Software Quality Journal (2023) 31:441–469
https://doi.org/10.1007/s11219-022-09604-2

Critical scenario identification for realistic testing of autonomous driving systems

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

Accepted: 2 October 2022 / Published online: 3 December 2022

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 since the potential test scenarios in real-world traffic are infinite, and covering large shares of them in the test is impractical. Thus, critical ones have to be prioritized. We have developed an approach for critical test scenario identification and in this study, we implement the approach and validate it on two real autonomous driving systems from industry by integrating it into their tool-chain. Our main contribution in this work is the demonstration and validation of our approach for critical scenario identification for testing real autonomous driving systems.

Keywords Critical scenario identification · Autonomous driving · Software testing · Test scenario generation

1 Introduction

While autonomous driving is expected to improve traffic capacity and reduce road accidents, testing of autonomous driving systems is a prerequisite to validate the reliability and safety of such systems (Song, Engström, et al., 2021). Inadequate or ineffective testing

✉ Qunying Song
qunying.song@cs.lth.se

Kaige Tan
kaiget@kth.se

Per Runeson
per.runeson@cs.lth.se

Stefan Persson
stefan.persson@volvocars.com

[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

could fail to discover potential defects and misbehavior in the systems and lead to severe accidents in the road traffic (Gambi, Mueller et al., 2019). The fatal accident caused by Uber's autonomous vehicle is an example where a pedestrian who walked her bicycle across the road was hit by the vehicle in Arizona, USA, in 2018 (Yang et al., 2020).

Current approaches for testing autonomous driving systems that rely on substantial real-world driving, or collecting real driving data at scale, are considered both inefficient and ineffective. They take an unpractical amount of time to complete and may still not cover rare traffic situations (Ponn et al., 2019), while the regular road traffic is considered non-critical most of the time (Klischat and Althoff, 2019). New approaches for testing autonomous driving systems based on critical scenario identification are increasingly demanded (Gambi, Mueller et al., 2019; Gambi, Huynh et al., 2019). We refer to *critical scenarios* here as scenarios that can lead to a collision or near-collision consequence or situation, and are of interest for testing autonomous vehicles.

Nevertheless, existing studies mostly present parts of an end-to-end solution for critical test scenario identification, for example, focusing on either simulation or optimization tasks of critical driving scenarios (Klück et al., 2019; Batsch et al., 2021). 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 highway scenarios (Ponn et al., 2020). Therefore, the feasibility of such approaches for testing different autonomous driving functions is unclear. In addition, previous studies tend not to validate their approaches on real driving functions from industry, but instead on basic implementations based on existing platforms like MATLAB Simulink (Iqbal et al., 2021), or using publicly available driving components like DeepDriving (Gambi, Mueller et al., 2019; Gambi, Huynh et al., 2019). The effectiveness of those approaches for testing real autonomous driving systems under real traffic conditions is not demonstrated.

To tackle the challenges mentioned above and facilitate testing of different autonomous driving systems, we have proposed a critical test scenario identification approach in our previous short paper (Song, Tan, et al., 2021). *We are detailing the approach in the current work and providing two real-world cases to demonstrate the implementation and result of this approach.* The two cases involve two autonomous driving functions developed by the automaker Volvo Cars — a parking function and a driving function. Our approach utilizes three existing engineering tools (requirement and verification management tool, SPAS simulation platform, and modeFrontier process optimization and automation tool). We implement this approach for the two autonomous driving systems and have identified numerous critical scenarios for them. Consequently, the identified scenarios can be used to design test cases for those systems in both simulated and real-world testing. To clarify our scope, the work does not aim to find the best optimization algorithm but to implement the approach and demonstrate the feasibility of it for testing real autonomous driving functions.

The contribution of this work is the implementation and validation of said approach for critical test scenario identification. The approach enables an end-to-end solution from the initial analysis of the system specifications until generating critical scenarios that can be used for testing. It is generic as the tools involved are exchangeable and is not subject to any particular driving function, simulator or application tools. Thus, the approach can, in principle, be used for critical scenario identification for testing any autonomous driving system. 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 and terms used in this study. Section 3 explains our the research approach. Sections 4 and 5 detail the case studies that use the proposed approach for identifying critical scenarios for testing two autonomous driving functions. Section 6 presents related literature on critical scenario identification for testing of autonomous driving systems. We present discussions and limitations of the study in Sect. 7, and conclude the paper in Sect. 8.

2 Terms and concepts

In this section, terms related to critical scenario identification for testing autonomous driving system are introduced. Also, we provide our definition of terms such as scenario, critical scenario, and scenario-based testing.

2.1 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 (Ulbrich et al., 2015). 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. (2015), Menzel et al. further refined the definition of scenario into three different abstraction levels — *functional*, *logical*, and *concrete* scenarios (Menzel et al., 2018). 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 instantiated from the logical scenarios by assigning concrete values for the parameters within the desired value range and distribution.

We adopt the definition of abstraction layers of scenarios in our work, proposed by Menzel et al. (2018), where *functional scenarios* are retrieved from the system specifications and analyzed to derive the parameters for *logical scenarios*. Subsequently, *concrete scenarios* are simulated and optimized to identify the critical ones for testing 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 such as pedestrians, cyclists, and vehicles of different types. We stick with the definition from Menzel et al. (2018) to use entities for defining scenarios.

2.2 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 impending collision (Zhang et al., 2020). In contrast, Kluck et al. focus more on the concrete scenario level and consider a scenario to be critical if underlying parameter values cause a malfunction of the autonomous driving system (Klück et al., 2019). 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) Hallerbach et al. (2018). Herein we take the interpretation from Kluck et al., and define critical scenarios as *scenarios with a parameter set that has a high probability of revealing unintended and unsafe behavior of the systems, which may cause a collision or near-collision situation for the vehicle and other entities in the road traffic* (Klück et al., 2019).

Furthermore, we need to differentiate critical scenarios from similar terms to avoid misunderstanding. Gambi et al. use *accident scenarios* from police reports as critical scenarios for testing autonomous vehicles (Gambi, Huynh et al., 2019), whereas 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 (Klischat & Althoff, 2019). *Challenging* scenarios and *complex* scenarios are often used as synonyms, and are related to critical scenarios. Riedmaier et al. (2020) claim that a scenario is critical if the behavior of the system is evaluated after the scenario has been executed either in real world or in simulation and the criticality being measured. In contrast, scenarios are challenging or complex if they are evaluated and classified as challenging or complex without being executed. Ponn et al. point out that challenging scenarios are not always necessarily critical ones but more often lead to critical ones when executed (Ponn et al., 2020). Lastly, we also differentiate the concept of critical scenarios from *corner-case scenarios* (also referred to as *edge cases*). Karunakaran et al. define an edge case as an unknown and unsafe scenario that is hard to predict during the test and can lead to severe results for the autonomous vehicle (Karunakaran et al., 2020). Since critical scenarios can either be known or unknown, 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.

2.3 Scenario-based testing

Scenarios are commonly used to substantiate test cases for autonomous driving systems (Erdogan et al., 2018). 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 (Klück et al., 2019). However, a test case should entail not only a scenario but also a pass-fail criterion to evaluate the resulting behavior of the system (Gambi, Mueller et al., 2019; Menzel et al., 2018). An example test case for an autonomous lane-keeping function, as given by Gambi, Mueller et al. (2019), is that the vehicle must follow a navigation path on a generated road map. The test fails if the vehicle cannot get to the destination or drives out of the lane. In this example, the test scenario consists mainly of the generated road map, and the pass/fail criteria is whether or not the vehicle under test gets to the destination and drives in the lane. The pass/fail criteria for a certain scenario could be derived based on the requirements of functional, safety, or performance standards etc. of the system.

Scenario-based testing plays a key role in safety validation of autonomous driving systems (Riedmaier et al., 2020). It tests the systems with different scenarios and examines the resulting behavior of the 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 (Batsch et al., 2021). The scenario-based testing approach aims to reduce the test effort to a manageable number of scenarios by limiting the testing to meaningful scenarios (Ponn et al., 2019). The number of concrete scenarios can be infinite due

to the combinatorial explosion of parameter values (Batsch et al., 2021), and identifying all possible scenarios is difficult regardless of which approach is used (Hallerbach et al., 2018). According to Batsch et al., scenario-based testing usually runs in simulation with Software-in-the-Loop (SIL). Still, it can also be carried out with Hardware-in-the-Loop (HIL), or in the real world with proving ground (also known as test tracks in some studies) or public road traffic (Batsch et al., 2021).

Despite the remarkable benefits of using scenario-based approaches for testing autonomous vehicles, identifying relevant scenarios for the system under test remains the prerequisite, especially those critical scenarios that violate the desired safety requirements (Riedmaier et al., 2020). Open questions still challenge us in regards to what constitutes good test scenarios and how to generate them systematically (Gambi, Huynh et al., 2019); how to define and collect realistic test scenarios (Erdogan et al., 2018); and how to identify the critical scenarios for testing (Zhang et al., 2020). Menzel et al. propose many different sources that can be used for deriving test scenarios, such as functional specifications, system boundaries, the operational environment, legal requirements, and real driving data collected (Menzel et al., 2018). 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 (Porres et al., 2020). That is the core of the current study — to implement and validate the feasibility of the approach we developed for critical scenario identification for testing real autonomous driving systems.

2.4 Objective functions

An integral part entailed in critical scenario identification is how can we quantify and evaluate a concrete scenario to be critical or not, thus the criticality of a scenario must be represented in a quantifiable way. In this study, we use the term *objective functions* to refer to the measurements of criticality of a scenario. Different surrogate measurements for safety evaluation of traffic conflicts can be used as objective functions, for example, Time-to-Collision (TTC), Post-Enchroachment Time (PET), and Time-to-Brake (Mahmud et al., 2017). Among these surrogate indicators, TTC is used the most, according to a review study by Laureshyn et al. (2016). As an example, Klück et al. (2019) measure and extract scenarios with TTC less than one second as critical scenarios for an autonomous emergency braking system.

Safety metrics extracted from industrial standards can also be used as objective functions, for example, ISO-15622 standard for adaptive cruise control (Riedmaier et al., 2020), ISO-26262 for general automotive development and test (Feng et al., 2020), and Responsibility–Sensitive Safety (RSS) for autonomous vehicles (Karunakaran et al., 2020). Several performance metrics, including safety, functionality, mobility, and drivers' comfort, are used for generating test scenarios for autonomous vehicles by Feng et al. (2020), and they use a combination of the maneuver challenge and exposure frequency as the measurements for evaluating critical scenarios.

Eventually, selecting objective functions must be specific to a particular driving system and the system specifications. In this study, we have selected *TTC* and *jerk* for the autonomous driving function, *distance to vehicles* and *angle of host vehicle* for the autonomous parking function, as objective functions. *TTC* represents the time remaining before collision, given the vehicles keep the current speed and direction. A shorter TTC means that a collision is more imminent. *Jerk* measures the change of acceleration rate of the vehicle, where a large jerk value means an aggressive change of acceleration of the vehicle and the

opposite of smoothness. *Distance to vehicles* simply measures and restricts that the host vehicle (also known as ego vehicle) should keep a safe distance (i.e., 0.3m) to the stationary vehicles after completing the parking maneuver, and *angle of host vehicle* requires the vehicle to keep a good orientation and be within a specific boundary of angles (i.e., ±3°) to the parking slots.

3 Research approach

In this section, we describe the context and method we used for the current study. Overall, we followed the design science paradigm in our research (Runeson et al., 2020). Design science paradigm is used as a frame to guide, describe, or communicate the research using three basic elements — problem conceptualization, solution design, and solution validation. The three elements are connected in the given order and can be performed in many iterations (Runeson et al., 2020).

We *investigated the problem* of critical test scenario identification by studying the existing literature and industrial practices in an earlier study (Song, Engström, et al., 2021). Then, we *designed the solution* for critical test scenario identification in another prior study (Song, Tan, et al., 2021). In the current work, we implement the approach we developed in an industrial context by integrating the existing engineering tools, and *validate* it using two real autonomous driving functions by collaborating with Volvo Cars, which we report in Sects. 4 and 5. Lastly, we *infer the potential usage* of our approach for testing autonomous driving systems in general, which is detailed later in Sect. 7.

3.1 Research context

The current study is based on the critical test scenario identification approach that we presented briefly in our previous work (Song, Tan, et al., 2021). As shown in Fig. 1, the approach defines a tool-chain with three different engineering tools and a workflow for critical test scenario identification. The approach enables an end-to-end solution from analyzing the system specifications until critical test scenarios are identified for a given autonomous driving system. The identified scenarios can then be used to substantiate test cases for the system.

The three engineering tools are as follows: (1) a requirement and verification management tool which is used for storing and analyzing the system specifications. By using this

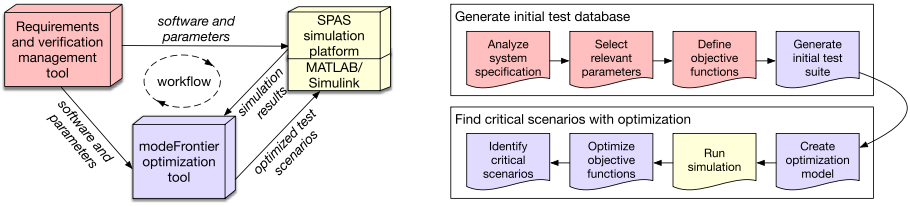


Fig. 1 Overview of the critical test scenario identification approach, consisting of interconnected tools (left) which are used in the workflow (right). The figure is a reprint from our previous work with minor adaptation, and we refer to Song, Tan, et al. (2021) for more details of the approach

tool, relevant parameters (e.g., speed and position of vehicles) and objective functions (e.g., TTC, PET) can be derived from the system specifications and operational environments; (2) an internal simulation platform SPAS, which is developed by Volvo Cars and used for early verification of the active safety and autonomous driving functions. We use SPAS to simulate the scenarios and record the simulation results; (3) a process automation and optimization tool modeFrontier, which is a commercial tool for process and design optimization. We create optimization models in this tool by using the parameters, objective functions, and an optimization algorithm (e.g., pilOPT), and the models explore critical scenarios based on the simulation result from SPAS.

The tools are exchangeable, meaning that we can substitute them with other similar tools to cope with different industrial environments or autonomous driving functions under test. For example, we can use other similar tools (e.g, SysWeaver (Bhat et al., 2018)) to replace tool (1) as it is an independent tool for studying the system functionality and extracting relevant parameters from its operational environment. We could also select a different simulator from tool (2), such as Carla (Dosovitskiy et al., 2017) or AirSim (Shah et al., 2018), to simulate the scenario execution. modeFrontier allows the integration of third-party applications in its optimization models and triggers the execution of the applications using e.g., available interfaces, command lines, or self-developed scripts in various programming languages. Thus, we just have to adapt the optimization flow in modeFroniter to incorporate a different simulator. Other simulators for autonomous driving are presented by Kang et al. (2019) and Rosique et al. (2019).

The workflow includes two main phases, see Fig. 1 (right). In the first phase, we start by analyzing the system specifications to understand the functionalities and the operational design domain (ODD) of the system. ODD is a concept that defines the operational environment where the system is designed to function (Gyllenhammar et al., 2020). The system specifications can be, for example, functional specifications, design documents, related standards or regulations. Based on that, we select relevant parameters that constitute a driving scenario and the value range and distribution of the parameters. Also, we define objective functions that measure the criticality of the executed scenarios and a threshold for each objective function for identifying critical 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 the SPAS simulator. The optimization model optimizes the scenario generation with respect to the objective functions and the simulation results. It executes the scenarios in the initial test suites in simulation, and continuously explores the parameter space based on the results of completed simulation. We also configure the number of iterations for the optimization model in modeFrontier based on the testing budget and computational resources available. Lastly, critical scenarios are identified if resulting objective function values surpass the thresholds. For example, scenarios with less than 0.3 m's distance to surrounding vehicles are critical for the autonomous parking function, which we describe in Sect. 5.

Our study is also relying on the collaboration with our industry partner — Volvo Cars, where they support us by providing access to the tools above and two autonomous driving functions, namely autonomous driving function and autonomous parking function. We implement our approach on these two functions for identifying critical scenarios for testing such systems, and demonstrate its feasibility for testing using realistic autonomous driving functions.

3.2 Research method

The current study is conducted in multiple steps for implementation and validation of the approach we proposed (Song, Tan, et al., 2021). We study each autonomous driving function in the first step. Then, we create the optimization models for the functions in steps 2 and 3. In the last step, we retrieve the optimization results and provide the identified critical test scenarios to the engineering teams.

1. For each autonomous driving function, we analyze the system specifications, through the requirement and verification management tool, to understand the functionality and the ODD of the system. The goal of this step is to identify the relevant parameters (e.g., speed and position of vehicles) and to define objective functions (e.g., TTC) for the autonomous driving function.
2. We explore the tool modeFrontier and create an optimization model by integrating the parameters and objective functions from step 1 and the SPAS simulation platform. We also configure the optimization algorithm (i.e., a multi-strategy searching algorithm pilOPT), size of the initial test suite, and the number of iterations based on the testing resource available.
3. We replicate the optimization model from step 2 and select a different algorithm (i.e., a heuristic searching algorithm MOSA) to compare how two algorithms differ in identifying critical scenarios.
4. We start the optimization models in modeFrontier, and debug the errors if the process fails or suspends, for example, the process gets timed out or software crashes due to any environment issues. After the optimization processes finish, we export and analyze the results such as the number of critical scenarios identified on each objective function, and how selected parameters are related to the identified critical scenarios, which we present in Sects. 4 and 5. The identified scenarios are provided to the engineering teams to test the systems and investigate potential flaws as well as to improve the specification, design, or implementation of the systems.

4 Case I: Autonomous driving function

This section describes the implementation of the approach which we present in Sect. 3 for the first case. 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 Analyze system specifications

ADF offers unsupervised in-lane driving in queued situations up to a specific speed limit $v_{\max}$, and enables 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) driving in a lane and (2) proactively adapting speed. These functionalities specify that the host vehicle shall stay in lane and maintain a safe longitudinal and lateral distance to

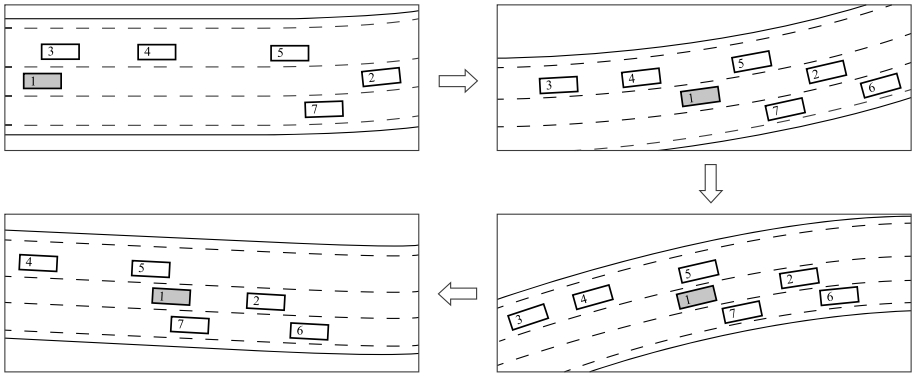


Fig. 2 A series of visualized scenes of the autonomous driving function (ADF). The gray vehicle (labeled 1) represents the host vehicle with ADF installed, and the white vehicles (labeled 2–7) represent the other vehicles on road

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 gray, while others are visualized in white. At the beginning of the scenario, the host vehicle drives at a relatively 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 decrease the speed gradually. At the end of the scenario, the host vehicle manages to adapt its speed to follow the front vehicle.

4.2 Select relevant parameters

With the requirement and verification management tool, the operational environment of ADF is analyzed, where it covers different kinds of influential factors on the road, including traffic, vehicle status, environment, infrastructure, other road users, and driver behavior. We select parameters from 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 on the parameter selection of movable entities as an example.

Road users include all kinds of vehicles, pedestrians, and animals. However, since the operational environment for ADF is on the highway where pedestrians and animals rarely appear. Thus, only vehicles are taken into consideration in this study. 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 and triggers 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 the 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 is of interest.

4.2.1 Initial position

The initial position of vehicles is selected, including the longitudinal and lateral positions along the road. Several constraints are defined to limit value selection of the initial position. First, each vehicle should keep a safe distance to others. The two-second rule, which is a rule of thumb estimating for safe distance at any speed for vehicles (Breyer et al., 2010), 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 simulation, 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 $3.5m$ (Mouhagir et al., 2016), 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 limit, i.e., $v_{V_h}(t) \leq v_{\max}$. To evaluate ADF's performance, the host vehicle should be able to detect 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 $v_{V_l}(t_0) \leq v_{V_h}(t_0)$, or otherwise, the ADF will be deactivated and switched to human maneuver mode. Moreover, to keep the traffic smooth on the highway, the minimum speed for all vehicles is set to a specified level $v_{\min}$. According to Abuelenin et al. (2014), the traffic velocity on the road approximately complies with a normal distribution, and thus normal distribution is used for velocity 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 real-world traffic data (Bokare & Maurya, 2017). Besides, the longest acceleration period is restricted to 3 seconds and 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. For a special case when there is only one lane in the road, the maximum number of vehicles could be placed on the road is 10, with subject to $d_{\min}^x$, $D_{\max}$ and the length of a vehicle. That is also the upper limit for the number of vehicles. The upper limit on vehicle number also reduces the design space and accelerates the scenario optimization process. To ensure there are enough vehicles on the road to formulate a critical scenario, the minimum 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: 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 an objective function that 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 (Feng et al., 2020). For this reason, jerk, measured as the rate of change in acceleration, is selected as another objective function to evaluate criticality of scenarios. Jerk value larger than $\pm 4m/s^3$ would not be acceptable for most vehicles (Bellem et al., 2016). 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 scenarios 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

To generate an initial test suite, we use MATLAB to translate the specifications in the requirement and verification management tool, and send the outputs to modeFrontier for generating a new design of experimentation (DoE) (Riedmaier et al., 2020). DoE is a systematic approach for analyzing the relationship between input parameters and output values and 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 the creation of new test scenarios. ModeFrontier has different approaches for design space exploration, and in this case, the Space filler DoE is leveraged for test scenario generation. This approach gives the most uniform filling of the design space, where the risk of missing corner cases can be mitigated. Latin Hypercube Sampling (LHS) is applied to generate random design configurations and an initial test suite (i.e., with 50 scenarios, according to the rule of thumb for DoE (Tan, 2019)) is generated.

4.5 Create optimization models

The optimization process in modeFrontier follows the scheme as discussed in Fig. 1. First, initial test scenarios are simulated in SPAS. Then, simulation results are recorded, saved, and analyzed by modeFrontier. Subsequently, new test scenarios are generated with distinct parameter values by the optimization model and executed in simulation. The entire optimization process ends after the specified number of iterations, and critical scenarios can then be extracted and analyzed. Figure 3 shows the modeFrontier optimization model for ADF. In short, the top part is parameters for generating new scenarios. The sub-parts inside represent sub-parameters that need to be specified for a scenario. The parameter values are transferred to the middle part, where the SPAS simulation is performed. Lastly, the bottom part is used to define the objective functions. The optimization is then based on the objective functions of the scenarios in SPAS simulation.

Two optimization algorithms, namely Multi-Objective Simulated Annealing (MOSA) (Ulungu et al., 1999) and pilOPT[1], are used for optimization purposes. The optimization models that use each algorithm are created separately, and Fig. 3 is an example of such models. pilOPT is an

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

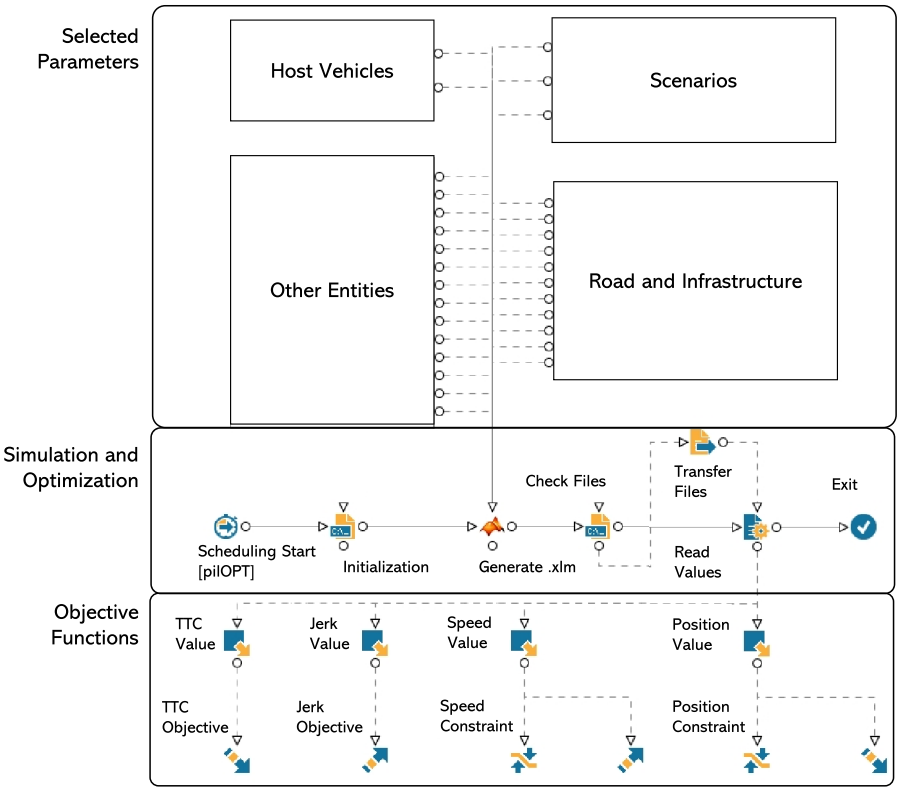


Fig. 3 modeFrontier optimization model for the ADF

in-house developed algorithm in modeFrontier, which can effectively handle the multi-strategy searching problem and minimize the amount of time and computational resources required. 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 models, 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 to extract and analyze the critical scenarios.

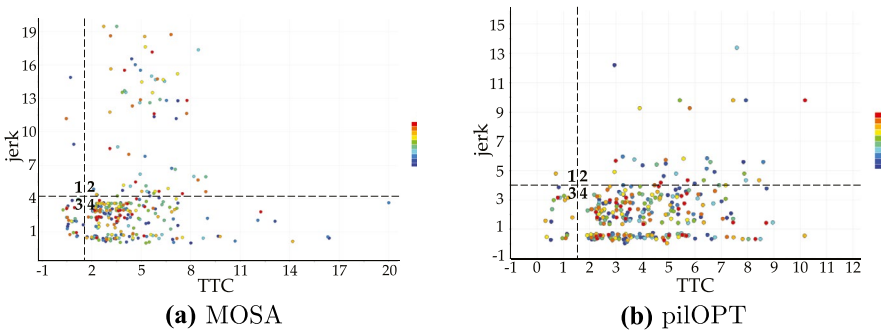


Fig. 4 modeFrontier optimization results from the autonomous driving function (ADF) with (**a**) MOSA and (**b**) pilOPT algorithms. Each dot represents a test scenario. The dashed lines are the criticality thresholds for TTC and jerk, respectively, while the precise scale of the two axis is left out for confidentiality reasons. Colors indicate the order of the scenario execution

4.7 Identify critical scenarios

Figure 4 shows the optimization results of critical scenario identification for ADF, with Fig. 4a using MOSA and Fig. 4b using pilOPT. A dot in the figure represents a test scenario, and the two dashed lines are the criticality thresholds for the objective functions TTC and jerk. By using these two dashed lines, we separate the scenarios into critical and non-critical sections. Thus, we see four quadrants in each sub-figure: (1) critical jerk and critical TTC, (2) critical jerk but non-critical TTC, (3) non-critical jerk but critical TTC, and (4) non-critical for both jerk and TTC. Therefore, scenarios in quadrants 1, 2 and 3 in each sub-figure are critical scenarios, since at least one objective function is critical in these quadrants. The colors merely indicate the sequence of the scenarios when they were executed in modeFrontier, blue means the earlier scenarios, and red means the later scenarios.

For the MOSA algorithm, we observed 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 scenarios with low jerk values, the TTC values are mostly over ΔT_{thres}, indicating that in those test scenarios, the host vehicle does not experience a sharp acceleration, thus being obvious safe scenarios. As scenarios are randomly distributed as in the figure, no distinct region feature and difference emerge with the optimization process.

In Fig. 4b, there is no obvious boundary on either axis, but the figure is divided into two groups. Test scenarios in the first group, located on the upper part of the figure, have remarkably high jerk values. The number of critical scenarios is summarized in Table 1 to compare the difference between MOSA and pilOPT. The number of scenarios caused by violating TTC and jerk constraints is not summing up the total amount since there are some

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

	MOSA	pilOPT
jerk	33	87
TTC	18	28
total	45	95

scenarios where both criteria are critical. We conclude that, in this case study, pilOPT has a better performance in finding critical scenarios than MOSA, especially with respect to jerk. This is, however, not our primary focus, and optimization algorithms can be further explored in future studies.

5 Case II: Autonomous parking function

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

5.1 Analyze 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 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 maneuver 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 maneuver 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 sets a brake torque to stop the vehicle and deactivates the parking function.

Figure 5 illustrates the function and operational scenes of APF. Specifically, five vehicles (numbered 2–6 and in white) are parked parallel to the road direction and remain stationary. The host vehicle (i.e., numbered with 1 and marked in gray — the vehicle with APF installed) first drives from the left and passes the stationary vehicles, it scans and

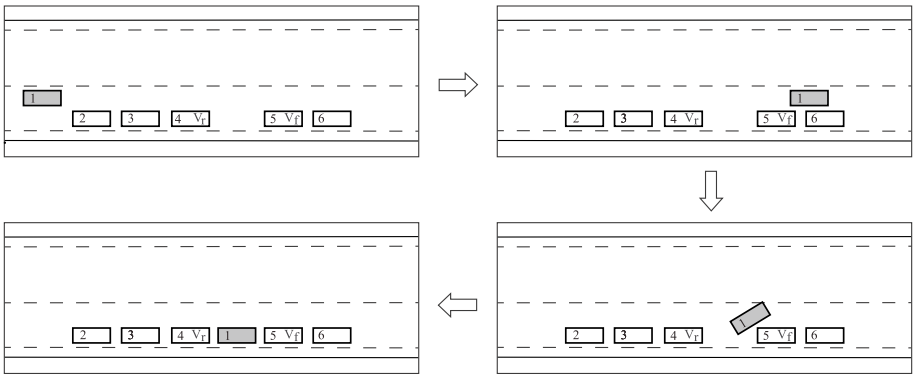


Fig. 5 A series of visualized scenes of the autonomous parking function (APF)

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 feasible position subject to physical constraints such as the slot length and the maximum steering angle the vehicle can perform. In an optimal situation, the host vehicle should stop 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.

5.2 Select relevant parameters

After analyzing the system specifications and current design of APF by using the requirement and verification management tool, we identify two relevant parameters 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 and Wegener (2003) adopted both the slot length and slot width as the two parameters that depict the parking space and used them to explore critical test scenarios for an autonomous parking system. Given the current design and implementation of APF, we presume a sufficient slot width in the current study and thus select slot length as a relevant parameter for defining scenarios.

Based on the setup in Fig. 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 slot length. The value range of slot length contains both a lower bound — the minimum slot length APF should handle without colliding the stationary vehicles, and an 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 vehicles 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 the most impact (referred as AnV_f). The value range for this parameter is set to $[-3°, 3°]$ according to the ISO-16787 standard (International Organization for Standardization, 2017) which is a standard 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 (International Organization for Standardization, 2017), 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 beyond these two criteria critical and should be identified. Based on these two criteria and the setup shown in Fig. 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

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

either of them. In addition, the yaw angle of the host vehicle (referred to as AnV_h) needs to be optimized to identify the scenarios that end with an angle beyond $\pm 3°$.

Nevertheless, we cannot have all the aforementioned objective functions in one optimization model due to the natural conflicts among 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 must 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 into two different optimization models as well. The resulting set of objectives is four, hence leading to four optimization models with two objective functions each as shown in Table 2.

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 (Tan, 2019), as shown in Eq. 1. 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 the parameters based on the intended distribution. However, the real distribution for both DtV_r and AnV_h are unknown and are difficult to model or predict. Hence, we generate the initial test scenarios with the Latin Hypercube Sampling (LHS) strategy and 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 (Batsch et al., 2021). LHS is considered superior to other sampling approaches like random sampling and ensures that the entire parameter space is covered as evenly as possible (Batsch et al., 2021). 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 what has been

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

presented in Fig. 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 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 tracks the objective functions' value. Then the optimization model optimizes the selection of parameters for another 72 iterations based on the completed simulation results. Lastly, we select the optimization algorithm in modeFrontier based on our previous experience (i.e., Sect. 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. We do not replicate all optimization models for MOSA, as the simulation and optimization are computationally expensive. To clarify again, we do not aim to find the best optimization algorithm in this study but to implement and validate the feasibility of the approach for critical test scenario identification for real driving functions.

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 are 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 due to possible problems such as software crashes or environmental issues.

5.7 Identify critical scenarios

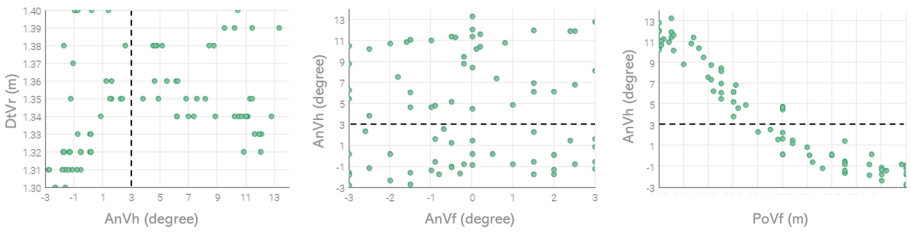
The result of the optimization models can be visualized in modeFroniter using different charts or statistical analysis tools and be exported in many different formats. As mentioned earlier, we created six optimization models for APF, and each model consists of 80 evaluation iterations. By filtering the 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, as we mentioned earlier, can be used to substantiate test cases for testing the function in different environments (e.g., in simulation, in testing tracks, or on public roads).

The resulting critical scenarios are found exclusively on one of the objective functions — AnV_h — and no critical scenarios found for both DtV_r and DtV_f. As shown in Fig. 6a — 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 can be found 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 ISO-16787. That indicates 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. An example is AnV_h gets 13.28° when AnV_f and PoV_f are at specific values. That is the considered critical since angle of host vehicle deviates too much after the parking maneuver. These scenarios should be investigated by the development and test teams to improve the system design, implementation, or test so it can handle them as intended and safely.

Furthermore, Fig. 6b 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 Fig. 6c, 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. This observation suggests that decreasing slot length can lead to more critical scenarios, and the smaller the slot length is, the larger deviation of the angle of host vehicle can be. For that reason, how slot length impacts the autonomous parking function should be analyzed and more tests by adapting the slot length may be conducted. Based on that, the system should be enhanced and re-tested using the identified critical scenarios.

The results are consistent when using other optimization models with different combinations of objective functions. We identify critical scenarios on AnV_h only, and the visualized results 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



(a) Optimization result of critical scenarios for both AnVh and DtVr

(b) Critical scenarios optimized on AnVf (i.e., angle of front vehicle)

(c) Critical scenarios optimized on PoVf (i.e., position of front vehicle)

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 dashed line in sub-figure (**a**) and above the dashed line in sub-figures (**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

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 that smaller slot length generally increases the angle of the host vehicle, which means a bad orientation to the parking slot and demonstrates there is a pattern of critical scenarios that the system does not meet.

Lastly, pilOPT generally identifies more critical scenarios than MOSA 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. Based on the results, pilOPT performs better than MOSA, while further comparison between these two algorithms is required. Since we do not aim to address the best optimization algorithm in the current study, we have demonstrated that our approach effectively identifies critical test scenarios and is general to different optimization algorithms.

6 Related work

In this section, we present related literature on critical test scenario identification for autonomous driving. The general idea of critical scenario identification, as described by Ponn et al., is that a concrete scenario is selected, executed, and evaluated with the criticality metrics (Ponn et al., 2019). The identified critical scenarios are then used for testing autonomous driving systems in different environments, such as simulation or testing tracks. As reported in the literature, there are different approaches for critical scenario identification, ranging from using expert knowledge, data extraction techniques, or search-based algorithms etc. We categorize the literature we surveyed based on our interpretation and compare them with our work separately in the following subsections.

Our approach falls into the third category of the approaches — search-based approaches as described in Sect. 6.3 — and uses search and optimization algorithms for identifying critical test scenarios from the scenario space. Unlike other studies that use similar approaches, we integrated tools and a workflow to provide an end-to-end approach for critical test scenario identification, and we validated our approach in an industrial context by using real autonomous driving systems. In contrast to studies that use different approaches such as knowledge-based approaches or data-driven approaches, our approach does not rely heavily on expert knowledge or substantial driving data collection, which is considered more efficient for testing autonomous driving systems.

For a complete literature overview, we refer to the systematic literature reviews by Zhang et al. (2021) for critical scenario identification, and Rajabli et al. (2020) for software verification and validation, as well as the survey by Riedmaier et al. (2020) for scenario-based approaches, all for assessment of safety of autonomous vehicles.

6.1 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 with respect to access to experts. As an example, Ponn et al. (2020) involved experts from the autonomous driving domain for selecting parameters of scenarios

and assessing the 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 catalogue of test scenarios (Riedmaier et al., 2020), yet the drawbacks are non-negligible. It requires expert involvement and is labor-intensive, and may lack the diversity and complexity of real-world scenarios, especially those accidents that impose complicated situations and rarely happen (Zhang et al., 2020). 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 (Ponn et al., 2019).

Compared to our approach, we do not rely on expert involvement, and identification of the critical scenarios is automated by integrating the existing engineering tools. Specifically, the selection of scenarios is based on optimizing the parameter space and simulation of the scenarios. Thus, it is not limited or biased by subjective knowledge acquired.

6.2 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, 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, Huynh et al. (2019) generated effective and critical test scenarios for autonomous driving by reconstructing crash accidents from police reports in simulation, using natural language processing. Zhang et al. (2020) introduced a toolkit for extracting critical scenarios based on real traffic accident videos and reproducing the scenarios in simulation. The extracted scenarios are then used for the safety assessment of autonomous vehicles. Erdogan et al. (2018) proposed 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 are 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. Ding et al. (2020) trained 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. (2020) 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. (2021) using Gaussian Processes to train and optimize the parameter selection towards the most critical scenarios on the performance boundary, and Jenkins et al. (2018) 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. (2020) used online supervised learning to train a generative model for searching and selecting critical scenarios for testing the autonomous maritime collision avoidance systems through the operation.

Even though diverse techniques for extracting or generating critical scenarios based on real driving data have been studied, limitations are described in these studies as well. A prerequisite of using such techniques is a data set that is comprehensive (Riedmaier et al., 2020), 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 (Karunakaran et al., 2020; Priisalu et al., 2021). As highlighted by Hallerbach et al. (2018), 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 (Ponn et al., 2019), the location where the data is collected, and the fact that rare-occurring situations are difficult to collect (Ding et al., 2020). After all, we still have to understand how to extract and select scenarios given massive data collected (Klitzke et al., 2019).

In contrast, our work does not rely on collecting data from different sources, and thus is not subject to the size or diversity of the data set. Instead, we first analyze the system functionalities and operational design domain based on the system specifications. Then, we create optimization models using the existing engineering tools. The optimization model optimizes scenario generation towards the objective functions for critical scenarios based on simulation results.

6.3 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 the 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. Our approach falls into this category.

Klischat and Althoff (2019) used evolutionary algorithms to optimize the drivable area of the vehicle to generate complex scenarios for testing the motion planning of the autonomous vehicles. Similar work is reported by Althoff and Lutz (2018) to generate safety-critical scenarios for collision avoidance of autonomous vehicles by optimizing the drivable area. Buehler and Wegener (2003) also employed evolutionary algorithms for generating critical scenarios for functional testing of an autonomous parking system. Specifically, genetic algorithms are a class of evolutionary algorithms commonly used for search and optimization problems. Gambi, Mueller et al. (2019) used genetic algorithms to evolve the generation of virtual road networks for testing the lane-keeping function. Klück et al. (2019) proposed an approach for test parameter optimization using genetic algorithms and have employed it for testing an autonomous emergency braking function. Felbinger et al. (2019) compared a genetic algorithm approach and a combinatorial testing approach for detecting critical scenarios for an 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 before the test and covering the entire parameter space is costly (Ponn et al., 2019). In addition, this approach does not rely on collecting substantial driving data and is easy to implement. However, some limitations are also stated in the existing studies. For example, generated scenarios may not be realistic in the real-world traffic, simulation of the scenarios is often computationally expensive, and only

low-dimensional scenarios can be handled effectively in optimization (Ding et al., 2020). To complement the said limitations, Beglerovic et al. (2017) simulated and optimized test scenarios based on a light-weighted surrogate model instead of the real system, Feng et al. (2020) established a sophisticated model of relevant parameters, metrics, and searching process for critical scenario generation, and Hallerbach et al. (2018) created 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 basic implementation of the autonomous driving function based on engineering tools like MATLAB Simulink (e.g., Ponn et al., 2020), or publicly available driving components such as DeepDriving and Beam.AI by Gambi, Mueller et al. (2019), and Gambi, Huynh et al. (2019), for validating the approaches, their effectiveness on realistic autonomous driving functions is not demonstrated. Besides, many of them are also function-specific, which is relevant to a particular function or operational domain for, e.g., parking system (Buehler & Wegener, 2003), motion planning (Klischat & Althoff, 2019), or highway scenario (Beglerovic et al., 2017). 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 system. We also demonstrate the effectiveness of this approach by using two real autonomous driving systems from the industry. As articulated by Hallerbach et al. (2018) and Ding et al. (2020), there exist very few studies that 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.

7 Discussion

In this section, we first summarize the main contributions and some limitations of the current work. Then, some future work items are included and discussed to improve or extend the approach further.

7.1 Contributions

In this paper, we present the implementation of an approach to critical test scenario identification for pilot validation of real autonomous driving systems. We argue that testing all possible driving scenarios in real road traffic is impractical, since it is expensive, time-consuming, and may still not cover all the rare-occurring traffic situations (Karunakaran et al., 2020; Zhang et al., 2020). In contrast to Kalra et al., who claim that millions or even billions of miles of driving tests are required to demonstrate the reliability of an autonomous vehicle (Kalra & Paddock, 2016), testing of autonomous driving functions must be based on a feasible number of test scenarios and focus on the most critical ones (Ponn et al., 2019; Klück et al., 2019). Using critical scenario identification and simulation is considered a good alternative to address the gaps as mentioned above and enable testing of autonomous driving functions in a more efficient way (Klück et al., 2019; Rajabli et al., 2020; Mauritz et al., 2016).

In our approach, we integrate the existing engineering tools and a workflow as an end-to-end 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 (Hallerbach et al., 2018). Our approach relies on optimizing 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 specific tools, techniques, or sensors employed in the function or simulation.

We demonstrate the feasibility 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 (Gambi, Mueller et al., 2019). Besides, many studies demonstrate the effectiveness of their approaches based on limited settings, such as a pedestrian step-out scenario (Batsch et al., 2021) and certain scenarios from Carla Scenario Runner Library (Ding et al., 2020). 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 Sects. 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 versions of the autonomous driving systems. Thus, the results are subject to the current design and specifications of the systems when conducting the study. Besides, we want to clarify that our approach can effectively generate some, but not all potential critical scenarios in the operational design domain. The goal is to find more critical scenarios using this approach, and possible tuning can still be performed regarding, for example, the sampling approach for the initial suite of scenarios and the number of optimization iterations. We can expect that the better the parameter space is sampled for the initial test suite, the more efficient the critical scenarios could be approached later in the optimization; the more optimization iterations can be assigned, the more critical scenarios could be identified, yet the cost comes in parallel from the computational resources needed for it.

7.2 Future work

Future improvement and extension of our approach regarding its design and implementation are multi-fold, including, e.g., scenario composition, parameter selection, realistic parameter distribution, and optimization algorithms.

First, the *composition and representation of scenarios* can be improved to include different driver behavior models and enable the definition of complex spatio-temporal interactions between different entities within the driving maneuver. As highlighted by Feng et al. (2020), 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. (2020) and Erdogan et al. (2018), to define static and dynamic elements in a full driving scenario in a structured way are good references to explore.

The second improvement of the approach is the comprehensive *parameter selection* to ensure all relevant parameters are identified and selected. The current approach is highly relying on analyzing the system specifications and extracting parameters based on its functionalities and operational design domain. Existing models for a structural description of

road traffic like Bagschik et al. (2018) or Scholtes et al. (2021) can be used to derive a complete list of parameters. The model from Schotles et al. includes entities and specifications of entities on six different layers such as road network, traffic infrastructure, moveable objects, weather conditions, and communication (i.e, information exchange between entities). Another alternative is to involve experts to identify relevant parameters, especially the significant ones, based on prior-established knowledge. Parameter selection could also be a continuously evolving step, where originally selected parameters might be removed due to irrelevance to a scenario's criticality and new identified parameters can be selected later.

Thirdly, *realistic distribution of the relevant parameters* selected should be investigated to improve the realism of the scenarios and real occurrence of the scenarios. As articulated by Batsch et al. (2021), 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 (Riedmaier et al., 2020); thus, the distribution of parameters is important and needs to be identified (Ding et al., 2020). Different sampling approaches such as adaptive sampling (Mullins et al., 2017), importance sampling (Feng et al., 2020), or modelling the distribution (Song et al., 2022) are a few candidates to be further studied.

Fourthly, we also propose to evaluate different *optimization algorithms* to best fit the generation of critical test scenarios for different autonomous driving systems and use parallelization to improve the efficiency of the simulation and optimization (Porres et al., 2020). 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 is more about the computational resources that can be allocated for optimization that matters. Our primary focus in this work is to implement the approach for critical test scenario identification and demonstrate the feasibility of the approach for real 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 the 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 (Koopman & Wagner, 2016; Knauss et al., 2017), the importance of using simulation and critical test scenario generation increases steeply (Jenkins et al., 2018). 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 is highly dependent on them (Beglerovic et al., 2017). Despite that sub-components within our approach can be further expanded and improved, we believe our work is worth the efforts and has 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 an end-to-end solution for critical test scenario identification that is general for different autonomous driving systems, according to Hallerbach et al. (2018).

8 Conclusion

Safety and reliability are indispensable properties for autonomous vehicles, yet there is no common standard way to test autonomous driving functions systematically and efficiently. Conventional requirements-driven testing approaches are impeded due to uncertainty of

the operational environment and the complexity of the driving scenarios. Therefore, support for identifying the critical scenarios for testing autonomous driving systems is needed.

We established an end-to-end approach with integrated tools and a workflow, and we implement it in this study to identify critical test scenarios for two real autonomous driving systems from Volvo Cars. The results suggest that our approach can effectively identify critical test scenarios. The identified scenarios can be used to substantiate test cases for autonomous driving systems either in simulation or in the real world.

Future extension of the approach aims to improve the scenario representation, incorporate the realistic distribution of the parameters, and compare the effectiveness of different optimization algorithms. Future work also includes integrating this approach into the practitioners' engineering practices and to observe how effective the approach, as well as the identified critical scenarios, could be for testing autonomous driving systems in a real industrial context.

The study provides a feasible and complete approach for critical test scenario identification for autonomous driving and a basis for building sub-components further. Given the widespread attention on autonomous driving and the challenges for testing the enabling functions, we shed light on testing different autonomous driving systems efficiently and effectively.

Acknowledgements Thanks to our colleagues in the Software Engineering Research Group at Lund University and Volvo Cars, and the anonymous reviewers of the journal for their review of earlier versions of the manuscript.

Funding Open access funding provided by Lund University. This work was supported in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP).

Data availability The datasets generated during and/or analyzed during the current study are not publicly available due to company confidentiality, protected by the Swedish Secrecy Act (SFS 2009:400, Ch. 24, §5).

Declarations

Conflict of interest The authors declare no competing interests.

References

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

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

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

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

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

Bellem, H., Schönenberg, T., Krems, J. F., & Schrauf, M. (2016). Objective metrics of comfort: developing a driving style for highly automated vehicles. *Transportation Research Part F: Traffic Psychology and Behaviour, 41*, 45–54.

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

Bokare, P., & Maurya, A. (2017). Acceleration-deceleration behaviour of various vehicle types. *Transportation Research Procedia, 25*, 4733–4749.

Breyer, G., et al. (April 2010). *Safe distance between vehicles*. Technical report, Conference of European Directors of Roads, Brussels, Belgium. https://www.cedr.eu/docs/view/60794fa6cf0c0-en

Buehler, O., & Wegener, J. (2003). 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*.

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

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

Erdogan, A., Kaplan, E., Leitner, A., & Nager, M. (2018). Parametrized end-to-end scenario generation architecture for autonomous vehicles. In *6th International Conference on Control Engineering & Information Technology (CEIT)* (pp. 1–6). IEEE.

Felbinger, H., Klück, F., Li, Y., Nica, M., Tao, J., Wotawa, F., & Zimmermann, M. (2019). Comparing two systematic approaches for testing automated driving functions. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)* (pp. 1–6). IEEE.

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

Gambi, A., Huynh, T., & Fraser, G. (2019). 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).

Gambi, A., Mueller, M., & Fraser, G. (2019). 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).

Gyllenhammar, M., Johansson, R., Warg, F., Chen, D., Heyn, H.-M., Sanfridson, M., Söderberg, J., Thorsen, A., & Ursing, S. (2020). 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)*.

Hallerbach, S., Xia, Y., Eberle, U., & Koester, F. (2018). 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.

International Organization for Standardization. (2017). *Intelligent transport systems – Assisted Parking System (APS) – Performance requirements and test procedures* (ISO 16787). International Organization for Standardization, Geneva, Switzerland.

Iqbal, M., Han, J. C., Zhou, Z. Q., & Towey, D. (2021). 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). IEEE.

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

Kalra, N., & Paddock, S. M. (2016). 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.

Kang, Y., Yin, H., & Berger, C. (2019). 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.

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

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

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

Klück, F., Zimmermann, M., Wotawa, F., & Nica, M. (2019). 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). IEEE.

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

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

Laureshyn, A., Johnsson, C., De Ceunynck, T., Svensson, Å., de Goede, M., Saunier, N., Włodarek, P., van der Horst, R., & Daniels, S. (2016). *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*.

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

Mauritz, M., Howar, F., Rausch, A. (2016). 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). Springer.

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

Mouhagir, H., Talj, R., Cherfaoui, V., Aioun, F., & Guillemard, F. (2016). 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). IEEE.

Mullins, G. E., Stankiewicz, P. G., & Gupta, S. K. (2017). 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). IEEE.

Ponn, T., Breitfuß, M., Yu, X., & Diermeyer, F. (2020). 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). IEEE.

Ponn, T., Gnandt, C., & Diermeyer, F. (2019). 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).

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

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

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

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

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

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

Scholtes, M., Westhofen, L., Turner, L. R., Lotto, K., Schuldes, M., Weber, H., et al. (2021). 6-layer model for a structured description and categorization of urban traffic and environment. *IEEE Access, 9,* 59131–59147.

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

Song, Q., Engström, E., & Runeson, P. (2021). 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).

Song, Q., Runeson, P., & Persson, S. (2022). A scenario distribution model for effective and efficient testing of autonomous driving systems. In *ACM 1st Workshop on Autonomous Software Testing (AUST)*. https://doi.org/10.1145/3551349.3563239

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

Tan, K. (2019). 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).

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

Ulungu, E. L., Teghem, J., Fortemps, P., & Tuyttens, D. (1999). Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multicriteria Decision Analysis, 8*(4), 221.

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

Zhang, X., Li, F., & Wu, X. (2020). CSG: Critical scenario generation from real traffic accidents. In *2020 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1330–1336). IEEE.

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. (2021). Finding critical scenarios for automated driving systems: A systematic literature review. Preprint retrieved from http://arxiv.org/abs/2110.08664

Qunying Song is a WASP-funded PhD student at the Department of Computer Science, Lund University, Sweden. His primary research interests lie in software testing of autonomous systems. Prior to his PhD study, he received his bachelor and master degree in Software Engineering, from Kristianstad University, in 2012 and 2013 respectively, and has been working as a software engineer in software industry for 6 years. Contact him at qunying.song@cs.lth.se.

Kaige Tan is a PhD student at the Mechatronics Devision, KTH Royal Institute of Technology, Sweden. His research focus is on predictive control and optimization of cyper-physical system using edge computing. Contact him at kaiget@kth.se.

Per Runeson is a professor of software engineering at Lund University, Sweden. He is the leader of the Software Engineering Research Group (SERG). His research interests include empirical research in industry on software development and management methods. He is particularly interested in studies on testing, and open source and data strategies. He is the principal author of "Case study research in software engineering", serves on the editorial board of IEEE Transactions on Software Engineering and Software Testing, Verification and Reliability. Contact him at per.runeson@cs.lth.se.

Stefan Persson is a senior engineer at Volvo Cars, Sweden. He works within active safety and his research is related to optimization strategies. He received a Ph. D. from KTH (Royal Institute of Technology), Sweden in 2004. Contact him at: stefan.persson@volvocars.com.