Critical Scenario Identification for Realistic Testing of Autonomous Driving Systems

Qunying Song (qunying.song@cs.lth.se)
Lund University, Sweden https://orcid.org/0000-0002-8653-0250
Kaige Tan
Royal Institute of Technology, Sweden
Per Runeson
Lund University, Sweden
Stefan Persson
Volvo Cars Corporation, Sweden

Research Article

Keywords: Critical scenario identification, Autonomous driving, Software testing, Test scenario generation

Posted Date: January 20th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1280095/v1

Critical Scenario Identification for Realistic Testing of Autonomous Driving Systems

Qunying Song[1*], Kaige Tan[2], Per Runeson[1] and Stefan Persson[3]

[1*]Department of Computer Science, Lund University, Box 118, SE-221 00 Lund, Sweden.
[2]Department of Mechatronics, Royal Institute of Technology, Brinellvägen 83, SE-100 44 Stockholm, Sweden.
[3]Volvo Cars Corporation, Assar Gabrielssons väg, SE-405 31, Göteborg, Sweden.

*Corresponding author(s). E-mail(s): qunying.song@cs.lth.se;
Contributing authors: kaiget@kth.se; per.runeson@cs.lth.se;
stefan.persson@volvocars.com;

Abstract

Autonomous driving has become an important research area for road traffic, whereas testing of autonomous driving systems to ensure a safe and reliable operation, remains an open challenge. Substantial real-world testing or massive driving data collection does not scale, as the potential test scenarios in real-world traffic are infinite and covering large shares of them in test is impractical, thus critical ones have to be prioritized. In this study, we establish a systematic approach for critical test scenario identification with integrated tools and a workflow, to explore the most critical test scenarios and facilitate testing of the autonomous driving functions. We also demonstrate the effectiveness of our approach by using two real autonomous driving systems from the industry by collaborating with Volvo Cars. Our main contribution in this work is a feasible and complete tool-chain for critical test scenario identification that is general for testing different autonomous driving systems.

Keywords: Critical scenario identification; Autonomous driving; Software testing; Test scenario generation

Statements and Declarations There is no financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

1 Introduction

While autonomous driving is expected to improve traffic capacity and reduce road accidents, testing of autonomous driving systems is prerequisite to ensure the reliability and safety of such systems [1]. Inadequate or ineffective testing could fail to discover potential defects and misbehavior in the system, and consequently lead to severe accidents in the road traffic [2]. The fatal accident caused by Uber's autonomous vehicle is a typical example of such failures where a cyclist in front was not detected and subsequently hit by the vehicle at an intersection in Arizona, US, in 2018 [3].

Current approaches for testing autonomous driving systems that rely on substantial real-world testing, or collecting real driving data at scale, are considered both inefficient and ineffective as they take an unpractical amount of time to complete, and may still not cover rare traffic situations [4]. The regular road traffic, most of the time, is considered non-critical [5]. Therefore, new approaches for testing autonomous driving systems based on critical scenario identification are increasingly demanded [2, 6]. Herein, critical scenarios are referred to as scenarios that can lead to collision or near-collision consequences or situations, and are of interest for testing of autonomous vehicles.

Nevertheless, existing studies mostly present parts of a complete solution for critical test scenario identification, for example, focusing on either simulation or optimization of driving scenarios [7, 8]. Also, the reported studies are in many cases function-specific, for example, by proposing interventions based on a particular function module, like motion-planning for the highway scenario [9]. Therefore, the feasibility of such approaches for testing different autonomous driving functions is unclear. In addition, previous studies tend to not validate their approaches on real driving functions from industry, but instead on some basic implementations based on existing platforms like MATLAB Simulink [10], or using publicly available driving components like DeepDriving [2, 6]. The effectiveness of those approaches for testing real autonomous driving systems, under real traffic conditions, is not demonstrated.

To tackle the aforementioned challenges and facilitate testing of different autonomous driving systems, we have proposed a critical test scenario identification approach in our previous short paper [11]. We extend the approach in the current work and generate critical test scenarios for two industrial autonomous driving functions by partnering with the automaker Volvo Cars, which include a parking function in the low-speed maneuvering domain and a driving function in the high-speed maneuvering domain. Our approach utilizes three existing engineering tools (requirement and verification management tool, SPAS simulation platform, and modeFrontier–process optimization and automation tool) and present a systematic approach for identifying critical driving scenarios through co-simulation with system implementations. The results of applying our approach on the two autonomous driving functions have demonstrated the effectiveness of this approach for identifying the most critical scenarios for testing. Consequently, the identified scenarios can be used

to substantiate test cases for autonomous vehicles in both simulated and real-world testing. To clarify our scope, the work does not aim to find the best optimization algorithm, but to present a systematic approach for critical test scenario identification, and to demonstrate the effectiveness of this approach for testing real autonomous driving functions.

Our work provides a complete solution by integrating different components into a feasible tool-chain as a whole, for critical test scenario identification for autonomous driving. It enables an end-to-end workflow from the initial analysis of the system specifications, until generating a set of critical scenarios that can be used for testing. The approach is generic as the tools involved are exchangeable, and is not subject to any particular driving function, development technique, simulator or application tools that might be intended. Thus, the approach can, in principle, be used for critical scenario identification for testing any autonomous driving systems. In addition, we provide evidence showing that the approach is effective in identifying critical scenarios for testing realistic autonomous driving functions. We also want to highlight that, due to industrial confidentiality concerns, only partial data and result analysis are presented, where sensitive information is removed, still demonstrating the principal outcomes.

The rest of the paper is organized as follows. Section 2 describes concepts, based on literature on critical scenario identification for testing of autonomous driving systems. Section 3 explains the research method and context we use for conducting the study. Section 4 and section 5 detail the case studies that using the proposed approach for identifying critical test scenarios for two industrial driving functions. We present discussions and limitations of the study in Section 6, and conclude the paper in Section 7.

2 Terms and Related Work

In this section, we first present the terms and concepts used in this study, then we summarize the literature on critical test scenario identification for autonomous driving and provide a comparison to our work.

2.1 Terms and Concepts

In this part, terms like scenario and critical scenario are introduced, and their composition as well as differentiation to other similar terms. Besides, relevant concepts predicated on those terms are described to form a basis of this study. Our interpretation and discussion of these terms and concepts are still based on the context of autonomous driving, with a particular focus on testing.

2.1.1 Scenario

According to Ulbrich et al., a *scenario* is defined as a temporal sequence of scenes, with actions and events of the elements that are involved within this sequence [12]. By actions and events, they mean, for example, maneuvers like

cut-out and avoid colliding with a vehicle ahead. Given this definition, a scenario consists of at least one scene with corresponding actions and events, and a scene, in this context, is embodied as the geo-spatially stationary environment, dynamic elements, and a self-representation of all actors and observers.

Based on the definition proposed by Ulbrich et al. [12], Menzel et al. further refined the definition of scenario into three different abstraction levels – *functional*, *logical*, and *concrete* scenarios [13]. Specifically, functional scenarios usually describe the involved entities and their behaviors using a natural language; logical scenarios specify the state space of the functional scenarios with the relevant parameters, parameter range and distribution; concrete scenarios are instantiation of the logical scenarios by assigning concrete values for the parameters within the desired value range and distribution. The relevant parameters are selected for logical scenarios to describe the environmental constitution of the function scenarios, the behavior of the elements involved, and the physical capabilities as well as constraints of the autonomous vehicle. Bagschik et al. [14], have proposed a five-layer model which defines the required parameters for the scenarios, including road-level, traffic infrastructure, temporary manipulation of the road and traffic infrastructure, objects, and environment. Yet, the value range and distribution of the parameters, as well as the possible relations between the parameters, have to be further investigated to instantiate realistic concrete scenarios.

We adopt the definition of scenario proposed by Menzel et al. [13] in our work, where functional scenarios are retrieved from the system specifications and analysed to derive the parameters for logical scenarios. Subsequently, concrete scenarios are simulated and optimized to identify the most critical ones for testing the autonomous driving systems. We have also observed that similar terms such as elements, entities, objects, and traffic participants are often used in the literature to refer to the different road users in the traffic, such as pedestrians, cyclists, vehicles of different types etc. We stick with the framework from Menzel et al. [13] to use entities within the definition of scenario.

2.1.2 Critical Scenario

There is no universal agreement as yet on what constitutes a *critical scenario*, although different interpretations in the literature share a high similarity. Zhang et al. describe a critical scenario as a dangerous road situation that may lead to an unsafe decision for the autonomous vehicle, and appropriate countermeasures must be taken immediately to avoid collision [15]. In contrast, Kluck et al. focus more on the concrete scenarios level, and consider a scenario to be critical if underlying parameter values cause a malfunction of the autonomous driving system [7]. Hallerbach et al. propose critical scenarios as the scenarios that need to be tested, which can be derived from both functional and non-functional requirements (e.g., traffic efficiency, driver comfort etc.) [16]. Herein we take the interpretation from Kluck et al., where critical scenarios are defined as the scenarios with parameter set that have high probability of revealing unintended and unsafe behavior of the systems, which may

cause a collsion or near-collision consequence of the vehicle and other entities on the road traffic [7].

An integral part entailed in critical scenarios is how we quantify and evaluate a scenario to be critical or not, thus the indication of criticality of a scenario must be represented in a quantifiable way. Different surrogate measurements for safety evaluation of traffic conflicts are used, for example, Time-to-Collision (TTC), Post-Enchroachment Time (PET), and Time-to-Brake etc [17]. Among these surrogate indicators, TTC is used the most according to a review study by Aliaksei et al. [18]. Safety metrics can also be extracted from industrial standards and used as the criticality indicators, for example ISO-15622 for adaptive cruise control [19], ISO-26262 for general automotive development and test [20], and Responsibility-Sensitive Safety (RSS) for autonomous vehicles [21]. A number of performance metrics including safety, functionality, mobility, and drivers' comfort are used for generating test scenarios for autonomous vehicles by Feng et al. [20], and they use a combination of the maneuver challenge and exposure frequency as the indicator of critical scenarios. Eventually, selection of the criticality indicators must be specific with respect to a particular driving system and the system specifications.

Furthermore, we have to differentiate critical scenarios from other similar terms to avoid misunderstanding. Gambi et al. use accident scenarios from police reports as critical scenarios for testing autonomous vehicles [6], while Klischat et al. argue that an accident by a human driver may not necessarily be a critical scenario and can be avoidable by others or autonomous vehicles [5]. *Challenging* scenarios and *complex* scenarios are often used alternately and one may consider they are the same as critical scenarios. Riedmaier et al. [19] claim a scenario is critical if the behavior of the system is evaluated after the scenario has been executed either in real-world or in simulator as well as the criticality being measured, and scenarios are challenging or complex only if the scenario itself is evaluated in someway and classified as being challenging or complex. Ponn et al. point out that challenging scenarios are not necessary always critical scenarios, but more often lead to critical scenarios when they are executed [9]. Lastly, we also differentiate the concept of critical scenarios from corner case scenarios (also referred as edge cases). Karunakaran et al. define an edge case as an unknown and unsafe scenario which is hard to predict during the test and can lead to severe result for the autonomous vehicle [21]. Since critical scenarios can either be known or unknown scenarios, we believe the edge cases are a subset of the critical scenarios that are of high interest for its identification and testing the autonomous driving systems.

2.1.3 Scenario-based Testing

Scenarios are commonly used to substantiate test cases for autonomous driving systems [22]. As stated by Kluck et al., a test case is the value assignments of all relevant parameters of the scenario, which essentially aligns with the definition of the concrete scenarios [7]. However, a test case should entail not only a scenario, but also a pass-fail criteria to evaluate the resulting behavior of the

system [2, 13]. An example test case for an autonomous lane-keeping function, as given by Gambi et al. [2], is that the vehicle must follow a navigation path on a generated road map. The test fails if the vehicle is not able to get to the destination or drives out of the lane.

Scenario-based testing is highly accepted and plays a key role for validation of safety of autonomous driving systems [19]. It is inherently connected to the concept of scenario as we have presented in the previous subsections, and it examines the resulting behavior of the autonomous vehicle in terms of interactions with the road infrastructure, with other road entities, and compliance with the functional specifications as well as the safety regulations [8]. The scenario-based testing approach aims to reduce the test effort to a manageable number of scenarios, by limiting the testing to meaningful scenarios based on the testing budget [4]. The number of concrete scenarios can be infinite due to the combinatorial explosion of parameter values [8], and identifying all possible scenarios is difficult regardless of which approach is used [16]. According to Batsch et al., scenario-based testing usually runs in simulation with Software-in-the-Loop (SIL), but can also be carried out with Hardware-in-the-Loop (HIL), or in real-world with proving ground (also known as test tracks in some studies) or regular road traffic [8].

Despite the remarkable benefits of using scenario-based approaches for testing autonomous vehicles, identifying relevant scenarios for the system under test remain the prerequisite, especially those critical scenarios that violate the desired safety requirements [19]. Open questions still challenge us in regards to what constitutes good test scenarios and how to systematically generate them [6]; how to define and collect realistic test scenarios [22]; and how to identify the critical scenarios for testing [15]. Menzel et al. propose many different sources that can be used for deriving test scenarios, which include, but are not limited to, functional specifications, system boundaries, operational environment, legal requirements, and real driving data collected [13]. While common and safe scenarios without significant actions can be easily identified and reduced, the success of a scenario-based testing approach is highly reliant on its ability to find more critical scenarios within a given testing budget [23]. This is the core of the current study – to establish a complete tool-chain for critical test scenario identification and to employ it for testing of real autonomous driving systems.

2.2 Critical Test Scenario Identification

The general idea of critical test scenario identification, as described by Ponn et al., is that a concrete scenario is selected, executed, and evaluated with the criticality metrics [4]. There are different approaches for critical scenario identification, as reported in the literature, ranging from using search-based algorithms, deep learning techniques, or using expert opinions etc. We categorize the literature that we surveyed based on our interpretation and compare them with our work in the following subsections.

For a complete literature overview, we refer to the systematic literature reviews by Zhang et al. [24] for critical scenario identification, and Rajabli et al. [25] for software verification and validation, as well as the survey by Riedmaier et al. [19] for scenario-based approaches, all for assessment of safety of autonomous vehicles.

2.2.1 Knowledge-based Approaches

The knowledge-based approaches leverage expert knowledge to generate, extract, or select scenarios for testing. This approach is not frequently reported in the literature due to its evident constraints. As an example Ponn et al. [9] involve experts from the autonomous driving domain for selecting parameters of scenarios and assessing weight of the parameters as well as evaluating resulting critical scenarios, for testing the autonomous driving systems.

The advantages of using this approach include the quick creation of an initial catalog of test scenarios [19], yet the drawbacks are non-negligible. It requires expert involvement and is labour-intensive, and may lack the diversity and complexity of real-world scenarios, especially those accidents that impose complicated situations and rarely happen [15]. In addition, the generation and selection of scenarios might be subjective, where simple scenarios are ignored but can still cause severe consequences. As a result, the derived scenarios are often considered lacking evidence for proof of safety in real traffic [4].

As a comparison to our approach, we do not require any expert involvement, and identification of the critical scenarios is automated by integrating the existing engineering tools. Specifically, selection of scenarios is based on optimization of the parameter space and the completed simulation results, and thus is not limited or biased by subjective knowledge that has been acquired.

2.2.2 Data-driven Approaches

The data-driven approaches extract critical scenarios based on available data sets that have been collected beforehand. The data can be presented in many different forms, for example, scenario libraries, police accident reports, or sensor data collected by test vehicles. Scenario extraction and selection techniques and tools are then used for identifying critical scenarios from the data.

Among the published studies, Gambi et al. [6] generate effective and critical test scenarios for autonomous driving by reconstructing crash accidents from police reports in simulation, using natural language processing. Zhang et al. [15] introduce a tookit for extracting critical scenarios based on real traffic accident videos and reproducing the scenarios in simulation. The extracted scenarios are then used for safety assessment of the autonomous vehicles. Erdogan et al. [22] propose an architecture to enable test scenario generation, where test scenarios are first extracted from a video stream that contains real-world sensor data, and then is stored in a structured database cluster with scenario definitions and the corresponding measurements. A user interface is implemented and included in this architecture to customize and adapt the conditions for test scenario generation based on the aforementioned scenario database.

Deep learning has been actively used for critical test scenario identification through the studies that we surveyed. Ding et al. [26] train a generative model for generating safety-critical scenarios by sampling through the parameters and rewarding the risky scenarios. The generative model gets a higher reward when a riskier scenario is generated. Another study that uses reinforcement learning is reported by Karunakaran et al. [21] for automatically generating scenarios and optimizing the learning towards the worst-case scenarios with respect to the RSS safety metrics. A few other studies that employ deep learning techniques include Batsch et al. [8] using Gaussian Processes to train and optimize the parameter selection towards the most critical scenarios on the performance boundary, and Jenkins et al. [27] using a recurrent neural network to generate accident scenarios for testing the autonomous driving systems based on the in-vehicle and vehicle-to-infrastructure data generated from simulators. In a related application domain, Porres et al. [23] experiment with online supervised learning to train a generative model for searching and selecting critical scenarios for testing the autonomous maritime collision avoidance systems through operation.

Even though diverse techniques for extracting or generating critical scenarios based on real driving data have been studied, limitations can be observed and are described in these studies as well. A prerequisite of using such techniques is a data set that is comprehensive [19], whereas it is well known that collecting real driving data at scale is both time-consuming and expensive, but still does not guarantee to include all corner cases [21, 28]. As highlighted by Hallerbach et al. [16], the major drawback of using recorded data is the incompleteness of the data set, thus we have to understand how the data is acquired and how representative it is. The quality of the data can be affected by various factors such as the type of sensors used and how they are installed [4], the location where the data is collected, and the fact that rare-occurring situations are difficult to collect [26]. After all, we still have to understand how to extract and select scenarios given massive data collected [29].

In contrast, our work does not rely on collecting data from different sources, and thus is not subject to the quantity or diversity of the data set. Instead, we first analyse the function and the operational design domain (ODD) of the system based on the system specifications. ODD is a concept that defines the operational environment of the autonomous vehicles, and is used to derive test scenarios and safety assessment of autonomous driving systems [30]. Then we create an optimization model using the existing engineering tools to integrate the parameters, the objective functions, and a simulation platform that runs the scenarios as well as to record the results. The optimization model optimizes scenario generation towards the objective functions for critical scenarios using design of experimentation (DoE) [19]. DoE is a systematic approach for analysing the relation between input parameters and output values as well as how the effect (output) changes over variation of the conditions (parameter sets). The DoE generation in our work represents the selection of parameter values and creation of new test scenarios.

2.2.3 Search-based Approaches

The search-based approaches employ search algorithms to optimize critical scenarios from the operational design domain of the autonomous driving system. This approach typically requires execution of the scenarios in simulation, and an objective function that measures the criticality of the scenarios. The search process evolves based on the parameter space and the objective function value of the executed scenarios. Also, it usually limits the search to a certain number of iterations based on the testing budget and computational resources available. In fact, our approach falls into this category.

Klischat et al. [5] use evolutionary algorithms to optimize the drivable area of the vehicle to generate complex scenarios for testing the motion planning of the autonomous vehicles, while similar work is reported by Althoff et al. [31] to generate safety-critical scenarios for collision avoidance of autonomous vehicles by optimizing the drivable area as well. Besides, Buehler et al. [32] also employ evolutionary algorithms for generating critical scenarios for functional testing of an autonomous parking system. Specifically, genetic algorithms is a class of evolutionary algorithms and is commonly used for search-and-optimization problems. Gambi et al. [2] use genetic algorithms to evolve the generation of virtual road networks for testing the lane-keeping function, and Kluck et al. [7] propose an approach for test parameter optimization using genetic algorithms and have employed it for testing an autonomous emergency braking function.

The advantages of using a search-based approach for solving optimization of critical scenarios for testing of autonomous driving systems are prominent, since the selection of parameter values is rather difficult prior to test and covering the entire parameter space is costly [4]. In addition, this approach does not rely on collecting substantial driving data and is easy to implement. Still, some limitations are also stated in the existing studies, including that generated scenarios may not be realistic in the real-world traffic, simulation of the scenarios are often computationally expensive, and only low-dimensional scenarios can be handled effectively in optimization [26]. Thus, other studies such as Beglerovic et al. [33] is conducted to simulate and optimize test scenarios based on a light-weighted surrogate model instead of the real system, Feng et al. [20] establish a sophisticated model of relevant parameters, metrics, and searching process for critical scenario generation, and Hallerbach et al. [16] create a complete tool-chain for critical test scenario identification for autonomous driving systems.

We believe that the search-based approach can well compensate for the scarcity of sensor data and generate critical scenarios that can be used to substantiate test cases for autonomous driving systems. While most of the existing studies use either a simple implementation of the autonomous driving function based on engineering tools like MATLAB Simulink (e.g. Ponn et al. [9]), or publicly available driving components such as DeepDriving and Beam.AI by Gambi et al. [2, 6], for validating the approaches, their effectiveness on realistic autonomous driving functions is not demonstrated. Besides,

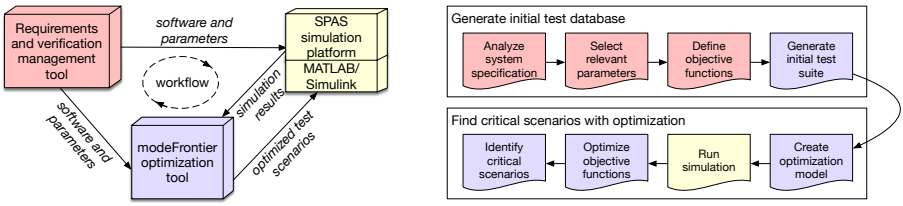


Fig. 1 Overview of the critical test scenario identification approach, consisting of interconnected tools (left) which are used in the workflow (right). The figure is a reprint from our previous work, and we refer to Song et al. [11] for more details of the approach.

many of them are also function-specific, which is relevant to a particular function or operational domain for, e.g. parking system [32], motion-planning [5], or highway scenario [33], and use only a limited set of scenarios for validation from e.g. NHTAS [15] and Euro NCAP [7]. Our approach is generic in that the tools involved are exchangeable and are not determined by the driving functions, so it can, in principle, be used for critical test scenario generation for any autonomous driving systems. We demonstrate the effectiveness of this approach by using two real autonomous driving systems from industry. As articulated by Hallerbach et al. [16] and Ding et al. [26], there exist very few studies that really provide a complete solution for critical test scenario identification which are generic to different autonomous driving systems. The major contribution of our work is to address such a gap and facilitate the testing of autonomous driving systems.

3 Research Context and Method

In this section, we describe the research context and method we use for the current study. We conduct the work on critical test scenario identification in the autonomous driving domain using the design science paradigm [34], in which we *formulate the problem* of critical test scenario identification by looking into the existing literature and industrial practices, then we *design the solution* by integrating the existing engineering tools and a workflow, and *validate* it in the industrial context using two real autonomous driving functions by collaborating with Volvo Cars. Lastly, we *infer the potential usage* of our approach for testing of autonomous driving systems in general.

3.1 Research Context

We base the current study on the critical test scenario identification approach that we presented briefly in our previous work [11] for testing autonomous driving systems. As shown in Figure 1, the approach integrates three different engineering tools and a workflow for optimization of critical test scenarios. The identified scenarios can then be used to substantiate test cases for testing the autonomous driving systems.

The three engineering tools are, (1) a requirement and verification management tool that is used for storing and analysing the system specifications;

(2) an internal simulation platform – SPAS, that is developed by Volvo Cars and used for early verification of the active safety and autonomous driving functions; and (3) a process automation and optimization tool – modeFrontier, which is a commercial tool for process and design optimization. The tools are exchangeable, meaning that we can substitute either of them with other similar tools to cope with different technical environment or autonomous driving functions under test. For example, we can use a different simulator like Carla [35] or AirSim [36] to simulate the scenario execution. Other simulators for autonomous driving are presented by Kang et al [37] and Rosique et al. [38], and Bhat et al. [39] discuss tools as well as methodologies for autonomous driving systems during different engineering stages.

The workflow includes two main phases, see Figure 1 (right). In the first phase, we start by analyzing the system specifications to understand the functionalities and the operational design domain of the system. The system specifications can be in different forms such as functional specifications, design documents, and related standards or regulations etc. Based on that, we select relevant parameters that constitute a driving scenario, and the value range as well as the distribution of each parameter. Also, we define objective functions that measure the criticality of the executed scenarios. Lastly, we generate an initial suite of scenarios by sampling through the parameter space based on the intended distribution and size of the initial test suite.

In the second phase, we create an optimization model in modeFrontier to integrate the selected parameters, objective functions, and a simulator. The optimization model is to optimize the scenario generation with respect to the objective functions and identify the most critical ones. It executes the scenarios in the initial test suites in simulation, and continuously explore the parameter space as well as evolving through the completed scenario simulation. We also configure the number of iterations for the optimization model in modeFrontier based on the testing budget and computational resources available.

We base our study on the collaboration with our industry partner – Volvo Cars, where they support us by providing access to the aforementioned tools and two autonomous driving functions, namely autonomous driving function and autonomous parking function. We replicate our approach on these two functions for identifying critical scenarios for testing such systems. By having access to the real industrial systems, we set up our approach and demonstrate the effectiveness of it for testing using realistic autonomous driving functions.

3.2 Research Method

We conduct the study under the design science paradigm [34] and have mainly conducted four steps as enumerated below. The problems of critical test scenario generation challenges are conceptualized in the industrial context [11]. We report our design of the critical test scenario identification approach (steps 1 and 2 below), and validate the approach using two autonomous driving functions from Volvo Cars (steps 3 and 4) and a potential usage context (step 4).

1. For each autonomous driving function, we analyse the system specifications and implementation through the requirement and verification management tool, to understand the functionality and the operational design domain of the system. We then identify the relevant parameters with the value range and distribution of each parameter, and define possible objective functions as well as the criticality thresholds for the autonomous driving function.
2. We explore the tool modeFrontier and create the optimization model there by integrating the parameters, objective functions, and the SPAS simulation platform. We design and include a MATLAB script in the optimization model to trigger the scenario simulation in SPAS and to record the execution result since the SPAS simulation platform is an external simulator to modeFrontier. Besides, we also configure the optimization algorithm, size of the initial test suite, and number of iterations for the optimization model.
3. We replicate the optimization model from the previous step by selecting a different optimization algorithm in modeFrontier. The purpose is to compare the results of two different optimization algorithms and show the generality of our approach to different optimization approaches. To clarify here, the contribution of the work is not to find the best algorithm, but to provide a complete approach for critical test scenario identification, and generate critical test scenarios for real autonomous driving functions from the industry.
4. We start the optimization process in modeFrontier, and debug the errors in case the process fails or suspends. After the optimization processes finish, we perform further analysis on the results in modeFrontier and export the findings in tables and figures, which we present in section IV and V. The identified critical scenarios are provided to the related engineering teams to test and investigate potential flaws in the autonomous driving functions.

4 CASE I: Autonomous Driving Function

This section describes the work and results for the first case that uses the proposed approach we present in Section 3. We aim to generate critical test scenarios for an early version of an autonomous driving function from Volvo Cars, which in this paper is referred to as the AD function (ADF).

4.1 Analyse System Specifications

ADF offers unsupervised in-lane driving in queue situations up to a specific speed limit $v_{\max}$, enabling the host vehicle to keep a safe distance to the preceding vehicle within the lane. The cardinal functionalities of ADF can be summarized as: (1) drive in lane, and (2) proactively adapt speed. These requirements specify that the host vehicle shall stay in lane and maintain a safe longitudinal and lateral distance to infrastructure, other vehicles and entities on the road. In addition, the host vehicle shall comfortably control speed to comply with the current speed limit.

Figure (2) shows snapshots in a scenario at different time steps. It is simulated in the SPAS platform and demonstrates the functionality of ADF. The host vehicle equipped with ADF is marked in red, while others are visualized in blue. At the beginning of the scenario, the host vehicle drives with a relative high speed compared to other vehicles. When driving around the bend, the host vehicle detects the front vehicle in the same lane, thus ADF drives the host vehicle to gradually decrease the speed. At the end of the scenario, the host vehicle manages to adapt its speed to follow the front vehicle.

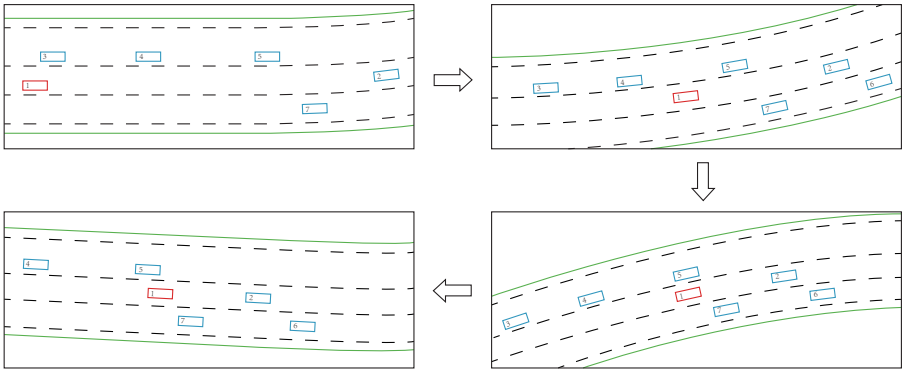


Fig. 2 A series of visualized scenes of the autonomous driving function (ADF)

4.2 Select Relevant Parameters

With the guidance of the requirement and verification management tool, the operational environment for ADF is characterized, where it covers all kinds of possible influential factors on the road, including traffic, vehicle status, environment, infrastructure, other road users, user input, and driver behavior.

This case study selects parameters in two domains: (1) movable entities, including dynamic behaviors of the host vehicle and other road users, and (2) road topology, including highway infrastructure and traffic conditions. In this section, we elaborate the parameter selection of movable entities as an example.

Road users include all kinds of vehicles, pedestrians and animals. Since the operational environment for ADF is on the highway, pedestrians and animals rarely appear. At the current stage, only vehicle models are taken into consideration for simplicity sake. The vehicles in the ADF function can be divided into three categories: host vehicle, lead vehicle, and other vehicles. We denote the vehicle set by $\mathcal{V} = \{V_h, V_l, V_{o1}, ..., V_{oN}\}, N \in \mathbb{Z}$, where V_h and V_l represent the host and lead vehicle, while $V_{o1}, ..., V_{oN}$ are other vehicles on the road. The simulation period of each scenario is set to T, unless a collision happens to trigger an early termination. In addition, parameters for each movable object are analyzed in four aspects to define a scenario: initial position, velocity, acceleration profile, and number of vehicles. In what follows, we

denote the position, velocity and acceleration of vehicle-i at time step-t by $\mathbf{p}_i(t) = [p_i^x(t), p_i^y(t)], v_i(t)$ and $a_i(t)$. Specifically, the velocity and acceleration are expressed as scalars since only the longitudinal information are of interests in the case.

4.2.1 Initial position

The initial locations of vehicles are selected, including the longitudinal and lateral positions along the road. Several constraints are defined to limit value selections. Each vehicle should keep a safe distance to others. The two-second rule, which a rule of thumb estimating safe distance at any speed for vehicles, is set as the baseline to deduce minimal initial longitudinal distance $|p_i^x(t_0) - p_j^x(t_0)| \geq d_{\min}^x, \forall i, j \in \mathcal{V}$. Also, to limit the scope of a scenario, we define a distance range between head-most vehicle and back-most vehicle in a simulation scenario, and its upper limit is denoted by $D_{\max}$. Regarding the lateral distance, the vehicle must leave a $d_{\min}^y = 1.5m$ space when considering regular road width for a freeway of $3.5m$ [40], i.e., $|p_i^y(t) - p_j^y(t)| \geq d_{\min}^y, \forall i, j \in \mathcal{V}$.

4.2.2 Velocity

ADF provides the nominal function only in situations when the host vehicle's velocity is lower than a specified level, i.e., $v_{V_h}(t) \leq v_{\max}$. In order to evaluate ADF's performance, the host vehicle should be able to detect, catch up and follow the lead vehicle. For this reason, the host vehicle should be in the right level of proximity to the lead vehicle, which allows the lead vehicle to be detected at the initial stage of a scenario. In addition, the initial velocity of the lead vehicle should follow $v_{V_l}(t_0) \leq v_{V_h}(t_0)$, otherwise the ADF will be deactivated and switched to human maneuver mode. Moreover, to enable driving properly on the highway, the minimum speed for all vehicles is set to a specified level $v_{\min}$. According to Abuelenin et al. [41], the traffic velocity on the road approximately complies to a normal distribution, and thus it is used to define the speed in scenario generation.

4.2.3 Acceleration

We restrict the acceleration of all vehicles on the road with $|a_i(t)| \leq 3m/s^2, i \in \mathcal{V}$ at each time step during the simulation, based on the real-world traffic data [42]. Besides, the longest acceleration period is restricted to 3 seconds. Acceleration values are sampled from a uniform distribution.

4.2.4 Number of vehicles

The number of vehicles on the road is jointly decided by $d_{\min}^x$, $D_{\max}$ and the length of a vehicle. Each scenario has $|\mathcal{V}| = N + 2$ vehicles. Considering the special scenario when there is only one lane in the road, to respect the safe distance, the vehicles on the simulated segment of the road cannot approximately exceed 10. The upper limit on vehicle number also benefits in reducing

the design space and accelerating the optimization process. Moreover, to make sure there are enough vehicles on the road to formulate a critical scenario, the minimal number of vehicles is set to 5. Thus, the number of vehicles defined in a scenario is given as $5 \leq |\mathcal{V}| \leq 10$.

4.3 Define Objective Functions

We define the objective function from two perspectives to extract critical scenarios, namely vehicle behavior and driver reaction. For vehicle behavior, criticality is defined as the closeness to an accident, of which TTC is used as an indicator. TTC is considered as an objective function which should be minimized. A threshold time (ΔT_{thres}) is set to distinguish critical scenarios from non-critical ones.

Regarding the driver reaction, ADF should ensure the driving comfort as much as possible. For this reason, jerk, measured as the rate of change in acceleration, is selected as another objective function to evaluate performance. When the jerk value is larger than $\pm 4m/s^3$, it would be not acceptable for most vehicles [43]. Thus, we try to maximize the absolute value of jerk to find the critical scenarios and set $|\dot{a}_{\text{thres}}| = 4m/s^3$ as a threshold for the corresponding scenario to be considered as critical.

Both TTC and jerk are evaluated at each scene and are updated by time frames. We select the extreme values of TTC and jerk within a simulation period to represent the criticality of a scenario. For this reason, the simulation will not be terminated prematurely if the value of TTC or jerk has exceeded the threshold, unless a collision is detected.

4.4 Generate Initial Test Suite

The DoE of this case study is formulated by the given parameters and constraints in Section 4.2. To generate a test suite, we apply MATLAB to translate the specifications in the requirement and verification management tool, and send the outputs to modeFrontier for DoE generation. ModeFrontier has different approaches for design space exploration, and in this study, the Space filler DoE is leveraged to guide the test scenario generation. This approach gives the most uniform filling of the design space, where the risk of missing corner cases can be mitigated. Regarding the sampling method, Latin Hypercube Sampling (LHS) is applied to generate random design configurations. In addition, an initial test set is generated as the input for the initial evaluation. The size of the initial tests is set to 50.

4.5 Create Optimization Models

The optimization process in modeFrontier follows the scheme as discussed in Figure 1. First, a test scenario in the SPAS platform is specified with initial parameters, including road, traffic, and vehicle information. These values are initialized in MATLAB and used for the SPAS simulation. At the end of the simulation, results of objective functions are sent to modeFrontier, where the

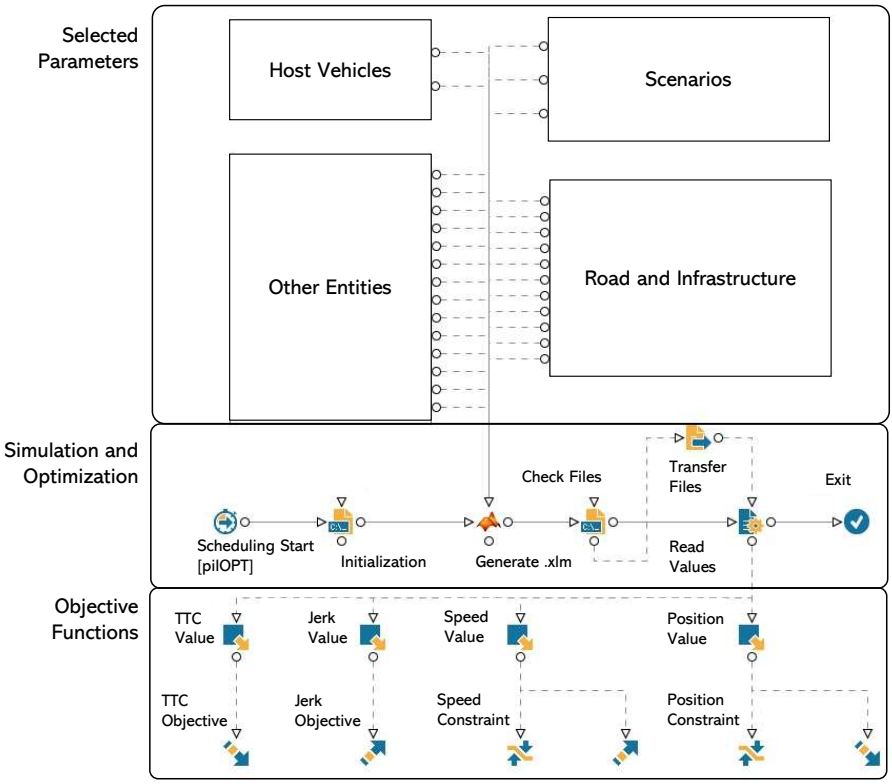


Fig. 3 modeFrontier optimization model for the ADF

data is parsed and analyzed. Subsequently, it generates a new scenario with distinct parameter values based on an optimization algorithm, which works as the next test scenario for the SPAS platform. After the specified number of iterations, the entire optimization process ends, with critical scenarios being analyzed and extracted. Figure 3 shows the modeFrontier optimization model in CASE I for the ADF. The block on top generates parameter values for a new scenario in modeFrontier by exploring the parameter space. The sub-blocks inside represent different aspects that need to be specified for a test scenario. Further, the values are transferred into MATLAB in the middle block, where the SPAS simulation is performed. In the end, the blocks in the bottom are used to determine optimization objectives. It carries out optimization based on the history simulation results.

Two optimization algorithms, namely Multi-Objective Simulated Annealing (MOSA) and pilOPT, are selected to perform optimization, respectively. The optimization models for algorithms are created separately, and Figure 3 is an example of the model how it looks. The results and performance are compared. pilOPT is an in-house developed algorithm in modeFrontier, which can effectively handle multi-strategy searching problem and minimizing the

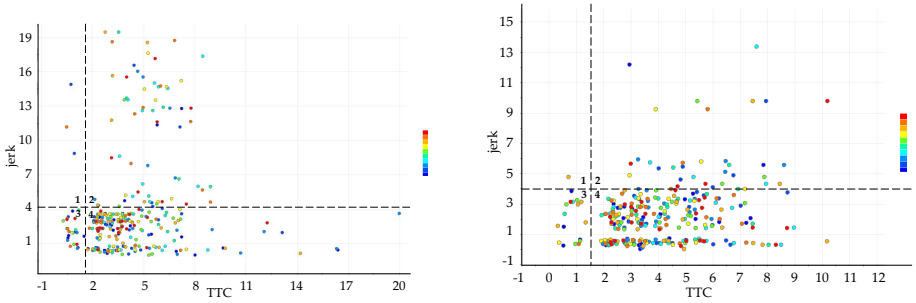
amount of time and computational resources required[1]. It combines the advantages of local and global search algorithms to get the optimum solutions. In contrast, MOSA is a heuristic searching algorithm, which is regarded as the benchmark algorithm to be compared with pilOPT.

4.6 Run Simulation and Optimization

After creating the optimization model and setting up the simulation environment, the optimization process is started in modeFrontier. The number of optimization iterations is determined mainly based on computational resources available and is set to 300 in this case. Higher intensive grid search can be performed with more powerful computing resources, although the number of available software licenses of commercial tools may also be limiting. After running the simulation and optimization, the results are saved for further analysis.

4.7 Identify Critical Scenarios

Figure 4 shows the simulation results for each test scenario during optimization by MOSA and pilOPT, and the relationship between TTC and jerk value are plotted. In the figure, sequence information of optimization iteration is represented by the change of colors (i.e. blue the earlier iterations, and towards red means the later iterations).



(a) MOSA (b) pilOPT

Fig. 4 modeFrontier optimization results from the autonomous driving function (ADF) with (a) MOSA and (b) pilOPT algorithms. Each dot represents a test scenario. The lines depict the borders of the critical scenarios, while scale labels are left out for confidentiality reasons. Colors indicate the sequence number of the iterated simulations.

For the MOSA algorithm, we observed that there is a clear boundary among test scenarios with very low jerk values. In addition, another boundary exists to partition test scenarios whether their TTC values exceed ΔT_{thres} or not. For test points with low jerk values, the TTC values are mostly over ΔT_{thres}, indicating that in those test scenarios, the host vehicle does not experience

[1] https://engineering.esteco.com/modefrontier/

the sharp acceleration process, thus being obvious safe cases. According to the definition of critical scenarios, if a test scenario has $|\dot{a}_i(t)| > |\dot{a}_{\text{thres}}|, i \in \mathcal{V}$, it is considered a critical scenario. Therefore, quadrants 1, 2 and 3 in Figure 4 (a) are critical scenarios. In the figure, points with different colors are randomly distributed, which means no distinct region feature and difference emerge with the optimization process.

In Figure 4 (b), there is no obvious boundary on either axis, but the figure is divided into two groups. Test scenarios in the first group, locating on the upper part of the figure, has remarkable high values of jerk. The number of critical scenarios are summarized in Table 1 to compare the difference between MOSA and pilOPT. The amount of scenarios caused by violating TTC and jerk constraints are not summing up the total amount, since there are some scenarios, where both criteria are critical. We conclude that, in this case study, pilOPT has a better performance in finding critical scenarios compared to MOSA, especially with respect to jerk. This is however not our primary focus here, and optimization algorithms have to be further explored.

Table 1 number of critical test scenarios with respect to the objective functions

	MOSA	pilOPT
jerk	33	87
TTC	18	28
total	45	95

5 CASE II: Autonomous Parking Function

In this section, we describe the work and result for the second case that uses the proposed approach we present in Section 3 to generate critical test scenarios for an early version of an autonomous parking function from Volvo Cars.

5.1 Analyse System Specifications

The Autonomous Parking Function (APF) aims to detect and park the vehicle into a feasible parking slot between two stationary vehicles autonomously, where a driver is not required. The function should be able to park the vehicle in both parallel and perpendicular slots, either reversely or forwardly.

The case study APF version supports only the rearward parking in parallel slots (i.e. parking slots that are parallel to the road direction) where the parking manoeuvre is performed mainly in three steps. First, the vehicle drives at a low speed and passively scans the empty slots using the ultrasonic sensors that are deployed on the front side of the vehicle. Second, the vehicle identifies the target slot and performs motion planning to park the vehicle in it without colliding the vehicles around. Lastly, the vehicle starts to actuate the rearward parking manoeuvre by controlling the steering wheel, propulsion, shifting

gear and braking, and follows the trajectory that is computed in the previous step. When the vehicle reaches the final position that has been planned, it deactivates the parking function and sets a brake torque to stop the vehicle.

Figure 5 illustrates the function and operational scenes of APF. Specifically, five vehicles (numbered 2–6 and in blue) are parked parallel to the road direction and remain stationary. The host vehicle (i.e. numbered 1 and in red – the vehicle with APF installed, also known as ego vehicle) first drives from the left and passes the stationary vehicles, it scans and identifies an empty slot between the rear vehicle (4, referred as V_r) and the front vehicle (5, referred as V_f). Then APF reversely parks the host vehicle into the slot without colliding with other vehicles and stops at a position that is feasible subject to the physical constraints such as the slot length and the maximum steering angle the vehicle can complete. In an optimal situation, the host vehicle ends at the center of the parking slot with a sufficient distance to both V_r and V_f, and the vehicle stands parallel to the parking slot.

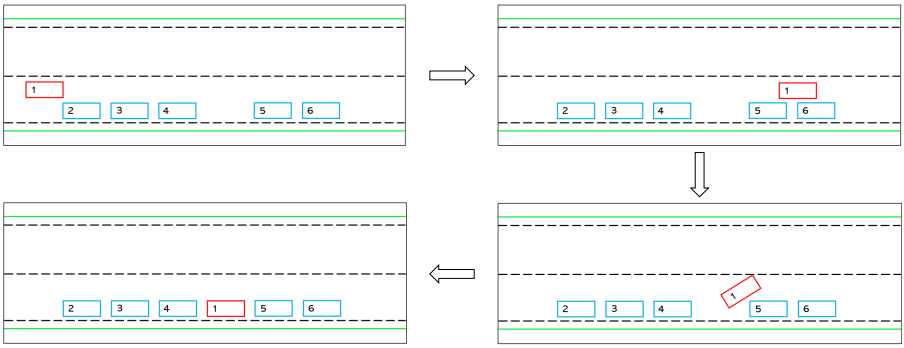


Fig. 5 A series of visualized scenes of the autonomous parking function (APF)

5.2 Select Relevant Parameters

After analysing the system specifications and current design of APF by using the requirement and verification management tool, we identify two parameters that are relevant for constituting a test scenario for APF, namely slot length and angle of the stationary vehicle.

Slot length describes the actual length of the parking slot and is the primary parameter that determines whether a parking slot is feasible or not. Buehler et al. [32] adopted both the slot length and slot width as the two parameters that depict the parking space, and use them to explore critical test scenarios for an autonomous parking system. Given the current design and the operational design domain of APF, we presume a sufficient slot width in the current

study, and thus it is not a valid parameter to optimize for critical scenario identification.

Based on the setup in Figure 5, slot length can be quantified and adjusted by changing the position of either V_r or V_f on the coordinate system of the simulation platform. Herein we select the position of V_f (referred as PoV_f) as the derived parameter for optimizing slot length. The value range of slot length includes both a lower bound – the minimum slot length APF should handle without colliding the stationary vehicles, and a upper bound – an adequate slot length that APF manages while keeping a sufficient distance to the stationary vehicles and a considerable yaw angle to the parking slot. Due to confidentiality concerns, we do not report the specific values here.

The angle of the stationary vehicle represents the yaw angle rate of the stationary vehicles (i.e. V_r and V_f), and is a parameter that determines the shape of the parking slot as well as the motion planning of APF. Since we here focus on rearward parking, and the slot length is generally larger than the standard parking slot length, we consider the yaw angle of V_f having most impact (referred as AnV_f). The value range for this parameter is set to $[-3°, 3°]$ according to the ISO-16787 standard [44] which is a standard specification for testing autonomous parking functions and is up to each nation to implement. According to this standard, a vehicle should remain within $[-3°, 3°]$ to the central line of the parking slot after completing the parking maneuver. Thereby, we take this standard specification as a reference for setting AnV_f.

5.3 Define Objective Functions

The basic acceptance criterion for a parking scenario, according to ISO-16787 standard [44], includes that the host vehicle should keep a minimal 0.3 m distance to other vehicles around and standstill with a yaw angle within $\pm 3°$ to the central line of the parking slot. We consider scenarios that are beyond these two criteria critical and should be identified as critical test scenarios for APF. Based on these two criteria and the setup shown in Figure 5, the distance to V_r (referred as DtV_r) and V_f (referred as DtV_f) should be minimized through optimization to identify the scenarios with less than 0.3 m distance to either of them. In addition, the yaw angle of the host vehicle (referred as AnV_h) needs to be optimized to identify the scenarios that end with an angle beyond $\pm 3°$.

Nevertheless, we cannot have all aforementioned objective functions in one optimization model due to the natural conflicts between them. For example, minimizing DtV_r is essentially maximizing DtV_f since these two vehicles are located on the two end sides of the parking slot. Thus, these two objective functions have to be separated into two different optimization models. In addition, we cannot maximize and minimize AnV_h at the same time to identify critical test scenarios that are greater than $3°$ and those lower than $-3°$. Thus, these two objective functions have to be separated in two different optimization models as well. The resulting set of objectives are four, hence lead to four optimization models with two objective functions each as shown in Table 2.

Table 2 modeFrontier optimization models and corresponding objective functions for APF

Model	Objective function 1	Objective function 2
1	minimize DtV_r	maximize AnV_h
2	minimize DtV_r	minimize AnV_h
3	minimize DtV_f	maximize AnV_h
4	minimize DtV_f	minimize AnV_h

5.4 Generate Initial Test Suite

We generate an initial set of test scenarios in modeFrontier to enable further optimization of the parameters towards the most critical scenarios. Based on the two parameters we select (i.e. PoV_f and AnV_f) and the objective functions we define, we first compute the size of the initial test suite using the rule of thumb for DoE [45], as shown in Equation 1, where N_{par} is the number of parameters and N_{obj} is the number of objective functions. As for APF, the size of the initial test suite is eight, given two parameters are selected and two objective functions are defined for each optimization model.

$$Initial\ suite\ size = 2 * N_{par} * N_{obj} \tag{1}$$

Next, an initial suite of test scenarios can be generated by sampling through the parameters based on the intended distribution. However, the realistic distribution for both DtV_r and AnV_h are unclear, and are difficult to model or predict, so we generate the initial set of test scenarios with the Latin Hypercube Sampling (LHS) strategy and a uniform distribution. In LHS, the parameter space is divided into equal parts with respect to the target sampling size (i.e. the size of the initial test suite) and the sampling position is randomly chosen according to the parameter distribution [8]. LHS is considered superior to other sampling approaches like random sampling and ensures that the entire parameter space is covered as evenly as possible [8]. As there is no such real distribution for the selected parameters provided, we also use the uniform distribution to assure every parameter value interval is equally likely.

5.5 Create Optimization Models

We create the optimization models in modeFrontier by integrating the selected parameters, the objective functions, and the SPAS simulation platform. Similar to Figure 3, the parameters are defined as inputs to the optimization model and are used to generate scenarios for simulation. An initial set of values for the parameters are sampled preliminary with LHS and are considered the initial test suite to enable further optimization of critical test scenarios. The objective functions are the output of the optimization model and are optimized based on the parameter space and the completed scenario simulation.

We configure the number of optimization iterations to 80 based on the testing budget and computational resources available. In other words, the optimization model first runs the initial test scenarios (i.e. 8 scenarios) in the SPAS simulation platform and track the value of the objective functions. Then the

optimization model optimizes the selection of parameters for another 72 iterations based on the completed simulation results. Parallelization of optimization is possible in modeFrontier, given enough computational resources are available. Lastly, we select the optimization algorithm in modeFrontier based on our previous experience (i.e., Section 4) where pilOPT was used. In addition, we also replicate two optimization models (1 and 3 in Table 2) using MOSA to compare two different optimization algorithms and demonstrate the generality of our approach in using different optimization strategies. Thus, we create six optimization models in total, as shown in Table 3. To clarify again, we do not aim to find the best optimization algorithm in this study, but to integrate the entire tool-chain and a workflow for critical test scenario identification.

Table 3 modeFrontier optimization models and results for APF. By results, we mean the number of critical test scenarios identified with respect to the objective functions.

Model	Objective function 1	Objective function 2	Algorithm	Iteration	Result
1	minimize DtV_r	maximize AnV_h	pilOPT	80	41
2	minimize DtV_r	maximize AnV_h	MOSA	80	40
3	minimize DtV_r	minimize AnV_h	pilOPT	80	35
4	minimize DtV_f	maximize AnV_h	pilOPT	80	40
5	minimize DtV_f	maximize AnV_h	MOSA	80	29
6	minimize DtV_f	minimize AnV_h	pilOPT	80	30

5.6 Run Simulation and Optimization

We start the optimization models in modeFrontier and the optimization process runs automatically. For each optimization iteration, the simulation result is recorded and optimized with respect to the objective functions. After all iterations completed, the optimization process terminates and full results are saved. Since scenarios are simulated in the SPAS simulation platform and are triggered from modeFrontier, we have set a maximum time for a single simulation session to avoid suspending the entire optimization process.

5.7 Identify Critical Scenarios

The result of the optimization models can be visualized in modeFroniter using different charts or statistical analysis tools, and be exported in many different formats. As mentioned earlier, we create six optimization models for APF and each model consists 80 evaluation iterations. By filtering the results with the criticality thresholds we define, the optimization models have identified 29 to 41 critical scenarios, as indicated by the last column (i.e. Result) in Table 3.

The critical scenarios are identified exclusively on one of the objective functions – AnV_h – and no critical scenarios identified for both DtV_r and DtV_f. As shown in Figure 6 (a) – the result of optimization model 1 from Table 3 for minimizing DtV_r and maximizing AnV_h using pilOPT, no critical scenario (i.e. $< 0.3\ m$) is identified in the DtV_r dimension as all scenarios resulted in a sufficiently large distance for DtV_r, which is considered as safe according to the

industrial standard. This indicates the the early implementation of the function we used is conservative on the distance to other vehicles. In contrast, 41 critical scenarios are identified based on the AnV_h which are greater than 3° to the central line of the parking slot.

Furthermore, Figure 6 (b) shows the correlation between parameter AnV_f and the objective AnV_h. The result indicates that AnV_f does not have a general effect on AnV_h and it is randomly distributed regardless the value of AnV_f. In contrast, an explicit pattern is drawn on PoV_f and AnV_h in Figure 6 (c), in which AnV_h keeps increasing when PoV_f decreases. When PoV_f is lower than a specific value, AnV_h is over the criticality threshold 3° and scenarios are identified as critical scenarios.

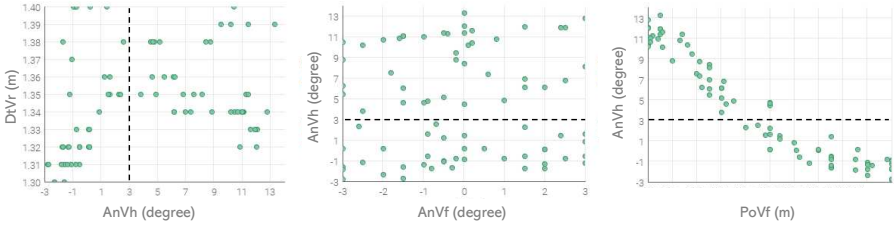


Fig. 6 Result of minimizing DtV_r and maximizing AnV_h using pilOPT. The dash line in the sub-figures is the criticality threshold for AnV_h and the dots are the scenarios executed in the simulation. Scenarios on the right side of the dash line in sub-figure (a) and above the dash line in sub-figure (b) and (c) are the critical scenarios identified on with AnV_h larger than 3°. The scale of PoV_f in sub-figure (c) is removed for confidentiality reasons.

The results are consistent when using other optimization models with different combination of objective functions. We identify critical scenarios on AnV_h only and the visualized results clearly indicate that AnV_h gets larger and exceeds the criticality threshold when PoV_f declines. The observations suggest that adapting the slot length and angle of the stationary vehicle does not generate critical test scenarios for APF with respect to the distance to the stationary vehicles. However, both of them lead to critical test scenarios where the angle of the host vehicle exceeds 3°. A clear trend is observed on the slot length that, smaller slot length generally increases the angle of the host vehicle, which means a bad orientation to the parking slot after the parking maneuver is done.

Lastly, pilOPT generally identify more critical scenarios than MOSA for APF in this case, although there are no significant differences between them consistently. For the optimization models that minimize DtV_r and maximize AnV_h, pilOPT identifies 41 critical scenarios and MOSA identifies 40. As for the models that minimize DtV_f and maximize AnV_h, pilOPT identifies 40

critical scenarios where MOSA identifies 29. An observation is that pilOPT performs better than MOSA according to the results, while further comparison between these two algorithms are required. Since we do not aim to address the best optimization algorithm in the current study, we have demonstrated that our approach is effective in identifying critical test scenarios and is general to different optimization algorithms or strategies as well.

6 Discussion

In this paper, we extend an approach for critical test scenario identification for autonomous driving and have used it for testing real autonomous driving systems. We argue that testing all possible driving scenarios in real road traffic is impractical, as it is expensive, time-consuming, and may still not cover all the rare-occurring traffic situations [15, 21], in contrast to Kalra et al. who claim that millions or even billions of miles of driving test are required to demonstrate the reliability of an autonomous vehicle [46]. Instead, testing of autonomous driving functions must be based on a feasible number of test scenarios and focus on the most critical ones [4, 7]. Using critical scenario identification and simulation is considered a good alternative to address the aforementioned gaps, and enable testing of autonomous driving functions in a more efficient way [7, 25, 47].

In our approach, we integrate the existing engineering tools and a workflow as a complete solution for critical test scenario identification. In contrast, existing studies mostly present a partial solution for critical scenario identification, and barely provide a complete tool-chain [16]. The application of a partial solution in practice may require additional work to integrate such an approach with the missing components. We integrate different tools and a workflow into a systematic approach, which is easy to use. The proposed approach relies on optimization of the parameter selection and simulation of the scenarios. As the tools involved are exchangeable, the approach is flexible and generic for testing different autonomous driving functions that are not subject to any specific tools, techniques, or type of sensors employed in the function or simulation.

We demonstrate the effectiveness of our approach for critical test scenario identification, using real autonomous driving functions in both high-speed and low-speed maneuvering domains. This is different from the most common approach, for validating proposed solutions for critical scenario identification in existing studies, which use a simple implementation of the autonomous driving function, or publicly available driving components like DeepDriving [2]. Besides, many studies demonstrate the effectiveness of their approaches based on limited settings, such as a pedestrian step-out scenario [8] and certain scenarios from Carla Scenario Runner Library [26]. Even though the potential of such approaches might be extended, the connection to real autonomous driving functions and to find critical scenarios in general is not explicitly provided.

The two cases we present in Sections 4 and 5 include the actual work we implement and the results achieved on real autonomous driving functions using

the proposed approach. While the results are generally effective in finding the critical test scenarios for the given autonomous driving systems, we would like to stress that they are merely early version of the autonomous driving systems. Thus, the results are subject to the current design and specifications of the systems when conducting the study. The main purpose here is to demonstrate the industrial relevance and applicability of the approach in practice.

Future improvement and extension of our approach regarding its design and implementation are multi-fold, including, e.g. scenario composition, parameter distribution, and optimization algorithms. First, the composition and representation of scenarios can be improved, to include different driver behavior models and enable definition of complex spatio-temporal interactions between different entities within the driving maneuver. As highlighted by Feng et al. [20], existing studies mostly handle only low-dimensional scenarios, whereas the actual operational design domain for the autonomous driving functions is much more complicated. OpenDrive and OpenScenario, as used by Zhang et al. [15] and Erdogan et al. [22], to define static and dynamic elements in a full driving scenario in a structured way are good references to explore.

Second, realistic distribution of the relevant parameters selected should be investigated to improve the realism of the scenarios and realistic occurrence of the scenarios. As articulated by Batsch et al. [8], scenario-based testing sampling requires a true distribution of the parameters. A shift in the distribution may impact the relevance and potential damage of the scenarios [19], thus distribution of parameters are important and need to be identified [26]. Different sampling approaches such as adaptive sampling [48], importance sampling [20], or modeling the distribution are few candidates to be further studied.

Thirdly, we also propose to evaluate different optimization algorithms to best fit the generation of critical test scenarios for different autonomous driving systems, and using parallelization to improve the efficiency of the simulation and optimization [23]. They are good directions to be sorted out in future research yet not the goals in the current study. Especially that parallelization is already a feasible option in optimization tools like modeFrontier, it's more about the computational resources that can be allocated count. Our primary focus in this work is to establish a complete approach for critical test scenario identification for autonomous driving, and to demonstrate the effectiveness of such an approach for realistic testing of autonomous driving systems. As a preliminary step, tools and a workflow are integrated, and critical test scenarios are generated for real autonomous driving systems from industry. Thus, it constitutes a basis for further exploration and refinement of the approach in practice.

Given the enormous challenges of testing autonomous driving systems we face [49, 50], the importance of using simulation and critical test scenario generation increases steeply [27]. Further, as stated by Beglerovic et al., selection of relevant parameters, objective functions, and appropriate evaluation criteria is a non-trivial task since each of them comes with its own challenges, and the quality of critical test scenario generation are highly depending on them [33].

Despite that sub-components within our approach can be further expanded and improved, we believe our work is worth the efforts and having a huge potential in the future in ensuring the safety and reliability of autonomous vehicles, particularly since very few studies have been reported for presenting a complete solution for critical test scenario identification that is general for different autonomous driving systems, according to Hallerbach et al. [16].

7 Conclusion

Safety and reliability are indispensable properties for autonomous vehicles, yet there is no common standard way to test the autonomous driving functions systematically and efficiently. Conventional requirements-driven testing approaches are impeded due to uncertainty of the operational environment and complexity of the driving scenarios. Thereby, identifying the most critical scenarios for testing the autonomous driving systems is developed.

We establish a complete approach with integrated tools and a workflow in this study, to enable the exploration of critical test scenarios and facilitate the testing of autonomous driving systems. As a pilot study, we implement the approach on two autonomous driving systems from industry by partnering with Volvo Cars, and the results suggest that our approach is effective in identifying critical test scenarios. The identified scenarios can be used to substantiate test cases for autonomous driving systems either in simulation, or in real world.

Future extension of the approach aims to improve the scenario representation, distribution of the parameters, and compare the effectiveness of different optimization algorithms. Eventually, the study provides a feasible and complete tool-chain for critical test scenario identification for autonomous driving, and a basis for building sub-components further upon. Given the widespread attention on autonomous driving and the corresponding challenges for testing the enabling functions, we shed light on testing of different autonomous driving systems in an efficient and effective way.

8 Acknowledgement

This work was supported in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP). Thanks to our colleagues in SERG and Volvo Cars for their review of earlier versions of the manuscript.

References

[1] Song, Q., Engström, E., Runeson, P.: Concepts in testing of autonomous systems: Academic literature and industry practice. In: IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), pp. 74–81 (2021)

[2] Gambi, A., Mueller, M., Fraser, G.: Automatically testing self-driving cars with search-based procedural content generation. In: Proceedings of the

28th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 318–328 (2019)

[3] Yang, B., Cao, X., Li, X., Yuen, C., Qian, L.: Lessons learned from accident of autonomous vehicle testing: An edge learning-aided offloading framework. IEEE Wireless Communications Letters **9**(8), 1182–1186 (2020)

[4] Ponn, T., Gnandt, C., Diermeyer, F.: An optimization-based method to identify relevant scenarios for type approval of automated vehicles. In: Proceedings of the ESV—International Technical Conference on the Enhanced Safety of Vehicles, Eindhoven, The Netherlands, pp. 10–13 (2019)

[5] Klischat, M., Althoff, M.: Generating critical test scenarios for automated vehicles with evolutionary algorithms. In: IEEE Intelligent Vehicles Symposium (IV), pp. 2352–2358 (2019). IEEE

[6] Gambi, A., Huynh, T., Fraser, G.: Generating effective test cases for self-driving cars from police reports. In: Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 257–267 (2019)

[7] Klück, F., Zimmermann, M., Wotawa, F., Nica, M.: Genetic algorithm-based test parameter optimization for ADAS system testing. In: IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), pp. 418–425 (2019). IEEE

[8] Batsch, F., Daneshkhah, A., Palade, V., Cheah, M.: Scenario optimisation and sensitivity analysis for safe automated driving using gaussian processes. Applied Sciences **11**(2), 775 (2021)

[9] Ponn, T., Breitfuß, M., Yu, X., Diermeyer, F.: Identification of challenging highway-scenarios for the safety validation of automated vehicles based on real driving data. In: 15th International Conference on Ecological Vehicles and Renewable Energies (EVER), pp. 1–10 (2020). IEEE

[10] Iqbal, M., Han, J.C., Zhou, Z.Q., Towey, D.: Enhancing euro NCAP standards with metamorphic testing for verification of advanced driver-assistance systems. In: IEEE/ACM 6th International Workshop on Metamorphic Testing (MET), pp. 37–41 (2021). IEEE

[11] Song, Q., Tan, K., Runeson, P., Persson, S.: An industrial workbench for test scenario identification in autonomous driving software. In: IEEE International Conference on Artificial Intelligence Testing (AITest), pp. 81–82 (2021). IEEE Computer Society

[12] Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., Maurer, M.: Defining and substantiating the terms scene, situation, and scenario for automated driving. In: IEEE 18th International Conference on Intelligent Transportation Systems, pp. 982–988 (2015). IEEE

[13] Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles. In: IEEE Intelligent Vehicles Symposium (IV), pp. 1821–1827 (2018). IEEE

[14] Bagschik, G., Menzel, T., Maurer, M.: Ontology based scene creation for the development of automated vehicles. In: IEEE Intelligent Vehicles Symposium (IV), pp. 1813–1820 (2018). IEEE

[15] Xinxin, Z., Fei, L., Xiangbin, W.: CSG: Critical scenario generation from real traffic accidents. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 1330–1336. IEEE

[16] Hallerbach, S., Xia, Y., Eberle, U., Koester, F.: 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)

[17] Mahmud, S.S., Ferreira, L., Hoque, M.S., Tavassoli, A.: Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. IATSS research **41**(4), 153–163 (2017)

[18] Laureshyn, A., Johnsson, C., De Ceunynck, T., Svensson, Å., de Goede, M., Saunier, N., Włodarek, P., van der Horst, R., Daniels, S.: Review of current study methods for VRU safety. appendix 6–scoping review: surrogate measures of safety in site-based road traffic observations: Deliverable 2.1–part 4. (2016)

[19] Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE access **8**, 87456–87477 (2020)

[20] Feng, S., Feng, Y., Yu, C., Zhang, Y., Liu, H.X.: Testing scenario library generation for connected and automated vehicles, part i: Methodology. IEEE Transactions on Intelligent Transportation Systems **22**(3), 1573–1582 (2020)

[21] Karunakaran, D., Worrall, S., Nebot, E.: Efficient statistical validation with edge cases to evaluate highly automated vehicles. In: IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–8 (2020). IEEE

[22] Erdogan, A., Kaplan, E., Leitner, A., Nager, M.: Parametrized end-to-end

scenario generation architecture for autonomous vehicles. In: 6th International Conference on Control Engineering & Information Technology (CEIT), pp. 1–6 (2018). IEEE

[23] Porres, I., Azimi, S., Lilius, J.: Scenario-based testing of a ship collision avoidance system. In: 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 545–552 (2020). IEEE

[24] Zhang, X., Tao, J., Tan, K., Törngren, M., Sánchez, J.M.G., Ramli, M.R., Tao, X., Gyllenhammar, M., Wotawa, F., Mohan, N., et al.: Finding critical scenarios for automated driving systems: A systematic literature review. arXiv preprint arXiv:2110.08664 (2021)

[25] Rajabli, N., Flammini, F., Nardone, R., Vittorini, V.: Software verification and validation of safe autonomous cars: A systematic literature review. IEEE Access, 4797–4819 (2020)

[26] Ding, W., Chen, B., Xu, M., Zhao, D.: Learning to collide: An adaptive safety-critical scenarios generating method. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2243–2250 (2020). IEEE

[27] Jenkins, I.R., Gee, L.O., Knauss, A., Yin, H., Schroeder, J.: Accident scenario generation with recurrent neural networks. In: 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 3340–3345 (2018). IEEE

[28] Priisalu, M., Pirinen, A., Paduraru, C., Sminchisescu, C.: Generating scenarios with diverse pedestrian behaviors for autonomous vehicle testing. In: 5th Annual Conference on Robot Learning (2021)

[29] Klitzke, L., Koch, C., Haja, A., Köster, F.: Real-world test drive vehicle data management system for validation of automated driving systems. In: VEHITS, pp. 171–180 (2019)

[30] Gyllenhammar, M., Johansson, R., Warg, F., Chen, D., Heyn, H.-M., Sanfridson, M., Söderberg, J., Thorsen, A., Ursing, S.: Towards an operational design domain that supports the safety argumentation of an automated driving system. In: 10th European Congress on Embedded Real Time Systems (ERTS) (2020)

[31] Althoff, M., Lutz, S.: Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In: IEEE Intelligent Vehicles Symposium (IV), pp. 1326–1333 (2018). IEEE

[32] Buehler, O., Wegener, J.: Evolutionary functional testing of an automated

parking system. In: Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT'03) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS'03), Florida, USA (2003)

[33] Beglerovic, H., Stolz, M., Horn, M.: Testing of autonomous vehicles using surrogate models and stochastic optimization. In: IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6 (2017). IEEE

[34] Runeson, P., Engström, E., Storey, M.-A.: In: Felderer, M., Travassos, G.H. (eds.) The Design Science Paradigm as a Frame for Empirical Software Engineering, pp. 127–147. Springer, Cham (2020)

[35] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on Robot Learning, pp. 1–16 (2017). PMLR

[36] Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics, pp. 621–635 (2018). Springer

[37] Kang, Y., Yin, H., Berger, C.: Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments. IEEE Transactions on Intelligent Vehicles **4**(2), 171–185 (2019)

[38] Rosique, F., Navarro, P.J., Fernández, C., Padilla, A.: A systematic review of perception system and simulators for autonomous vehicles research. Sensors **19**(3), 648 (2019)

[39] Bhat, A., Aoki, S., Rajkumar, R.: Tools and methodologies for autonomous driving systems. Proceedings of the IEEE **106**(9), 1700–1716 (2018)

[40] Mouhagir, H., Talj, R., Cherfaoui, V., Aioun, F., Guillemard, F.: Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation. In: IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1114–1119 (2016). IEEE

[41] Abuelenin, S.M., Abul-Magd, A.Y.: Empirical study of traffic velocity distribution and its effect on VANETs connectivity. In: International Conference on Connected Vehicles and Expo (ICCVE), pp. 391–395 (2014). IEEE

[42] Bokare, P., Maurya, A.: Acceleration-deceleration behaviour of various vehicle types. Transportation research procedia **25**, 4733–4749 (2017)

[43] Bellem, H., Schönenberg, T., Krems, J.F., Schrauf, M.: Objective metrics of comfort: developing a driving style for highly automated vehicles. Transportation research part F: traffic psychology and behaviour **41**, 45–54 (2016)

[44] International Organization for Standardization: ISO 16787. Intelligent Transport Systems – Assisted Parking System (APS) – Performance Requirements and Test Procedures. Geneva, Switzerland (2017). International Organization for Standardization

[45] Tan, K.: Building verification database and extracting critical scenarios for self-driving car testing on virtual platform. Master's thesis, KTH, School of Industrial Engineering and Management (ITM) (2019)

[46] Kalra, N., Paddock, S.M.: 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)

[47] Mauritz, M., Howar, F., Rausch, A.: Assuring the safety of advanced driver assistance systems through a combination of simulation and runtime monitoring. In: International Symposium on Leveraging Applications of Formal Methods, pp. 672–687 (2016). Springer

[48] Mullins, G.E., Stankiewicz, P.G., Gupta, S.K.: Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1443–1450 (2017). IEEE

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

[50] Knauss, A., Schröder, J., Berger, C., Eriksson, H.: Paving the roadway for safety of automated vehicles: An empirical study on testing challenges. In: IEEE Intelligent Vehicles Symposium (IV), pp. 1873–1880 (2017). IEEE