.

Scenario exploration involves searching for a set of concrete scenarios within a logical scenario by exploring a predefined range of parameters or distributions that define the scenario and actor

behavior within each executed scenario. The set of all possible parameter combinations within a logical scenario is referred to as the *scenario space*. Scenario exploration (Section 2.3.3) is a category of scenario acquisition (see Figure 2.7 and Figure 2.8). Figure 2.8 shows the transition from logical scenario to concrete scenario, where one logical scenario leads to n concrete scenarios found during the exploration. The following concept aims to include a result evaluation that has not been previously considered in Section 2.3. The key steps involved in scenario exploration, as illustrated in Figure 4.7, are as follows:

(1) Exploration of all parameter ranges and distributions within the scenario space to identify relevant and distinct areas.

(2) Identification of critical areas within the parameter space of the logical scenario to ensure that no critical scenarios are overlooked.

(3) Reduction of simulation cost by optimizing the set of scenarios that need to be simulated with the help of optimization algorithms.

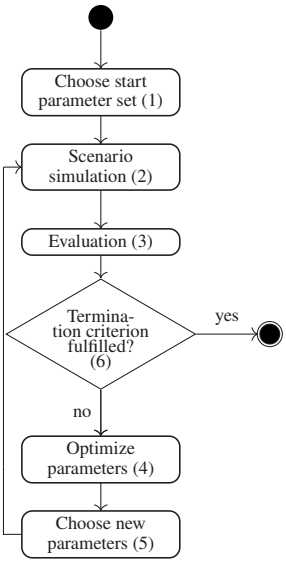


Figure 4.8: Activity diagram of the scenario exploration workflow.

The optimization process is an iterative technique, as shown in the activity diagram in Figure 4.8, which corresponds to item (1) of the list of characteristics provided. Initially, a starting parameter set is chosen (1), and a concrete scenario is simulated (2). The results are then evaluated using criticality metrics (3), and an optimization algorithm (4) selects a new parameter set (5) to be simulated again (2). This procedure is repeated until a termination criterion is met (6). Steps

3-6 are performed by the optimization algorithm. Other possible approaches are evolutionary algorithms as described by Schütt et al. [11].

Item two in the list above presents a goal for scenario exploration: to search for critical areas in the parameter space. It is necessary to find both critical and non-critical scenarios to represent the logical scenario from the perspective of the ego vehicle. Criticality is always measured for the ego vehicle and can be distributed differently for other driving functions (see Section 5.1.1) or revisions. Item three is closely related to item two. Some exploration methods require fewer scenario simulations than others, such as grid search versus optimization, which will be explained later in this section. Therefore, the objective is to minimize the exploration cost. Throughout this work, Bayesian optimization is used since it also maintains a surrogate model for further investigations. However, similar experiments have also been made with EAs as part of a supervised Master's thesis [166] and a resulting conference paper [11].

4.5 Concrete Scenario Clustering

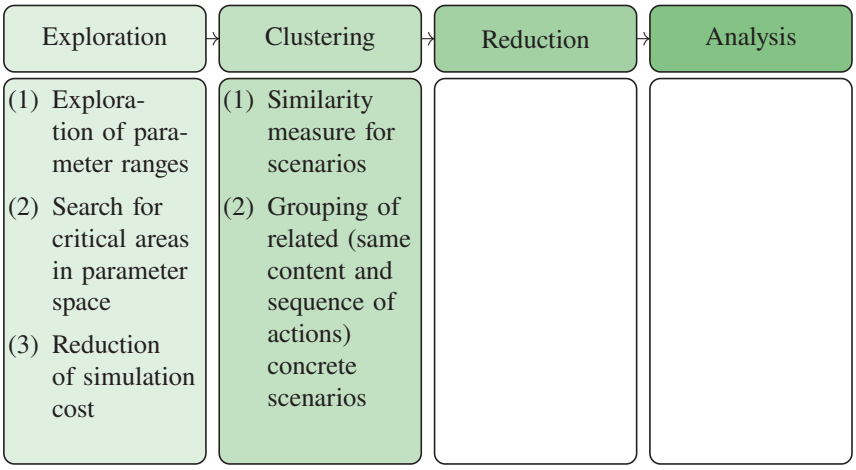


Figure 4.9: Scenario clustering step: measure scenario similarity and find clustering.

Scenario clustering is a technique that identifies and categorizes concrete scenarios within a logical scenario based on content. It summarizes similar sequences of scenes, trajectories, or concrete scenarios into one scenario description that represents a range of parameters [108]. The process of clustering involves two main steps, as shown in Figure 4.9:

(1) Measuring the similarity between scenarios, either behavior-based (see Section 4.5.1.2) or criticality-based (see Section 4.5.1.1).

(2) Grouping concrete scenarios based on their content and sequence of actions.

The primary objective of scenario clustering is to organize a given collection of scenarios based on their content and the specific objectives guiding the clustering process. Notably, the outcomes of simulated scenarios, including criticality and trajectories, depend on the tested ego vehicle or SUT, ADS, or ADAS. According to Greenhill et al. [68] and Nidhra and Dondeti [112], the role of the ego vehicle in the optimization process can be viewed as a black box. This is because the optimization process only considers the output, which is the vehicle's performance in a specific scenario, without taking into account any internal mechanisms.

Similar to other optimization algorithms, BO must find a balance between exploration and exploitation when selecting parameters [45]. As it can be seen in the experiments (Chapter 5 and Figure 3.8), BO adopts a narrow sampling approach by focusing on concrete scenarios in proximity to critical areas or areas characterized by high levels of parameter uncertainty. Consequently, these algorithms often find parameter sets (concrete scenarios) that result in minor variations in scenario outcomes. Therefore, clustering based on the distance between concrete scenarios in the parameter space is not recommended.

4.5.1 Computation of Scenario Distance

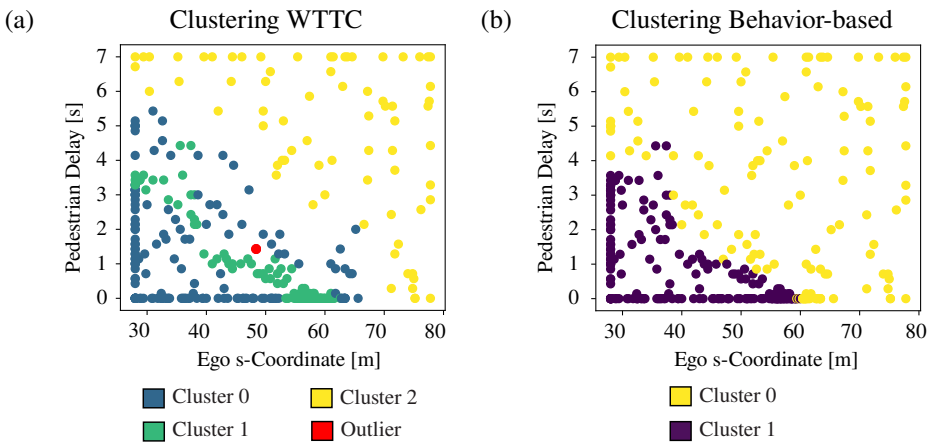


Figure 4.10: (a) Example for criticality-based clustering; (b) Example for behavior-based clustering.

Scenario clustering is based on comparing concrete scenarios with each other. However, the concrete scenarios chosen by the optimization algorithm within the parameter space of a logical scenario do not follow a defined distribution. Additionally, as discussed in the previous paragraph, they concentrate on critical areas. Clustering can be performed based on various features, e.g., sequences of scenes, trajectories, or the course of scenario criticality throughout the simulation. It has to be established which method is the best to compare concrete scenarios from one logical scenario. Additionally, there are multiple methods available for comparing scenarios and assessing their similarity [3]. In the scope of this work, DTW is used to compare different time series (see Section 3.2.4). The measured scenario distances are then used to create a distance matrix (see Section 4.5.1.3).

Figure 4.10 illustrates the comparison of two clustering approaches: criticality-based (a) and behavior-based (b). It is the same parameter space as already shown in Figure 3.9, but colored regarding the cluster assignments. Figure 4.10 (a) shows a cluster comprising exclusively critical scenarios enclosed by a larger cluster encompassing less critical scenarios. Additionally, the yellow cluster contains non-critical scenarios. The blue and green dots in Figure 4.10 (b) are the color assignment of two scenario clusters, with their boundaries converging at the region of highest criticality. This convergence is due to a discernible shift in ego behavior, such as the ego vehicle passing an intersection before or after another actor.

4.5.1.1 Criticality-based Similarity

The initial approach is criticality-based clustering. This involves clustering scenarios based on their level of criticality, which allows for the grouping of high-risk and low-risk scenarios. DTW is used to compare different time series of criticality curves (see Section 3.2.4) and finally to cluster similar scenarios into one group. Most criticality metrics can be computed for each simulation step (nanoscopic), e.g., distance or WTTC, resulting in a distinct criticality curve or trajectory. Additionally, these stepwise calculated metrics can be summed up to microscopic criticality values, e.g., $min(distance)$ or $min(WTTC)$. These microscopic criticality values are shown in Figure 3.9 for an exemplary 2-dimensional parameter space. It can be observed that some metrics have a higher gradient (e.g., WTTC) when approaching critical areas than others (distance) (further explained in Section 5.3.1). However, in most cases, they all identify critical regions in the same areas of the scenario space.

For instance, Figure 4.10 (a) displays three clusters of scenarios: critical scenarios within less critical scenarios that surround this cluster. Additionally, the yellow cluster contains non-critical scenarios. In this example, the clustering was based on the criticality metric WTTC. This example is discussed further in the experiments in Section 5.3.2.

4.5.1.2 Behavior-based Similarity

A method for determining scenario similarity is to analyze vehicle trajectories extracted from various types of data, e.g., recorded naturalistic driving data sets [91, 3]. This trajectory-based approach also is also applicable to recorded simulation data after the exploration phase. The underlying concept of this approach is that significant differences in concrete scenarios are unlikely to exist as long as the trajectory of the ego vehicle remains constant. For instance, if the ego vehicle does not need to apply brakes in response to another car, the exact distance of that car (e.g., 40 or 50 meters away) becomes irrelevant. Similarly, if the ego vehicle has to brake for an actor, it does not matter if it is always the same actor as long as the braking occurs at the same place in time and space during the concrete scenario. Figure 4.10 (b) depicts a two-dimensional scenario parameter space featuring two distinct clusters. The borders of the clusters converge in an area linked to critical scenario outcomes. In the given example, the ego vehicle and another actor cross the same intersection. Notably, the purple cluster comprises scenarios where the other actor passes the intersection before the ego vehicle, while the yellow cluster comprises scenarios where the ego vehicle passes first. The scenarios bordering each cluster are highly critical.

4.5.1.3 Distance Matrix

Exploration algorithms must determine the distance between concrete scenarios. However, a challenge faced by optimization algorithms when exploring logical scenarios is the spatial distribution of these scenarios in the parameter space. It does not provide any meaningful insights into the scenario outcome since the optimization algorithm determines the selection of parameter combinations, which draws from parameter ranges regarding its exploration and exploitation model. Therefore, it is necessary to establish scenario similarity using measures other than the Euclidean distance in the parameter space. Braun et al. [3] describe various methods for computing scenario similarity. One of these approaches, DTW [28], was utilized in the following step for computing a distance matrix D since the simulated scenarios in this work provide recorded data (e.g., position or criticality) over a temporal course (cf. Section 3.2.4). DTW can handle temporal distortions, variation in the duration, and the speed of trajectories. The trajectory of the ego vehicle adjusts at different times during the simulation due to varying adversary actor behavior. These reactions result in different outcomes and, therefore, less similar scenarios. Other possible distance matrices can be computed using microscopic criticality metrics, such as PET, WTTC, Traffic Quality, or the distance between actors (see Section 5.1.2).

A distance matrix contains the pairwise distances between all simulated scenarios (see Section 3.2.4). Pairwise distance, in this case, refers to the DTW between two ego trajectories or the course of criticality of two concrete scenarios. The matrix in Figure 4.11 shows the distance

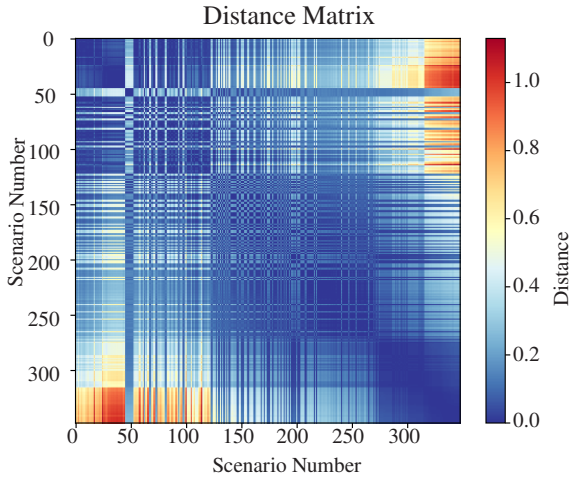


Figure 4.11: Distance matrix colored regarding the concrete scenarios' distances. The diagonal compares each scenario with itself, resulting in a distance of 0.0. The unit of the distance depends on what was compared, e.g., meters or seconds.

between two scenarios, with dark blue indicating similar scenarios and red indicating different scenarios. This matrix was taken from the same experiments (see Section 5.3.2) as the clustering depicted in Figure 4.10 (b), which is based on the behavior-based distance matrix in Figure 4.11. The axes display scenario identifiers, with each number representing a parameter combination. The diagonal displays the smallest distances, as it compares each scenario to itself. The distance between two trajectories is measured using DTW for behavior-based comparisons or the course of criticality in criticality-based approaches.

4.5.2 Dimensionality Reduction with Kernel PCA

The distance matrix is used for clustering the scenarios. Between the computation of the matrix and the clustering, there is one step in between before the clustering can take place: dimensionality reduction (also called projection into a low-dimensional space) by the utilizing a kernel PCA with a replaced distance matrix in the RBF kernel (see Section 3.2.3). In order to do so, the distance matrix is used to compute a Gaussian RBF kernel for a kernelPCA. Nonetheless, in the case of concrete scenarios within a logical scenario, the dimensions of the original feature space are unknown. However, the distance matrix discussed in Section 4.5.1.3 provides sufficient information to compute a kernel PCA.

4.5.3 Clustering

The next step is to cluster the specific scenarios in order to group them based on their content. Notably, the projection of scenarios into the kernel space, as demonstrated in Figure 3.10 (c) in Section 3.2.3, can enhance the distinguishability of clusters. Utilizing kernel-based techniques can help make clusters more distinguishable from each other. This allows for clearer differentiation between scenarios. The clustering process reveals the inherent behavioral and criticality structure and patterns within the scenario data, which helps to uncover the number of distinct clusters that depict the underlying scenario variations of a single logical scenario.

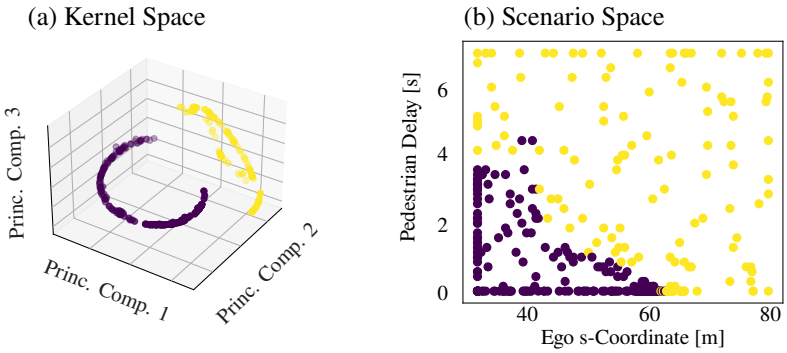


Figure 4.12: (a) Example for behavior-based clustering with DBSCAN in the kernel space. The axes are the three most influential principal components; (b) Cluster assignments from the kernel space taken back to the original scenario space.

The proposed approaches for projection and clustering can be combined and used for concrete scenarios. Figure 4.12 (a) shows an example of a behavior-based kernel space clustered into two distinct clusters with DBSCAN (see Section 3.2.5) In this example, both clusters have a C-shaped pattern in the kernel space and are orthogonal to one another. DBSCAN was used since the number of clusters is unknown. The axes depict the three most influential principal components after the projection into the kernel space. This cluster assignment can be transferred to the original scenario parameter space (Figure 4.12 (b)).

In Figure 4.12 (b), the highest criticality is found in the area where the two clusters intersect, as shown in Figure 4.10. The scenario itself and the resulting outcomes are further explained in Section 5.2.1. The purple cluster encompasses situations in which a pedestrian crosses an intersection before the ego vehicle. The yellow cluster, on the other hand, contains scenarios where the ego vehicle crosses first. Additionally, the simulation results can depend on interactions between different parameters. For instance, the starting position of the ego vehicle can range from close to the pedestrian to further away, and the pedestrian may have different waiting times before

moving. In this case, achieving the same scenarios is possible with different parameter sets. For example, a close starting point with no waiting time or a far starting point with a long waiting time.

4.6 Reduction by Archetypal Analysis

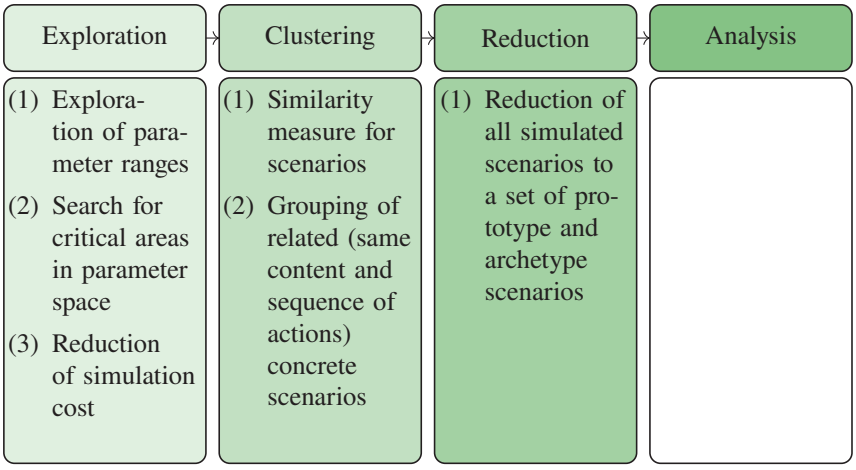


Figure 4.13: Scenario exploration step: reduction of scenario set.

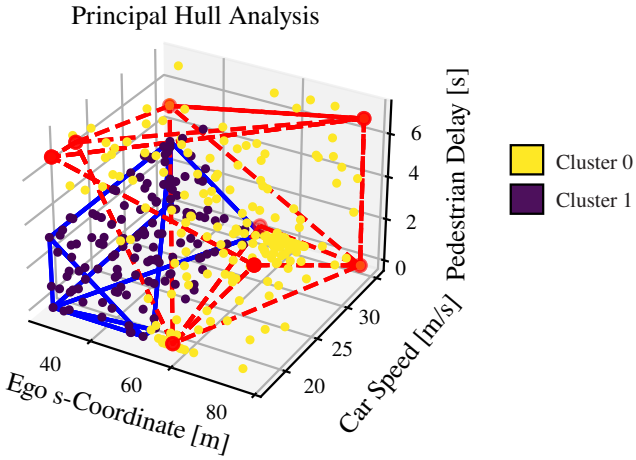


Figure 4.14: Principal convex hull for both clusters.

The third step involves scenario set reduction, as listed in Figure 4.13:

- The reduction of all simulated concrete scenarios from one logical scenario to a reduced set of arche- and prototypes representing the logical scenario.

The preceding exploration and clustering steps result in different scenario sets, some of which may contain redundancies. The goal of this step is the reduction of such redundancies and repeating concrete scenarios. Prototypes and archetypes can be extracted from each cluster and used as a representative set for the logical scenario to further reduce the dataset.

In this context, archetypes are extreme examples or corner cases within a cluster representing distinct patterns or characteristics. The similarity to an archetype is determined by measuring the distance between a data point and the archetype itself. When a data point shows similar distances to more than one archetype, it indicates a mixture. The data point's characteristics change towards an archetype the closer it gets. To identify these archetypes in convex clusters, a principal hull analysis can be used (see Section 3.2.6). This analysis represents the archetypal corner cases of the cluster, and the archetypes can be found as the vertices of this hull. Figure 4.14 displays the principal convex hull of each cluster's eight data points. In addition, a prototype can be selected. Prototypes are the most average concrete scenarios within a cluster. They can be defined by mean or median parameter sets or by existing scenarios that are closest to them.

It is important to acknowledge that the proposed exploration and clustering approach produces different outcomes for behavior-based and criticality-based clustering. The selected distance type plays a crucial role in the clustering results, and as a result, scenario sets are reduced. Behavior-based clustering identifies cluster borders and corresponding archetypes in regions of the scenario space where criticality is high in case cluster borders meet. Conversely, criticality-based clustering identifies borders around critical or non-critical clusters. Furthermore, Figure 4.10 demonstrates that clustering may lead to clusters nested within other clusters depending on the dimensionality and specified parameters. If a cluster does not occupy a convex space in the scenario parameter space, the proposed algorithms cannot analyze or reduce it.

4.7 Probability Analysis of the Results

The final phase involves an analysis of the surrogate model resulting from the exploration with Gaussian processes in step 1 (see Section 4.4). This phase encompasses two distinct steps:

(1) The surrogate model allows for predicting criticality across the entire logical scenario space. By leveraging this model, predictions about criticality metrics can be generated for every point in space. Each prediction is subject to some noise depending on the certainty at that specific point.

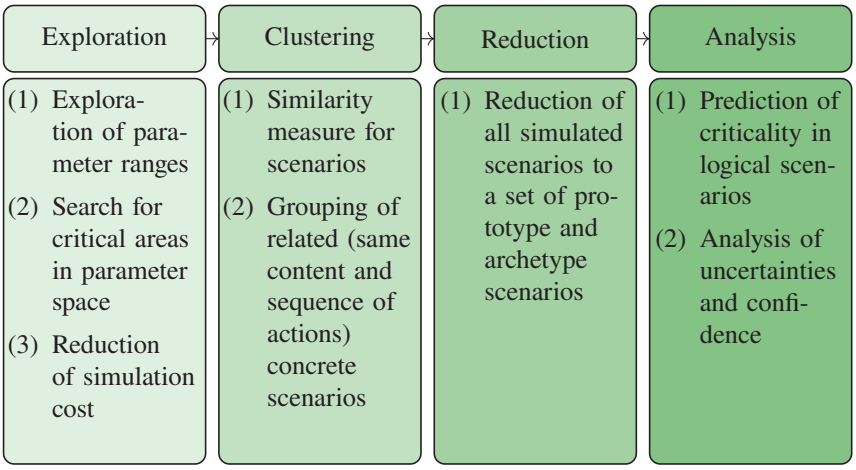


Figure 4.15: Analysis step: prediction of criticality and analysis of uncertainty

(2) The prediction enables the analysis of uncertainty and confidence within the scenario space by evaluating the variance among predictions made for each spatial point. The certainty of a model affects the noise in the predictions for each point in space. The variance between different predictions at the same point can be used to determine the confidence level of a model.

As outlined by Lim et al. [94], the Gaussian process regression used in Bayesian optimization serves as a surrogate model that represents the objective function through Bayesian inference. This allows for additional statements and insights into the underlying characteristics and validity of the objective function model. After the training phase, the Gaussian process can predict values for any point within the input space. The model can forecast the criticality metric it was trained on, such as traffic quality, for different parameter configurations within the logical scenario.

Due to the statistical nature of the anticipated criticality values, there may be variability for a given input set. By utilizing these predicted values, it is possible to compute both the mean and standard deviation, which facilitates the extraction of subsequent insights:

- Complete prediction over the input space.
- Quantification of uncertainty associated with each point in the parameter space.
- Determination of a confidence interval, as illustrated in Figure 3.8 (b) or Figure 4.16.

Figure 3.8(b) provides an example of a confidence interval within a one-dimensional space. However, in most logical scenarios where optimization is necessary, the parameter space has

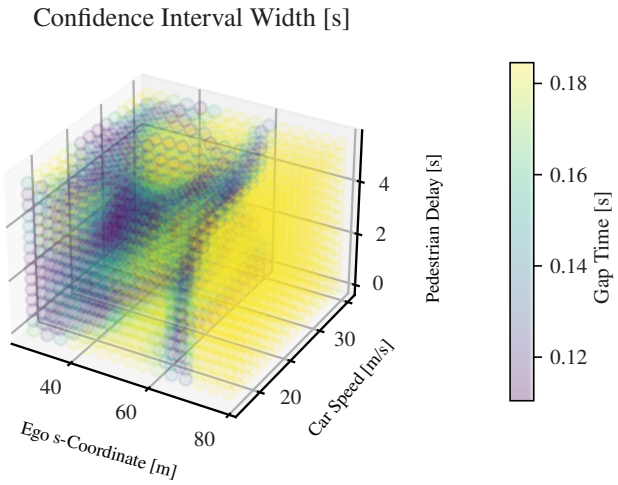


Figure 4.16: Example of a 3-dimensional plot that illustrates the confidence interval of the metric gap time at each point. The purple color corresponds to a narrow interval, indicating high confidence, while the yellow color represents a wider interval, indicating lower confidence.

more than one variable. For a single parameter, it is possible to simulate a grid of concrete scenarios instead of using an optimization approach. Therefore, when dealing with a 2- or 3-dimensional space, alternative techniques are required to visualize the confidence interval. An option is to employ color coding for points in space, as exemplified in Figure 4.16. The confidence interval of the gap time is presented, with the smallest width being around $0.11\,\mathrm{s}$ and the largest around $0.185\,\mathrm{s}$. Upon closer examination of the scenario criticality results, it is evident that the area with higher criticality and more samples during exploration corresponds to the smaller confidence interval (refer to Section 5.3.1 for the results presented in a 2-dimensional space). In this context, the color indicates the extent of the confidence interval. Shades of yellow represent lower confidence, while shades of purple represent higher confidence.

4.8 Traffic Density Potential

Most metrics are only suitable to measure the criticality between two traffic participants, e.g., TTC or PET (cf. section 2.2.5). However, in certain situations, such as automated scenario exploration or generation, it may be unclear which traffic participants these metrics should be applied to. One possible solution is to measure the chosen metric between the ego vehicle and every other traffic participant in a scene and to use the worst measured value by utilizing *min* or *max*. To overcome this problem, a new scene metric is proposed in this work that can evaluate

a complete traffic scene. This new metric is used throughout this work and compared to other metrics.

Hallerbach et al. [70] proposed a metric called traffic quality that is measured within a given scene, which addresses this problem for highway scenarios (Section 2.2.5). The proposed Traffic Density Potential (TDP), described in detail further down, is based on the traffic quality approach but incorporates some changes to make it suitable for urban scenarios. In contrast to [70], the final score does not need trained weights for each of the four sub-metrics. Furthermore, the final score increases as the traffic situation around the ego vehicle becomes denser. The names of the sub-metrics of the traffic quality as proposed by Hallerbach et al. [70] were partly changed for a more consistent naming scheme and to avoid confusion with the quality taxonomy from Section 2.2. Both the TDP and the taxonomy share the names of macro- and microscopic sub-metrics (TDP) and macro- and microscopic levels of resolution (taxonomy) but are not related to each other in terms of content. The TDP has four sub-metrics, while the taxonomy has three levels of resolution. In fact, in terms of the quality taxonomy, the TDP is a nanoscopic scenario quality metric. Additionally, TDP avoids information from previous time frames as much as possible to simplify computation. [2].

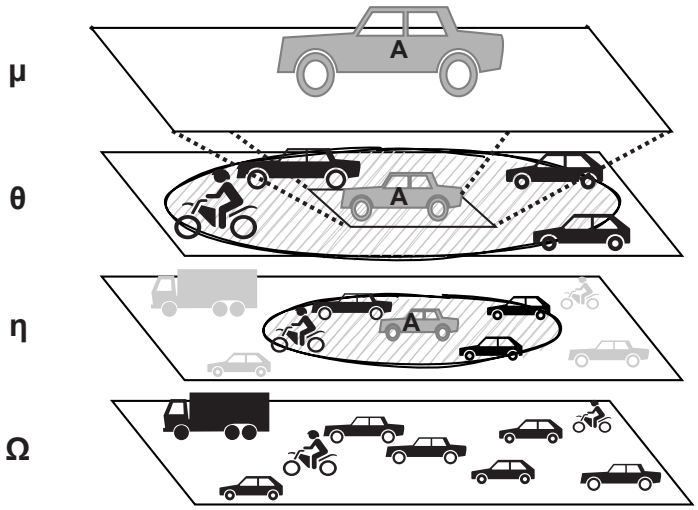


Figure 4.17: Schematic overview of the different areas of interest of the Traffic Density Potential (TDP) on an exemplary traffic scene, starting with macroscopic resolution at the bottom and metascopic at the top. The gray vehicle, denoted as A, is the ego vehicle. The black ellipse symbolizes the area relevant for the sub-metric. Black cars are considered for a sub-metric, whereas light gray cars are outside the area of consideration.

[2] Parts of this chapter have already been published in [8] and [12]

The TDP consists of four sub-metrics like the original traffic quality formula and is schematically depicted in Figure 4.17: macroscopic (Ω) (complete scene), metascopic (η), mesoscopic (Θ), and microscopic (μ) (only ego vehicle). All four parts evaluate different aspects within the given scene and are used to get a final score.

4.8.1 Macroscopic Density Potential

The macroscopic TDP sub-metric calculates the coefficient of variation of the velocities of all vehicles in a scene. A higher macroscopic value indicates more unsteady movement among all traffic participants. Uniform traffic is considered uncritical, regardless of the average velocity of all participants. The macroscopic TDP component for a complete scene is calculated as

$$\mathrm{TDP}_{\Omega} = \frac{\sigma_{sc}}{\bar{\nu}_{sc}} \tag{4.1}$$

where σ_{sc} describes the standard deviation of the velocities and $\bar{\nu}_{sc}$ the mean velocity of all vehicles V_t within one scene at time t. The macroscopic value is the same for all traffic participants within one scene. Congestion situations, such as merging lanes or traffic lights, where most cars are stationary and only a few are in motion, as well as roundabouts with vehicles waiting to enter and a high number of parked cars within a map, are typical scenarios with high macroscopic values. The macroscopic sub-metric does not take into account spatial traffic density. For example, a map with few vehicles traveling at different velocities may still have high macroscopic values.

4.8.2 Metascopic Density Potential

The metascopic TDP sub-metric is the ratio between all vehicles on the scene and the ones within the estimated braking distance of an ego vehicle A:

$$\mathrm{TDP}_{\eta}(A) = \frac{V_t^{br}(A)}{|V_t|} \tag{4.2}$$

where $|V_t|$ represents the set of all vehicles found on the scene at time t. $V_t^{br}(A) \subseteq V_t$ where $V^{br}(A)$ denotes all vehicles which are within the breaking distance of A at the time t. The metascopic value always ranges from 0 and 1 and increases with the velocity of the ego vehicle until the braking distance encompasses the entire scene. High metascopic values are typical in situations involving rapid movements across a map or when leaving a scene with high acceleration.

4.8.3 Mesoscopic Density Potential

The mesoscopic TDP sub-metric is computed in a similar way to the macroscopic TDP sub-metric. However, it only takes into account the coefficient of variation within the braking distance of an ego vehicle.

$$\mathrm{TDP}_{\theta}(A) = \frac{\sigma_{br}}{\bar{\nu}_{br}} \tag{4.3}$$

where σ_{br} describes the standard deviation of the velocities and $\bar{\nu}_{br}$ the mean velocity of all vehicles within the braking distance of the ego vehicle $V^{br}(A)$. Typical situations with high mesoscopic values occur when the ego vehicle passes or approaches a group of slower-moving vehicles.

4.8.4 Microscopic Density Potential

The microscopic TDP formula is the only sub-metric that uses data from previous time frames. It only considers the ego velocity and acceleration development over the last few seconds. It consists of two ratios combined:

$$\mathrm{TDP}_{\mu}(A) = \frac{\frac{\bar{a}_A}{a_{ref}} + \frac{\bar{\nu}_A}{\nu_{ref}}}{2} \tag{4.4}$$

where $\bar{a}_A$ describes the mean acceleration and $\bar{\nu}_A$ the mean velocity of the ego vehicle over a certain time in the past. Values a_{ref} and ν_{ref} are reference values, e.g., $\nu_{ref} = 50\frac{km}{h}$ and $a_{ref} = 1.5\frac{m}{s^2}$ for urban traffic. Both reference values can be adjusted based on the evaluated scenes. For instance, traffic-calmed zones may require different adjustments. Situations with high microscopic scores typically involve fast-moving vehicles or situations where the ego vehicle accelerates or brakes.

4.8.5 Final Score

The last step is calculating the final score from the four different sub-metrics. A spacial-related approach instead of finding weights for each term was used [70]. Therefore, the l_2-Norm of all four sub-metrics is used:

$$\mathrm{TDP}_{co} = \sqrt{\mathrm{TDP}_{\Omega}^2 + \mathrm{TDP}_{\eta}^2 + \mathrm{TDP}_{\theta}^2 + \mathrm{TDP}_{\mu}^2} \tag{4.5}$$

However, TDP_{co} is very sensitive and in several cases marks uncritical situations as critical. In order to avoid this problem, a penalty term ρ_x is introduced:

$$\mathrm{TDP}_{\rho_x} = \rho_x \mathrm{TDP}_{co} \tag{4.6}$$

where ρ_x can be replaced by a penalty or reward depending on the metric's goal. In this work, three possible penalty terms with different focus aspects are proposed. All three terms are based on the distance between the ego vehicle and the nearest vehicle within a scene and punish empty space around an ego vehicle:

$$\rho_1 = \frac{1.5}{d_{min}} \tag{4.7}$$

$$\rho_2 = e^{-\frac{d_{min}}{5.0}} \tag{4.8}$$

$$\rho_3 = e^{-\frac{d_{min}-1.0}{10.0}} \tag{4.9}$$

where d_{min} is the distance to the closest vehicle, respectively. The proposed criticality threshold for TDP_{co} is 1.5 and for three combinations with the penalty term 1.0, where critical scenes achieve a value above the threshold.

ρ_1 prioritizes distance, which means that a vehicle in close proximity to the ego has a greater impact on criticality, potentially neglecting the overall traffic quality of the scene. However, two vehicles that are close together or moving slowly may be considered less critical than when the ego vehicle is further away but moving quickly towards a potential collision. To address this, ρ_2 and ρ_3 are introduced, with ρ_3 being less sensitive to the ego passing by stationary vehicles.

A in-depth evaluation can be found in the Appendix B.2.

5 Scenario Exploration and Analysis Experiments

5.1 Scenario Exploration Setup

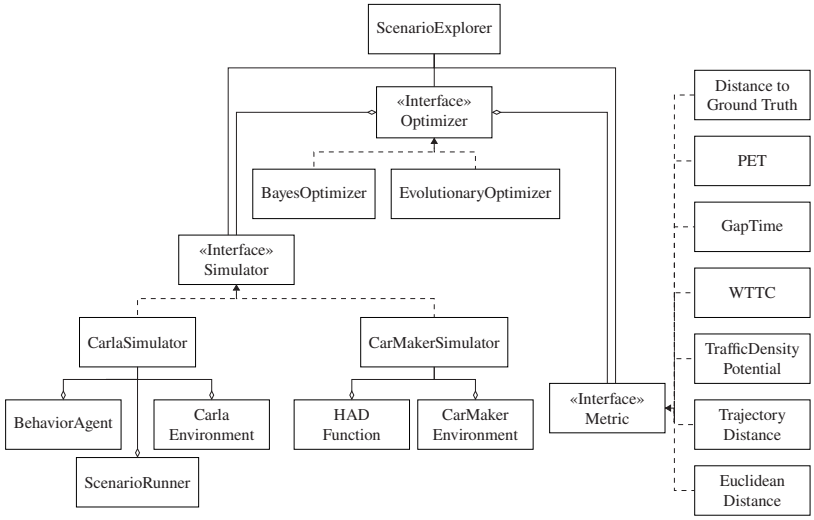


Figure 5.1: Class diagram of the simulation tool setup.

Figure 5.1 illustrates the class diagram of the simulation framework and portrays the underlying class relationships. The implementation of the complete tool setup is programmed in Python 3.9. It consists of three main components: the simulator (Section 5.1.1), the optimizer (Section 4.4), and the metrics module (Section 5.1.2). During startup, the **ScenarioExplorer** creates three objects: the simulator, metric, and optimizer objects. There is a corresponding interface for each object:

- **Simulator**: The simulator provides an interface for using various simulation tools for the exploration experiments. This includes different simulation tools such as Carla or IPG

CarMaker, as explained in Section 5.1.1. The driving functions used for the scenario exploration experiments are part of the simulation environment.

- **Optimizer**: The optimizer interface enables the use of various optimization algorithms to explore the logical scenario space. This work mainly uses Bayes Optimization with Gaussian Processes (see Section 3.2.2), but other methods are possible, e.g., evolutionary algorithms. Section 4.4 explains the optimization process in-depth in Figure 4.8.
- **Metric**: The metric interface allows the integration of different criticality metrics (see Section 5.1.2) that are used for the optimization. The goal is to find the most critical concrete scenarios within the logical scenario.

After initialization, the simulator and metric objects are passed to the optimizer. The boxes **CarlaEnvironment** and **CarMakerEnvironment** represent the entire Carla or CarMaker simulator software without providing further details, since both are treated as black boxes and the interface that is part of this software only communicates with them. The **BehaviorAgent** and **HADFunction** represent the driving functions of each tool (Section 5.1.1). Seven different metrics were used for all experiments (Section 5.1.2). Each component serves as an interface to replace parts without losing interaction possibilities. They interact with each other through the **ScenarioExplorer**. In general, the ScenarioExplorer starts a simulation, the optimizer selects the next best concrete scenario, and the metric determines the criticality or fitness of a concrete scenario.

The EvolutionaryOptimizer was implemented in a supervised Master's thesis [166] and a resulting conference paper Schütt et al. [11]. The thesis uses EA to derive new concrete scenarios from two logical scenarios. Unlike the experiments in the current thesis, the master thesis uses the Carla scenario description based on behavior trees. A parent generation of concrete scenarios is used to generate a child generation. Each individual in a generation is simulated and evaluated for its fitness. The fitness is defined by the criticality of a scenario, the higher the criticality, the fitter an individual is. The metrics used were WTTC, distance, and traffic density potential. In EAs, scenario variations are derived using two methods based on genetic evolution. The first method of scenario modification is mutation and the second is cross-over. In mutation, scenario parameters are changed, e.g. the speed of a vehicle. The second step is cross-over. Here, subtrees of behavior trees are swapped with each other, thus changing the behavior (sub)sequences of actors. The creation of a new child generation is repeated until termination criteria are met, e.g. all scenarios are below a defined criticality threshold. In the thesis, experiments with different generation sizes are conducted for both example scenarios and the results are evaluated in terms of the number of generations needed to complete the search.

5.1.1 Simulation Tool Setup

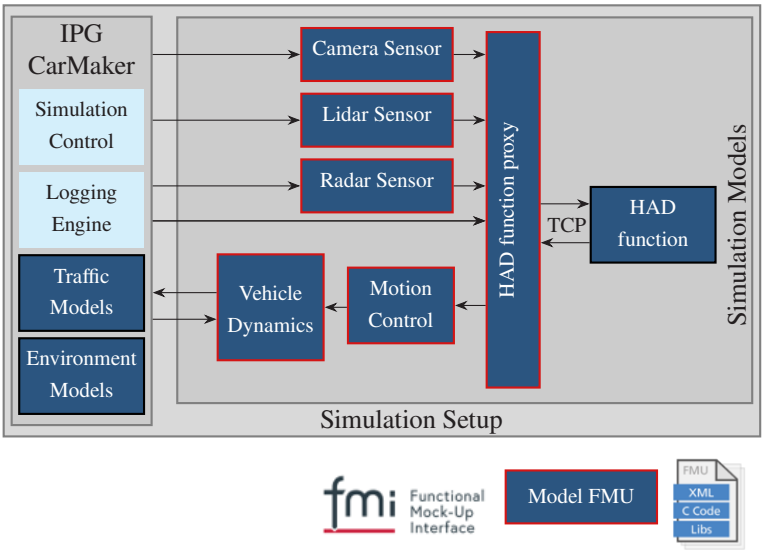


Figure 5.2: Modular structure of the IPG CarMaker simulation environment.

The simulation tool is included in the experiments via the **Simulator** interface shown in Figure 5.1. The first simulation setup, IPG CarMaker 8.0.2 [78] with the extension from the SET Level project[1], served as a foundation for the exploration and is shown in Figure 5.2. CarMaker operates as an open integration and testing platform. It orchestrates a centralized control unit for running a closed-loop simulation. The system encompassing system includes both proprietary and external models tailored to specific scenarios and constraints. The proprietary models constitute the comprehensive simulation environment, encompassing road infrastructure, environmental factors, and traffic models. The road, environment, and traffic simulation utilized proprietary models. In contrast, six external models were integrated as Functional Mock-up Units (FMUs) via an extended version of the Open Simulation Interface (OSI) [22]. These external models encompassed three sensor models (camera, lidar [95], and radar), an autonomous driving function, a motion control model, and a vehicle dynamics model, all of which were developed in the SET Level project. As previously mentioned, the Highly Automated Driving (HAD) functions are included into the simulation environment and further explained later in this section.

[1] `https://setlevel.de/`

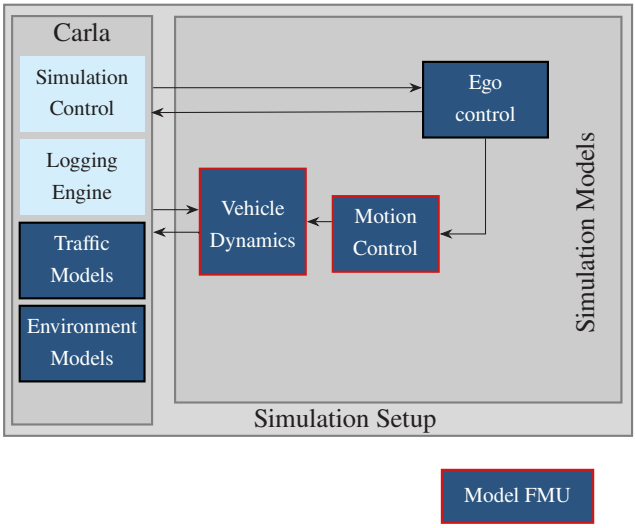


Figure 5.3: Modular structure of the Carla simulation environment.

The CarMaker setup described earlier was customized for the SET Level project's specific scenario and cannot be used for other scenarios without modifications to the proprietary tool chain. Therefore, to conduct additional experiments, another simulation tool was required. The second set of experiments used the Carla simulator [49] version 0.9.13 with the scenario runner. The scenario runner was modified to allow the ego vehicle to start simultaneously with a specific scenario. Only Carla's internal vehicle dynamics and motion control models and communication were used for the experiments, and instead of sensor models, object lists were utilized which makes the simulation environment setup simpler than the CarMaker setup and is depicted in Figure 5.3.

Another notable difference pertains to the physics simulation.

CarMaker does not use a physics engine, which means that objects can pass through each other without collision simulation. On the other hand, Carla is implemented in the game engine Unreal 4 and uses its physics engine to simulate crashes. As a result, in scenarios simulated in Carla, all simulations stop after a crash is recorded. In contrast, in CarMaker, actors successfully reach their designated destination points after each simulation run, and collisions are identified solely by criticality metrics.

All scenarios used in the experiments were detailed either in CarMaker's format or, in the case of Carla, in the OpenSCENARIO format [21]. A new scenario file is generated for each simulation

loop in both cases before the simulator executes this concrete scenario. Section 5.1.1 shows a comparison of both used tools with and ideal (non existing) environment.

Table 5.1: Comparison of both environments with an ideal environment in this section mentioned categories. (y)es for environment provides given functionality or (n)o if not.

	CarMaker Setup	Carla Setup	Ideal Simulation
OSI interface for sensor models	y	n	y
OSI interface for driving function	y	n	y
OSI interface for vehicle dynamics	y	n	y
OSI interface for motion control	y	n	y
Behavior simulation	n	y	y
Physics Engine	n	y	y
OpenSCENARIO	n	y	y
OpenDRIVE	n	y	y

Throughout the work, Bayesian optimization and Gaussian processes were used. It is acknowledged that other optimization algorithms may also be suitable. However, unlike other optimization algorithms, Bayesian optimization with Gaussian processes uses models that can provide data for the probability analysis step described in Section 4.7. It is important to note that the primary focus of this work was not solely on the optimization algorithm and process itself. The optimization experiments were performed using the open-source project Common Bayesian Optimization Library (COMBO) [145]. COMBO utilizes Bayesian optimization with automatic hyperparameter tuning, Thompson sampling to select the next best candidate, and random feature maps to enhance performance. The decision to use COMBO was mainly influenced by its automatic hyperparameter tuning capabilities.

Driving Functions The simulator setup includes the driving functions since the architecture for each simulation environment may vary, as illustrated Section 5.1.1.

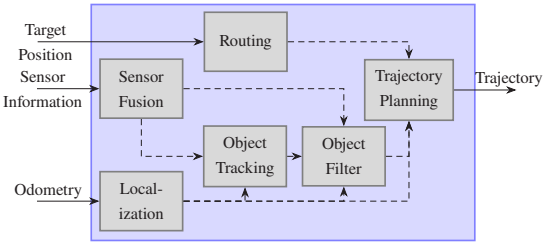


Figure 5.4: Modular structure of the driving function used with CarMaker [7].

The computation of the ego vehicle's trajectory in CarMaker experiments is done by a lightweight and highly automated driving function, as shown in Figure 5.4. It was adjusted and integrated via OSI [22] during the SET Level project. This function revolves around a modified, curvature-aware version of the Intelligent Driver Model (IDM), which was previously employed in [162] and initially introduced in [143]. This function is implemented using the Robot Operating System (ROS) framework [121] to foster a modular and flexible system architecture.

In experiments run with Carla, the Carla behavior agent[2] was used. The agent available in three modes: cautious, normal, and aggressive. It can navigate to a destination point by following waypoints defined by the localization class within the abstract agent class. The agents' different behaviors result (among others) from reaction time for braking or maximum speed.

It has to be noticed that the driving function connected via OSI to CarMaker is not deterministic. Therefore, simulations conducted with identical parameter sets may result in slight discrepancies, such as $0.01\,\mathrm{m}$, in the resulting criticality metric (cf. Appendix B.1).

5.1.2 Metrics

All the following metrics were implemented by using the metric interface mentioned in Figure 5.1 and used for the experiments. Criticality is measured using specific metrics to assess the potential for critical situations in a given scenario. The objective of the scenario exploration is to identify critical scenarios involving the ego vehicle and other traffic participants. Therefore, the optimization focus is on the criticality between the ego vehicle and other traffic participants. Most metrics can only evaluate the criticality between two traffic participants, namely the ego and a second

[2] `https://carla.readthedocs.io/en/0.9.13/adv_agents/`

actor within a scenario. Several criticality metrics[3] were utilized as the objective function to be optimized by the Bayesian optimization:

- **Distance**: Euclidean distance between the center of mass of two vehicles or, alternatively, the distance between the two closest points of the bounding boxes of both vehicles.
- **Trajectory Distance**: Distance between two traffic participants along their trajectories and road network.
- **Worst-time-to-collision** (WTTC): Metric based on time-to-collision (TTC) [73], but without the TTC's limitation to car-following scenarios [149]. WTTC tends to classify less critical situations as critical in order not to overlook true critical situations.
- **Gap time** (GT): The predicted distance in time between the two traffic participants crossing an intersection point [16].
- **Post-encroachment time** (PET): The actual distance in time between the two traffic participants crossing an intersection point [16]. Unlike GT, PET can only be calculated after both vehicles crossed the intersection.
- **Traffic Density Potential (TDP)**: The overall assessment of the traffic within the scene where the ego vehicle is situated. This metric is further explained in Section 4.8.
- **Distance of ground truth and sensor data**: The measured distance between adversary actor positions and their ground truth position recorded by the simulation environment, calculated with DTW [28].

All metrics listed above, except for TDP and the distance between ground truth and sensor data, indicate criticality by finding the smallest value optimization within logical scenarios. In contrast, maximizing TDP is necessary to achieve the same goal. It is important to interpret each metric in its individual result space, and single values should not be compared with values from other metrics.

5.2 Experimental Scenario Setups

This section introduces two intersection scenarios used in the experiments conducted in this work. The first scenario, 1A, was derived from the SET Level project and involves four traffic participants.

[3] Parts of this chapter are published in [8] and the implementation of all used criticality metrics can be found here: `https://github.com/fzi-forschungszentrum-informatik/scene-fingerprint`

Scenario 1B replicates 1A but with different starting and ending positions, and an ego vehicle turning left instead of right. Scenario 2 was specifically created to explore alternative methods of parameterizing logical scenarios, with a focus on the parameters that define synchronization points. Scenarios 1A and 1B are designed for CarMaker, while scenario 2 is tailored for Carla.

The scenario setups have distinct characteristics, particularly in their parameters and physics simulation. Scenarios 1A and 1B demonstrate a transparent and human-understandable relationship between the parameters and the behaviors and actions of the actors involved. This transparency is due to the incorporation of attributes such as speed and position, which are comprehensible by humans. It is important to note that in the CarMaker experiments, all actors, with the exception of the ego vehicle, follow predetermined trajectories and do not react to the presence of other actors, nor do they support collision detection. As a result, no physics simulation is done.

In contrast, the second scenario used in Carla experiments introduces the possibility of hidden parameters that are affected by the ego vehicle's behavior and interactions among actors. Complexity arises due to synchronization points and diverse actor behaviors. Unlike scenarios 1A and 1B, in these experiments, each actor can respond to the behavior of others, potentially triggering unforeseen chains of reactions among actors.

5.2.1 Scenario 1A and 1B (CarMaker)

The intersection scenario for CarMaker comes in two different versions: Version 1A and 1B. Scenario 1A features a right-turning ego vehicle, while Version 1B features a left-turning ego vehicle. Both scenarios have the same actors. In scenario 1A, the ego vehicle makes a right turn, crossing the trajectories of the pedestrian and the truck, but not crossing the trajectory of the other car. However, in both scenarios, all actors have different start and target positions. These two scenarios prompt different behaviors from the ego vehicle, as it responds dynamically to the presence of other actors obstructing its intended route. Unlike the other actors, who strictly follow predefined trajectories, the ego vehicle is the only participant in both scenarios with autonomous decision-making capabilities guided by its behavioral model. Figure 5.5 shows a traffic sequence chart of scenario 1A with all four alternative orders where participants may pass the intersection.

- **First Alternative**: The simulations reveal a first type of collision, which may be unrealistic in some cases. This occurs when the pedestrian model is unaware of the ego vehicle and only follows the trajectory of the ego vehicle. In this scenario, even though the ego vehicle is waiting for the pedestrian, the pedestrian either crashes into the ego vehicle (unrealistic) or passes very close in front of it (realistic). Most criticality metrics identify this situation as highly critical.

- **Second Alternative**: The second collision involves the ego vehicle colliding with the pedestrian as the pedestrian crosses the street. The pedestrian begins to walk while the object detection system of the ego vehicle fails to identify the pedestrian as a potential hazard. By the time the system recognizes the danger, it is too late and the ego vehicle is unable to brake in time.
- **Third Alternative**: The ego passes the intersection before the pedestrian.
- **Fourth Alternative**: The ego passes the intersection after the pedestrian.

Scenario 1B was designed with changes in the actors' start and end positions and maneuvers, as is shown in Figure 5.6. The ego vehicle executes a left turn, resulting in an intersection between the ego trajectory and the trajectories of all three opposing traffic participants. This scenario does not involve an imminent collision, but rather presents near-collision situations.

During simulation, three potential outcomes emerge:

- **First Alternative**: The intersection ahead of the ego vehicle was obstructed by the other car C, which prevented the ego driver from seeing the pedestrian. This created a highly critical situation that required abrupt braking to avoid a collision.
- **Second Alternative**: The ego vehicle passes before the other car. However, the pedestrian is faster in crossing the street, and the ego can brake without problems since no other actor interrupts the view.
- **Third Alternative**: The ego vehicle passes before the pedestrian begins crossing the street.

In all scenarios, the ego vehicle may encounter situations where it must wait for the truck to pass through the intersection, which also hides the pedestrian from the view of the ego vehicle. However, the obstruction is not significant enough to result in a collision.

The following ranges were chosen as parameter ranges for scenarios 1A and 1B, for which the optimal critical parameter sets have to be found during the scenario exploration:

- **Pedestrian delay**: The pedestrian waits for a given time $t_{P_{delay}}$ in s before crossing the road, where $t_{P_{delay}} \in \{0.0, ..., 7.0\}$.
- **Ego position**: The ego vehicle starts at a given s-coordinate $s_{E_{start}}$ in m along the road, where $s_{E_{start}} \in \{27.99, ...77.99\}$.
- **Car speed**: The maximum speed $v_{C_{max}}$ that the other car is allowed to achieve in $\mathrm{m/s}$, where $v_{C_{max}} \in \{12.5, ..., 30.0\}$.

No parameters varying the truck's behavior or trajectory were used. In a grid for this scenario parameter space with 50 samples for the pedestrian delay, 250 samples for the ego position, and 50 samples for car speed, the grid holds 625.000 scenarios. If each scenario takes approximately $30\,\mathrm{s}$ for simulation execution in CarMaker and calculation of criticality metrics, the execution of all 625.000 scenarios takes more than 217 days of non-stop simulation on a single machine (Intel Core i5 8350U with 1.7GHz, 8GB DDR4). Furthermore, it is important to note that simulation time increases exponentially as scenarios are given additional parameters and parameter ranges.

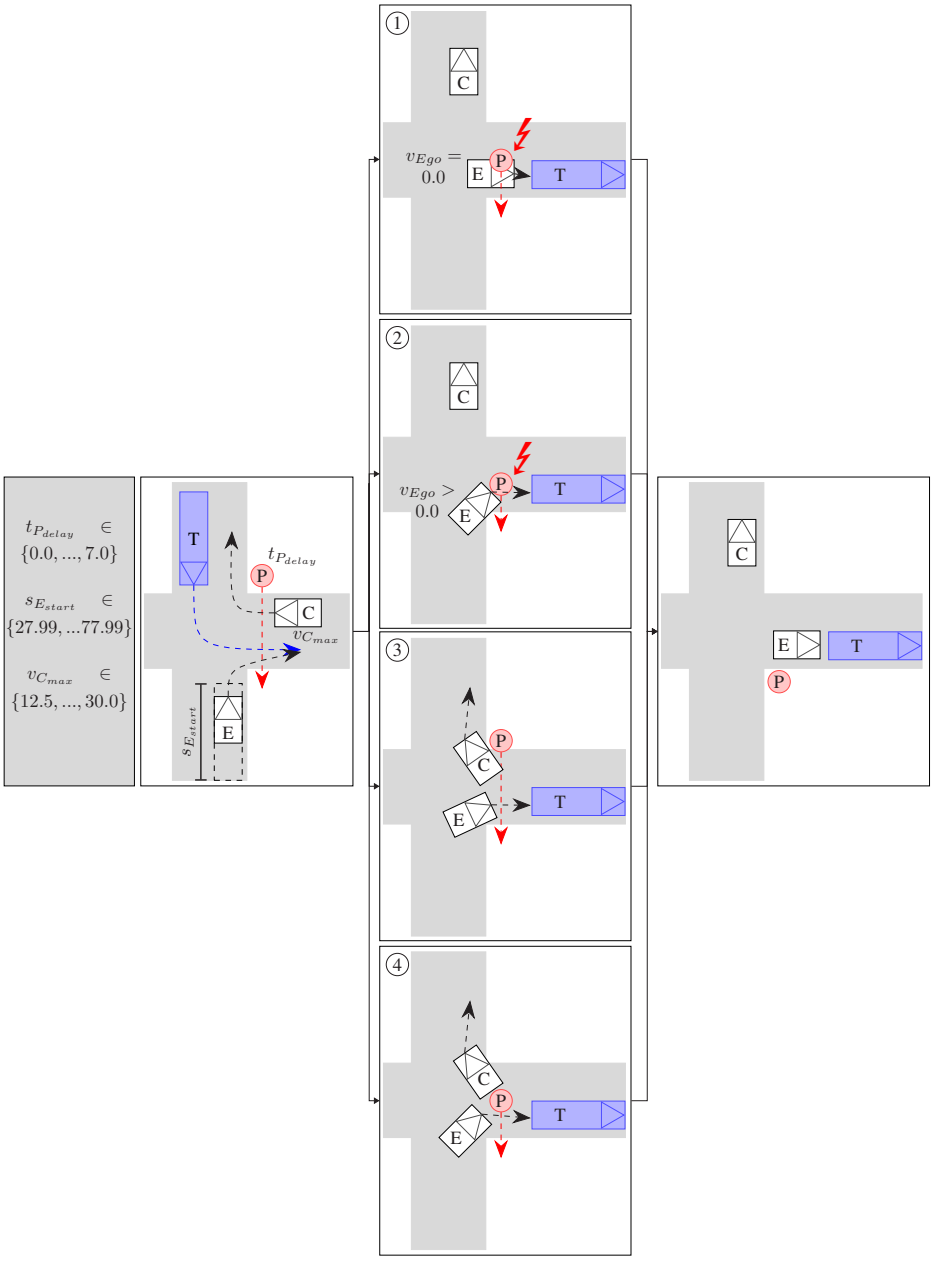


Figure 5.5: Scenario 1 A: Traffic Sequence Charts with all four alternative outcomes: (1) pedestrian runs into ego vehicle, (2) ego vehicle drives into pedestrian, (3) pedestrian passes after all ego vehicle (and all others) crossed the intersection, (4) pedestrian crosses the intersection before ego vehicle. C: car, E: ego vehicle, P: pedestrian, T: truck, $v_{Ego} = 0.0$: ego velocity is zero.

Figure 5.6: Scenario 1 B: Traffic Sequence Charts with all three alternative outcomes: (1) pedestrian and car C cross the intersection before ego vehicle, (2) pedestrian crosses the intersection before ego vehicle and C is behind ego, (3) ego passes before pedestrian and C. C: car, E: ego vehicle, P: pedestrian, T: truck, $v_{Ego} = 0.0$: ego velocity is zero.

5.2.2 Scenario 2 (Carla)

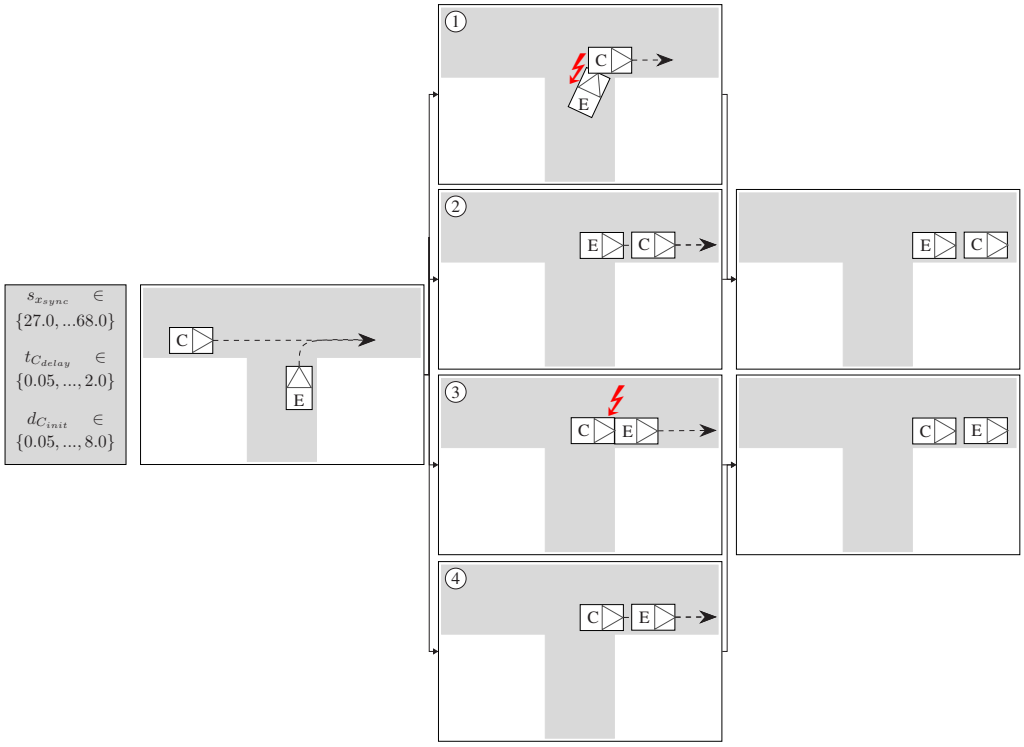


Figure 5.7: Second experimental scenario: ego turns right with sync points with four different outcomes: (1) ego crashes into car, (2) ego follows behind car, (3) car crashes into ego, (4) car follows behind ego. C: car, E: ego vehicle.

The second scenario involves a T-intersection where the ego vehicle (E) turns right, while a second car (C) proceeds straight across the intersection. Both vehicles have synchronization parameters, and the second car aims to reach its synchronization point at the same time as the ego vehicle.

The outcomes of scenario 2 are depicted in the traffic sequence chart shown in Figure 5.7. It can manifest in four different alternatives:

- **First Alternative**: Ego vehicle collides with C, which already has passed or is in the middle of the intersection.
- **Second Alternative**: Ego turns right after C already passed the intersection.
- **Third Alternative**: Ego turns right and C collides with E by driving into the ego vehicle's rear. Most of these scenario outcomes happen in a constellation similar to car-following scenarios and, therefore, are not suitable for all types of criticality metrics.

- **Fourth Alternative**: Ego turns right before C crosses the intersection.
- **Target world position**: The synchronization point for the other car. The x-coordinate range in m is defined as $s_{x_{sync}} \in \{27.0, ...68.0\}$.
- **Car delay**: The other car has to wait for a given time $t_{C_{delay}}$ in s before accelerating, where $t_{C_{delay}} \in \{0.05, ..., 2.0\}$.
- **Init distance**: The ego vehicle has to travel for a certain distance $d_{C_{init}} \in \{0.05, ..., 8.0\}$ in m before C's routing is initialized.

The two behavior models utilized for E and C introduce hidden parameters that are not part of the scenario parameterization but influence the scenario outcome. For example, the ego vehicle brakes for C because C is too close for the ego vehicle to continue driving. However, this proximity also causes C to brake, resulting in both vehicles slowly approaching the intersection. These outcomes depend on scenario and behavior parameters that are not included in the parameterization and can influence each other. For instance, such parameters may include the minimum distance allowed between two actors or the maximum speed at which an actor is permitted to drive. In addition, some behaviors may only occur in specific scenarios when certain parameters are present, such as reactions to other actors. These dependencies and influences are considered hidden parameters and are not present in scenarios 1A and 1B, where other actors do not affect trajectory followers. At this intersection, according to traffic rules, the ego vehicle coming from the right has the right of way over C. However, it is important to note that both driver models are not aware of all traffic rules, including the rule to give way. All simulations for scenario 2 were conducted using the Carla Simulator. The simulation of the grid search with 3374 simulated scenarios took 39.87 hours on a machine with an Intel Core I9-9900k with 3.6GHz, 32GB DDR4, and an 8GB RTX 2080S graphic card.

5.3 Experiments

Three sets of experiments were conducted. The first set of experiments only used scenarios 1A and 1B, and scenario exploration was conducted under different circumstances. In the second set, cluster analysis was performed for scenarios 1A and 2. In the third set, probability and uncertainty were computed for all three scenarios:

- Experiment 1
  - 1-dimensional parameter space exploration of scenario 1A and 1B

  - 3-dimensional parameter space exploration of scenario 1A and 1B, which can be reduced to a 2-dimensional space
  - Forcing influence of negligible parameter
- Experiment 2
  - Clustering and archetype analysis of scenario 1A
  - Exploration and clustering of scenario 2
  - Comparison of exploration and clustering results to grid search
- Experiment 3
  - Probabilistic evaluation of scenario 2
  - Probabilistic evaluation of scenarios 1A and 1B

5.3.1 Experiment 1

Experiment 1 aims to explore and optimize the criticality of scenarios 1A and 1B from Section 5.2.1. The focus is on the situation between the ego vehicle and the pedestrian, who is the most vulnerable road user in these contexts. The simulations were performed using the tool described in Section 5.1.1, which utilizes CarMaker and the driving function integrated via OSI. The optimization was done using Bayesian optimization and Gaussian processes in the optimization process (Section 4.4). All metrics were taken on a nanoscopic scenario quality level (every simulation step) and reduced to a maximum or minimum value depending on the metric (microscopic scenario quality) (Section 4.3.2). However, the optimization was only done for Euclidean distance, trajectory distance, PET, WTTC, and gap time (Section 5.1.2).

All parameters were varied and optimized according to the parameter ranges (Section 5.2.1). The exploration of scenarios 1A and 1B involved approximately 100 executions in total for the 1-dimensional experiment. For the 3-dimensional experiment, there were 373 executions of concrete scenarios within the logical scenario in total for 1A and 449 for 1B. These numbers represent the sum of all concrete scenario executions of all optimization runs. One optimization run was started for each metric (Euclidean distance, trajectory distance, PET, WTTC, gap time). The termination criterion was set to 105 scenario executions or when the internal model of the scenario space could no longer increase its precision with further runs. All explorations took 105 steps, except for trajectory distance, which stopped early. If the same parameter sets were selected for different metrics, a second simulation run was not conducted.

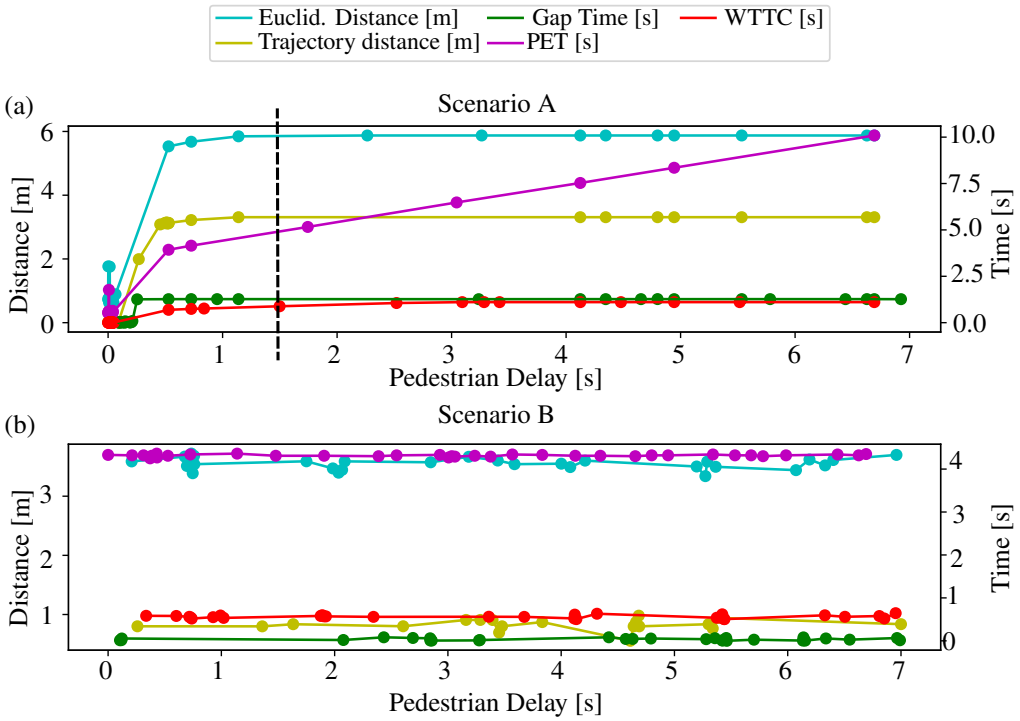


Figure 5.8: Bayes optimization results of both scenarios on a one-dimensional parameter space. The metrics used for optimization are Euclidean and trajectory distance, gap time (GT), post-encroachment-time (PET), and worst-time-to-collision (WTTC). Each dot corresponds to the worst measured criticality of a concrete scenario.

Scenario 1: 1-dimensional Exploration and Optimization

In the initial sub-experiments, exploration occurred in a one-dimensional parameter space, with the sole variable optimized across all simulations being **pedestrian delay** $t_{P_{delay}}$. The other two parameters were set to constant values: the ego vehicle s-coordinate of the start position was set to $s_{E_{start}} = 60.0\,\mathrm{m}$ in scenario 1A and $s_{E_{start}} = 67.0\,\mathrm{m}$ in scenario 1B. Additionally, a constant value of $v_{C_{max}} = 15.0\,\mathrm{m/s}$ was applied in both scenarios. Figure 5.8 (a) and (b) present the results for scenarios 1A and 1B, respectively Each dot represents the most severe criticality measured for each metric during the simulation of a concrete scenario with the given pedestrian delay. All metrics are shown on a scale in seconds or meters. It is important to note that critical situations between the ego vehicle and other traffic participants, besides pedestrians, may arise in certain scenarios. However, the analysis has omitted these specific scenarios due to the metric selection. One-on-one metrics are typically used to estimate the criticality of scenarios. Furthermore, it is not possible to directly compare the results of different metrics. For example,

in the 3-dimensional experiments, a scenario marked with a yellow dot in Figure 5.9 (b) may not have the same level of criticality as a yellow scenario data point in Figure 5.9 (c). Critical scenarios were identified in scenario 1A with pedestrian delays near $0.0\,\mathrm{s}$. These can be seen in the bottom right corner of Figure 5.8 (a), where all criticality values are around 0.0. The criticality in Figure 5.8 (a) reaches its maximum from a delay of $0.0\,\mathrm{s}$ to $1.5\,\mathrm{s}$. The threshold is indicated by a dashed line in Figure 5.8 (a). In Figure 5.8 (b) there is no such observable threshold and, therefore, not line indicating it. However, beyond a delay of approximately $1.5\,\mathrm{s}$, no changes in criticality were observed for all metrics except PET. The PET values increased beyond this point, indicating that the scenarios were no longer critical. Notably, Allen, Brian, L. et al. [16] set the threshold for critical scenarios at $PET < 1.5\,\mathrm{s}$. In scenario 1B, the criticality metrics remained constant despite changes in pedestrian delays. Therefore, the outcome of scenario 1B was not affected by pedestrian delays, considering the chosen values for the other two parameters.

Scenario 1: 3-dimensional Exploration and Optimization

In the second sub-experiment, a parameter space exploration was conducted in a 3-dimensional scenario. The experiments were carried out in two runs, one with scenario 1A and the other with scenario 1B The results for the experiments are shown in Figure 5.9, where (a) shows results for the Euclidean distance, (b) trajectory distance, (c) post-encroachment time, (d) worst-time-to-collision, (e) gap time, (f) distance between ground truth and sensor data measured with dynamic time warping (DTW), and (g) inverse universal TDP. The color scale on the right is inverted for (f) and (g) since high values indicate high criticality for both metrics. It is important to note that all values presented indicate criticality, with lower values indicating higher criticality.

In scenarios 1A and 1B, the **car speed** has little to no visible influence on the simulation results regarding the criticality of the pedestrian's situation. Therefore, a three-dimensional plot can be reduced to the two dimensions of **pedestrian delay** and **ego s-coordinate**. However, this does not imply that there is no influence at all, and outliers or deviations in the plot that are not congruent with other values in their neighborhood might be influenced by the car's speed. One possible explanation for the lack of influence is that in scenario 1A, the trajectories of the ego vehicle and the car do not intersect, and the ego vehicle does not need to react to the car. Additionally, the other car starts close to the intersection in both scenarios, which may not have given it enough time to accelerate until it reaches the intersection and consistently passes the ego vehicle at the same speed.

The recordings of all experiments allow a replay that humans can view. These replays were used for further analysis of what happened in different areas. In general, scenario 1A has four different outcomes regarding the criticality of the pedestrian's situation, which also can be seen in the

traffic sequence charts in Figure 5.5: the ego vehicle reaches the intersection where it stops for the pedestrian crossing the street. This result can be observed in scenarios at the bottom left corner in Figure 5.9 (a)-(f), where the criticality decreases for most metrics.

In the second result, the ego vehicle passes the intersection before the pedestrian or the truck. These scenarios are primarily located at the top and right halves of Figure 5.9 (a)-(f). The other scenarios involve the ego vehicle arriving at the intersection just after the truck, with the pedestrian out of sight. In these situations, the ego vehicle stops too late, resulting in contact with or hitting the pedestrian. These critical scenarios are located in the diagonal area near the bottom left corner of Figure 5.9 (a)-(f). This diagonal reaches from $t_{P_{delay}}$ between 3 and 4 and $s_{E_{start}} = 27.99$ to $t_{P_{delay}} = 0.0$ and $s_{E_{start}}$ between 52 and 62.

The criticality in Figure 5.9 (g) differs from the rest since it describes the TDP as explained in Section 4.8. This metric considers not only pedestrians but also other traffic participants, such as trucks and cars. Therefore, critical situations between the ego vehicle and the truck are also included, leading to different criticality patterns in the parameter space. The truck has a more significant influence on criticality than the pedestrian: Previously found critical situations between pedestrians and ego vehicles often occur when the ego vehicle is stationary or nearly stationary while the pedestrian passes by. Using distance as a metric still identifies highly critical situations, even when the distance is almost $0\,\mathrm{m}$. The truck is constantly moving at $40.0\,\mathrm{m/s}$, which leads to a faster rise of the TDP than the pedestrian's behavior. The middle part of Figure 5.9 (g) shows critical scenarios where the ego vehicle starts with the distance of $60.0\,\mathrm{m}$ or more to the intersection. The starting distance refers to the point at which the ego must react to the truck's entry into the intersection or when all actors are crossing the intersection in close proximity. In cases where the ego vehicle starts closer to the intersection with a smaller s-coordinate, it crosses the intersection before the truck and other participants.

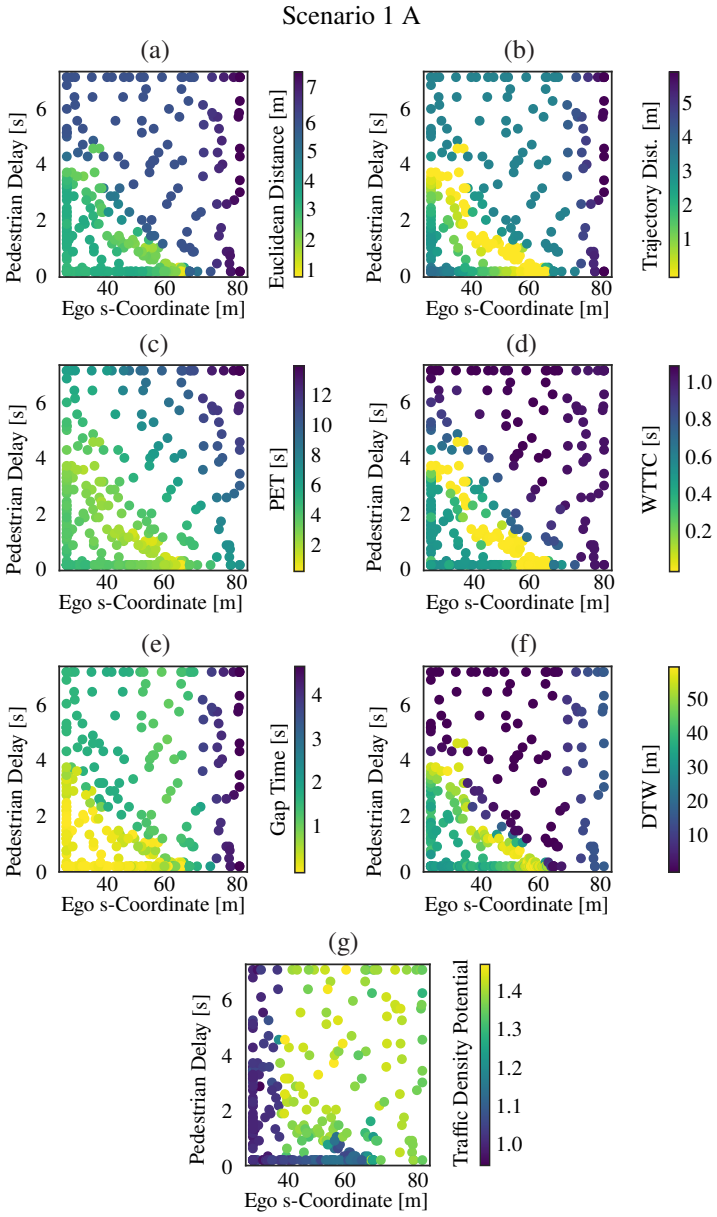


Figure 5.9: Bayes optimization results of scenario 1A. Each dot corresponds to one concrete scenario. The values of the metrics are the most critical value measured with the regarding metric. The metrics used for optimization are Euclidean (a) and trajectory (b) distance, post-encroachment time (PET) (c), worst-time-to-collision (WTTC) (d), gap time (e), distance (DTW) between ground truth and sensor data (f), and inverse universal TDP (g). Yellow indicating critical and dark blue or purple indicating non-critical scenarios. The color scale is inverse for (f) and (g) since high values here indicate high criticality. For the other values low values indicate high criticality.

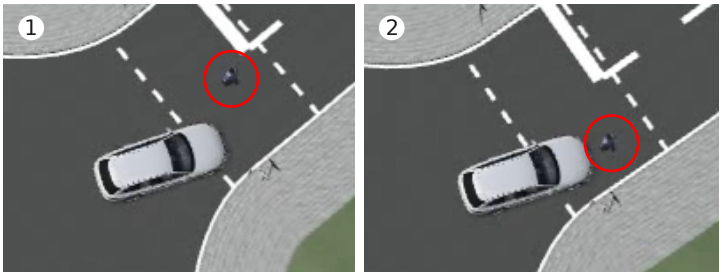


Figure 5.10: Screenshot from IPG CarMaker 8.0.2 showing a less critical scenario (left) and critical scenario where the ego vehicle almost hits the pedestrian (right).

The replays of scenario 1B demonstrate that the ego vehicle responds to the pedestrian, the truck, and the other car. The results of scenario 1B are presented in the traffic sequence chart in Figure 5.6. The different outcomes are localized in the parameter space as follows: The less critical area in the bottom left corner, which can be observed in Figure 5.11 (b) and (d), results from a parameter set where the ego vehicle intersects the pedestrian's path after they cross the road.

To enhance comprehension of the simulation, Figure 5.10 displays two screenshots of the scenario. Both show the pedestrian passing in front of the ego vehicle. However, in image (1), the ego vehicle can brake comfortably for the pedestrian, whereas in image (2), it has to brake strongly to avoid a crash. These situations are variations of the first alternative in Scenario 1B, as shown in Figure 5.6.

When looking at the plots in Figure 5.11 and the corresponding replays, several observations can be made: In Figure 5.11, the ego vehicle passes the intersection before the pedestrian on the right side of each plot where the s-coordinate has its highest values. This is followed by a critical line around an s-coordinate of $70\,\mathrm{m}$ with near-collisions. These near-collision scenarios, which are detected in plots (a), (b), (d), and (e) in Figure 5.11, result from a change in the order in which all actors cross the intersection. When the s-coordinate is small enough (around $65\,\mathrm{m}$ - $70\,\mathrm{m}$), Car C is not following the ego vehicle anymore but driving right in front of it. Thus, the ego vehicle is unable to detect the pedestrian and can only apply short-term braking. Car C crosses the intersection before the ego vehicle with sufficient distance in the middle and left part of the ego s-coordinate, while the ego vehicle waits for the pedestrian to pass.

The slightly more critical cluster in (a) and (c), at delay $2\,\mathrm{s}$ and s-coordinate around $30\,\mathrm{m}$ results from interference with the truck, which hinders the ego vehicle from turning left and arrives at the same time as the pedestrian for whom the ego has to wait. Interestingly, (f) shows that in this cluster, the DTW is smaller than in the surrounding concrete scenarios, which means that

the deviation between the sensor data and the ground truth is smaller, too. This deviation can be explained by the pedestrian not walking while the truck obstructs the ego's view.

Figure 5.11 (g) shows this scenario's TDP. In contrast to scenario 1A, scenario 1B shares more similarities with the patterns in (a)-(f), although some parts are inverse to other criticalities. This divergence arises from the fact that in 1B, the ego must react to all three traffic participants, resulting in more interconnection between their behaviors. Furthermore, the TDP reacts to dense traffic, particularly with varying velocities, and tends to increase when actors are in close proximity to each other. In general, TDP indicates critical situations on the left in the plot of Figure 5.11 (g) when the ego has to react to the truck and on the right when the other car is driving right behind it. The pedestrian's influence in this scenario is low since it moves slower and does not act unexpectedly. Finally, it must be mentioned that pedestrians have a low impact on TDP and that TDP was developed with data that does not contain pedestrians (see Section 4.8 and Appendix B.2). One way to increase the impact on pedestrians is to also increase their weight in the calculation. The color scale on the right of Figure 5.11 (f) and Figure 5.11 (g) is inverted compared to the others since, for both metrics, high values indicate high criticality. For all others, the lower the value, the higher the criticality.

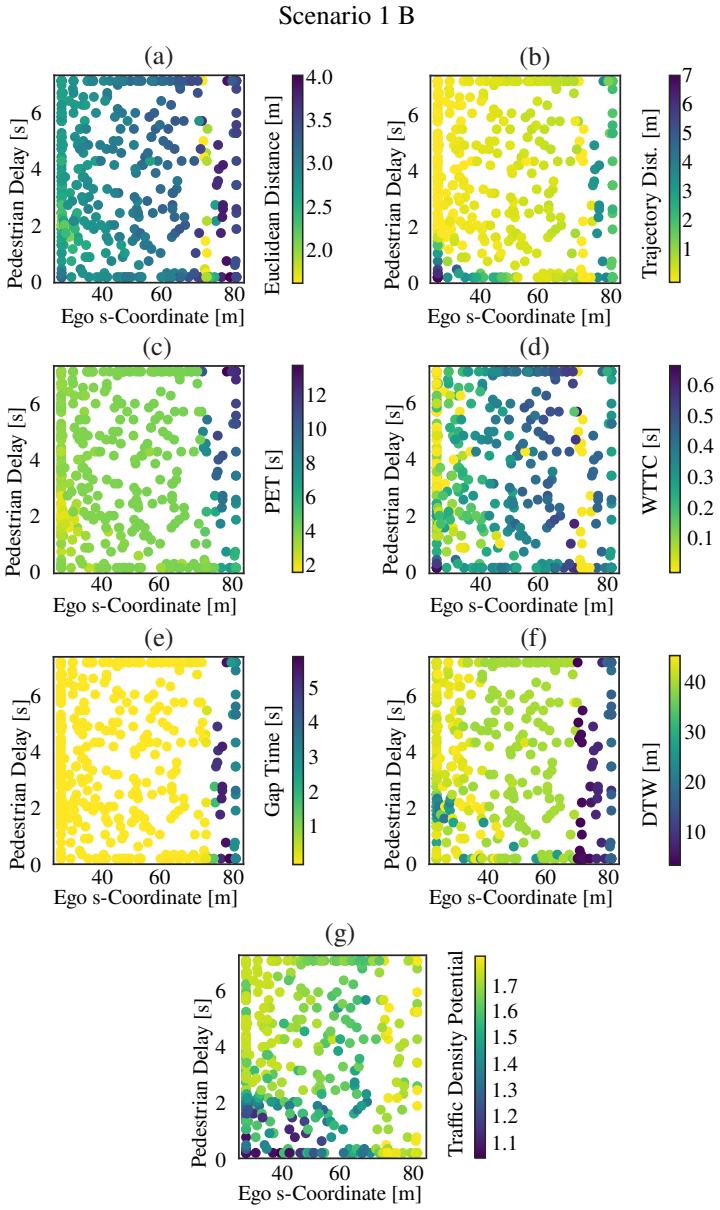


Figure 5.11: Bayesian optimization results for scenario 1B. ach dot represents a concrete scenario, and the corresponding metric values are shown.The metrics used for optimization include Euclidean (a) and trajectory (b) distance, post-encroachment time (PET) (c), worst-time-to-collision (WTTC) (d), gap time (e), distance (DTW) between ground truth and sensor data (f), and TDP (g). The color scale used in this context employs yellow to indicate critical scenarios and dark blue or purple to indicate non-critical scenarios. The color scale is inverted for (f) and (g) as high values in these cases indicate high criticality.

Scenario 1B: Forcing Influence of Negligible Parameter

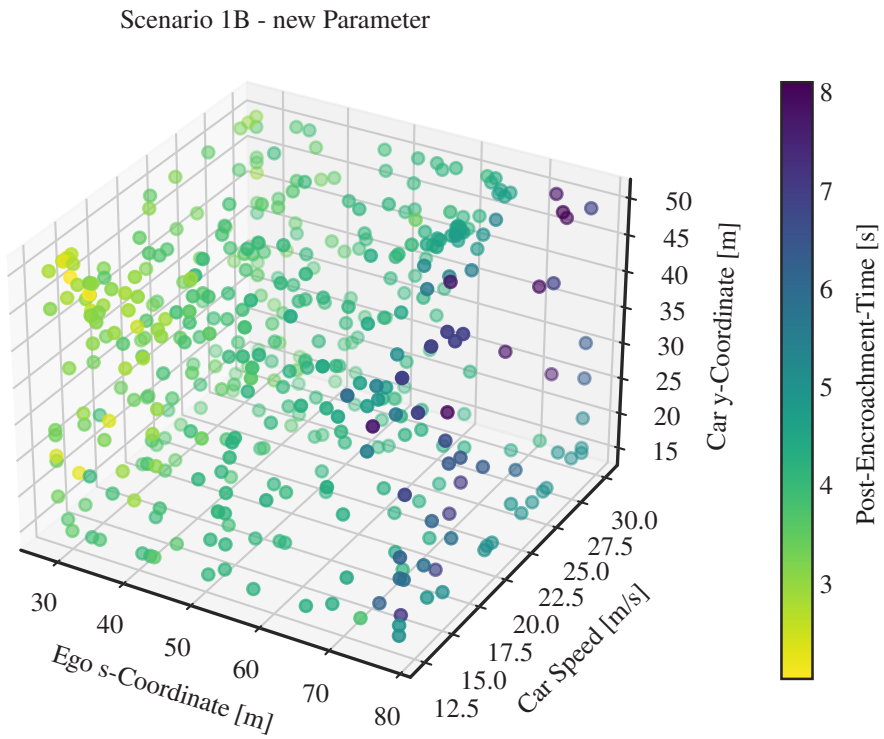


Figure 5.12: Exemplary Bayesian optimization results of scenario 1B on a three-dimensional parameter space for the metric post-encroachment-time with a replacement.

In the third setup, the pedestrian delay $t_{P_{delay}}$ was replaced by a new variable $y_{C_{start}}$ to see if the car speed has more influence on the scenario outcome:

- **Car position**: The y-coordinate $y_{C_{start}}$ in m fulfils $y_{C_{start}} \in \{15.0, ..., 50.0\}$. 50 samples with a step size of $0.7\,\mathrm{m}$ were taken.

All three variables were varied and optimized, and the five previously mentioned metrics were used. The results of scenario 1A indicate that neither the car's speed nor its y position have an impact on the outcome. The outcome is not surprising, as there is no intersection between the trajectory of the ego vehicle and the car. The results indicate that the outcome is solely dependent on the s-coordinate. As Figure 5.12 shows, the ego s-coordinate remains the most influential factor. However, in certain areas of the parameter space, car speed and y-coordinate also impact scenario criticality.

Conclusion of Experiment 1

The comparison of the results of both scenarios leads to the following conclusions:

(1) Scenario 1A exhibits a higher variance in criticality for distance, WTTC, and DTW than scenario 1B. Trajectory distance, PET, and gap time are equally scaled in both scenarios. This is because 1A contains more critical situations, including collisions. However, the TDP reaches higher criticality in 1B due to the presence of more intersection trajectories and all actors coming closer together than in 1A.

(2) The speed of Car C has no significant impact on scenario 1A and minimal impact on 1B. In scenario 1A, Car C does not even intersect the ego vehicle's trajectory.

(3) Some metrics are more sensitive, e.g., trajectory distance and gap time. Both find more highly critical concrete scenarios than the other metrics.

(4) The metrics have varying gradients, which is evident in scenario 1A where PET has a less steep gradient compared to trajectory distance or gap time.

(5) All used one-on-one metrics lead to similar patterns in criticality, e.g., bottom left corner in scenario 1A or scenarios with a small ego vehicle s-coordinate in scenario 1B.

(6) An universal metric like TDP can show different patterns resulting from other critical situations being overlooked by other metrics since no restrictions regarding the considered traffic participants have to be made. However, pedestrians in particular have less influence than vehicles due to their lower maximum speed.

5.3.2 Experiment 2

The first part of experiment 2 focuses on the clustering and archetype analysis of scenario 1A. These techniques are based on the exploration of scenario 1A (Section 5.3.1). The second part consists of a clustering of scenario 2 (Section 5.2.2). An exploration of scenario 2 was done equally to the explorations of experiment 1 as a pre-step for the clustering. The simulations of the exploration were executed using Carla. The optimization was done using Bayesian optimization and Gaussian processes in the optimization process (Section 4.4). Four of the metrics mentioned in Section 5.1.2 were taken on a nanoscopic scenario quality level (every simulation step) and reduced to a maximum or minimum value depending on the metric (microscopic scenario quality) (Section 4.3.2). These metrics are Euclidean distance, trajectory distance, WTTC, and TDP. The other metrics were omitted since some collisions in this scenario take place in a car-following scenario where these three metrics were not usable due to the vehicle constellation.

Scenario 1A: Clustering and Archetype Analysis

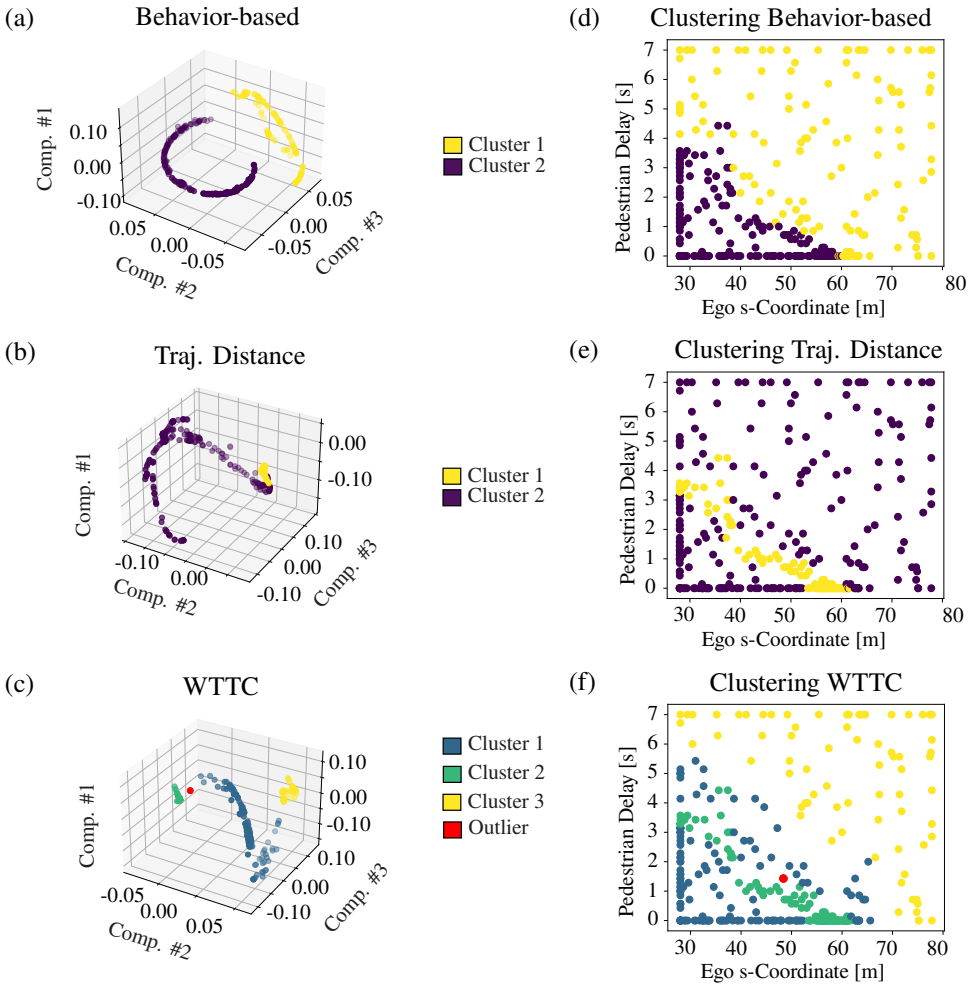


Figure 5.13: (a)-(c) Projections into different 3D kernel spaces. (d)-(f) Clustering from kernel spaces visualized in scenario parameter spaces.

The available data resulting from the exploration step can be used for clustering. The following steps were done with the results of scenario 1A. First, the parameter space data must be projected into the kernel space (Section 4.5.2). The projection is necessary for two reasons: (1) the concrete scenarios were selected by the optimization algorithm and, thus, selected with a high density around critical areas but no relation to what was happening, and (2) the distribution of scenarios in the parameter space is not related to any clusters regarding the sequence of situations as depicted in the traffic sequence charts. The distance matrix for describing the similarity

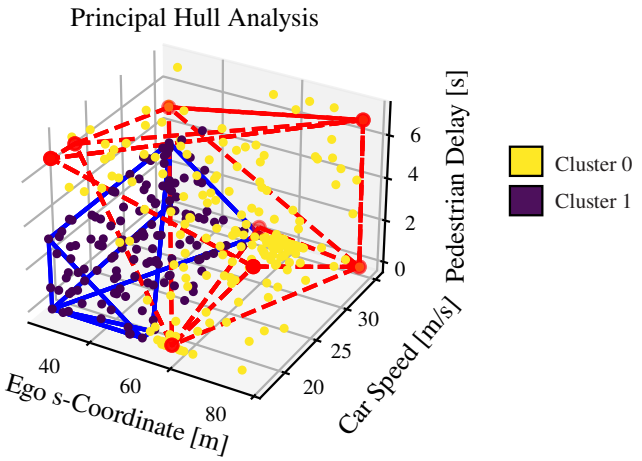


Figure 5.14: Principal Hull Analysis (PHA) to identify archetypal scenarios.

between single concrete scenarios has to be computed (see Section 4.5.1.3). There are two different possible approaches: behavior-based distance (Section 4.5.1.2) and criticality-based distance (Section 4.5.1.1). DBSCAN was used to cluster within the kernel space as explained in Section 4.5.3. Finally, an archetype analysis was done as depicted in Section 4.6.

Figure 5.13 (a)-(c) display the projected concrete scenarios in the kernel space, respectively. After the projection, the clustering was done in the new space. Each image indicates the clusters in the kernel's space by different colors. Additionally, Figure 5.13 (d)-(f) illustrate the same cluster assignments in the original parameter space reduced to two dimensions for better visualization of the scenarios, given the negligible influence of $p1\_speed$.

As mentioned before for scenario 1A, the critical diagonal reaches from $t_{P_{delay}}$ between 3 and 4 and $s_{E_{start}} = 27.99$ to $t_{P_{delay}} = 0.0$ and $s_{E_{start}}$ between 52 and 62. Near this critical diagonal, scenarios depict situations where the car either collides with or closely approaches the pedestrian. Finally, above the diagonal are scenarios where the pedestrian crosses the intersection after the ego vehicle. Among these three categories, scenarios lying further away from the diagonal are generally considered less significant as they involve minimal interaction between the ego vehicle and pedestrian, thereby suggesting a potential for reducing the number of concrete scenarios.

Figure 5.13 (a) depicts the results of the scenarios clustered regarding the ego vehicle's trajectory (behavior-based) in the kernel space after performing a kernel PCA ($\gamma = 100.0$). At the same time, (d) shows the cluster assignments in the original space. Interestingly, the cluster borders converge at the area with the highest criticality. Pairs (b) and (e) represent the kernel space where the trajectory distance (criticality-based) was utilized as a distance metric for computing

the kernel matrix. In contrast, (c) and (f) represent the kernel space using WTTC (criticality-based) as the distance metric. Clustering with behavior-based distance identifies two distinct clusters: high-criticality (yellow) scenarios and low-criticality (purple) clusters. On the other hand, criticality-based distance like WTTC can differentiate between high criticality (green), medium criticality (blue), and low criticality (yellow). A comparison of Figure 5.13 (e) and (f) and Figure 5.9 (b) and (d) demonstrates that the clustering closely aligns with the measured criticality gradient throughout the simulation. Figure 5.14 shows an additional principal hull analysis (red and blue vertices and edges) that encompasses both scenario clusters in the scenario parameter space. The resulting archetypes of the hull analysis are a suitable foundation for a reduced scenario set, which can be supplemented with prototypical scenarios from the center of each cluster. For the implementation of the experiments, the python library py_pcha [23] was used. This reduced set is represented by all archetype scenarios (corner cases) combined with the prototypes (average) of each cluster. For scenario 1, where eight scenarios for each cluster is one of the smallest working principal hulls in this 3-dimensional parameter space, this leads to 18 scenarios as a reduced set representing logical scenario 1.

Scenario 2: Exploration and Clustering

Scenario 2 was explored through 315 concrete scenario executions, with the termination criterion being set at 105 scenario executions or when the internal model of the scenario space could no longer increase its precision with further runs. In this experiment, all three metrics stopped the exploration after 105 executions. Both actors are equipped with bounding boxes and collision detection for physics simulation, which may prevent them from reaching their final destination. In simulations 1A and 1B, collision detection is not included, allowing the simulation to continue after a collision. In contrast to scenarios 1A and 1B, all three parameters influenced the scenario outcomes, as depicted in Figure 5.15. Scenario 2 demonstrated three distinct outcomes as shown in Figure 5.7:

(1) Vehicle C crosses the intersection before the ego vehicle, corresponding to scenarios classified as less critical by the applied criticality metrics with parameters $time$ and $distance$ close to zero.

(2) Collisions between the two vehicles.

(3) The ego vehicle passing before vehicle C, represented by scenarios in the top right region of the parameter space with high $time$ and $distance$ values.

In Figure 5.15, all plots (a)-(f) of scenario 2 show a critical region marked as the cluster in the middle. Unlike Figure 5.5 and Figure 5.6, Figure 5.15 does not show the criticality gradient and,

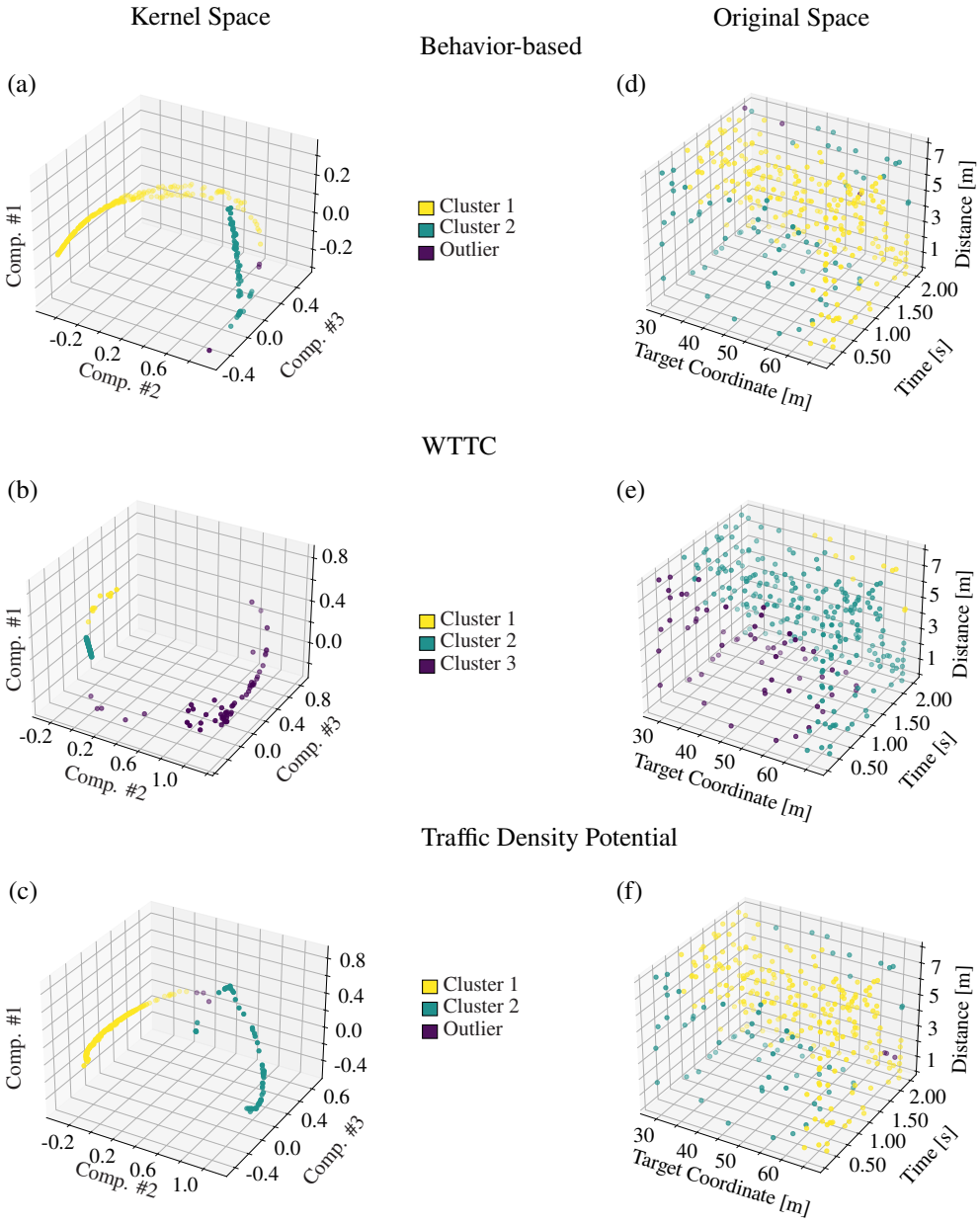


Figure 5.15: (d)-(f) Concrete scenarios on the original scenario space, colored by their assigned clusters. (a)-(c) Projections into the 3D kernel spaces, depending on the chosen distance matrix, and colored by cluster assignment.

therefore, the color is not connected to the criticality value, e.g., yellow does not indicate high criticality and stands for a clusters assignment. This region is critical as it includes scenarios

where accidents are caused by both the ego vehicle and vehicle C. It is important to note that critical scenarios near the point where vehicle C crosses involve the ego vehicle colliding with vehicle C, which then leads to accidents caused by vehicle C. Unlike scenarios 1A and 1B, which employ DTW to measure distances between ego trajectories, the clustering process distinguishes accidents as a separate cluster due to the simulation terminating upon a crash. As a result, typical clusters emerge from the clustering results:

(1) Non-crash situations represented by clusters 1 and 3 (3 only existing in Figure 5.15 (b)). WTTC distinguishes between who is crossing the intersection first.

(2) Accidents at the intersection denoted by cluster 1.

(3) A few outliers marked as purple.

In Figure 5.15 (b), the accident-free scenarios are projected and divided into two separate clusters, unlike (a) and (c).

The critical region, identified as blue cluster in the parameter space in (d) and yellow in (e) and (f), encompasses scenarios where accidents occur. Specifically, scenarios in the vicinity of the region where vehicle C crosses first involve the ego vehicle colliding with vehicle C, leading to subsequent accidents caused by vehicle C. The clustering process utilizing DTW distances between ego trajectories effectively distinguishes accidents as a separate cluster, differentiating them from non-crash scenarios. This is in contrast to the clustering outcome observed in Scenario 1A. In summary, the clustering analysis produces two primary clusters: (1) non-crash situations and (2) accidents at the intersection.

Further insights can be gained from the kernel space. In Figure 5.15, all critical scenarios are closely situated together. Moreover, clusters 1 and 2 exhibit the most distinct boundaries in the kernel space where TDP was employed. This observation is attributed to TDP exhibiting a steep gradient in areas where critical and non-critical clusters intersect, thereby enabling clear demarcation between the clusters.

A principal hull analysis was not conducted on the clusters of Scenario 2 due to the non-convex nature of the identified critical clusters. These results allow future research to develop methods and reduce the scenario set for non-convex cluster shapes.

Comparison of Exploration and Clustering Results to Grid Search

One advantage of the chosen Bayes optimization is that it provides information about the confidence of the criticality at the areas around simulated data points, unlike other optimization

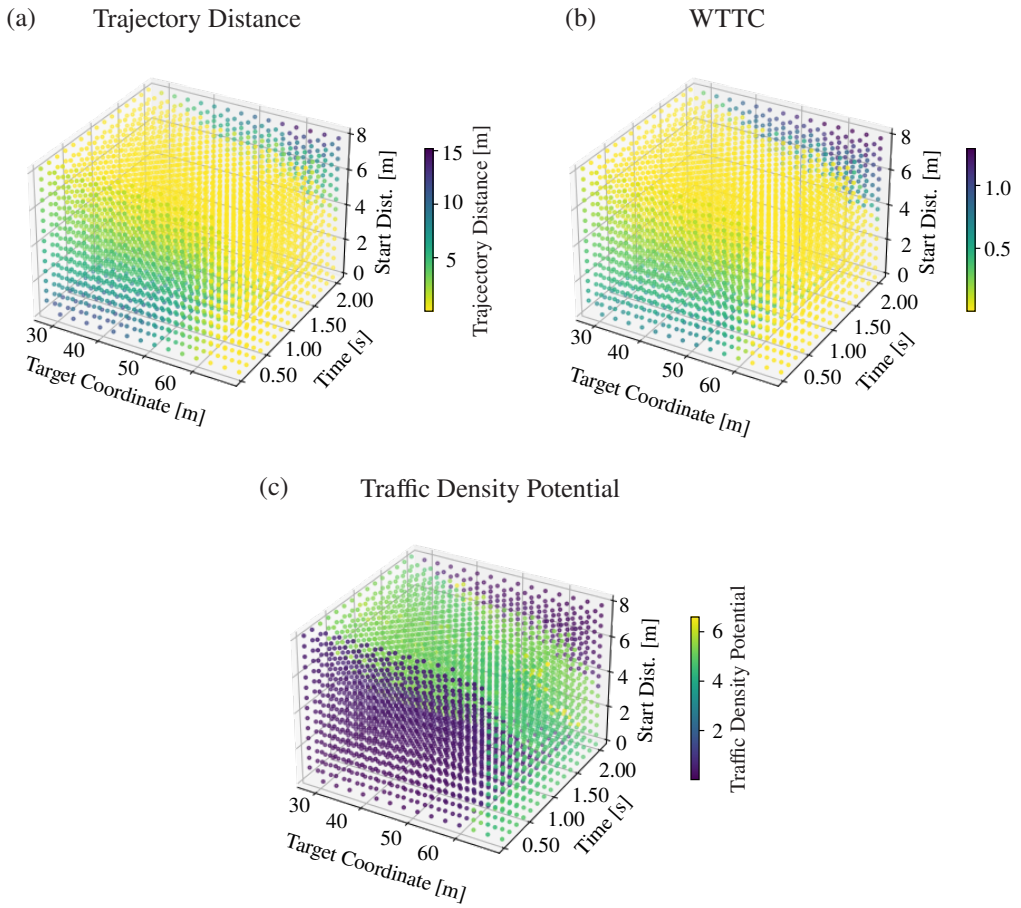


Figure 5.16: Grid of scenario 2 colored regarding the criticality of different metrics: trajectory distance, WTTC, and TDP.

algorithms such as genetic algorithms. However, even though this information is given, it is based on assumptions made in the model design, e.g., choosing a Gaussian RBF kernel. Another possible approach is to use a grid for parameter combinations. Grid search is a systematic method used to identify critical areas within the parameter space by simulating all parameter combinations through brute force. Typically, a set step size is used for each parameter. This approach results in exponentially growing calculation and simulation time when additional parameters are introduced into the scenario space. Mori et al. [106] suggests that critical scenarios within a grid may be overlooked. Additionally, in a scenario space defined by three parameter ranges, with 15 steps per parameter, the simulation effort already includes 3375 scenarios (39.5h of simulation time), more than ten times the optimization needed for scenario 2.

Figure 5.16 displays all 3375 simulated scenarios color-coded based on their criticality, measured using a different metric for each plot, as demonstrated in scenarios 1A and 1B in Figure 5.9 and Figure 5.10. The number of simulations greatly impacts the evaluation of clustering Calculating the distance matrix for one of the criticality metrics takes about 3 hours for approximately 300 concrete scenarios. However, the computation time grows exponentially, taking almost 16 days for 3375 scenarios The lengthy calculation time makes clustering analysis unfeasible for grid search with a narrow grid size, and expanding the grid size could potentially overlook critical scenarios.

Conclusion of Experiment 2

The comparison of the results from scenario clustering and archetype analysis of 1A, and clustering of 2, leads to the following conclusions:

(1) The absence of collision detection in 1A results in a behavior-based clustering that allows the algorithm to divide the scenario space into two different orders: one in which the ego vehicle crosses the intersection before the pedestrian, and another in which the pedestrian crosses first. In scenario 2, where the simulation stops after a collision, the results show a distinct cluster for collision situations.

(2) Archetype analysis can only be performed on convex clusters using principal hull analysis. Therefore, it cannot be performed for scenario 2. It is only determined after clustering whether archetype analysis can be performed for a given scenario or not.

(3) The criticality metrics and behavior-based clustering produced matching results for scenario 2. However, WTTC was able to further divide the non-critical cluster based on their outcomes.

(4) In scenario 2, where there are only two actors, TDP finds the same critical area as other metrics since there are no situations other metrics potentially overlook.

5.3.3 Experiment 3

This experiment utilizes the likelihood models resulting from Bayes optimization with Gaussian processes, as described in Section 4.7. The surrogate model resulting from the Gaussian regression can predict the outcome of a given concrete scenario (parameter set within the parameter space). These results may vary based on the model's standard deviation for this point in space. This information enables the evaluation of the mean and standard deviation for each model and the

entire parameter space of a logical scenario. In scenario 2, the predicted values are compared to the grid search results mentioned earlier.

Scenario 2: Probabilistic Evaluation

The concluding experiment evaluates the assessment of the objective function model using the approach explained in Section 4.7. The model is used to predict the criticality of all metrics, including trajectory distance, WTTC, and TDP. The surrogate model, which is the optimized objective function, anticipates the criticality results within the logical scenario spaces derived from simulations, as shown in Figure 5.16. It is important to recognize that comparing simulations to a grid may not always be possible due to the high dimensionality of the scenario parameter space and the time required to run a single simulation.

For scenario 2, a comparative analysis of the predicted results is possible based on the grid simulations executed in the preceding section. Figure 5.17 ((a), (c), and (e) present the prediction outcomes in a grid format, which can be compared with the computed grids shown in Figure 5.16. The results of all three trained models exhibit parallel patterns with the simulated data. However, the TDP patterns exhibit the most significant deviation. This is because the original data is less smooth than the prediction. It should be noted that the predicted results' scales differ from the simulated grid results in Section 5.3.2, as shown in Table 5.2. The evaluation shows that the predicted trajectory distance is more critical than the empirical concrete scenarios. In contrast, WTTC and TDP render predictions with less divergence from the simulated data. Noteworthy, Figure 5.17 (b), (d), and (f) delineate the computed standard deviation corresponding to each predicted mean presented in (a), (c), and (e). The trajectory distance exhibits a maximum deviation of approximately $0.15\,\mathrm{cm}$, WTTC of $0.024\,\mathrm{s}$, and 0.25 for TDP. Furthermore, critical regions for trajectory distance and WTTC are identified with higher confidence in certain segments of the parameter space compared to non-critical domains. This is due to the optimization algorithm typically selecting more concrete scenarios in these areas.

Table 5.2: Scenario 2: Comparison of simulated and predicted criticality values.

Metric	Simulated Range	Predicted Range
Traj. Distance	0.0 m to 15.0 m	0.0 m to 8.0 m
WTTC	0.0 s to 1.4 s	0.0 s to 1.75 s
TDP	0.0 to 6.5	0.0 to 6.2

Table 5.3 presents a comprehensive overview of the most and least optimal confidence interval widths collected from all three models. These interval widths have been calculated using a confidence level of 95%. The entries within the table denote that the predicted mean for each

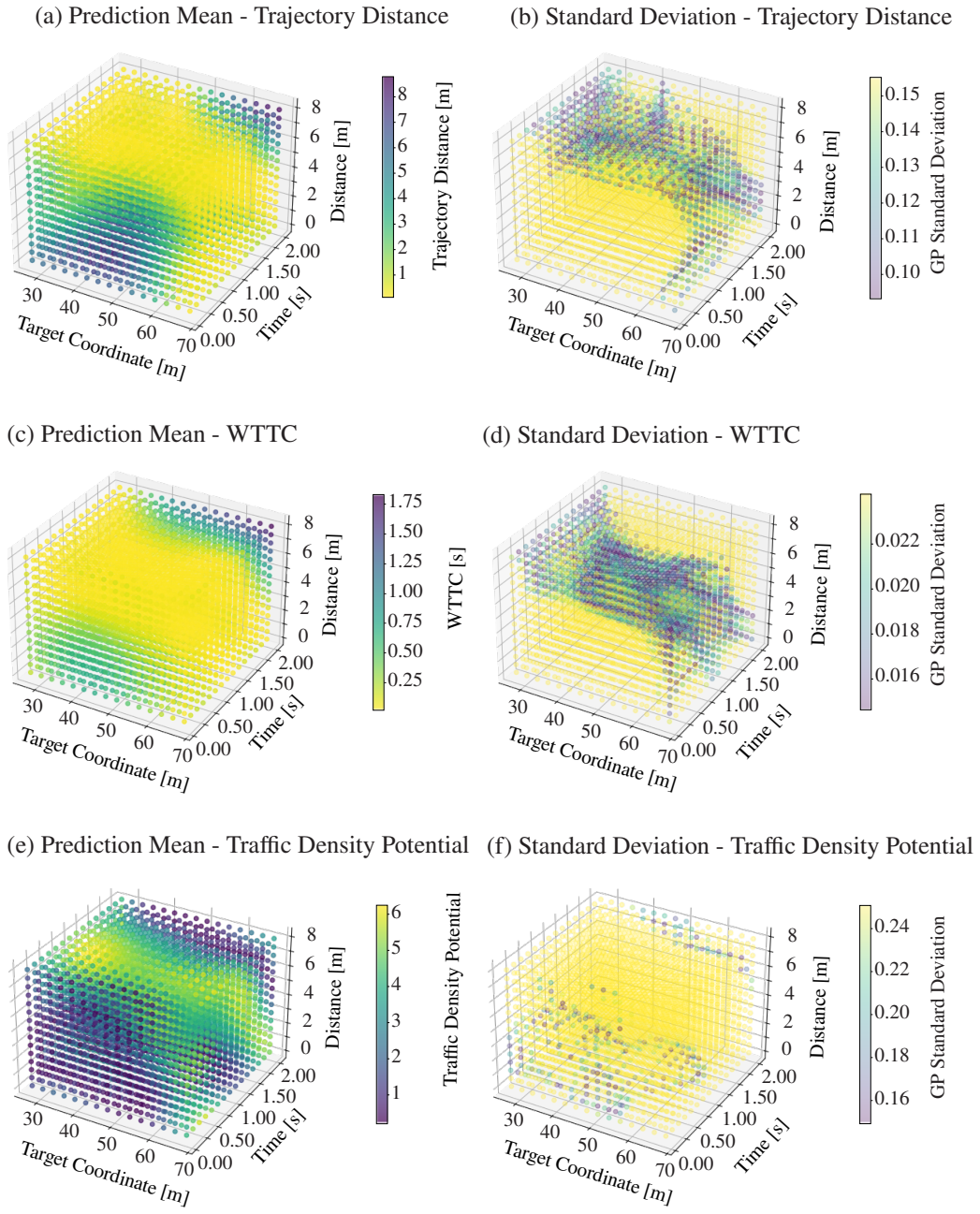


Figure 5.17: Scenario 2: (a), (c), (d): predicted mean as grids for trajectory distance, WTTC, and TDP. (b), (d), (f): standard deviation for all three metrics.

Table 5.3: Scenario 2: Best and worst confidence interval widths.

Metric	Width Best	Width Worst
Traj. Distance	0.37 m	0.62 m
WTTC	0.06 s	0.09 s
TDP	0.59	0.98

corresponding parameter set resides within the 95% confidence interval across all predictions. To illustrate, the best entry for the Worst Time-to-Collision (WTTC) predicts deviations of no more than 0.03 seconds in 95% of cases for parameter points within the interval width. A plot with the confidence interval can be found in Appendix C, as the confidence intervals are related to the standard deviation and the plots show similar patterns. Increasing the number of simulations can enhance the level of certainty, particularly when areas in the parameter space were neglected during the optimization. However, there is always a trade-off between result quality and calculation time.

Table 5.4: Confidence Level 99%

	Below (%)	Within Interval (%)	Above (%)
Distance	35.83	45.02	19.15
WTTC	18.35	67.87	13.78
TDP	30.56	44.87	24.57

Table 5.5: Confidence Level 95%

	Below (%)	Within Interval (%)	Above (%)
Distance	37.55	42.85	20.60
WTTC	25.55	56.46	27.99
TDP	32.25	36.75	31.00

Table 5.6: Critical Values (Confidence Level 95%)

	Below (%)	Within Interval (%)	Above (%)
Distance < 5.0	50.73	41.68	7.60
WTTC < 0.7	32.02	55.19	12.79
TDP > 1.0	4.17	51.06	44.87

In the final part of this experiment, the predicted values were compared to those derived from the simulated grid and evaluated in terms of their accuracy. Table 5.4 and table 5.5 present the

Table 5.7: Non-Critical Values (Confidence Level 95%)

	Below (%)	Within Interval (%)	Above (%)
Distance > 5.0	14.77	8.96	76.27
WTTC > 0.7	24.14	13.79	62.07
TDP < 1.0	71.77	26.97	1.26

confidence intervals for three distinct criticality metrics at confidence levels of 99% and 95%, respectively.

As illustrated in these tables, approximately half of the predictions fall within the confidence intervals. This outcome could be attributed to the limited number of simulations, as only around 120 simulations were performed per metric. The scarcity of simulations is particularly noticeable in non-critical regions, where the optimizer tends to sample fewer points. Consequently, even when deviations remain within the non-critical region, the quality of predictions is suboptimal.

To address this issue, the criticality prediction was analyzed separately for critical and non-critical regions, as shown in table 5.6 and table 5.7. In critical areas, the results are comparable to those reported in the earlier tables. However, in non-critical areas, the surrogate models exhibit a tendency to predict less critical values than those observed in the simulations. Overall, these results indicate, that more simulations might be needed to get a better surrogate model.

Scenario 1: Probabilistic Evaluation

In scenarios 1A and 1B, it is important to acknowledge the non-deterministic nature of the driving function. It should be recognized that simulations conducted with identical parameter sets may result in slight discrepancies, such as 0.01 m, in the resulting criticality metric. When examining the probabilistic results, it is important to keep in mind this fact as it affects the predictions, standard deviations, and accumulates with the uncertainties from the predicted data.

A comparative analysis of the grid-based exploration for scenarios 1A and 1B was omitted due to the long simulation times (approximately 160 days). Furthermore, in contrast to the preceding two-dimensional depictions of scenarios 1A and 1B, the subsequent visualizations adopt a three-dimensional perspective, as depicted in Figure 5.18, Figure 5.19, Figure 5.20, and Figure 5.21. Except for WTTC, which differs from the simulated scenario criticality, the remaining metrics in However, it is important to note that, except for the gap time metric in scenario 1A and the gap time, PET, and trajectory distance metrics in scenario 1B, the predictive outcomes do not exhibit consistent patterns when compared to the outcomes of the exploratory analysis. Moreover, as illustrated in Table 5.8, metrics demonstrating inadequate predictive performance

are characterized by relatively uniform standard deviations and confidence interval widths across the entire parameter space. Upon closer examination of the standard deviations, it is evident that, with the exception of Euclidean distance and PET in scenario 1A, the standard deviation remains within narrow limits: < 0.0952 s for gap time, < 0.0852 m for trajectory distance, and < 0.0252 s for WTTC. For PET, the standard deviation is not completely uniform (non-yellow data points) like Euclidean distance. Further simulations may increase accuracy. In scenario 1B, Euclidean distance and PET simulation results are equally uniform, and further simulations are not expected to yield better results. Typically, the better-performing metric models have a low standard deviation or at least a decreasing standard deviation in critical regions compared to the non-critical zones within the scenario parameter space. This trend signifies a concentration of the optimization process on scenarios characterized by heightened criticality. Appendix C contains an additional plot with the confidence interval.

Table 5.8: Scenario 1A and 1B: Best and worst confidence interval widths and standard deviations.

Metric	Width Best Scenario 1A	Width Worst Scenario 1A	Width Best Scenario 1B	Width Worst Scenario 1B
Eucl. Distance	1.07 m	1.07 m	0.34 m	0.34 m
Gap Time	0.23 s	0.37 s	0.45 s	0.75 s
PET	1.58 s	1.74 s	1.07 s	1.07 s
Traj. Distance	0.21 m	0.34 m	0.35 m	0.59 m
WTTC	0.06 s	0.10 s	0.06 s	0.10 s

Metric	Std Best Scenario 1A	Std Worst Scenario 1A	Std Best Scenario 1B	Std Worst Scenario 1B
Eucl. Distance	0.27 m	0.27 m	0.08 m	0.08 m
Gap Time	0.06 s	0.09 s	0.11 s	0.19 s
PET	0.45 s	0.45 s	0.27 s	0.27 s
Traj. Distance	0.05 m	0.08 m	0.09 m	0.15 m
WTTC	0.02 s	0.03 s	0.02 s	0.03 s

Conclusion of Experiment 3

The results of experiment 3 lead to the following conclusions:

- Surrogate models can predict outcomes across the entire parameter space of the scenario.
- The predictions can be utilized to compute the predicted mean for each point, along with its standard deviation, uncertainty, and confidence interval.

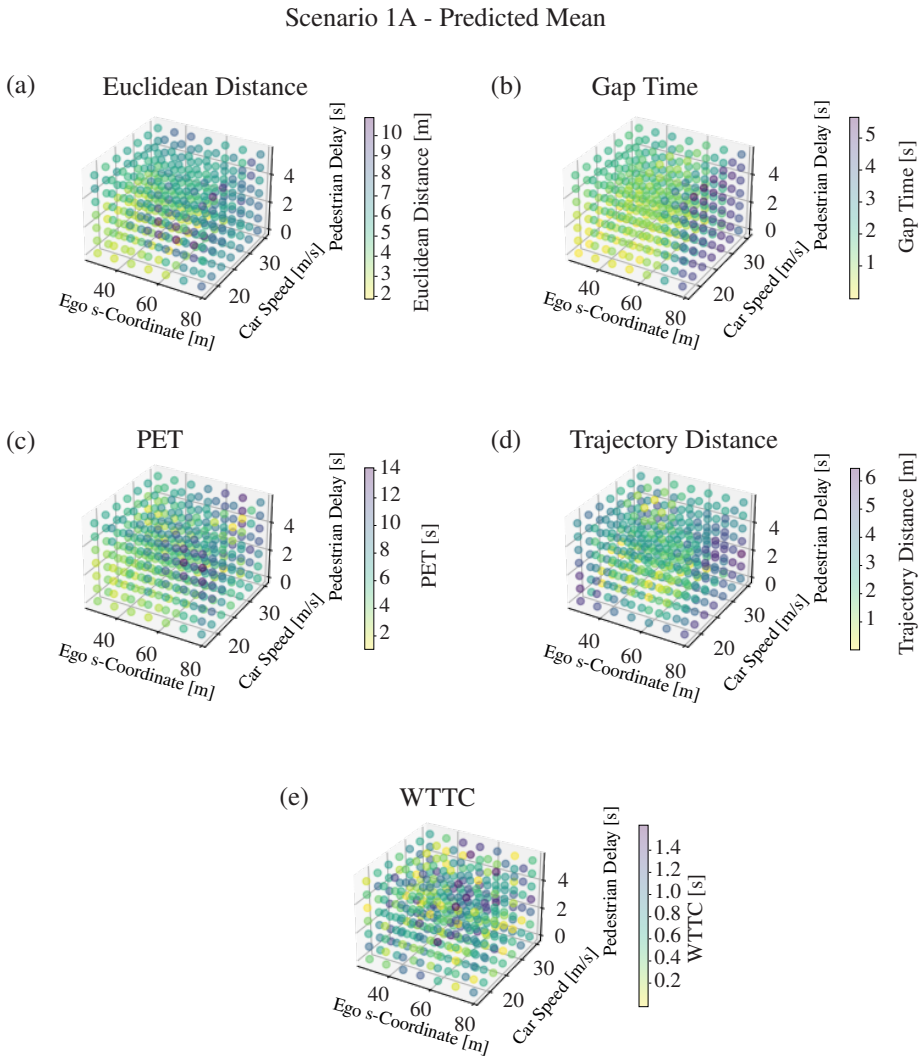


Figure 5.18: Scenario 1A: predicted mean as grids for Euclidean distance, gap time, post-encroachment time, trajectory distance, and WTTC.

- The optimization algorithm BO tends to select more concrete scenarios around critical areas. As a result, the standard simulated values show the same critical areas. However, as Table 5.2 and Table 5.3 show, the maximum values can deviate.

However, it is debatable whether all critical scenarios were found since the optimization only handles probabilities. Despite the probability indicating otherwise, statistically, critical regions

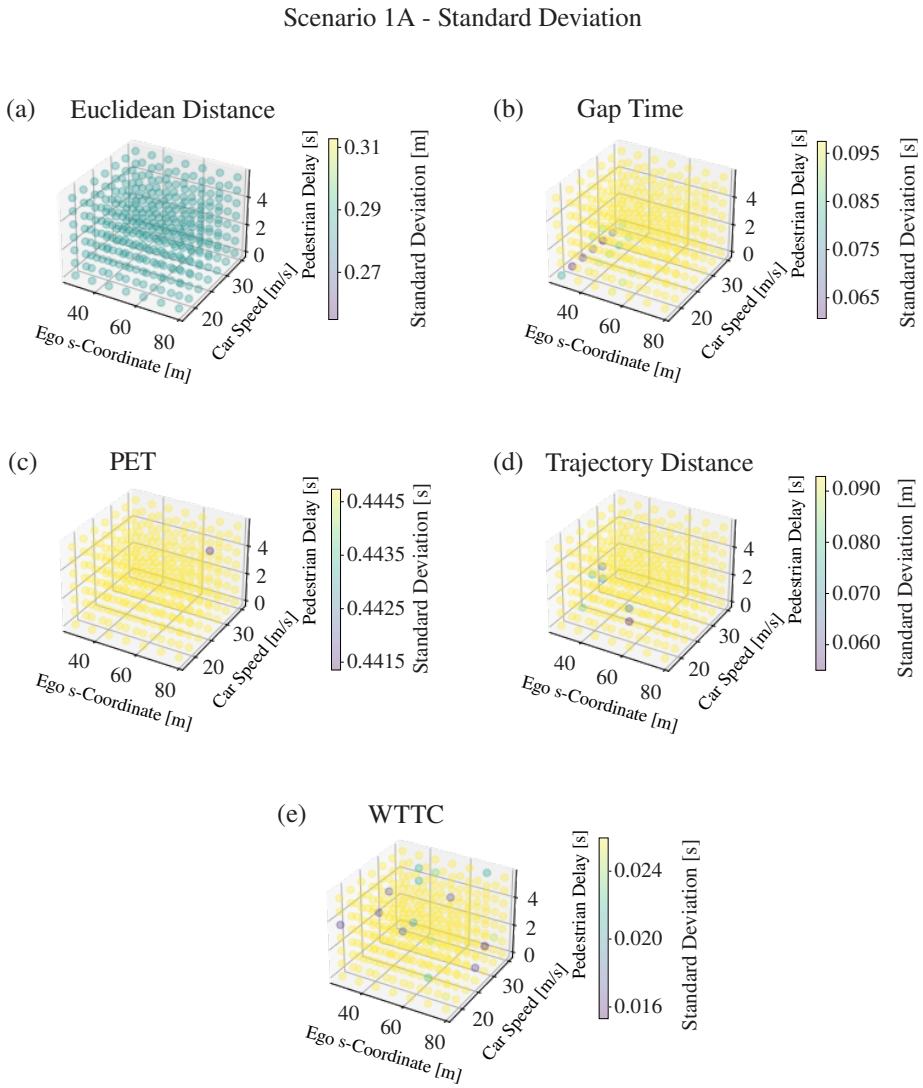


Figure 5.19: Scenario 1A: standard deviation as grids for Euclidean distance, gap time, post-encroachment time, trajectory distance, and WTTC.

may still be overlooked. This issue, however, is a result of the optimization algorithm and not the criticality metric, and it affects all used metrics. It should be noted that the critical situations found are only critical with respect to the metric used. A human expert may identify other critical situations.

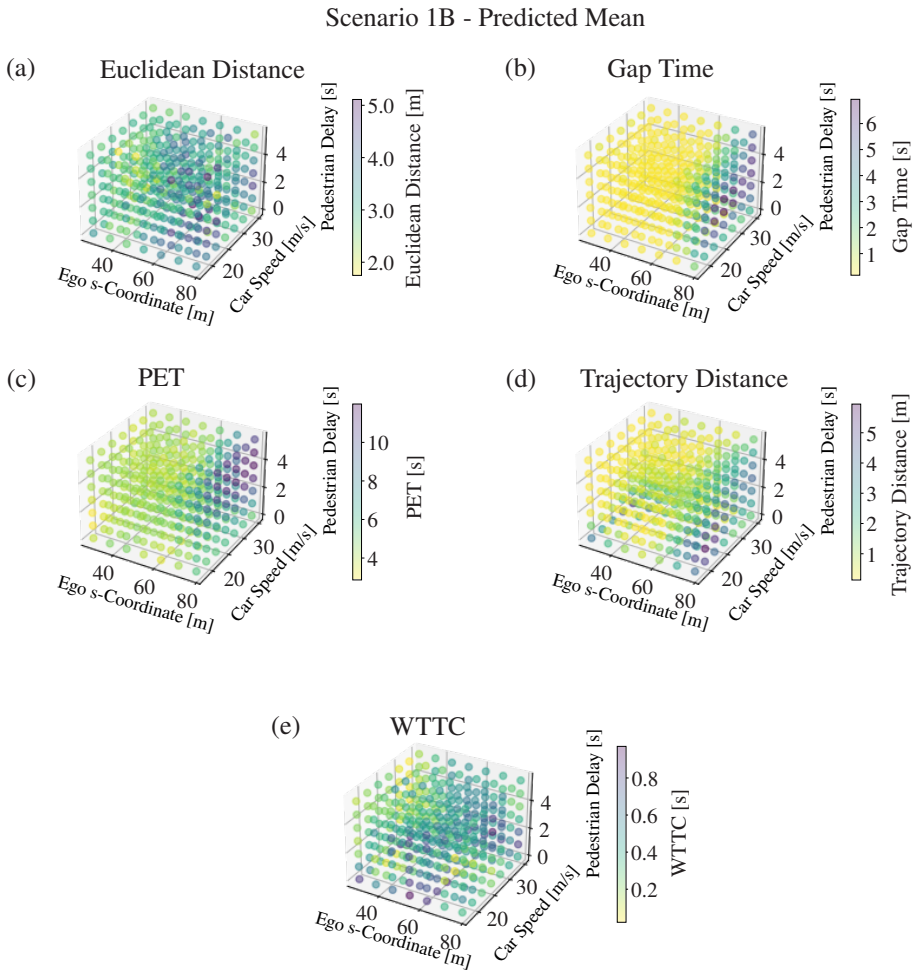


Figure 5.20: Scenario 1B: predicted mean as grids for Euclidean distance, gap time, post-encroachment time, trajectory distance, and WTTC.

5.4 Categorization within the Quality Taxonomy

In addition to scenario exploration, this work focuses on different quality categories. As established in Section 4.3, the assessment of scenario quality can be classified into three distinct levels of resolution. Moreover, criticality metrics can be interpreted as measures of quality across different domains, such as the evaluation of the System Under Test (SUT), the sensor model, or the quality of scenarios themselves. These metrics indicate the extent to which specific regions within a parameter range require testing, given the employed driving function.

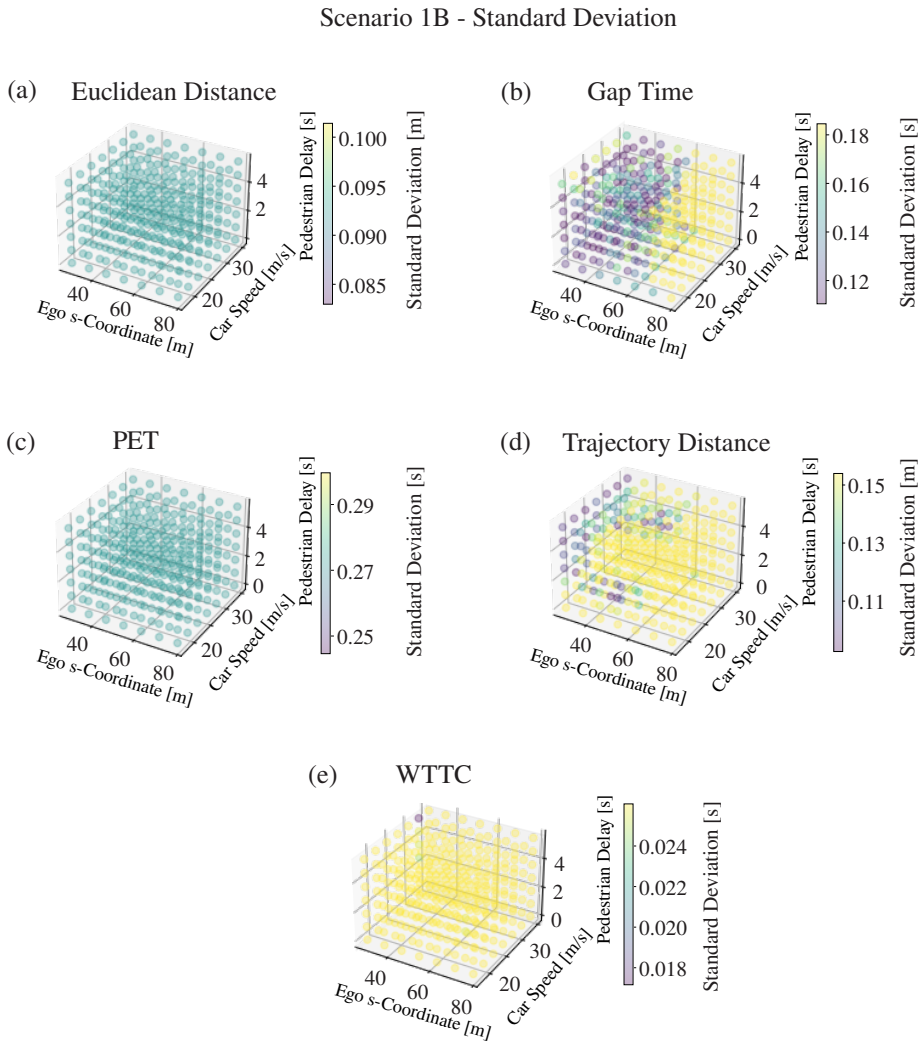


Figure 5.21: Scenario 1B: standard deviation as grids for Euclidean distance, gap time, post-encroachment time, trajectory distance, and WTTC.

This work uses conventional criticality metrics, such as WTTC and trajectory distance, to evaluate a single time step during a simulation, thereby characterizing the nanoscopic quality of scenarios. Additionally, a comprehensive metric, i.e., TDP, is introduced to address the challenge of determining which aspects of a scenario require evaluation. This is important because most metrics focus primarily on the ego vehicle and a single additional actor. In multi-actor scenarios, such as scenarios 1A and 1B, critical scenarios may be overlooked or metrics may need to be computed

for every possible combination of ego and other actors. Depending on the simulation's goal, it may even be necessary to perform multiple optimizations depending on the observed actor. By using the TDP metric, the evaluation includes the ego vehicle's perspective of a scene regardless of the number and position of other actors. This provides a precise measure of scenario quality. Therefore,TDP allows the user to obtain a more comprehensive scenario quality metric.

Subsequently, the assessment of microscopic quality is conducted in a secondary phase by aggregating the nanoscopic outcomes mentioned earlier. This process involves summarizing the results, such as identifying the maximum or minimum measured value obtained during the simulation of a concrete scenario or using microscopic metrics like PET. Most previous plots use microscopic scenario metrics to indicate the criticality of a specific scenario within the parameter space. This evaluation includes elements such as optimal and suboptimal standard deviations, as detailed in Section 5.3.3 3.2, and the determination of confidence intervals for a given logical scenario. See Figure 5.16 for an example. These macroscopic properties allow conclusions to be drawn about the adequacy of simulations and the need for further investigation.

5.5 Advantages of Traffic Density Potential

The main goal of TDP is to overcome the problem that most metrics (e.g., PET, TTC, gap time) are measured between two distinct actors, e.g., ego vehicle and one further actor (see Section 4.8 and appendix B.2). However, the approach of only observing two actors can introduce several problems:

- If a scenario involves more than two actors, it must be determined whether the metric should be calculated between each pair of actors or at least between the ego and every other actor. Additionally, it must be determined how the different measured values are weighted and whether critical values carry more weight than non-critical values.
- Critical situations may be overlooked if only two actors are observed among a group of actors.
- In datasets like the INTERACTION dataset Zhan et al. [161], where there is no defined ego vehicle and metrics are calculated for each actor pair, it is possible to overlook critical situations if only two actors are observed among a group of actors. To avoid this, it is important to consider all actors in the dataset. Additionally, it is important to note that the cost of calculating this metric grows exponentially. The calculation time can increase depending on the metric and its calculation method.

TDP can overcome these issues: The calculation is performed for the entire scene, regardless of the number of other vehicles present. This comprehensive approach eliminates the need to determine which actors should be included or how individual measurements should be combined into a final score. As a result, critical situations between the ego vehicle and other actors are less likely to be overlooked. In large datasets, the cost only grows linearly as long as no other metrics, such as distance, are used for the penalty in the final score (see Section 4.8.5).

If one of the actors close to the ego vehicle is a pedestrian, it is important to note that they cannot raise the criticality value as high as other cars due to their comparatively low velocity. In this situation, the criticality should be measured from the point of view of the pedestrian. Alternatively, pedestrians can be given a higher weight than other cars. However, the dataset used in Appendix B.2 did not offer pedestrian data; therefore, it was impossible to determine such a weight.

6 Conclusion

6.1 Summary

This work focuses on exploring and evaluating scenarios and the derived findings. The initial chapters provide as a foundational backdrop, with Section 2.1 establishing scenario-based testing as the basis for the following work. It acquaints readers with metric standards and techniques for acquiring scenarios in Section 2.2 and Section 2.3. The conceptual framework, as explained in Sections 4.4 and 4.7, presents a structured approach that includes the sequential stages of **Exploration**, **Clustering**, **Reduction**, and **Analysis**. Section 4.3 serves as an insertion to introduce a quality taxonomy for further used metrics and a comprehensive metric called Traffic Density Potential (TDP) to assess the quality of a scene from the point of view of an ego vehicle in Section 4.8. The mentioned concept is then further explained and evaluated with concrete experiments in two different logical scenarios in Chapter 5.

6.2 Research Questions and Requirements

Section 1.1.3 introduced three challenges, which were then refined into research questions in Section 1.2. Further, in Section 4.2 four requirements were formulated regarding the research questions.

The first set of research questions addresses scenario evaluation:
RQ1.1: What is the definition of scenario quality or criticality, and how do they differ and resemble known criticality metrics?
RQ1.2: Are different levels of abstraction of criticality needed during the development and testing process?
RQ1.3: What metrics are appropriate for providing a comprehensive assessment of the criticality of a scene or scenario?

The first two derived requirement belongs to this block of research questions:
REQ.1: The used criticality metrics shall be capable of identifying critical situations regardless

of the actor constellation. Metrics that are only suitable for car following scenarios are unsuitable for scenarios with more than two actors.
REQ.2: The scenario exploration tool shall find all critical scenarios within the scenario space using metrics that meet REQ.1

The classification system introduced in Chapter 4.3 systematically defines scenario qualities across different levels of simulation resolution. It emphasizes the importance of scenario evaluation throughout the development and testing phases, offering insights into the System Under Test (SUT) quality and delineating the appropriate focus of evaluation, e.g., the SUT or where in the scenario parameter space more simulations are needed (RQ1.1). In this context, the different abstraction levels for different types of quality help to differentiate on what level a criticality or quality statement is made, e.g., a complete logical scenario cannot be evaluated by the criticality metrics measured at one single time step or from one concrete scenario. Clarifying the level of abstraction, resolution, and domain of interest at which a statement is made, significantly contributes to the comprehension of the examined functionalities (RQ1.2). In addition to the taxonomy, a comprehensive metric, namely TDP, is proposed.

The use of the novel TDP metric as a scenario quality metric enhances the evaluation approach by providing an ego-scene-centric perspective that exceeds traditional 2-actors-centric metrics. A vehicle within a group of other actors will exhibit lower quality, indicating a more dangerous situation, than a vehicle outside this group or in an unoccupied part of the map. As exhibited in Figure 5.8(g) when compared to (a)-(f), scenes with more than one additional actor to the ego vehicle can have other criticality patterns. This is due to the fact that a traditional criticality metric can only assess the criticality between the ego and a second actor. Therefore, relying solely on conventional metrics may result in overlooking critical situations involving other actors in scenarios 1A and B, where critical situations between the ego vehicle and the truck or the second car may go unnoticed. However, when only two actors are involved, as shown in Figure 5.15, TDP can identify the same patterns as metrics that focus solely on the two actors.

Based on the results, it can be concluded that the TDP meets REQ.1. Furthermore, experiments conducted in scenario 1 (see Section 5.3.1) demonstrate that the TDP can identify critical areas beyond the metrics that are limited to two actors and constellations such as car following or intersections (Section 5.5). In addition, the TDP can identify the same critical situations as conventional metrics in one-on-one situations, as shown by experiments conducted in scenario 2, thereby fulfilling REQ.2.

The second set of research questions focuses on the exploration of a logical scenario and its result evaluation:
RQ2.1 Which algorithms can be found and used for a fast scenario exploration to find

critical parameter sets for logical scenarios?
RQ2.2 Which influence does the criticality metric have on the results of the scenario exploration and evaluation?
RQ2.3 How can further analysis help reduce and summarize the set of found concrete scenarios?

Following these research questions two further requirements were found:
REQ.3: The logical scenario shall be represented by a reduced set of representative concrete scenarios.
REQ.4: The logical scenario space shall be analyzed regarding its credibility and uncertainties.

Any optimization algorithm can be used for scenario exploration, such as BO [7] or evolutionary learning [9]. This work demonstrates that the chosen BO approach, employing GP, effectively serves as a tool for rapid scenario exploration. The surrogate models generated not only facilitate efficient optimization but also enable the analysis of uncertainty, allowing for the formulation of empirical statements regarding confidence intervals and standard deviations (RQ2.1).

As shown in experiments in Section 5.3.1 and Section 5.3.2, the main results using alternative criticality metrics do not differ substantially when identifying critical regions in a scenario parameter space. However, the surrogate models generated from these metrics show different performances in prediction and probability evaluation, depending on the scenario parameterization and number of simulated scenarios (RQ2.2). Furthermore, the data obtained during the optimization process can be used to cluster and further reduce the identified set of scenarios. The use of principal hull analysis highlights the potential for scenario reduction, which contributes to the synthesis of a representative set of critical scenarios (RQ2.3).

The scenario clustering and the reduction to a representative set of concrete scenarios were shown in the experiment in Section 5.3.1 and Section 5.3.2. To enable behavior- and criticality-based clustering, projection into the kernel space via kernel PCA was utilized. DBSCAN was employed as a clustering approach since it does not require knowledge of the number of clusters. In scenario 1A, the reduction is feasible, and REQ.3 is satisfied because it resulted in a substantial reduction in specific scenarios. However, it is important to note that the reduction is only possible for logical scenarios with convex clusters. This indicates that REQ.3 is only partially satisfied, and additional reduction techniques for concave clusters must be identified.

The credibility analysis is possible thanks to the surrogate models resulting from the selected optimization algorithm. The surrogate model can predict criticality for the entire range of scenario parameters. Therefore, it can be used to make empirical statements about the confidence interval and standard deviation. Therefore, BO with GP satisfies REQ.4, unlike other methods such as EA.

6.2.1 Place Among Current Approaches of Scenario-based Testing

In Section 2.1.2, the process of scenario derivation according to EU Regulation 2019/2144 was shown. This work is located in the scenario derivation part. It is assumed that logical scenarios are primarily used in knowledge-based approaches, such as ODD or safety analysis. However, it is also possible to analyze and derive natural driving data sets in the real world, as demonstrated by [91, 84]. For further steps, there are two possible options:

- the reduced scenario set can be used for nominal and critical scenario tests, or
- the exploration and evaluation can serve as a type of test itself among other testing approaches.

Additionally, the outlined method can be used during the design phase to identify potential issues and opportunities related to the intended functionality of the system or function being considered. This proactive approach allows for early identification of challenges and opportunities in development, facilitating informed decision-making and effective mitigation strategies. Early identification not only facilitates well-informed decisions but also enables the implementation of effective mitigation strategies, thereby enhancing the overall development process and final outcomes.

6.2.2 Limitations

The utilization of BO with GP introduces certain limitations in anticipating the capabilities of the surrogate model. While this approach offers the advantage of predictive modeling and uncertainty estimation, it is essential to recognize that the underlying training data inherently constrains the surrogate model's predictions. Consequently, the surrogate model's ability to anticipate outcomes depends on the representativeness and diversity of the training data. One limitation arises from the potential bias introduced by the training data distribution. If the training data predominantly encompasses specific regions of the parameter space, such as critical areas, the surrogate model may struggle to extrapolate predictions beyond those regions accurately. This limitation can lead to overconfidence or inaccuracies when predicting outcomes in unexplored areas.

Additionally, the GP surrogate model relies on the assumption of smoothness in the modeled underlying function. Consequently, when confronted with functions exhibiting non-smooth behavior or sharp discontinuities, the surrogate model might struggle to capture and predict these abrupt changes accurately. This can be observed in the experiments with scenario 2 and TDP.

Figure 5.16 shows a high gradient between critical and non-critical areas, whereas the prediction in Figure 5.17 has much smoother transitions between both regions. As mentioned, the principal hull analysis has the limitation, that it is only possible for convex clusters. As stated before, other approaches need to be found. Other possible limitations that could not be observed during the experiments are that the projection into the kernel space and, thus, the clustering might only work for some possible logical scenarios. This should be kept in mind when further experiments are done. One issue with BO libraries is their limitation in handling high dimensions. During testing, all libraries struggled to handle more than three dimensions, primarily due to memory constraints. However, this is a limitation of the implementation and not of the method itself.

6.3 Outlook

Many methodologies employed in this study offer opportunities for substitution with alternative algorithms and techniques. For example, scenario exploration (cf. Section 3.2.2) can be accomplished through several different approaches. Neural networks, reinforcement learning, linear regression, or other optimization methods can be utilized instead of BO. For concrete scenarios, the DTW (cf. Section 3.2.4) similarity measure could be substituted with alternative metrics, such as the Hausdorff distance or the Frechet distance proposed by Braun et al. [3]. Furthermore, other clustering approaches (cf. Section 3.2.5), such as OPTICS or hierarchical clustering (Chapter 3), could replace the DBSCAN clustering algorithm, as the objectives and circumstances of scenario exploration and clustering may differ from those in this work. Further experiments and scenario exploration with more than three parameter ranges can be performed. To support this, a possible approach is to replace the simulation with surrogate models, as demonstrated by Abdessalem et al. [14, 29] in their work on maximizing the criticality between an ego vehicle and a pedestrian crossing the street for reinforcement learning. This technique can also be applied to identify concrete scenarios and reduce time-consuming simulations, e.g., dimensionality higher than three.

The presented work on scenario result evaluation opens avenues for future research that is closely related to the coverage analysis of logical scenarios. Among other things, scenario coverage addresses the question of whether concrete scenarios are suitable for representing logical scenarios, which is a crucial component of scenario exploration. OpenSCENARIO DSL [21] and Foretellix [60] proposed to measure scenario coverage by the grid search approach [106] and defines concrete scenarios stepwise for a given parameter range. Hauer et al. [72] proposed an alternative approach that compares the problem of *Have all scenario types been tested?* with the Coupon Collector's Problem (CCP), a model from probability theory and related to the urn problem. There are

given sets of coupons, baseball cards, or similar items, and a collector has to draw each type or individual with a constant probability. From this, the question

(a) If there are N cards, how many cards are needed to draw (with replacement) to have at least one of each?

(b) What is the probability of having a complete collection if S cards were drawn?

There are two variations of the classical CCP: (1) the probability differs for some types (e.g., McDonald's Monopoly); (2) the number of types or individual is not known a priori (a butterfly collector comes to an unknown island and does not know who many butterfly types there are). Unfortunately, there is no analytical solution for (b) [72], and Monte Carlo methods must be used. For (2), there are several approaches: computing confidence intervals for the unknown number of coupons [59, 48]. The clustering of concrete scenarios within the logical scenario could be used to determine the different scenarios and the surrogate model to define the certainty of the results. The use of clustering in scenario space raises the question of whether it can be used to make assumptions about criticality. Archetypes and prototypes may then be used to represent clusters and aid in coverage. For instance, a cluster of non-critical concrete scenarios may be represented by only a few scenarios, as per ISO/PAS 21488, since they are all known to be non-hazardous. Another important step is to make statistical statements about the certainty of the scenario results and the coverage that still needs to be explored. Another question is how many iterations of optimization are necessary to achieve reliable results and a high degree of certainty.

In conclusion, the use of scenario exploration techniques in scenario space not only evaluates criticality but also provides potential pathways for efficient representation through archetypes and prototypes. This dissertation contributes to the advancement of scenario-based testing by developing and applying systematic scenario exploration, clustering, reduction, and analysis. Moving forward, it is essential to conduct statistical analyses to quantify the certainty of scenario outcomes and to determine the extent of coverage. The evaluation of scenario quality and the methodologies employed in scenario exploration enhance the understanding and optimization of critical scenarios, thereby contributing to the broader landscape of software testing and validation.

A State of the Art

A.1 Scenario Acquisition

A.1.1 Scenario Generation - Examples

Paranjape et al. [118] have presented a novel approach for generating maps and agents (vehicles and pedestrians) in real-time, which is organized into layer one and layer four, respectively. The agents are controlled using behavior trees, and the modular behavior concept enables the generation of a broader range of scenarios for testing throughout the simulation. The procedural nature of this approach allows for a more flexible and customizable scenario generation and allows the generation of a broader range of scenarios for testing throughout the simulation.

A.1.1.1 Data-driven Scenario Generation - Examples

Goss et al. [67] propose a modular scenario framework that utilizes simple logical scenarios extracted from accident data, which are stored as atomic blocks/scenarios. These atomic blocks can be recombined in the second step to obtain new, more complex scenarios that differ from the existing concrete scenarios.

A.1.2 Scenario Alteration - Examples

Abeysirigoonawardena et al. [15] trained an automated driving function by reinforcement learning to avoid hitting pedestrians. In this work, they used Bayes Optimization and Euclidean distance to generate training situations where pedestrians' trajectories led them in front of the ego vehicle to get new training scenarios. They enhanced their distance evaluation by a term to punish parallel movement to avoid situations with a small Euclidean distance where the ego vehicle and the pedestrian have parallel trajectories. In this work, the term scenario is used for pedestrians' trajectories and not in the way scenario was previously defined in Section 2.1.3. Additionally, the state of the driving function changes throughout experiments since the driving function's

performance increases. In this work no classical scenarios for testing were generated, and the experiments and training of the driving function, the simulator CARLA was used.

Nonnengart et al. [115] suggested a formal method-based approach. Instead of algorithm-based, it is an information-based alteration method where the information about a scenario is considered. In their framework CriSGen, they used a composition of formalized abstract models for scenario description and maneuvers for virtual training of an ADS. The example scenario contains an ego vehicle approaching a pedestrian crossing the street. To get critical scenarios, they utilized uncritical and formally abstracted scenarios via reasoning on their non-linear arithmetic constraint formulas to alter the maneuvers towards a more dangerous situation for the pedestrian. They claim, their approach is complete and guarantee this approach does not overlook critical scenarios.

A.1.3 Scenario Exploration - Examples

In the PEGASUS project, Bussler et al. [34] used an evolutionary algorithm to find relevant parameter sets for the verification and validation of automated driving systems. They utilized a parameterized logical scenario of an ego vehicle performing a left turn, crossing the trajectory of a pedestrian and another vehicle at an intersection in Hamburg. The evolutionary algorithm explored an abstract parameter space and selected, mutated, and recombined individuals (i.e., concrete scenarios) from a starting pool to obtain a new generation of individuals. Fitness functions based on Euclidean distance and time-to-collision were used to evaluate the scenarios suggested by the learning framework, and these functions were referred to as criticality metrics because they quantitatively assessed the quality of the system under test (SUT), i.e., the ego vehicle. A higher fitness score indicated a higher criticality of the individual (scenario) with respect to the criticality metrics. The paper did not provide further information about the simulation framework or scenario description language used.

Baumann et al. [27] proposed a related approach, where a reinforcement learning algorithm explored the parameter space of a logical scenario to find concrete scenarios. Their example scenario involved an ego vehicle attempting to overtake a leading vehicle in oncoming traffic. After analyzing the factors influencing the driving system of the ego vehicle, test scenarios were generated and executed in a simulation, and the simulation results were evaluated using criticality metrics that included headway, time-to-collision, and required longitudinal acceleration (a_x). The evaluation was used as a reward function for generating new test cases, where the higher the criticality, the higher the reward given to the generation. The ϵ-greedy algorithm was used to prevent local maxima by randomly selecting parameter sets with low probability and parameter sets with the highest expected reward with high probability. The critical scenarios were stored as a set of critical test cases, and the simulator used in the experiments was CarMaker.

Abdessalem et al. [14] proposed an evolutionary scenario exploration method where classification decision trees were used to guide the search for new test scenarios. Their training scenario involved a pedestrian crossing the street in front of an approaching ego vehicle. Two criticality metrics were used: the first metric combined the Euclidean distance between the ego vehicle and the pedestrian with the field of view of the ego vehicle, and the second metric calculated the speed of the ego vehicle at the time of collision and returned -1 if no collision occurred. The guidance algorithm accelerated the search for critical scenarios using an evolutionary algorithm to 78% of a baseline evolutionary algorithm.

A.1.4 Scenario Extraction Naturalistic Driving Data - Examples

A.1.4.1 Extraction from Naturalistic Driving Data

Zofka et al. [163] used recorded sensor data with certain traffic participant maneuvers, such as lane changes, to create new traffic participant trajectories for more critical scenarios. The scenarios are derived and parameterized from traffic scenarios out of recorded sensor data. In the next step, the actors' trajectories can be modified to get more critical scenarios for testing, which makes it a combination of scenario extraction and alteration. The scenarios found can then be simulated by simulation tools, such as IPG CarMaker.

Furthermore, concrete scenarios in urban environments can be extracted from datasets and aggregated to logical scenarios based on maneuver recognition [84]. In the first step, the actors are categorized regarding their role within a scene, i.e., they interact with the ego vehicle or pass by. This leads to three different types of interaction: merging, diverging, and crossing. After this, a maneuver recognition is used to classify and extract maneuvers from the dataset. Before the extracted concrete scenarios can be aggregated to logical scenarios, they have to be identified. It is possible that a single driving scene contains more than one scenario, and all of them are extracted and identified. If maneuver sequences are the same, they are categorized as derived from the same logical scenario. The resulting logical scenario sets consist of a maneuver sequence, the assigned concrete scenarios, and parameter ranges and distributions. Another related approach is proposed by Langner et al. [91], where concrete scenarios were extracted from real-world-driving data and then, in a second step, clustered into sets of logical scenarios.

Most of the scenario extraction methods available in the literature are designed to tackle highway use cases. However, to address the scarcity of scenario datasets in urban intersections, Weber et al. [155] proposed an unsupervised machine learning pipeline that can extract concrete scenarios and cluster them into logical scenarios from an urban intersection. The proposed pipeline consists

of three main steps. In the first step, the naturalistic dataset is filtered to reduce the noise, and a post-encroachment time threshold is applied when the actors are close enough to each other in a spatial sense to ensure a small temporal distance for all possible trajectory combinations. In the second step, a principal component analysis is conducted to reduce the feature set for further processing. In the third and final step, a grid-based model of the ego environment is used for feature extraction, and the clustering is performed. The proposed method results in a set of logical scenarios that can be used for further testing and development of autonomous driving systems.

A.1.4.2 Scenario Extraction from Crash Data - Examples

CSG, a toolkit proposed by Xinxin et al. [158], extracts scenarios from real traffic accident videos for re-simulation using simulation software such as Carla and OpenDRIVE. They use Deep Neural Networks for scenario extraction and evaluate each scenario based on a weighted sum of safety indicators, such as deceleration rate to avoid collisions, TTC, and time-integrated TTC. In re-simulation, where a system under test is used instead of the original ego trajectory, other actors can adjust their velocity and trajectory to make a scenario more critical.

Gambi et al. [63] present an unusual approach using naturalistic language processing and related ontologies to automatically extract scenarios from police reports for simulation. Their method assumes that real accidents and crashes contain the most critical scenarios.

Esenturk et al. [53] propose an approach for crash report analysis, where they analyze the pre-crash conditions and their relation to accident severity. They use this information to model a modified logistic regression model to extract useful data, which is then used to generate high-risk scenarios using the same logistic regression model.

A.1.5 Table with all Mentioned Examples of Scenario Acquisition

Table A.1: Overview of mentioned scenario acquisition methods.

Category	Sub-category	Examples	Main Features
Generation	From Scratch	[93], [125], [118]	• More than parameter values are changed • New concrete scenarios from different logical scenarios
	Data-driven	[67], [58], [65] , [139], [140]	
Alteration	Algorithm-based	[15], [152], [81], [64], [17], [88]	• Concrete starting point • No min or max values necessary • Typically changes in trajectory or velocity
	Information-based	[115]	
Exploration	Optimization-based	[34], [14], [47], [27], [142]	• Range as starting point • Optimization problem
	Analytical	[76]	
Extraction	Naturalistic data	[67], [32], [84], [71], [163], [155], [91], [90]	• Based on naturalistic data • Scenario is subsection of data
	Crash data	[158], [63], [36]	
Expert		[44], [6], [49], [50], [78], [117], [107], [21]	• Human designed scenario
Aggregation		[84], [155], [91]	• Summary of concrete scenarios to form one logical
Combinations		[67], [32], [84], [163], [155], [91], [33]	• Combination of categories to get end result

B Scenario Quality and Criticality

B.1 Taxonomy for Quality in Simulation

B.1.1 Simulation Environment

Simulation environment quality refers to the quality a simulation environment and its components and modules have or are presumed to have. The simulation models, their couplings, or the complete simulation infrastructure are the elements from which the qualities are quantified. Before a simulation model or environment can be used, it must be ensured that it approximates its real-world equivalent or functionality sufficiently in all relevant aspects.

B.1.2 System Under Test Quality

The quality of a system under test evaluates the observable behavior and performance of a system under test with respect to the desired or intended functionality according to predefined requirements. Figure 4.4 shows a selection of different types of SUT quality. One possible subclass is safety quality, which evaluates how safely a system under test can handle a specified situation. Safety can be functional safety according to ISO 26262 [79], where it is defined as “absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems”.

Another approach is to assess system under test behavior in critical scenarios, where critical situations are derived from a prior criticality analysis [109]. A common example is a near-collision situation evaluated with metrics like the time-to-collision (TTC) metric [73]. Additionally, in the early stages of the development process, where proof of concepts and ideas are tested, an automated driving system’s behavioral performance might play a more significant role for developers than safety. It is important to note that different quality metrics can contradict each other: improving comfortable braking might also lead more likely to crashes in near-collision situations.

B.1.3 Examples for Quality Types

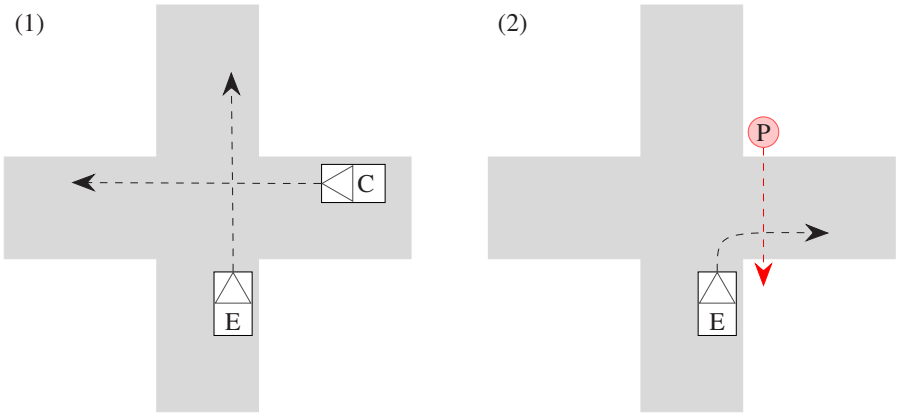


Figure B.1: Two example scenarios: (1): Ego vehicle (E) and adversary (C) crossing the intersection; (2):Ego vehicle (E) takes a right turn, while pedestrian (P) crosses the intersection

The following examples take place at an urban intersection without traffic lights or signs and are depicted in Figure B.1. In the first scenario (1), both the ego vehicle (E) and the adversary car (C) attempt to cross the intersection without taking a turn. A synchronization point is placed in the middle of the intersection to ensure that both vehicles arrive at the same time. In the second scenario (2), the ego vehicle turns right and intersects the path of a pedestrian crossing the road. Some of the following quality examples relate to the events that occur during the scenario, such as the criticality between the ego vehicle and the pedestrian, while others evaluate only a specific part of the scenario, such as a simulation model, in terms of quality. Certain parameters were modified across different scenarios, including the maximum velocity v_{max} of the ego vehicle, the time it takes for the pedestrian to cross the road t_{cross}, and the minimum distance d_{start} between the ego vehicle and pedestrian that triggers the pedestrian's movement.

B.1.3.1 Simulation Environment Quality

The reliability of three different simulation setup outputs was tested as an example of simulation quality. The execution results from ten executions of the same scenario repeated with one of the simulation setups were compared to make a statement about the macroscopic simulation quality. These comparisons were based on an exemplary requirement regarding the used simulation setup:
RQ1: *If a concrete scenario is executed twice or more within the same simulation environment, the deviation of traffic participants' trajectories measured with dynamic time warping (DTW)*

Table B.1: Simulation environment quality in literature

Level	Description	Publication
Nanoscopic	simulation model V&V methods survey	[123]
	steering system evaluation	[62]
	scenario-based model evaluation	[89]
Microscopic	accuracy, effort, efficiency	[69]
Macroscopic	simulation model V&V methods survey	[123]
	ISO 26262 Standard: tool qualification	[79]

shall not deviate further than 10.0 m *for a trajectory with 2500 data points. Since Carla 0.9.11 does not support deterministic simulation results, a small deviation can result in a difference of* 10.0 m *or more. The threshold* 10.0 m *was chosen to allow these minor deviations without allowing new trajectories for the ego vehicle.*

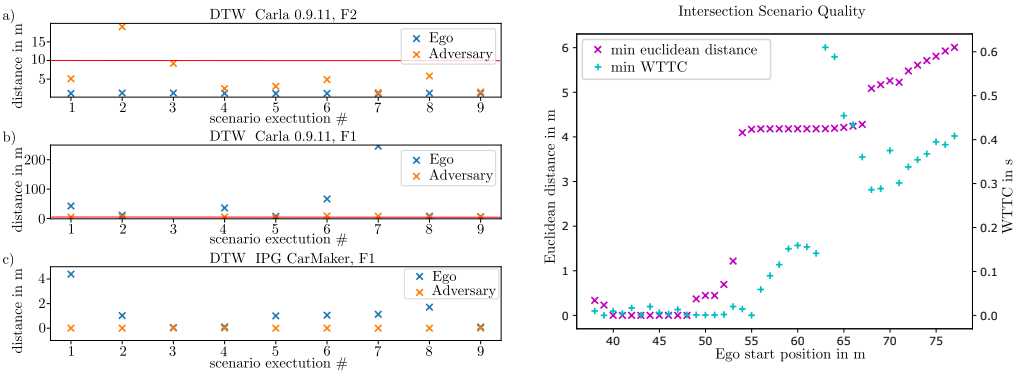


Figure B.2: a), b), c) on the left show examples of nanoscopic simulation environment quality, and the right side depicts an example of microscopic scenario quality by using euclidean distance and worst-time-to-collision (WTTC).

Two simulation environments were used: Carla 0.9.11 and CarMaker 8.0.2 from IPG combined with two driving functions, F1 and F2, respectively. F1 is an external driving function connected via the open simulation interface (OSI) [20] to both simulation environments, and F2 is the BasicAgent from Carla's PythonAPI [49]. The same intersection scenario, i.e., scenario (1), was executed ten times for each setup, resulting in thirty scenario executions, where the first execution was used as a reference point for the following results.

The trajectories of the ego vehicle and an adversary vehicle were compared with the reference scenario's trajectories via DTW, a metric used to calculate the distance between two time series [124] as defined in RQ1. Ideally, the distance should be 0.0 m or close to 0.0 m for nearly identical

simulation results, indicating deterministic results. We used a threshold of 10.0 m, indicated by the red lines on the left-hand side of Figure B.2 a) and b). In c), no threshold is shown since all values lie beyond this threshold. If a simulation consists of 2000 steps, the average trajectory deviation per step should be smaller than 0.005 m.

Among the actors involved in the simulated intersection scenarios, the adversary vehicle from CarMaker is the only one with completely deterministic behavior. This is demonstrated in Figure B.2 c), where the CarMaker software's trajectory follower produces consistent results. The largest DTW distances are observed for F1 in Figure B.2 b), particularly in scenario 3 where the ego vehicle deviated significantly from the reference trajectory, resulting in a trajectory deviation of over 5000 for the ego vehicle and over 9000 for the adversary vehicle. Both values are not shown in Figure B.2 b) since the scale would make it even more difficult to read other results. In this scenario, the ego vehicle came off the course and drove a circle at the intersection, which results in a very different trajectory than the reference trajectory. Additionally, it influences the opponent's behavior, e.g., an adversary vehicle might need to wait longer until it can pass the intersection if it is occupied by the ego vehicle. This scenario highlights the need for high-quality microscopic simulation, since such deviations can affect the behavior of other actors in the simulation. The high DTW distances could be caused by coupling methods of low quality, resulting in low microscopic simulation quality, or a non-deterministic physics model, which is inappropriate when a deterministic model is needed and also states low nanoscopic quality.

Figure B.2 a) shows a DTW value around 1.0 for each F2 trajectory, which is reasonable given that Carla 0.9.11 uses the non-deterministic physics model from its Unreal engine. This suggests that although there may be minor differences between the trajectories, the overall results are close to each other. However, the adversary vehicle, which responds to the ego vehicle, shows higher DTW values, indicating greater deviations in trajectory. In contrast, Figure B.2 c) demonstrates that CarMaker's adversary vehicle is capable of producing deterministic behavior. Although F1 still produces non-deterministic results, all values are below the threshold of 10.0 m.

Based on these observations, further experiments are needed to evaluate the microscopic simulation environment quality, particularly with regard to the coupling of the driving function, which could be affected by issues with the PID controller or the message interface (OSI). Additionally, the quality of the system under test (i.e., driving function F1) needs to be evaluated across all three levels of resolution. Table B.1 lists some examples for the simulation environment quality evaluation.

Table B.2: System under test quality in literature

Level	Description	Publication
Nanoscopic	Time-to-collision	[73], [46]
	Gap Time	[16]
Microscopic	PET, ET	[16]
	min. Time-to-collision	[46]
Macroscopic	collision probability	[46]

B.1.3.2 System Under Test Quality

The system under test quality example measures the nanoscopic and microscopic quality of the scenario intersection (2) described above. For exemplary purposes, a simple requirement is being formulated.
RQ2: *The ego vehicle shall not cause an accident or hit pedestrians.*

For each time step during simulation, the nanoscopic metrics braking distance and braking time were calculated for the ego vehicle and distance and gap time (see Section 2.2.3) for the criticality between the pedestrian and the ego vehicle. All four are shown in Figure B.3 and measured for the intersection scenario (2) with $v_{\text{max}} = 32\,\text{km/h}$, $t_{\text{cross}} = 5\,\text{s}$, and $d_{\text{start}} = 16\,\text{m}$. Gap time describes the predicted distance in time between vehicle and pedestrian passing the intersection of their trajectories.

The fact that it is going towards 0 s means there is a near-collision situation or even a collision. Gap time cannot be measured anymore when one actor passes the intersection of both trajectories. Therefore, the green gap time graph stops after about 29.2 s when the application period of gap time has passed. The gaps in the graphs of braking time and distance show the system under test either stood still or tried to accelerate (where braking time and distance approach infinity) in between braking phases since the application period condition for both metrics is a negative acceleration. Additionally, threshold values can be defined for the considered quality metrics, e.g., Gap Time $> 2\,\text{s}$. In Figure B.3 a), the gap time and distance approach a value of 0.0 s and 0.0 m, respectively. Both metrics indicate that the ego vehicle gets dangerously close to the pedestrian. The ego vehicle is already standing at the time of the shortest distance since both metrics reach their minimum at different times. The ego vehicle fulfilled its goals in terms of not hitting the pedestrian. An example of performance quality is to evaluate comfortable braking behavior.

In the second step, the microscopic quality was evaluated for each executed scenario to compare the criticality throughout a set of scenarios. Figure B.3 b) and c) show the metric results of post-encroachment time and encroachment time (see Section 2.2.3), respectively. Each value

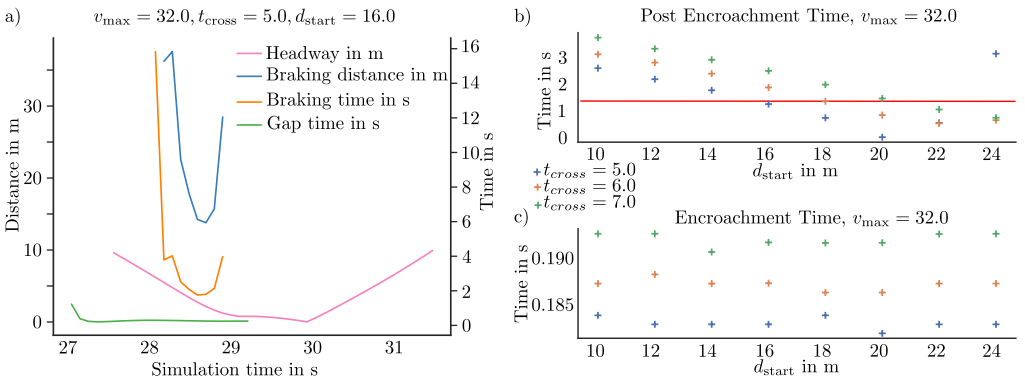


Figure B.3: a) Example for nanoscopic metrics. b) + c) Examples for microscopic metrics.

in the figures corresponds to a specific scenario derived from the logical scenario in the urban intersection example. According to Allen, Brian, L. et al. [16], post-encroachment time is the actual time gap between two traffic participants passing the intersection point or area of their trajectories. Encroachment time describes the time an actor occupies the intersection point or area and is exposed to a possible accident. For the evaluation of the microscopic quality, PET and ET (see Section 2.2.3) were used. As shown in Figure B.3 b), encroachment time increases slightly with the time the pedestrian needs to cross the street (t_{cross}), but, as expected, the ego vehicle's starting distance and speed have no impact as they are not related to the pedestrian's movement. The metric results can be evaluated with the example threshold values PET $> 1.5\,\text{s}$ [16] and ET $> 2.0\,\text{s}$. In Figure B.3 b) it is shown that the ego vehicle falls below the given threshold in several scenarios, which indicates critical scenarios. Table B.2 lists some examples for the system under test quality evaluation.

B.1.3.3 Scenario Quality

Scenario quality metrics can provide evidence about a scenario's criticality or relevance for a test series. The difference between the system under test quality and scenario quality is the focus of the evaluation. Metrics related to the system under test pertain to its performance, while scenario quality focuses on various properties of a scenario, such as weather conditions, light, criticality, actor behavior, and other aspects relevant to testing the system. However, system under test quality can also be part of a scenario, and quality criteria for scenarios can be a composition of different quality metrics or the same metrics taken from different traffic participants and evaluated together. The experiments in Section B.1.3.2 also determine the scenario quality from various points of view. The scenario quality, in particular the scenario criticality, can be used as a way measure to decide which scenarios might be interesting for further use or do not give any new insights.

Table B.3: Scenario quality in literature

Level	Description	Publication
Nanoscopic	TTC	[86]
	TDP	[70]
	WTTC	[149]
Microscopic	scenario uniqueness	[90]
	search-based techniques	[131] (min. distance), [86] (min. TTC)
Macroscopic	coverage	[72]

Section 4.4 uses scenario metrics on a nanoscopic and microscopic level to generate a set of scenarios. A macroscopic example is scenario and scenario parameter coverage, which is essential in overall test coverage. Scenario coverage can be viewed on different levels, e.g., a set of concrete scenarios sufficiently represent a logical scenario, or the scenario set or database sufficiently represents all situations within an operational design domain. Reducing mileage is one of the goals of scenario-based testing, so it is crucial to make a statement about the coverage. Table B.3 shows a short list with examples of scenario quality from literature.

B.2 Traffic Density Potential Evaluation

The best way to evaluate a new metric would be to compare it with a labeled (urban) traffic dataset where the ground truth of all participants is available regarding their current position, movement, and criticality situation. However, this approach comes with a set of problems. There is no available urban traffic dataset with labels regarding the objective criticality of single vehicle positions or scenes. Additionally, large datasets such as the INTERACTION dataset [161] only contain a handful of near-collision and collision situations.

The second problem is that apart from collisions and near-collision situations, it is hard to define an objective and universal set of rules for critical traffic situations. The desired criticality often depends on the question at hand or special features within recorded data, e.g., road maps, country-dependent traffic rules, or the driver. Therefore, the evaluation task is defined as follows:

Task: Critical scenes that can be used for resimulation within the INTERACTION dataset shall be found.

This task is used to derive the following rules regarding the labeling of critical scenes:

(i) A collision or near-collision occurred.

(ii) Situations have a high crash potential, i.e., minor alterations in velocity or trajectory can lead to collisions in resimulation (e.g., moving cars pass by with a distance of less than 0.5 meters).

The first part of this evaluation in Appendix B.2.1 compares the results of the TDP to a human expert labeled sub-set of the INTERACTION dataset. The labeled dataset consists of 29.569 scenes, of which 4.263 (14.42%) are labeled critical. The results of the TDP are then compared to other established criticality metrics.

B.2.1 Data-Centered Evaluation

Table B.4: Comparison of results for selected metrics, showing best entries in bold and worst underlined. Abbreviations are explained in Appendix B.2.1.

	Dist	ET	GT	PET	PTTC	TTC	WTTC	TQρ_1	TQρ_2	TQρ_3	TQ$_{co}$
TP	1162	3700	745	2374	975	605	**4029**	3083	2149	3530	3041
TN	22364	6212	21592	16261	**22366**	23626	7373	16627	21475	13130	13494
FP	2942	19094	3714	9045	2940	**1680**	17933	8679	3831	12176	11812
FN	3101	563	3518	1889	3288	3658	**234**	1180	2114	733	1222
ACC	0.796	0.335	0.755	0.630	0.789	**0.819**	0.386	0.667	0.799	0.563	0.559
MR	0.204	0.665	0.245	0.370	0.211	**0.181**	0.614	0.333	0.201	0.437	0.441
TPR	0.273	0.868	0.175	0.557	0.229	0.142	**0.945**	0.723	0.504	0.828	0.713
FPR	0.116	0.755	0.147	0.357	0.116	**0.066**	0.709	0.343	0.151	0.481	0.467
TNR	0.884	0.245	0.853	0.643	0.884	**0.934**	0.291	0.657	0.849	0.519	0.533
FNR	0.727	0.132	0.825	0.443	0.771	0.858	**0.055**	0.277	0.496	0.172	0.287
PRE	0.283	0.162	0.167	0.208	0.249	0.265	0.183	0.262	**0.359**	0.225	0.205
CoK	0.159	0.040	0.027	0.117	0.116	0.094	0.086	0.220	**0.302**	0.164	0.121
F1S	0.278	0.273	0.171	0.303	0.238	0.185	0.307	0.385	**0.419**	0.354	0.318
MCC	0.579	0.547	0.514	0.572	0.558	0.550	0.595	0.636	**0.654**	0.622	0.587

The performance of TDP is compared to that of other established metrics, such as TTC and potential time-to-collision (PTTC) [2] a TTC deviate, Euclidean distance and WTTC, or PET, GT, and ET (see Section 2.2.3 for all mentioned metrics). For each of these metrics, established thresholds were used, resulting in a binary classification of critical and non-critical situations: $1.5s$ for time-based metrics (except WTTC, where $0.47s$ was used), and $1.0m$ for Euclidean distance.

Table B.4 presents the results for various performance measures, including true positives (TP), true negatives (TN), false negatives (FN), false positives (FP), accuracy (ACC), miss-classification rate (MR), true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), precision (PRE), Cohen's kappa (CoK), F_1-score (F1S), and Matthews correlation coefficient (MCC). The best performance in each category is indicated by bold entries, while the worst are underlined. However, it should be noted that only 14.42% of the imbalanced dataset is labeled as critical. Therefore, some bold values may not necessarily indicate good performance of the criticality metric. For instance, WTTC simply classifies most parts of the data as critical.

TTC is another well-performing metric (five bold entries in Table B.4), but its results are due to the opposite problem: most of the data is classified as uncritical, and with more than 85% being truly uncritical, some performance measures (e.g., accuracy and misclassification rate) indicate good performance. Therefore, precision, Cohen's kappa, F_1-score, and Matthews correlation coefficient were considered, where values grow with a better performance. In all four performance metrics, the TDP metric has four bold entries in Table B.4 and is one of the best-performing metrics since, unlike TTC and WTTC, it has no underlined values showing the worst-performing entry for this metric.

Cohen's kappa is a statistical measure that assesses the level of agreement between two raters or classifiers, such as the ground truth and the criticality metric being evaluated in this case [153]. Its value ranges from -1 to 1, where 0 indicates that both raters agree by chance, and negative or positive values indicate less or more than chance agreement, respectively. Cohen's kappa is calculated by dividing the observed relative agreement (p_o or accuracy) by the hypothetical probability of agreement by chance with data labels randomly assigned (p_e), as follows: [153]:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{B.1}$$

According to Wang et al. [153], a value of 0.359, which is the best performance among all the metrics evaluated, is categorized as fair agreement. However, the TDP metric offers an acceptable trade-off between correctly classifying true critical scenarios as critical without including too many uncritical ones in the set of critical scenarios.

Another evaluation metric is the F_1-score [38], which is a value between 0 and 1 that indicates the quality of a binary classification result. It is calculated using the following formula [38]:

$$\frac{2}{recall^{-1} + precision^{-1}} = F_1 = \frac{2TP}{2TP + FP + FN}, \tag{B.2}$$

where TP represents true positives, FP false positives, and FN false negatives. However, the F_1-score has limitations in dealing with class imbalance.

To address this problem, Chicco and Jurman [38] proposed the Matthews correlation coefficient (MCC), which is calculated using the following formula:

$$MCC = \frac{(ab - cd)}{\sqrt{(a+c)(a+d)(b+c)(b+d)}}, \tag{B.3}$$

where $a = TP$ are the true positives, $b = TN$ true negatives, $c = FP$ false positives, and $d = FN$ false negatives. The normalized version of MCC ranges from 0 to 1, with 0 indicating the worst performance, 1 indicating the best, and 0.5 indicating a random assignment of classes. It should be noted that the Matthews correlation coefficient may decrease when only one class is recognized, as is the case with TTC or WTTC.

The main reason for the found false positive hits, especially for TDP with ρ_2, lies in situations where vehicles drive towards each other or next to each other with proximity at high velocity even though they are not in the same lane. The false negatives result from the problem that sometimes the metric is less sensitive to criticality than the labeled data, i.e., the metric recognizes a situation as critical a few timestamps later and categorizes them as uncritical earlier than the labeled data. This problem in sensitivity arises from the trade-off between too many false positives vs. false negatives, and it depends on the task at hand if the threshold has to be higher to get fewer false positives or lower to get fewer false negatives. Another way of changing the outcome is to use different penalty terms, where ρ_1 emphasizes the criticality close to the vehicle and neglects things that happen further away, ρ_3 which takes in more of its surrounding vehicles, and ρ_2 as a trade-off in between both options.

It has to be kept in mind, that each metric has its special situations to recognize, and it depends on the question regarding what criticality is measured and which one shall be used. The primary benefit of the TDP is its versatility, as it is not limited to specific vehicle constellations, like TTC (car-following) or PET (intersection). Hence, it can be applied to recorded data or simulations without requiring prior knowledge of the scenes or scenarios.

B.2.2 Typical Detected Situations and Examples

B.2.2.1 Typical Critical Situations

Upon examining the traffic in the INTERACTION dataset, typical situations have been detected where TDP indicates criticality. In general, stationary objects are considered less critical than moving objects since they are assumed to have less influence on the outcome of a situation than moving objects. Often, these situations exhibit *high-tension*, characterized by dense traffic with

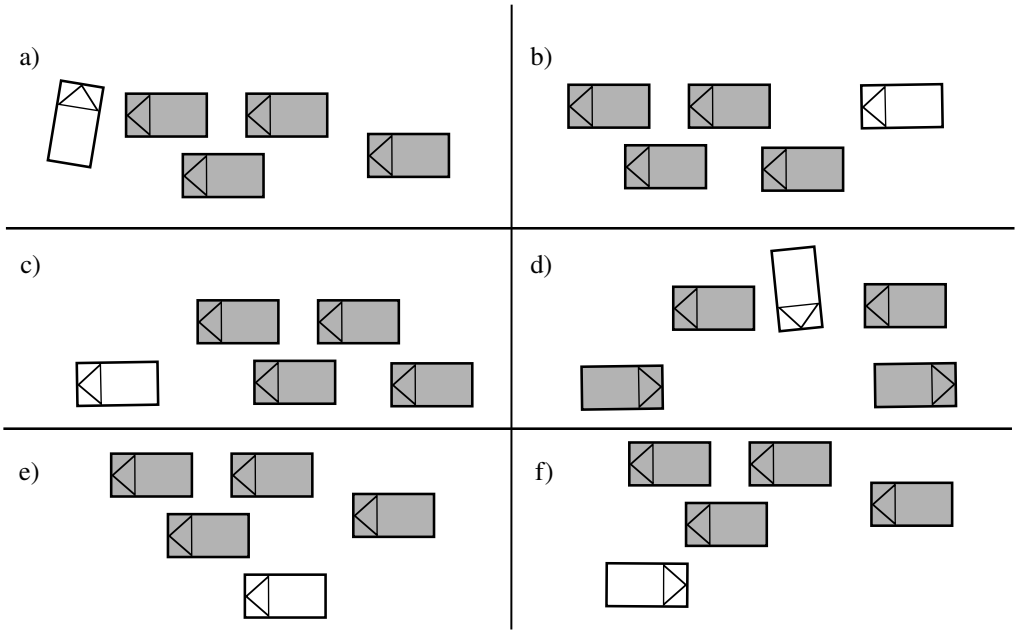


Figure B.4: Typical scenes classified as critical. The faster the ego vehicle (white) or the higher the difference in velocity among all traffic participants, the higher the criticality.

varying velocities of different traffic participants, as shown in Figure B.4. These *high-tension* situations typically include:

(i) The ego vehicle passing by or driving around one or more stationary or slow-moving objects:

- Other vehicles waiting to take a left or right turn (see Figure B.4 (a)).
- The ego vehicle approaching the end of already waiting vehicles (see Figure B.4 (b)) or accelerating at the start of a traffic jam (see Figure B.4 (c)).
- The ego vehicle moving around parked or slow cars (see Figure B.4 (d)).

(ii) Vehicles moving towards or next to each other in the same lane (see Figure B.4 (e)) or oncoming directions (see Figure B.4 (f)). This scenario also includes cases where both vehicles are on different lanes. Since no map data is used due to the simplified calculation, it is impossible to distinguish if both vehicles are in the same lane.

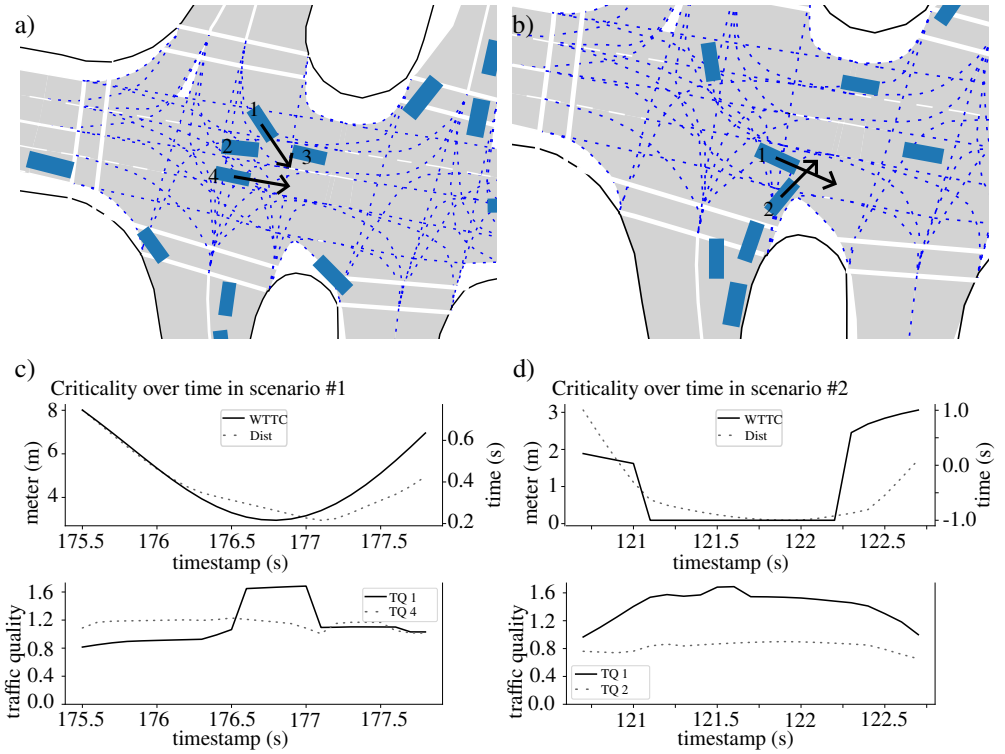


Figure B.5: (a) shows a snapshot of a near-collision situation (scenario #1). (b) shows a snapshot of a collision with two intersection trajectories (scenario #2). (c) and (d) show the progression of criticality metrics (distance, WTTC, TDP) over time, respectively.

B.2.2.2 Near-collision Example

In the near-collision scenario depicted inFigure B.5 (a), two vehicles (129, 139) are stationary at an intersection. While vehicle 135 attempts to cross the intersection between both mentioned vehicles, vehicle 142 passes 129 and 139 in an oncoming lane. All of this occurs in less than three seconds, and 142 cannot see 135 early on. Both vehicles barely avoid a collision. Figure B.5 (c) presents the measured distance and WTTC between 135 and 142, respectively. The minimum distance between the two is approximately $2.75m$, which, in slow urban traffic, does not indicate a critical situation. However, their WTTC drops to $0.22s$, representing a critical situation. The TDP value for vehicle 135 begins to increase steeply when it passes through the gap between 129 and 139 and remains slightly above 1.0 throughout the maneuver. The TDP value for 142 stays around 1.2, with a slight dip when vehicle 135 comes to a halt and accelerates again.

B.2.2.3 Collision Example

The second example, illustrated in Figure B.5 (b), is taken from the same dataset and involves vehicle 81 coming from the left side of an intersection and turning right onto the bottom arm, while vehicle 167 from the left bottom arm intends to turn right as well. In this case, vehicle 81 collides with vehicle 164, and both vehicles pull over to the side of the road, which is a common post-collision behavior. Figure B.5 (d) shows the distance curves between vehicle 81 and 167, which decrease to $0m$ and result in a WTTC value of $0s$ at timestamp 121.1. The negative WTTC value indicates that after the crash, both vehicles came to a stop, and the system of equations used to calculate the WTTC cannot be solved. The TDP of vehicle 81 remains constantly high throughout the critical maneuver and collision, whereas the value of vehicle 167 remains below 1.0 as it is still stationary and, therefore, has a lower impact on the outcome of the situation.

C State of the Art

C.1 Confidence Interval Plots

The following plots depict the width of the confidence intervals for scenario 1 A and B and scenario 2. The color indicates the width, with yellow implying a wider width and blue or purple a smaller width for a scenario parameter set.

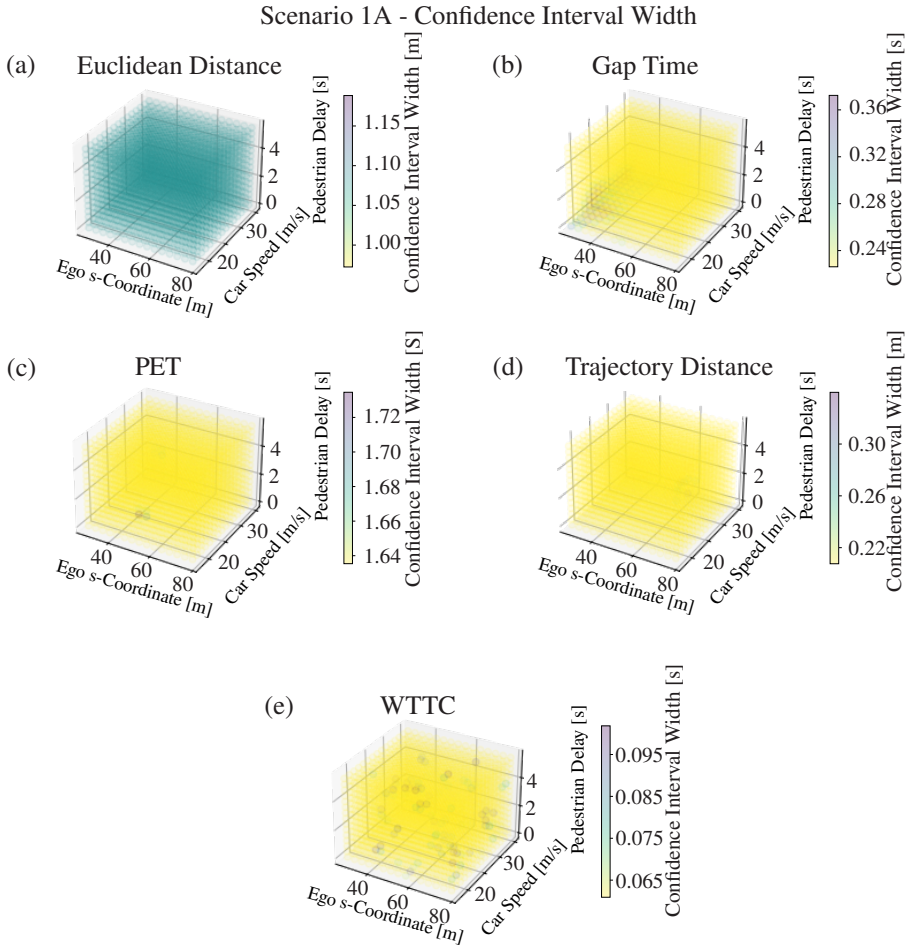


Figure C.1: Confidence intervals as grid for scenario parameter space of scenario 1 A.

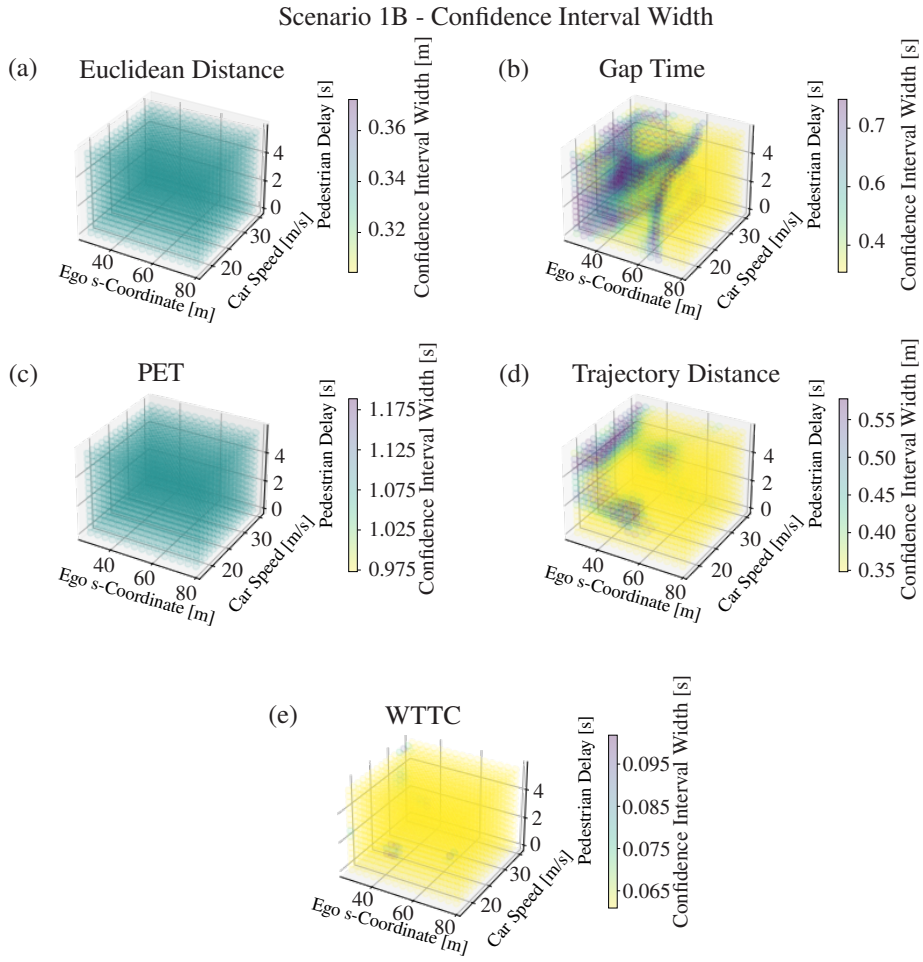


Figure C.2: Confidence intervals as grid for scenario parameter space of scenario 1 B.

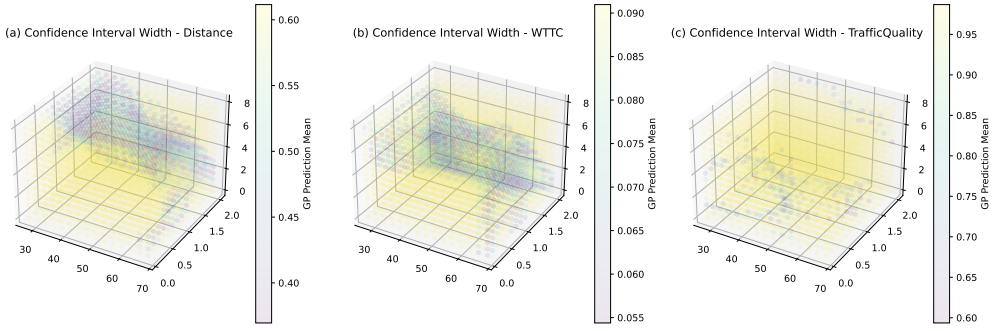


Figure C.3: Confidence intervals as grid for scenario parameter space of scenario 2.

List of Figures

List of Tables

List of Publications

Journal articles

[1] Barbara Schütt, Markus Steimle, Birte Kramer, Danny Behnecke, and Eric Sax. A taxonomy for quality in simulation-based development and testing of automated driving systems. *IEEE Access*, 10:18631–18644, 2022.

[2] Lukas Westhofen, Christian Neurohr, Tjark Koopmann, Martin Butz, Barbara Schütt, Fabian Utesch, Birte Neurohr, Christian Gutenkunst, and Eckard Böde. Criticality Metrics for Automated Driving: A Review and Suitability Analysis of the State of the Art. Technical report, Springer Nature, June 2022. URL `https://doi.org/10.1007/s11831-022-09788-7`.

Conference contributions

[3] Thilo Braun, Julian Fuchs, Felix Reisgys, Lennart Ries, Johannes Plaum, Barbara Schütt, and Eric Sax. A Review of Scenario Similarity Measures for Validation of Highly Automated Driving. In *26th IEEE International Conference on Intelligent Transportation Systems (ITSC 2023)*. IEEE, 2023.

[4] Jannis Erz, Barbara Schütt, Thilo Braun, Houssem Guissouma, and Eric Sax. Towards an ontology that reconciles the operational design domain, scenario-based testing, and automated vehicle architectures. In *2022 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2022.

[5] Joshua Ransiek, Barbara Schütt, Adrian Hof, and Eric Sax. Generation of adversarial trajectories using reinforcement learning to test motion planning algorithms. In *26th IEEE International Conference on Intelligent Transportation Systems (ITSC 2023)*. IEEE, 2023.

[6] Barbara Schütt, Thilo Braun, Stefan Otten, and Eric Sax. Sceml: a graphical modeling framework for scenario-based testing of autonomous vehicles. In *Proceedings of the 23rd*

ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pages 114–120, 2020.

[7] Barbara Schütt, Marc Heinrich, Sonja Marahrens, J. Marius Zöllner, and Eric Sax. An Application of Scenario Exploration to Find New Scenarios for the Development and Testing of Automated Driving Systems in Urban Scenarios. In *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems*, pages 338–345, 2022. ISBN 978-989-758-573-9. doi: 10.5220/0011064600003191.

[8] Barbara Schütt, Maximilian Zipfl, J Marius Zöllner, and Eric Sax. Fingerprint of a traffic scene: an approach for a generic and independent scene assessment. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–8. IEEE, 2022.

[9] Barbara Schütt, Stefan Otten, and Eric Sax. Clustering-based criticality analysis for testing of automated driving systems. *26th IEEE International Conference on Intelligent Transportation Systems (ITSC 2023)*, 2023.

[10] Barbara Schütt, Joshua Ransiek, Thilo Braun, and Eric Sax. 1001 ways of scenario generation for testing of self-driving cars: A survey. In *IV2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023.

[11] Barbara Schütt, Meng Zhang, Christian Steinhauser, and Eric Sax. Evolutionary behavior tree generation for dynamic scenario creation in testing of automated driving systems. In *ICSRS2023*. IEEE, 2023.

[12] Barbara Schütt, Maximilian Zipfl, J Marius Zöllner, and Eric Sax. Inverse universal traffic quality - a criticality metric for crowded urban traffic scenes. In *IV2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023.

[13] Maximilian Zipfl, Barbara Schütt, and J Marius Zöllner. Scene-extrapolation: Generating interactive traffic scenarios. 2024.

Bibliography

[14] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 1016–1026. IEEE, 2018.

[15] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE, 2019.

[16] Allen, Brian, L., B. Tom Shin, and Cooper, Peter, J. Analysis of Traffic Conflicts and Collisions. *Transportation Research Record*, 667:67–74, 1978.

[17] Matthias Althoff and Sebastian Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.

[18] Matthias Althoff, Olaf Stursberg, and Martin Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):299–310, 2009. Publisher: IEEE.

[19] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999. Publisher: ACM New York, NY, USA.

[20] ASAM e.V. Welcome to Open Simulation Interface's documentation!, 2023. URL `https://opensimulationinterface.github.io/osi-documentation/index.html`.

[21] ASAM OpenSCENARIO. ASAM OpenSCENARIO, 2020. URL `https://www.asam.net/standards/detail/openscenario/`.

[22] ASAM OSI. ASAM OSI® (Open Simulation Interface), 2021. URL `https://opensimulationinterface.github.io/osi-documentation/index.html`.

[23] Ulf Aslak. py_pcha, 2023. URL `https://github.com/ulfaslak/py_pcha`.

[24] Johannes Bach, Stefan Otten, and Eric Sax. Model based scenario specification for development and test of automated driving functions. In *2016 IEEE Intelligent Vehicles Symp. (IV)*, pages 1149–1155. IEEE, 2016.

[25] Gerrit Bagschik, Till Menzel, and Markus Maurer. Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1813–1820. IEEE, 2018.

[26] Max Paul Bauer, Anthony Ngo, and Michael Resch. The YASE Framework: Holistic Scenario Modeling with Behavior Trees. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7. IEEE, 2021.

[27] Daniel Baumann, Raphael Pfeffer, and Eric Sax. Automatic Generation of Critical Test Cases for the Development of Highly Automated Driving Functions. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5. IEEE, 2021.

[28] Richard Bellman and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959. Publisher: IEEE.

[29] Raja Ben Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing Advanced Driver Assistance Systems Using Multi-Objective Search and Neural Networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016, pages 63–74, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-3845-5. doi: 10.1145/2970276.2970311. URL `https://doi.org/10.1145/2970276.2970311`. event-place: Singapore, Singapore.

[30] Florian Bock, Christoph Sippl, Aaron Heinzz, Christoph Lauerz, and Reinhard German. Advantageous usage of textual domain-specific languages for scenario-driven development of automated driving functions. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2019.

[31] 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. doi: 10.1109/IV47402.2020.9304839.

[32] Thilo Braun, Lennart Ries, Franziska Körtke, Lara Ruth Turner, Stefan Otten, and Eric Sax. Collection of Requirements and Model-based Approach for Scenario Description. In *VEHITS*, pages 634–645, 2021.

[33] Thilo Braun, Lennart Ries, Moritz Hesche, Stefan Otten, and Eric Sax. Maneuver-based Visualization of Similarities between Recorded Traffic Scenarios. In *Proceedings of the*

11th International Conference on Data Science, Technology and Applications - Volume 1: DATA,, pages 236–244. SciTePress, 2022. ISBN 978-989-758-583-8. doi: 10.5220/0011140600003269. Backup Publisher: INSTICC.

[34] Andreas Bussler, Lukas Hartjen, Robin Philipp, and Fabian Schuldt. Application of Evolutionary Algorithms and Criticality Metrics for the Verification and Validation of Automated Driving Systems at Urban Intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 128–135. IEEE, 2020.

[35] Oliver Bühler and Joachim Wegener. Evolutionary functional testing. *Computers & Operations Research*, 35(10):3144–3160, 2008. Publisher: Elsevier.

[36] Yi Cao, Lingyun Xiao, Honglei Dong, Yan Wang, Xiaobo Wu, Pingfei Li, and Yuanchao Qiu. Typical pre-crash scenarios reconstruction for two-wheelers and passenger vehicles and its application in parameter optimization of aeb system based on nais database. In *26th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, 2019.

[37] Carla Simulator Community. Core concepts, 2023. URL `https://carla.readthedocs.io/en/latest/core_concepts/`.

[38] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020. Publisher: Springer.

[39] Michele Colledanchise and Petter Ögren. How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees. *IEEE Transactions on robotics*, 33(2):372–389, 2016. Publisher: IEEE.

[40] Council of European Union. Regulation (EU) 2018/858 of the European Parliament and of the Council of 30 May 2018 on the approval and market surveillance of motor vehicles and their trailers, and of systems, components and separate technical units intended for such vehicles, amending Regulations (EC) No 715/2007 and (EC) No 595/2009 and repealing Directive 2007/46/EC, 2018. URL `https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX%3A32018R0858`.

[41] Council of European Union. Regulation (EU) 2019/2144 of the European Parliament and of the Council of 27 November 2019 on type-approval requirements for motor vehicles and their trailers, and systems, components and separate technical units intended for such vehicles, as regards their general safety and the protection of vehicle occupants and vulnerable road users, amending Regulation (EU) 2018/858 of the European Parliament and of the Council and repealing Regulations (EC) No 78/2009, (EC) No 79/2009 and (EC) No 661/2009 of the European Parliament and of the Council and Commission Regulations

(EC) No 631/2009, (EU) No 406/2010, (EU) No 672/2010, (EU) No 1003/2010, (EU) No 1005/2010, (EU) No 1008/2010, (EU) No 1009/2010, (EU) No 19/2011, (EU) No 109/2011, (EU) No 458/2011, (EU) No 65/2012, (EU) No 130/2012, (EU) No 347/2012, (EU) No 351/2012, (EU) No 1230/2012 and (EU) 2015/166, 2019. URL https://eur-lex.europa.eu/eli/reg/2019/2144/oj.

[42] Council of European Union. COMMISSION IMPLEMENTING REGULATION (EU) 2022/1426 of 5 August 2022 laying down rules for the application of Regulation (EU) 2019/2144 of the European Parliament and of the Council as regards uniform procedures and technical specifications for the type-approval of the automated driving system (ADS) of fully automated vehicles, 2022. URL https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32022R1426#d1e41-20-1.

[43] Adele Cutler and Leo Breiman. Archetypal analysis. *Technometrics*, 36(4):338–347, 1994. Publisher: Taylor & Francis.

[44] W Damm, S Kemper, E Möhlmann, T Peikenkamp, and A Rakow. Traffic sequence charts-from visualization to semantics. Technical report, AVACS Technical Report 117 (Oct 2017), 2017.

[45] George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, 2021. Publisher: ACM New York, NY.

[46] Erwin de Gelder and Jan-Pieter Paardekooper. Assessment of automated driving systems using real-life scenarios. In *Proc. 2017 IEEE Intell. Vehicles Symp. (IV)*, pages 589–594, 2017.

[47] Wenhao Ding, Baiming Chen, Minjun Xu, and Ding Zhao. Learning to collide: An adaptive safety-critical scenarios generating method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2250. IEEE, 2020.

[48] Gregory Dobson and Tolga Tezcan. Optimal sampling strategies in the coupon collector's problem with unknown population size. *Annals of Operations Research*, 233:77–99, 2015. Publisher: Springer.

[49] 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.

[50] dSPACE GmbH. dSPACE - SIMPHERA Web-based solution for simulation and validation in autonomous driving development, January 2022. URL https://www.dspace.com/de/gmb/home/products/sw/simulation_software/simphera.cfm.

[51] dSpace GmbH. Scenario-Based Tests with Radar Sensors, 2022. URL https://www.dspace.com/en/pub/home/products/systems/darts/scenariobasedtests.cfm.

[52] Nils Gustaf Eissfeldt. *Vehicle-based modelling of traffic . Theory and application to environmental impact modelling*. PhD Thesis, Universität zu Köln, 2004. URL https://kups.ub.uni-koeln.de/1274/.

[53] E Esenturk, Siddartha Khastgir, A Wallace, and P Jennings. Analyzing real-world accidents for test scenario generation for automated vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 288–295. IEEE, 2021.

[54] esmini. esmini: a basic OpenSCENARIO player, 2023. URL https://github.com/esmini/esmini.

[55] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. Issue: 34.

[56] ASAM e.V. OpenSCENARIO XML, March 2024. URL https://www.asam.net/standards/detail/openscenario-xml/.

[57] ASAM e.V. OpenSCENARIO DSL, March 2024. URL https://www.asam.net/standards/detail/openscenario-dsl/.

[58] Lan Feng, Quanyi Li, Zhenghao Peng, Shuhan Tan, and Bolei Zhou. TrafficGen: Learning to Generate Diverse and Realistic Traffic Scenarios. *arXiv preprint arXiv:2210.06609*, 2022.

[59] Mark Finkelstein, Howard G Tucker, and Jerry Alan Veeh. Confidence intervals for the number of unseen types. *Statistics & Probability Letters*, 37(4):423–430, 1998. Publisher: Elsevier.

[60] Foretellix. Measurable Scenario Description Language Reference, July 2020. URL https://www.foretellix.com/wp-content/uploads/2020/07/M-SDL_LRM_OS.pdf.

[61] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[62] Dirk Frerichs and Matthias Borsdorf. Quality for vehicle system simulation. In *VDI-Kongress" SIMVEC-Simulation und Erprobung in der Fahrzeugentwicklung*, 2018.

[63] Alessio Gambi, Tri Huynh, and Gordon Fraser. Generating effective test cases for self-driving cars from police reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 257–267, 2019.

[64] Briti Gangopadhyay, Siddartha Khastgir, Sumanta Dey, Pallab Dasgupta, Giovanni Montana, and Paul Jennings. Identification of test cases for automated driving systems using bayesian optimization. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1961–1967. IEEE, 2019.

[65] Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 157–164. IEEE, 2021.

[66] Google Waymo. Waymax: An Accelerated, Data-Driven Simulator for Large-Scale Autonomous Driving Research, 2023. URL `https://waymo.com/research/waymax/`.

[67] Quentin Goss, Yara AlRashidi, and Mustafa İlhan Akbaş. Generation of modular and measurable validation scenarios for autonomous vehicles using accident data. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 251–257. IEEE, 2021.

[68] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian Optimization for Adaptive Experimental Design: A Review. *IEEE Access*, 8: 13937–13948, 2020. doi: 10.1109/ACCESS.2020.2966228.

[69] Felix Günther. *Beitrag zur Co-Simulation in der Gesamtsystementwicklung des Kraftfahrzeugs*. PhD Thesis, Techn. Univ. München, München, Germany, 2017.

[70] Sven Hallerbach, Yiqun Xia, Ulrich Eberle, and Frank Koester. Simulation-based identification of critical scenarios for cooperative and automated vehicles. *SAE International Journal of Connected and Automated Vehicles*, 1(2018-01-1066):93–106, 2018.

[71] Lukas Hartjen, Fabian Schuldt, and Bernhard Friedrich. Semantic Classification of Pedestrian Traffic Scenarios for the Validation of Automated Driving. In *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pages 3696–3701. Institute of Electrical and Electronics Engineers Inc., October 2019. ISBN 978-1-5386-7024-8. doi: 10.1109/ITSC.2019.8917485.

[72] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *Proc. 22st Int. Conf. Intell. Transp. Syst. (ITSC)*, pages 2950–2955, 2019.

[73] John C. Hayward. Near miss determination through use of a scale of danger. In *Unknown*, 1972.

[74] Errol R Hoffmann and Rugolf G Mortimer. Drivers' estimates of time to collision. *Accident Analysis & Prevention*, 26(4):511–520, 1994. Publisher: Elsevier.

[75] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. *Kernel methods in machine learning*. 2008.

[76] Hardi Hungar. A concept of scenario space exploration with criticality coverage guarantees. In *International Symposium on Leveraging Applications of Formal Methods*, pages 293–306. Springer, 2020.

[77] Christer Hydén. The development of a method for traffic safety evaluation: The Swedish Traffic Conflicts Technique. *Bulletin Lund Institute of Technology, Department*, 70, 1987.

[78] IPG - Automotive GmbH. IPG - Automotive GmbH - Everything about virtual test driving, 2023. URL `https://ipg-automotive.com/en/products-solutions/software/carmaker/`.

[79] ISO. Road vehicles – Functional safety (ISO26262), 2018. Type: Norm.

[80] Nidhi Kalra and Susan M 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, 2016. Publisher: Elsevier.

[81] Dhanoop Karunakaran, Stewart Worrall, and Eduardo Nebot. Efficient statistical validation with edge cases to evaluate highly automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[82] Sebastian Mathias Keller, Maxim Samarin, Fabricio Arend Torres, Mario Wieser, and Volker Roth. Learning extremal representations with deep archetypal analysis. *International Journal of Computer Vision*, 129:805–820, 2021. Publisher: Springer.

[83] Christian King, Lennart Ries, Christopher Kober, Christoph Wohlfahrt, and Eric Sax. Automated Function Assessment in Driving Scenarios. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, pages 414–419. IEEE, 2019.

[84] Christian King, Thilo Braun, Constantin Braess, Jacob Langner, and Eric Sax. Capturing the Variety of Urban Logical Scenarios from Bird-view Trajectories. In *VEHITS*, pages 471–480, 2021.

[85] Jack PC Kleijnen. Verification and validation of simulation models. *European journal of operational research*, 82(1):145–162, 1995. Publisher: Elsevier.

[86] Florian Klück, Martin Zimmermann, Franz Wotawa, and Mihai Nica. Performance comparison of two search-based testing strategies for ADAS system validation. In *IFIP Int. Conf. on Testing Software and Systems*, pages 140–156. Springer, 2019.

[87] Philip Koopman, Beth Osyk, and Jack Weast. Autonomous vehicles meet the physical world: Rss, variability, uncertainty, and proving safety. In *International Conference on Computer Safety, Reliability, and Security*, pages 245–253. Springer, 2019.

[88] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.

[89] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symp. (IV)*, pages 204–211. IEEE, 2017.

[90] Jacob Langner, Johannes Bach, Lennart Ries, Stefan Otten, Marc Holzäpfel, and Eric Sax. Estimating the uniqueness of test scenarios derived from recorded real-world-driving-data using autoencoders. In *Proc 2018 IEEE Intell. Vehicles Symp. (IV)*, pages 1860–1866, 2018.

[91] Jacob Langner, Hannes Grolig, Stefan Otten, Marc Holzäpfel, and Eric Sax. Logical Scenario Derivation by Clustering Dynamic-Length-Segments Extracted from Real-World-Driving-Data. In *VEHITS*, pages 458–467, 2019.

[92] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014. Publisher: SpringerOpen.

[93] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[94] Yee-Fun Lim, Chee Koon Ng, US Vaitesswar, and Kedar Hippalgaonkar. Extrapolative Bayesian optimization with Gaussian process and neural network ensemble surrogate models. *Advanced Intelligent Systems*, 3(11):2100101, 2021. Publisher: Wiley Online Library.

[95] Clemens Linnhoff, Philipp Rosenberger, and Hermann Winner. Refining Object-Based Lidar Sensor Modeling — Challenging Ray Tracing as the Magic Bullet. *IEEE Sensors Journal*, 21(21):24238–24245, 2021. doi: 10.1109/JSEN.2021.3115589.

[96] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner,

and Evamarie Wießner. Microscopic Traffic Simulation using SUMO. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL https://elib.dlr.de/124092/. Journal Abbreviation: IEEE Intelligent Transportation Systems Conference (ITSC).

[97] SM Sohel Mahmud, Luis Ferreira, Md Shamsul Hoque, and Ahmad Tavassoli. Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS research*, 41(4):153–163, 2017. Publisher: Elsevier.

[98] Sumbal Malik, Manzoor Ahmed Khan, and Hesham El-Sayed. CARLA: Car Learning to Act—An Inside Out. *Procedia Computer Science*, 198:742–749, 2022. Publisher: Elsevier.

[99] Ryan Marcotte and Howard J Hamilton. Behavior trees for modelling artificial intelligence in games: A tutorial. *The Computer Games Journal*, 6(3):171–184, 2017. Publisher: Springer.

[100] Anthony D McDonald, Hananeh Alambeigi, Johan Engström, Gustav Markkula, Tobias Vogelpohl, Jarrett Dunne, and Norbert Yuma. Toward computational simulations of behavior during automated driving takeovers: a review of the empirical and modeling literatures. *Human factors*, 61(4):642–688, 2019. Publisher: SAGE Publications Sage CA: Los Angeles, CA.

[101] Andrew McHutchon and Carl Rasmussen. Gaussian process training with input noise. *Advances in neural information processing systems*, 24, 2011.

[102] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1821–1827. IEEE, 2018.

[103] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2383–2390. IEEE, 2019.

[104] Microsoft Research. Welcome to AirSim, 2023. URL https://microsoft.github.io/AirSim/.

[105] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.

[106] Ken T Mori, Xu Liang, Lukas Elster, and Steven Peters. The Inadequacy of Discrete Scenarios in Assessing Deep Neural Networks. *IEEE Access*, 2022. Publisher: IEEE.

[107] MSC Software. Virtual Test Drive (VTD) - Complete Tool-Chain for Driving Simulation, 2023. URL https://www.mscsoftware.com/product/virtual-test-drive.

[108] Demin Nalic, Tomislav Mihalj, Maximilian Bäumler, Matthias Lehmann, Arno Eichberger, and Stefan Bernsteiner. Scenario Based Testing of Automated Driving Systems: A Literature Survey. In *Proc. FISITA Web Congr.*, page 30, 2020.

[109] Christian Neurohr, Lukas Westhofen, Martin Butz, Martin Bollmann, Ulrich Eberle, and Roland Galbas. Criticality Analysis for the Verification and Validation of Automated Vehicles. *IEEE Access*, 2021. Publisher: IEEE.

[110] Daiheng Ni. A framework for new generation transportation simulation. In *Proc. 2006 Winter Simulation Conf.*, pages 1508–1514. IEEE, 2006.

[111] Nick Bastone. Elon Musk Says Tesla Will Roll Out 1 Million Robo-Taxis by Next Year. Here's How He Plans on Doing It, 2019. URL https://www.inc.com/business-insider/tesla-is-taking-direct-aim-at-uber-and-lyft-with-plans-to-roll-out-one-million-robo-taxis-by-next-year.html.

[112] Srinivas Nidhra and Jagruthi Dondeti. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, 2 (2):29–50, 2012.

[113] David Nistér, Hon-Leung Lee, Julia Ng, and Yishou Wang. An Introduction to the Safety Force Field (Nvidia), 2019. URL https://developer.download.nvidia.com/driveworks/secure/docs/DRIVE_8.0_Release_Docs/the-safety-force-field.pdf?IwXpblapg2B6NbhQTUgbvNTqDl7IVRzWr46XC4F3d4r6uc5ErD7pFjZUWGR1Fsxa1aUKSu_GMgOpqJAKkkYIvYEMpo8l7xzmmiUaYNGzzz8X_iKVVxuxOdYlkwnxaptq6KZASxMu-8AkgynG0-hB-PMBgExeiEYdOAVNAhxXCZKh71YcrGNQ.

[114] David Nistér, Hon-Leung Lee, Julia Ng, and Yishou Wang. An Introduction to the Safety Force Field (Nvidia), 2019. URL https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/an-introduction-to-the-safety-force-field-v2.pdf.

[115] Andreas Nonnengart, Matthias Klusch, and Christian Müller. CriSGen: Constraint-based generation of critical scenarios for autonomous vehicles. In *International Symposium on Formal Methods*, pages 233–248. Springer, 2019.

[116] NVIDIA Corporation. END-TO-END SOLUTIONS FOR AUTONOMOUS VEHICLES, 2022. URL https://developer.nvidia.com/drive#:~:text=It's%20modular%20and%20open%2C%20empowering,monitoring%2C%20and%20natural%20language%20processing.

[117] openPASS. openPASS - Target Objectives, 2022. URL https://openpass.eclipse.org/target_objectives/.

[118] Ishaan Paranjape, Abdul Jawad, Yanwen Xu, Asiiah Song, and Jim Whitehead. A modular architecture for procedural generation of towns, intersections and scenarios for testing autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 162–168. IEEE, 2020.

[119] Pegasus Project. Pegasus Method - An Overview, 2019. URL https://www.pegasusprojekt.de/files/tmpl/Pegasus-Abschlussveranstaltung/PEGASUS-Gesamtmethode.pdf.

[120] PTV Group. Traffic Simulation Software - PTV Vissim - PTV Group, 2023. URL https://www.ptvgroup.com/en/solutions/products/ptv-vissim/.

[121] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, and others. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009. Issue: 3.2.

[122] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. 026218253X, 2006.

[123] Stefan Riedmaier, Benedikt Danquah, Bernhard Schick, and Frank Diermeyer. Unified Framework and Survey for Model Verification, Validation and Uncertainty Quantification. *Archives Comput. Methods Eng.*, pages 1886–1784, 2020. Publisher: Springer.

[124] Lennart Ries, Philipp Rigoll, Thilo Braun, Thomas Schulik, Johannes Daube, and Eric Sax. Trajectory-Based Clustering of Real-World Urban Driving Sequences with Multiple Traffic Objects. In *Proc. 24th IEEE Int. Conf. Intell. Transp. Syst.* IEEE, 2021.

[125] Sebastian Rietsch, Shih-Yuan Huang, Georgios Kontes, Axel Plinge, and Christopher Mutschler. Driver dojo: A benchmark for generalizable reinforcement learning for autonomous driving. *arXiv preprint arXiv:2207.11432*, 2022.

[126] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3): 303–304, 2008. Publisher: Nature Publishing Group US New York.

[127] Rodney Brooks. Predictions Scorecard, 2021 January 01, 2021. URL https://rodneybrooks.com/predictions-scorecard-2021-january-01/.

[128] Philipp Rosenberger, Jan Timo Wendler, Martin Friedrich Holder, Clemens Linnhoff, Moritz Berghöfer, Hermann Winner, and Markus Maurer. Towards a Generally Accepted Validation Methodology for Sensor Models - Challenges, Metrics, and First Results. In *Grazer Symposium Virtuelles Fahrzeug*, May 2019. URL http://tuprints.ulb.tu-darmstadt.de/8653/.

[129] SAE. J3016 - SURFACE VEHICLE RECOMMENDED PRACTICE - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2021. URL https://www.sae.org/standards/content/j3016_202104/.

[130] Eric Sax. *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie*. Hanser, 2008.

[131] Manuel Schiller, Marius Dupius, Daniel Krajzewicz, Andreas Kern, and Alois Knoll. Multi-resolution traffic simulation for large-scale high-fidelity evaluation of VANET applications. In *Simulating Urban Traffic Scenarios*, pages 17–36. Springer, 2019.

[132] Maike Scholtes, Lukas Westhofen, Lara Ruth Turner, Katrin Lotto, Michael Schuldes, Hendrik Weber, Nicolas Wagener, Christian Neurohr, Martin Herbert Bollmann, Franziska Körtke, and others. 6-layer model for a structured description and categorization of urban traffic and environment. *IEEE Access*, 9:59131–59147, 2021. Publisher: IEEE.

[133] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018. Publisher: Elsevier.

[134] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. Publisher: MIT Press.

[135] scikit learn. 2.3.9. OPTICS, December 2023. URL https://scikit-learn.org/stable/modules/clustering.html#optics.

[136] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008.

[137] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[138] Markus Steimle, Gerrit Bagschik, Till Menzel, Jan Timo Wendler, and Markus Maurer. Ein Beitrag zur Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. In ITS Niedersachsen, editor, *AAET - Automatisiertes und vernetztes Fahren*, pages 10–32, Brunswick, Germany, 2018.

[139] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

[140] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901, 2021.

[141] Technical Committee ISO/TC 22, Road vehicles, Subcommittee SC 32, Electrical and electronic components and general system aspects. ISO/PAS 21448:2022(en) Road vehicles — Safety of the intended functionality, 2022. URL `https://www.iso.org/obp/ui/#iso:std:iso:pas:21448:ed-1:v1:en`.

[142] Silvia Thal, Roman Henze, Ryo Hasegawa, Hiroki Nakamura, Hisashi Imanaga, Jacobo Antona-Makoshi, and Nobuyuki Uchida. Generic Detection and Search-based Test Case Generation of Urban Scenarios based on Real Driving Data. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 694–701. IEEE, 2022.

[143] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Phys. Rev. E*, 62(2):1805–1824, August 2000. doi: 10.1103/PhysRevE.62.1805. URL `https://link.aps.org/doi/10.1103/PhysRevE.62.1805`. Publisher: American Physical Society.

[144] TÜV SÜD AG. Assessment of automated vehicles with scenario-based testing, 2020. URL `https://www.tuvsud.com/en/-/media/global/pdf-files/brochures-and-infosheets/mobility-and-automotive/tuvsud-av-scenario-testing.pdf?la=en&hash=25C763437E68B5A17CEBF5F286E66554`.

[145] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: an efficient Bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016. Publisher: Elsevier.

[146] Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *2015 IEEE 18th Int. Conf. Intelligent Transportation Systems*, pages 982–988. IEEE, 2015.

[147] VVM Project. Homepage: VVM Project, 2022. URL https://www.vvm-projekt.de/en/.

[148] Walther Wachenfeld and Hermann Winner. The release of autonomous vehicles. In *Autonomous driving*, pages 425–449. Springer, 2016.

[149] Walther Wachenfeld, Philipp Junietz, Raphael Wenzel, and Hermann Winner. The worst-time-to-collision metric for situation identification. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 729–734. IEEE, 2016. doi: 10.1109/IVS.2016.7535468.

[150] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Back. A new acquisition function for Bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512. IEEE, 2017.

[151] Jie Wang. An intuitive tutorial to Gaussian processes regression. *Computing in Science & Engineering*, 2023. Publisher: IEEE.

[152] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021.

[153] Juan Wang, Yongyi Yang, and Bin Xia. A simplified Cohen's Kappa for use in binary classification data annotation tasks. *IEEE Access*, 7:164386–164397, 2019. Publisher: IEEE.

[154] Waymo. Waymo Safety Report, 2020. URL https://storage.googleapis.com/sdc-prod/v1/safety-report/2020-09-22-safety-report.pdf.

[155] Nico Weber, Christoph Thiem, and Ulrich Konigorski. UnScenE: Toward Unsupervised Scenario Extraction for Automated Driving Systems from Urban Naturalistic Road Traffic Data. *arXiv preprint arXiv:2202.06608*, 2022.

[156] Hermann Winner, Stephan Hakuli, and Gabriele Wolf. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Springer-Verlag, 2011.

[157] Matthew Wood, Philipp Robbel, Michael Maass, R Duintjer Tebbens, M Meijs, M Harb, and P Schlicht. Safety first for automated driving. *Aptiv, Audi, BMW, Baidu, Continental Teves, Daimler, FCA, HERE, Infineon Technologies, Intel, Volkswagen*, 2019. URL https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf.

[158] Zhang Xinxin, Li Fei, and Wu Xiangbin. Csg: Critical scenario generation from real traffic accidents. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1330–1336. IEEE, 2020.

[159] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005. Publisher: Ieee.

[160] Ansar-Ul-Haque Yasar, Yolande Berbers, and Davy Preuveneers. A computational analysis of driving variations on distributed multiuser driving simulators. In *IASTED International Conference on Modelling and Simulation*, pages 178–186. ACTA Press; Canada, 2008. Issue: 19.

[161] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*, 2019.

[162] Marc Rene Zofka, Florian Kuhnt, Ralf Kohlhaas, and Johann Marius Zöllner. Simulation framework for the development of autonomous small scale vehicles. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR 2016, San Francisco, CA, USA, December 13-16, 2016*, pages 318–324, 2016. URL `https://doi.org/10.1109/SIMPAR.2016.7862413`.

[163] Marc René Zofka, Florian Kuhnt, Ralf Kohlhaas, Christoph Rist, Thomas Schamm, and J Marius Zöllner. Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 1422–1428. IEEE, 2015.

Supervised Student Theses

[164] Felix Deumlich. Identifikation kritischer Szenarien für das Testen automatisierter Fahrfunktionen anhand der Bewertung der Verkehrsqualität. Bachelor thesis, HTW Berlin, 2022.

[165] Johannes Gundlach. Binning Algorithm of Operational Design Domain Coverage for Testing Highly Automated Trucks. Master thesis, KIT - Karlsruher Institut für Technologie, 2024.

[166] Meng Zhang. Scenario Generation to Find New Scenarios for the Development and Testing of Automated Driving Systems in Urban Traffic. Master thesis, Technische Universität Berlin, 2023.