# A taxonomy of validation strategies to ensure the safe operation of highly automated vehicles

Batsch, F., Kanarachos, S., Cheah, M., Ponticelli, R. & Blundell, M.



**Original citation & hyperlink:**
Batsch F, Kanarachos S, Cheah M, Ponticelli R, Blundell M. 2020. A taxonomy of validation strategies to ensure the safe operation of highly automated vehicles. Journal of Intelligent Transportation Systems.
https://dx.doi.org/10.1080/15472450.2020.1738231

DOI 10.1080/15472450.2020.1738231
ISSN 1547-2450
ESSN 1547-2442

Publisher: Taylor and Francis

***This is an Accepted Manuscript of an article published by Taylor & Francis in Journal of Intelligent Transportation Systems. on 20/03/2020, available online: http://www.tandfonline.com/10.1080/15472450.2020.1738231***

# A Taxonomy of Validation Strategies to Ensure the Safe Operation of Highly Automated Vehicles

Felix Batsch[a,b], Stratis Kanarachos[a], Madeline Cheah[b], Roberto Ponticelli[b], Mike Blundell[a]

[a]Institute for Future Transport and Cities, Coventry University, Coventry CV1 5FB, United Kingdom; [b]HORIBA MIRA, Nuneaton CV10 0TU, United Kingdom

**ARTICLE HISTORY**
Compiled March 1, 2020

**ABSTRACT**
Self-driving cars are on the horizon, making it necessary to consider safety assurance and homologation of these autonomously operating vehicles. In this study, we systematically review literature that proposes new methods for these areas. The available methods were categorized into a novel taxonomy, dividing them into the strategies of combinatorial testing, robustness testing and search-based testing. We analyzed the literature in regard to modeling capabilities, targeted automation subsystem, targeted driving task level and the metrics used for criticality evaluation and coverage of the scenario space. We found that there are significant differences and shortcoming in the modeling capabilities of the existing research and that methods of each strategy usually target a specific driving task level. Additionally the criticality assessment of scenario-based validation methods was examined, revealing the need for more comprehensive metrics to assess complex scenarios. The developed taxonomy furthers the understanding in different scenario-based testing approaches for automated vehicles and serves as a guide for future research.

**KEYWORDS**
taxonomy; scenario based testing; highly automated driving; test case generation; intelligent transportation systems

## 1. Introduction

Automated vehicles (AVs) of SAE level 3 and higher (SAE, 2018) offer the potential to significantly reduce accident rates, traffic congestion and greenhouse gas emissions (Brown, Gonder, & Repac, 2014; Greenblatt & Saxena, 2015; Ross & Guhathakurta, 2017; Tsugawa & Kato, 2010). Ensuring that AVs are safe in every situation remains an open problem, as the automation in an unrestricted environment introduces a number of challenges.

Most research has been directed towards the development of automated systems, fuelled by the emergence of new players from the technology industry. Recent accidents involving vehicles with different levels of automation show however, that the technology is not yet infallible and fails in ordinary highway-driving scenarios, which could easily be handled by a vigilant human driver (Duarte & Ratti, 2018). Furthermore, incidents regarding AVs in public tests could potentially lead to distrust and a generally

CONTACT. Felix Batsch Email: batschf@coventry.ac.uk

dismissive attitude by the public, which could significantly impede a contemporary introduction (T. Zhang et al., 2019).

While the electronic stability control (ESC), which is mandatory in all new vehicles, is validated through a single sine-with-dwell test (NHTSA, 2007), the same techniques cannot be used for AVs as they have to operate through a much wider variety of scenarios. Statistical assessments in studies by Watzenig and Horn (2017) and Kalra and Paddock (2016) estimate the necessary test drive distance to validate that AVs are better than current human drivers in the magnitude of $10^6$ to $10^9$ kilometers. These estimates can be compared to the total annual road traffic in Germany which amounted to 757 billion kilometers in 2015 (ADAC, 2016). With the cost for real world testing estimated by Wachenfeld (2017) at 2.65 Euro per kilometer, the testing expenditures would exceed economic justification.

Furthermore, the emergence of rules to guide the development and introduction of automated vehicles, although necessary, adds further complexity to the topic, as most governments have released guidelines or code of practices for automated vehicle development and testing. An ethics commission appointed by the German government for example proposes a pool of scenarios, controlled by the government, from where the software is tested (Di Fabio et al., 2017). Several countries such as the United Kingdom, Germany, China and several states in the United States have additionally introduced legislation and code of practices to allow for testing of AVs on public roads, where they can encounter realistic traffic scenarios. All legislation requires a driver being ready and able to take over control of the vehicle if necessary, but especially the British code of practice for automated vehicle trialing (Department for Transport, 2019) and the California code of regulations on testing of autonomous vehicles (Department of Motor Vehicles, 2018) do not necessarily require the safety driver to be within the vehicle.

It is not viable, to include all possible circumstances that an automated vehicle might encounter in a fixed set of scenarios, as the amount of scenarios an automated vehicle could encounter is indefinite (Schuldt, Saust, Lichte, Maurer, & Scholz, 2013). This entails that an automated vehicle's software code cannot include all these scenarios in hard code. On the other hand, all the scenarios still have to be accounted for in the validation (Koenig, Gutbrod, Hohmann, & Ludwig, 2017).

The recent emergence of machine learning algorithms, and their end-to-end use in automated vehicles increases the validation problem. These enhanced vehicles are able to outsmart predefined tests, similar to the events in the recent diesel emissions scandal, where vehicles could identify when they were driven on a test bench, instead of the road (Zhao, 2016). New ways to test AVs which deploy machine learning are therefore indispensable.

Findings can also be transferred from other fields in robotics, like unmanned underwater vehicles (UUV) (Mullins, Stankiewicz, Hawthorne, & Gupta, 2018), or aerospace and aviation (Calefato, Ferrarini, Kutila, & Landini, 2015). Furthermore, combinations of different software testing methods could yield valuable research which could be transferred onto the AV validation problem (Anand et al., 2013).

Research effort in the area of AV validation has gained traction, introducing various methods to validate AVs. In this paper, we introduce a novel taxonomy of how virtual validation methods for AVs can be differentiated. This aids researchers to identify gaps in existing research. The validation methods are allocated to three overarching strategies by identifying similarities in the underlying theories of the methods. The main difference from previous taxonomies like Stellet et al. (2015) is the focus on virtual validation strategies, which are indispensable for the validation of AVs.

We selected published research found through keyword searches of IEEE and Scopus

databases, as well as publications from key authors and institutions for our review. As initial keywords we used "scenario-based testing", "validation" and "autonomous vehicles", scanning the reference sections of found papers for further relevant literature. We limited our review to literature from the past 25 years.

The remainder of this section lays out a summary of terminology commonly used in literature, pointing out overlaps and redundancies. Section 2 describes the validation methods that are typically used in vehicle validation and where virtual validation methods are integrated in the overall validation process. In Section 3 we describe the proposed overarching virtual validation strategies and their use in automated vehicle validation. In Section 4 we discuss the findings from the literature analysis and highlight gaps in the existing research. Finally, in Section 5 we summarize the proposed taxonomy. Additionally a tabular overview over the covered literature and its analysis is appended.

### *1.1. Terminology*

In order to provide a common basis of discussion and make the understanding of our work easier, we start by setting out the terminology we use in this paper. We point out how terminology is used in existing research, but the definitions given in this paper are not definitive and will certainly evolve as the only recently emerging field of AV validation evolves.

#### *1.1.1. Scenario, Situation, Scene*

A commonly used classification for the terms *scenario*, *situation* and *scene* has been elaborated by Geyer et al. (2014) and was improved by Ulbrich, Menzel, Reschka, Schuldt, and Maurer (2015): A *scene* is therein defined as a snapshot of the spatial layout of the vehicle and its environment, without any temporal attributes. Furthermore, a *situation* is derived from a *scene* to include mission-specific goals and values. It is therefore always from the ego vehicle's point of view. Finally the *scenario* includes the temporal aspect, by combining several *scenes* to a progression.

A more detailed classification of the term *scenario* is presented by Menzel, Bagschik, and Maurer (2018). To facilitate the use of *scenarios* alongside the development process according to ISO-26262, they differentiate the *scenario* description into *functional*, *logical* and *concrete scenarios*. The *functional scenario* is described in natural language for use in the concept phase. The *logical scenario* describes parameter spaces within the state space for all necessary parameters, while the *concrete scenario* details a specific set of parameters with concrete values for each parameter. The *logical* and *concrete scenario* description is used in the development and testing phase respectively.

The terminology for *scene*, *situation* and *scenario* by Ulbrich et al. (2015) is adopted in this paper, making use of the finer distinction by Menzel et al. (2018) whenever necessary.

Note that the term *maneuver* is sometimes used (Nagel, Enkelmann, & Struck, 1995; Najm, Smith, & Yanagisawa, 2007; Waymo, 2017), and is analogous to the term *situation*. However, since the definitions are similar to that of *situation*, we will not be using it in this paper.

Furthermore the actors and general setting of a *scenario* need to be described:
The term *ego vehicle* or *ego* describes the vehicle that is being examined or observed in the *scenario*, usually the AV. It is sometimes also referred to as the system-under-test (SUT).

*Dynamic objects* describe all objects other than the *ego vehicle*, that change their state over the course of the *scenario*. These are for example other vehicles, pedestrians or objects like traffic signals. *Static objects* are objects that are stationary and do not change its state during the *scenario*, for example traffic signs, trees and buildings.

The *environment* describes the road layout and lane network with the lane markings for example. Additionally it describes environmental states, such as lighting and weather conditions (i.e. rain, snow, fog, etc.).

#### *1.1.2. Edge, Boundary or Corner Case Scenario*

When scenarios are used in the domain of testing, further differentiation is used to separate different types of scenarios.

Current literature uses a variety of different terms to describe test scenarios that have the potential to be safety critical for an AV, such as edge, boundary or corner case scenarios. 9 out of the 47 papers considered in this review mention one or more of the three designations for scenarios, but no definition is given (Galko, Rossi, & Savatier, 2014; Hutchison et al., 2018; Khastgir et al., 2017; Koopman & Wagner, 2016; D. R. Kuhn, Kacker, & Lei, 2010; Mullins et al., 2018; Nie & Leung, 2011; Tuncali, Fainekos, Ito, & Kapinski, 2018; Tuncali, Pavlic, & Fainekos, 2016). Generally the terms describe a test scenario where the outcome is on the boundary between safe and unsafe, according to the used metric to assess the safety. These scenarios are especially interesting for AV validation, as a small change in one parameter might result in the scenario becoming unsafe.

The term boundary case scenario should not be confused with boundary value analysis, where a systems is analyzed with parameters at their maximum or minimum value. An edge, boundary or corner case scenario can, but does not necessarily require one or multiple operating parameters at their maximum or minimum values.

#### *1.1.3. Rare Event*

Another term often used in research regarding scenario-based validation is the term *rare event*. From the 47 papers considered in this review, 11 refer to rare events (Gietelink, 2007; Gietelink, De Schutter, & Verhaegen, 2005, 2006; Huang, Lam, & Zhao, 2017; Hutchison et al., 2018; Koopman & Wagner, 2016; Shortle & L'Ecuyer, 2011; Zhao, 2016; Zhao, Huang, Peng, Lam, & LeBlanc, 2018; Zhao et al., 2017; Zlocki, Eckstein, & Fahrenkrog, 2015).

According to Lagnoux (2006), a rare event is defined by a very low probability of occurrence, between $10^{-9}$ and $10^{-12}$. The probability threshold to classify an event as rare varies from discipline to discipline. If we take a traffic accident with personal injury as a rare event, we can construct the following (frequentist) deduction from accident statistics (Zlocki et al., 2015): In 2015, 305,659 accidents with personal damage occurred in Germany, on a total driven mileage of $7.57 * 10^{11}$ kilometers (ADAC, 2016; Statistisches Bundesamt (Destatis), 2018). This equals a mean distance in-between accidents of $2.47 * 10^{6}$ kilometers or a mean accident occurrence of $4.04 * 10^{-7}$ accidents per kilometer.

## 2. Automotive Validation Methods

This section gives an overview over the different validation methods that are used today. In the automotive context, validation assures that the functionality and safety during the intended use of the vehicle can be guaranteed. This is traditionally done by using the staged V-Model validation process as defined in ISO26262 and displayed in Figure 1. The left side of the V-Model represents the design phase and the right side the corresponding hardware tests conducted to validate the design. The test methods range from field operational tests (FOTs), which represent reality as close as possible, to validation in simulation, where the focus is mainly on the software.

In the following we will describe these test methods in the light of AVs and highlight published research that falls within the test methods of the classical V-Model, as well as the emerging methods of scenario-based testing.

### *2.1. Field Operational Test*

FOTs are the most realistic of the validation methods, as they are held with the prototype vehicle in public or confined areas. The high degree of reality requires more effort and thus comes at a higher cost. Traditionally manufacturers have therefore always sought to reduce the amount of testing in FOTs as much as possible.

A methodology for conducting FOT studies is presented by Barnard et al. (2016). In the past, several large-scale FOT studies have been concluded, for example euroFOT which focused on intelligent vehicles systems (Kessler et al., 2012) and DRIVE C2X focusing on cooperative systems (Schulze et al., 2014). Further studies are TeleFOT (Mononen et al., 2013) and FOTsis (Alfonso, Sánchez, Menéndez, & Cacheiro, 2015). A non-exhaustive list of FOTs with objectives around ADAS (Advanced Driver Assistance Systems) and highly automated driving can be found in Table 1.

Waymo, the self-driving car spin-off of Alphabet Inc., has pursued a FOT-oriented development and validation. They have been testing automated vehicles since 2009 and amassed up to 16 million kilometers on public roads (Waymo, 2017). Since then many more demonstrations of autonomous driving in real traffic have been carried out, for example the fully automated drive by "Bertha", a prototype Mercedes S-Class, through urban and rural Germany (Ziegler et al., 2014).

FOT data is often used as initial data set, from which test scenarios are derived into X-in-the-Loop (XiL) testing or simulation (Zhao, 2016; Zhao et al., 2018, 2017; Zlocki et al., 2015). Another approach is to run the automated driving functions passively alongside human drivers in FOTs, thus unable to execute the complete driving task. This open loop setup allows for a realistic recording of critical scenarios and a simultaneous evaluation of the automated driving functions (Koenig et al., 2017). The implementation of this approach is difficult and costly however, as it requires the deployment of expensive sensory equipment for the sole purpose of recording as well as a large fleet of vehicles to gather a sufficient amount of data. This approach is therefore mainly feasible for OEMs on customer vehicles, piggy-backing on existing sensors. Additionally, it doesn't give an indication of how an automated driving function executes after it defers from the human driver's execution of the driving task, due to the missing feedback loop.

Field operational test drives in real traffic are the final stage of vehicle validation, as shown in Figure 1. Due to their high expenditure their use should be reduced as much as possible, for example with a predominant use of simulation tools and XiL

**Table 1.** A Chronology of FOTs Targeting Intelligent Transportation Systems[1]

| **Name of the FOT** | **Objective** | **Year** | **Utilized Vehicles / Drivers** | **Covered Distance** |
|---|---|---|---|---|
| ACAS (Ervin, 2005) | ADAS (CAS, ACC, FCW) | 1999 - 2004 | 11 / 96 | 220,480 km |
| Volvo Intelligent Vehicle Initiative (Battelle Memorial Institute, 2007) | ADAS (CAS, ACC, AEB) | 1999 - 2005 | 100 / >1000 | N/A |
| Road Departure Crash Warning (LeBlanc et al., 2006) | ADAS (LDW) | 2001 - 2006 | 11 / 78 | 133,575 km |
| Sky Project | V2X, ADAS | 2004 - 2009 | 5200 / 5600 | N/A |
| Assisted Driver (Alkim et al., 2007) | ADAS (ACC, LDW) | 2005 - 2006 | 19 / 19 | N/A |
| IVBSS (Sayer et al., 2011) | ADAS (CAS) | 2005 - 2011 | 26 / 126 | 1,311,862 km |
| AOS (Potters, 2009) | ADAS (ACC, FCW, LDW) | 2007 - 2009 | 2402 / N/A | 77,000,000 km |
| euroFOT (Kessler et al., 2012) | ADAS | 2008 - 2012 | ∼1200 / ∼1200 | 35,000,000 km |
| TeleFOT (Mononen et al., 2013) | ADAS | 2008 - 2012 | 616 | 5,613,566 km |
| Waymo (Waymo, 2017) | AV | ongoing since 2009 | N/A | >16,000,000 km |
| FOTsis (Nieto et al., 2015) | V2X | 2011 - 2014 | N/A | N/A |
| Drive C2X (Schulze et al., 2014) | V2X | 2011 - 2014 | >200 / 750 | >1,500,000 km |
| SPMD (Bezzina & Sayer, 2015) | V2V | 2011 - 2014 | 115 / N/A | >2,735,885 km |
| Bertha Benz Memorial Route (Ziegler et al., 2014) | AV | 2013 | 1 / 1 | 103 km |

[1] CAS: Collision Avoidance System; ACC: Adaptive Cruise Control; FCW: Forward Collison Warning; AEB: Autonomous Emergency Braking; LDW: Lane Departure Warning, V2X: Vehicle-to-X Communication

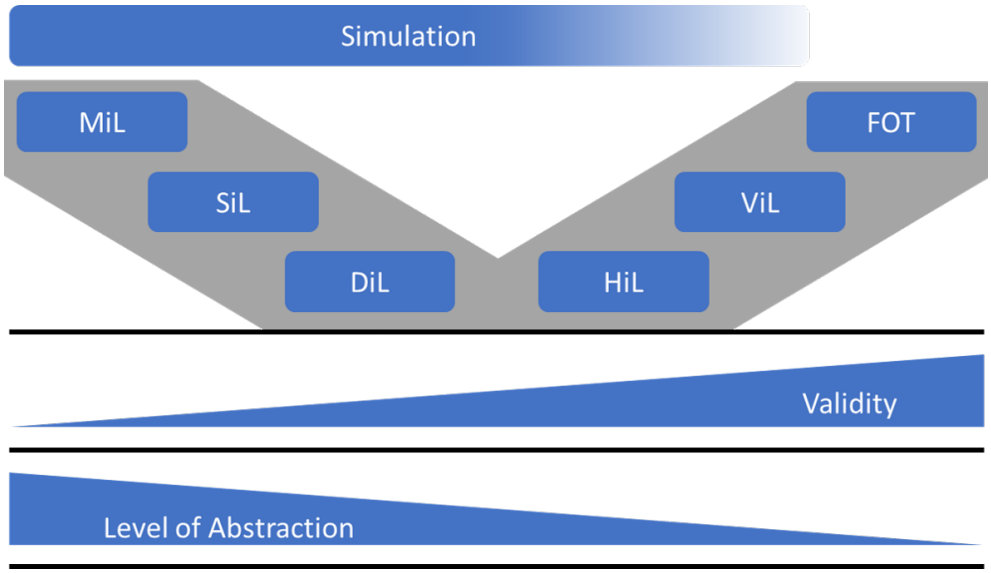


**Figure 1.** Validation Methods in the V-Model
MiL: Model-in-the-Loop, SiL: Software-in-the-Loop, DiL: Driver-in-the-Loop,
HiL: Hardware-in-the-Loop, ViL: Vehicle-in-the-Loop, FOT: Field Operational Test

methods. Further analysis of FOT based validation is therefore beyond the scope of this article and can be found in FESTA (2016), which is the result of the FESTA (Field opErational teSt supporT Action) project.

### *2.2. X-in-the-Loop Methods*

X-in-the-Loop describes a variety of validation techniques, where part of the reality is abstracted and simulated on a computer. They are often used in early development stages to verify and validate the functional requirements of specific components, when a more realistic validation is not yet possible due to the absence of adequate prototypes (Feilhauer, Haering, & Wyatt, 2016). Modeling and subsequent simulation makes these validation methods less representative than FOTs, although being cheaper to carry out. Due to the use of physical hardware, all XiL methods (except Software-in-the-Loop (SiL) and Model-in-the-Loop(MiL)) have to be conducted in real-time, resulting in complications regarding their scalability. The different XiL methods and their integration in the V-Model are shown in Figure 1.

Vehicle-in-the-Loop (ViL) is the XiL method with the lowest level of abstraction, as only other traffic participants or objects of interest are simulated (Bock, 2008; Bock, Maurer, & Farber, 2007; Uberbacher, Wolze, & Burtsche, 2017). If necessary, surrogate targets can be utilized to recreate safety critical maneuvers (Winner, Hakuli, Lotz, & Singer, 2016, Chapter 11). That allows for safe and comprehensive testing of ADAS (advanced driver assistance systems) and highly automated driving features or scenarios for AVs, with the disadvantage that tests are still setup-intensive.

Interesting approaches in Hardware-in-the-Loop (HiL) validation of ADAS and AVs are presented by Feilhauer et al. (2016). They propose the injection of virtual data in different layers of the ADAS architecture, allowing a validation of particular components on HiL test benches. Galko et al. (2014) present an example of this methodology, where generated data is injected into vehicle sensors on a ViL test bench. Both meth-

ods represent examples of HiL validation that test the robustness of ADAS and AVs. They will be described in more detail in Section 3.4.1.

A Software-in-the-Loop implementation for ADAS with HiL capabilities is presented by Zhou, Schmied, Sandalek, Kokal, and del Re (2016), where FOTs are automatically imported in the simulation software IPG Carmaker to set up an identical scenario in simulation. Safety critical scenarios can however not be utilized in this method, as they would require the critical scenario to be carried out in reality. A systematic approach to allocate test scenarios to the different XiL methods is given by Schuldt, Menzel, and Maurer (2015).

At this point two additional XiL methods are mentioned for completeness: MiL is often used in early development stages, to test models without the need to compile them into the target software.

Driver-in-the-Loop (DiL) systems are colloquially known as driving simulators, where everything but the human driver and the Human-Machine-Interface (HMI) are physically realized. DiL focuses on the human driver and can thus be used for takeover scenarios in ADAS. Their application in AV testing is less significant, as AVs aim at eliminating the human from preforming the designated driving task. DiL can however be used to operate other vehicles in the virtual domain and thus evaluate how the AV interacts with human drivers. Through this the human driver that is "in-the-loop" can exhibit a wide range of behaviors toward the SUT, from aggressive to defensive.

### *2.3. Scenario-based Testing*

Alongside the described test methods of the traditional V-Model, the principle of scenario-based testing is emerging for AVs (Menzel et al., 2018). This stems from the transition of the designated driving task from the human driver to the vehicle. As the execution of the designated driving task becomes part of the vehicle, according to ISO26262, this has to be tested in regard to environmental and driving situations. The high complexity of the environment and the automated system thus leads to the necessity of AVs being tested based on comprehensive scenarios. As described in Section 1, the number of scenarios that AVs could possibly encounter is indefinite. This section presents research that describe scenario-based validation approaches to manage the scale of this validation problem.

A methodology to generate and organize situations for FOTs is presented by Zlocki et al. (2015). They propose to populate a database from accident data, FOT data and staged driving situations. Furthermore, scenarios can then be created via simulation, where automated vehicles are included in the situations and then simulated stochastically with Monte Carlo simulations (c.f. Section 3.5.1). There are however two difficulties with this approach: Firstly, basing the database on recorded accident data will exclude near miss scenarios that did not lead to an accident, but are nonetheless highly safety critical. Additionally, the introduction of stochastic into the scenario generation process leads to a tremendous amount of scenarios, if this process is not limited and directed towards significant scenarios.

A similar methodology is defined by Alvarez et al. (2017), where a modified situation is generated from a reference situation to evaluate a specific event of interest, which they call a target situation. Comparable to Zlocki et al. (2015), these target situations are extracted from accident databases. Alvarez et al. (2017) use the term situation, but as this is a subset of the term scenario, their methodology can be extended to scenarios in general.

The work presented by Rueger, Sieber, Siegel, Siedersberger, and Faerber (2015) studied the controllability of a braking-and-evasion scenario through FOTs and validated a ViL simulation with the results. Even though they targeted the interaction and overruling of an evasion assistance system by the human driver, their work is a typical scenario-based validation. It demonstrates, that ViL and FOTs are especially limited by safety aspects however, making scenarios, which are on the boundary of collision/no collision impossible to test. Simulation methods like MiL and SiL do not have this limitation, but on the other hand lack in validity as shown in Figure 1.

To overcome this problem, Zofka, Kohlhaas, Schamm, and Zollner (2014) developed a validation method, where simulation is fused with data from FOTs, increasing the reliability of pure SiL simulation. Another approach by the same authors, that focuses on generating scenarios from sensor data for open-loop simulation is presented in Zofka et al. (2015).

## 3. Taxonomy of Different Validation Strategies in Simulation-based Testing

Validation of a system aims at proving that the intended purpose of the system is met and it is safe to be used by the intended user. This is done by comprehensively testing the system on whether the actual and intended behaviors deviate from one another. Similar to validation in the aerospace industry, the validation of highly automated vehicles has to evaluate the behavior in the event of a failure as well, since undetected failures can lead to significant financial and personal damages. Currently, validation of AVs is based on the development of test scenarios, derived from either expert knowledge or empirical data (Stellet et al., 2015). The SAE Level 5, also known as full driving automation, defines an unlimited operational design domain however, which leads to an indefinite number of possible scenarios (SAE, 2018). Since most scenarios of day-to-day driving are uneventful, effort has to be made to automatically generate challenging scenarios and prove formally that coverage of the intended testing space is obtained.

The automated generation of test scenarios is only possible in combination with vehicle simulation, while FOTs, ViL and HiL are indispensable to thereafter validate these simulations. FOTs, ViL and HiL are limited by their requirement to be run in real-time, making them impractical for large-scale testing. Their use should therefore be reserved for a small subset of significant and representative scenarios, which could be found through simulation and simultaneously validate the simulation.

This section inspects recent literature in the field of simulation-based testing and identifies three principal strategies to find and prune the amount of scenarios for validating highly automated vehicles as shown in Figure 2.

### *3.1. Examined Aspects*

This section describes the aspects that were considered to divide the papers into the previously mentioned virtual validation strategies. The reviewed literature and the examined aspects are listed in Table 2, at the end of this article. The table is organized by the type of strategy that the reviewed articles are allocated to, which is indicated in columns 2 - 6. The order is consistent with the description of the strategies in sections 3.3-3.5.

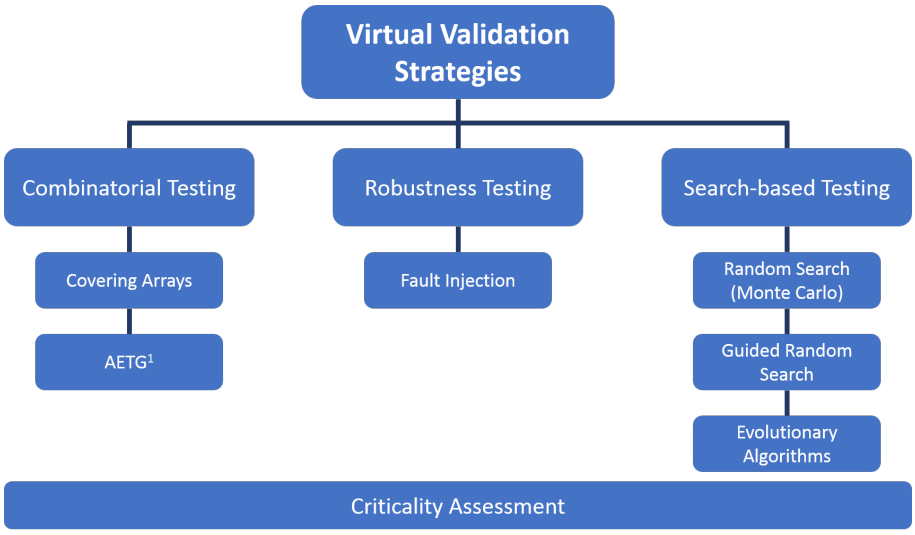


**Figure 2.** Taxonomy of Virtual Validation Strategies
[1]Automatic Efficient Test Generator

The columns 7 - 10 indicate the scenario modeling capabilities as described in Section 1.1.1. Here we considered whether the paper included the ego vehicle, static and dynamic objects and the environment in the modeling of the scenario.

Columns 11 - 13 indicate the driving task levels the respective method aims at validating. The three-level hierarchy of the driving task was proposed by Winner et al. (2016) and consists of the navigation task, the vehicle guidance task and the vehicle stabilization task. The three tasks are derived from the human behavior while operating a vehicle and have to be executed by the systems on an AV.

In columns 14 - 17 the subsystems of the AV that is being tested by the respective method is indicated. To that end, the AV was decomposed into the following functional subsystems: perception, sensor fusion, decision making and actuation.

The second half of the table lists the features of the particular methods:

- The case studies or examples where the method has been applied and the simulation environment that was used (Column 18).
- Whether the method proposes a general framework for validating AVs, or only aims at validating specific parameters (Columns 19 and 20).
- Whether the method incorporates a sequential approach, where a scenario can change over time, or only considers a time invariant, static scenario (Columns 21 and 22).
- The metrics used to evaluate the criticality of the scenario, i.e. the way it is judged if a scenario outcome is positive or negative (Column 23). This evaluation is a crucial aspect of scenario-based testing and thus described in more detail in Section 3.2
- How the effectiveness of the method is measured, i.e. the coverage of the scenario space. Often this includes comparisons to other methods (Column 24).
- An indication if the publication presents a novel method or a review of previously described methods (Columns 25 and 26).

### *3.2. Criticality Assessment*

An essential part of testing is the assessment of the results. In order to compare results across different validation strategies and methods, it is necessary to use the same criticality metrics. The assessment metrics here are referred to as criticality metrics (Sippl, Bock, Wittmann, Altinger, & German, 2016), as the scenario-based validation is evaluated by how critical a scenario is to either the safety of the ego vehicle, or to the safety of other traffic participants.

This evaluation can be done manually by examining every test individually, if the number of conducted validation tests is small. For larger amounts of tests, conducted in simulation (e.g. MiL, SiL), manual evaluation of the test outcomes becomes unfeasible. Therefore, the evaluation has to be automated to find safety critical scenarios. These critical scenarios could then be further investigated through other XiL methods or FOTs as previously described.

In software engineering the problem of matching the right evaluation to the results is called the oracle problem and is a big restraint in test automation (Barr, Harman, McMinn, Shahbaz, & Yoo, 2015). Evaluating the scenario outcome on a binary scale depending on one parameter is a simple task, for example detecting a collision with a Boolean value. Using such a basic function might however not be sufficient for evaluating an AV, especially in its interaction with an environment that includes humans (Stellet et al., 2015). A scenario where a pedestrian is just missed by a few inches for example, might not flag up as precarious with a function that just distinguishes between collision/no collision.

The following metrics have been used to assess the criticality of scenarios and can easily be automated in a simulation setup (de Gelder & Paardekooper, 2017; Koenig et al., 2017; Roesener et al., 2017; Sippl et al., 2016; van der Horst & Hogema, 1993):

- Time-to-collision (TTC)
- Time-to-brake (TTB)
- Time-headway (THW) or Gap time (GT)
- Post-encroachment-time (PET)
- Proportion of Stopping Distance (PSD)
- Initially-Attempt-Post-Encroachment-Time (IAPET)

Metrics from accident research, which take into account multiple factors have also been proposed to rate scenarios (Cunto & Saccomanno, 2008; Sobhani, Young, Sarvi, & Bahrololoom, 2013; Young, Sobhani, Lenné, & Sarvi, 2014). Such metrics include the *crash potential index* (CPI) which takes into account the deceleration rate to avoid a crash and the maximum deceleration rate, and the *road safety index* (RSI) which is a function of injury severity score (ISS), traffic volume and number of crashes. Using the ISS on its own as suggested by Alvarez et al. (2017) is not regarded as expedient, as it can only be applied once a crash with injury has happened.

Hallerbach, Xia, Eberle, and Koester (2018) suggest to complement TTC and TTB with a combination of multiple traffic quality metrics, which take microscopic and macroscopic traffic data like the change of traffic density and velocity fluctuations into account. Combinations of multiple criticality metrics reduce the amount of false-positive and false-negative evaluations and should therefore be favored.

A criticality assessment metric with a clear physical interpretation is introduced by Stellet, Vogt, Schumacher, Branz, and Zollner (2016). They use a collision energy reduction metric, which evaluates scenarios based on how the collision energy is reduced. This metric is however only applicable if a crash is unavoidable and cannot compare

the criticality of non-crash scenarios to each other.

As Sippl et al. (2016) mention, the TTC or GT based metrics listed above are not accounting the fact, that dynamic objects in scenarios can alter their path or speed. Thus they only give valid measurements for the criticality if the conflict is imminent. Furthermore, none of these safety or criticality metrics take into account how close proximity passing of dynamic objects, such as pedestrians or cyclists, is accounted for, since these are often perceived as threatening by the pedestrian/cyclist. The evaluation of what is perceived as critical by a human is subjective and a universal criticality metric is therefore difficult, if not impossible to formulate.

### *3.3. Combinatorial Testing*

Combinatorial testing (CT) is a simple, yet effective way for testing software and systems, sometimes also known as interaction or t-way testing. It has been in practical use for many years and can be applied for any XiL method. Theoretically CT could also be used in FOT, but is not practicable due to its large number of test cases. CT evolved after a number of studies found that failures occur due to the specific combination of a limited number of parameters, also known as *interaction failures* (Anand et al., 2013; D. R. Kuhn et al., 2010; Nie & Leung, 2011; J. Zhang, Zhang, & Ma, 2014). For example, a large portion of failures can be found by covering all pairs of parameter combinations for all possible values each parameter can take. This is a big improvement over exhaustive testing, where every parameter combination is tested for every possible value. In addition to the described pairwise testing, where all 2-way combinations are covered, the failure detection can be increased by covering all $t$-way combinations, which devolves into exhaustive testing, if $t = N$, $N$ being the number of all parameters. A CT for a system with 10 parameters, each with two possible values would result in only 13 tests to cover all 3-way combinations, compared to $2^{10} = 1024$ for exhaustive testing. However, the number of detected failures and the amount of combinations tested are inversely proportional, thus testing up to 6-way combinations has shown to be sufficient for a number of hardware and software applications as can be seen from Figure 3 (D. R. Kuhn et al., 2010). From a computational cost perspective, the number of $t$-way tests scale proportional to $v^t log(N)$ with $v$ being the number of possible values for each parameter (Cohen, Dalal, Fredman, & Patton, 1997; D. R. Kuhn et al., 2010).

#### *3.3.1. Configuration versus Input Parameter Testing*

For the application in testing AV scenarios, two forms of CT can be distinguished (D. R. Kuhn et al., 2010): Examining combinations of scenario configuration parameters, such as parameters for environmental setup and static objects of the scenario. Or combinations of scenario input parameters, which define the dynamic objects of the scenario. For the testing of automated vehicles based on scenarios, both types are equally critical, as a failed scenario can emerge from a combination of either *a)* a t-way combination of configuration parameters; *b)* a t-way combination of input parameters: or *c)* a t-way combination of configuration *and* input parameters. This augments testing effort, as especially the combination of configuration and input parameters massively increases the necessary number of tests. Thus it has been suggested to combine and enhance combinatorial testing with other techniques (Nie & Leung, 2011).

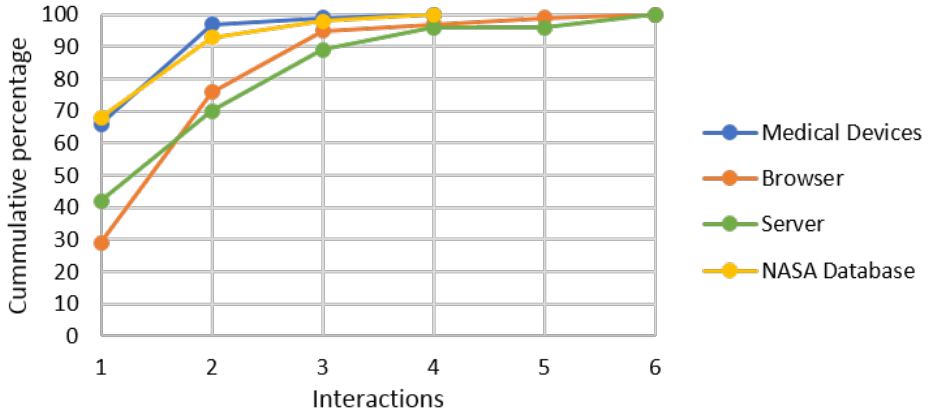


**Figure 3.** Fault Detections for up to 6-way Interactions in Different Applications after D. Kuhn et al. (2004)

### *3.3.2. Covering Arrays*

Combinatorial tests are usually designed in the form of *covering arrays*, where each row represents a single test, containing the set of parameter values represented as columns (Nie & Leung, 2011). All rows together must contain all t-way combinations of parameters to form a complete covering array. The construction of covering arrays has been subject of much research, as it is not a trivial task and was shown to be non-deterministic polynomial-time (NP) hard in some examples (Anand et al., 2013). Furthermore, the upper bound grows logarithmically with the number of parameters (D. R. Kuhn et al., 2010). Covering arrays can be constructed by a wide range of methods, like One-Test-at-a-Time (e.g. Automatic Efficient Test Generator (AETG) (Cohen et al., 1997)), In-Parameter-Order (IPO) (Rocklage, Kraft, Karatas, & Seewig, 2017), evolutionary computation (Shiba, Tsuchiya, & Kikuno, 2004), particle swarm optimization (Mahmoud & Ahmed, 2015) and backtracking search (Rocklage et al., 2017). Overviews over the algorithms are given in Anand et al. (2013); Colbourn (2004); Hartman and Raskin (2004); J. Zhang et al. (2014).

Covering arrays have been used to generate scenarios to test automated vehicles in multiple publications: Rocklage et al. (2017) build on the 4-layer-model proposed by Schuldt et al. (2013), but use CT for all layers combined. They generate the necessary covering array for a parametrized merging scenario through non-recursive backtracking, using a trajectory planner as feasibility checker for the generated trajectories. While they incorporate weather and lighting conditions, it remains questionable if their approach can be scaled up for more complex urban scenarios, with a higher number of traffic participants.

A different use of covering arrays is applied by Tuncali et al. (2018). They use 2-way covering arrays to generate initial populations of scenarios, from which they then proceed to find further, more challenging scenarios (cf. Sec. 3.4).

Betts and Petty (2016) use all possible combinations (full combinatorial testing) to establish ground truth and an upper bound as benchmark for Monte Carlo and search-based (genetic algorithm) scenario simulation strategies of a UAV, both of which will be described in subsequent sections (c.f. Sections 3.5.1 and 3.5.3).

#### *3.3.3. Conclusion of the Combinatorial Testing Strategy*

Combinatorial testing achieves good domain coverage, but is prone to combinatorial explosion as explained in Section 3.3. Mere combinatorial testing for complex automated driving scenarios is therefore not adequate, but could be incorporated with other methods to truncate the amount of tests.

From Column 20 of Table 2 it can be seen that all combinatorial testing methods propose a general framework for testing. The disadvantage of most of the considered methods is that they can only be used on static scenarios (8 out of the 11 combinatorial testing methods considered, Column 21 of Table 2). Only three are able to model sequential scenarios (Column 22, Table 2). Including sequential scenarios further increases the problem of combinatorial explosion.

Comparing combinatorial testing with random testing (c.f. Section 3.5.1) shows that the efficiency of failure detection is dependent on the problem size. For scenarios with a high number of variables, research suggests that random testing achieves a coverage close to CT, but a 100% coverage is unlikely due to the inherent randomness. Thus if 100% coverage is necessary and the dimensionality of the SUT is fairly low, CT is to be preferred (D. R. Kuhn et al., 2010).

### *3.4. Robustness Testing*

In software engineering, robustness is defined as the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions (ISO/IEC/IEEE 24765:2010(E), 2010). This can be extended to robustness testing of automotive systems, where invalid or exceptional input values are used to test the system, in the hope that these lead to increased failure rates. The failures can then be analyzed to find susceptibilities in the system, e.g. weak spots that would not have been found with valid inputs. The Ballista testing methodology was one of the first to follow this assumption (Kropp, Koopman, & Siewiorek, 1998).

Robustness is a key aspect of all AVs, as they are supposed to work in a wide range of operating conditions, which includes exceptional and unforeseeable rare events. The function of the automated system should therefore not be fatally impeded by faulty inputs or failing components (e.g. flat tire, disconnected wire, ECU going offline) as these are likely to occur at some point.

As can be seen from Table 2, robustness testing is mainly aimed at testing the perception systems of the AVs. This is achieved through adversarial examples and faulty data, which is injected into the vehicle sensors (c.f. Section 3.4.1). Only Koopman and Wagner (2016) propose a method for testing the actuation system, and specifically the stabilization level of the driving task.

A system level approach to robustness testing is described in Section 3.4.2.

#### *3.4.1. Fault Injection*

The key method in robustness testing is fault injection (Koopman & Wagner, 2016; Kropp et al., 1998). Especially with systems of higher complexity, multiple subsystems can be tested for robustness by injecting faults at different layers, e.g. feeding tampered or invalid inputs into subsystems. For ADAS and AVs, this has been realized in HiL, as described in Sec. 2.2. Data can be injected at three different layers of the ADAS/AV architecture: the sensor channel, the raw data stream and the processed data stream (Feilhauer et al., 2016). The injected data can of course be modified and enriched with

faults, thus testing for robustness. The different layers each target a specific subsystem of the ADAS/AV: injection in the sensor channel layer tests the physical sensors, while injection in the raw data stream layer targets subsequent data processing algorithms, like object recognition and motion estimation. Injection of data in the processed data stream layer tests sensor fusion and behavior generation algorithms like trajectory planning.

An example for sensor channel data injection is presented by Galko et al. (2014), where a screen is used to inject a simulated video feed into the camera sensor of a vehicle running on a ViL test bench. Additionally, their setup allows for an uncomplicated addition of weather conditions like rain, by sprinkling water in between the screen and the camera sensor, thus affecting the perception system.

Another method to test camera sensors regarding their robustness towards rain is proposed by Hospach, Mueller, Rosenstiel, and Bringmann (2016). Their method allows to overlay a recorded video feed with simulated rain of variable strength and thus test the image processing algorithms in a SiL setup.

Robustness has been shown to be especially problematic in modern image recognition algorithms using Neural Networks (Eykholt et al., 2018; Goodfellow, Shlens, & Szegedy, 2015). Eykholt et al. (2018) achieved 100% misclassification of a stop sign using minimal perturbation in form of graffiti-like stickers, fooling the algorithm into classifying it as a speed limit sign. Adversarial sample generation for machine learning systems has been extensively studied but is beyond the scope of this paper and will therefore not be covered.

While fault injection in camera sensors has been widely applied, testing other automotive sensors like Radar and LiDAR in the same way is more complicated. Radar HiL data injection is commercially available as a component test bench, but structured data injection into LiDAR sensors has yet to be achieved. First steps in the direction of fault injection into LiDAR sensor have been taken by means of crude attacks as described by Petit, Stottelaar, Feiri, and Kargl (2015) and Shin, Kim, Kwon, and Kim (2017). It is to be expected that a closed loop LiDAR sensor test bench will be available in the near future.

Hutchison et al. (2018) propose a methodology to test the robustness of automated systems based on the previously mentioned Ballista Method. Since many automated systems are distributed systems consisting of subsystems communicating through a bus, they approach the testing process by injecting faulty values into bus messages, thus being the only paper that aims at testing the sensor fusion of an AV (c.f. Column 15, Table 2). The instructions on which values are modified come from a database of exceptional values, which was build up by the authors empirically through extensive previous work. This reliance on experience for obtaining the exceptions database, along with the need for manual test specification which requires significant system knowledge, makes inferences on test coverage and efficiency difficult however.

#### *3.4.2. System-level Robustness Testing*

A system level approach of finding critical scenarios based on a robustness metric is presented by Tuncali et al. (2018, 2016). In Tuncali et al. (2018) they use a robustness metric as fitness function for their search algorithm, that follows a combinatorial generation of scenarios, as described in Sec. 3.3. Aiming to minimize the robustness value, they anticipate to find scenarios that are on the boundary between success and failure, so called boundary or edge cases. Examples for such scenarios are critical just-miss scenarios of other traffic participants or low-speed collisions.

#### *3.4.3. Conclusion of the Robustness Testing Strategy*

10 out of 12 robustness testing methods test for specific parameters (cf. Column 19, Table 2). It can therefore be concluded that the robustness testing strategy is mainly used for single component/parameter tests and when the failure types are roughly known to the expert conducting the tests. Furthermore, fault injection on its own only aims at evaluating specific events or failures. It does not provide sufficient means to achieve comprehensive coverage of the test space, unless it is combined with other strategies. This is amplified by the need of manual test specification for some robustness metrics, which requires exceptional knowledge of the SUT.

8 out of the 12 robustness testing methods can be applied to sequential scenarios, which is better than the limitation to static scenarios that many combinatorial testing methods have (Column 22, Table 2).

The robustness testing strategy also aims predominantly at testing and validating the perception system of AVs, as 8 of 12 methods target this AV subsystem (cf. Column 14 of Table 2) and 9 out of 12 methods target the guidance driving task (cf. Column 12, Table 2).

Adversarial perturbation attacks on image recognition algorithms, as shown by Goodfellow et al. (2015), are far from reality however, and their significance towards judging the performance in real-life situations is not always given.

### *3.5. Search-based Testing*

Search algorithms are often used in engineering to find an optimal solution to a problem. The search is guided by either a fitness or objective function, which is either maximized or minimized (Saeed, Ab Hamid, & Mustafa, 2016). This approach can be extended to the problem at hand: the search for challenging scenarios, that can be used to validate automated vehicles. An objective function would then for example target the maximization of a criticality score, as described in Section 3.2.

Search algorithms aim to explore a predefined search space in order to find a solution that best fits the fitness/objective function. In the following we describe three search strategies: The exploration can be completely random (c.f. 3.5.1), or preferably follow a strategy to operate more efficiently and save computation time (c.f. 3.5.2). Additionally, the application of evolutionary algorithms to find scenarios is described (c.f. 3.5.3).

#### *3.5.1. Random Search (Monte Carlo Sampling)*

Completely random exploration of the search space is usually referred to as Monte Carlo sampling, or Monte Carlo simulation. In a Monte Carlo simulation, a large number of similar experiments are carried out, by sampling certain parameters from a uniform probability density function. According to the law of large numbers, the relative probability of a random event then stabilizes around the theoretical probability of said random event. Monte Carlo methods can be used for decision problems as well as search problems, however the focus in this paper lies on their use to solve the latter. Due to their long history, Monte Carlo methods have been thoroughly researched and are commonly applied to problems in vehicle safety analysis (Betts & Petty, 2016; Gietelink, 2007; Zhao, 2016), as well as in a wide range of other fields, such as physics and finance. Generally Monte Carlo methods show the following pattern (Althoff & Mergel, 2011) (see also Figure 4):

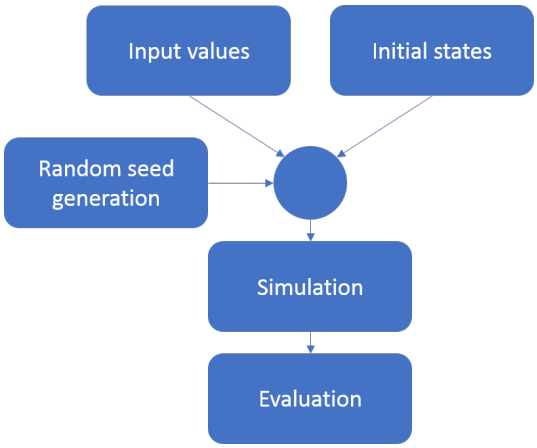


**Figure 4.** Monte Carlo Scenario Generation

(1) Random seed generation and randomization of inputs and initial states
(2) Simulation of the experiment with the randomly generated inputs and states
(3) After finishing a predetermined (large) number of experiments, the results are aggregated and evaluated

For the use case of simulating a driving scenario, initial data can either be manually created or taken from FOTs (de Gelder & Paardekooper, 2017; Zhao, 2016). A Monte Carlo simulation can be run based on this data, by randomizing the characteristics of the ego vehicle and other traffic participants (Alvarez et al., 2017). From the results of such an analysis, the safety of an automated driving system can be estimated, and conflicts can be detected.

The high dimensionality of automated driving scenarios raises the amount of necessary simulations quickly, making it impractical (de Gelder & Paardekooper, 2017). The problem is, that the complete randomization also leads to the simulation of many non-safety-critical, "boring" scenarios, like straight driving on a road without other traffic participants interacting with the ego vehicle (Zhao, 2016).

Monte Carlo simulations have been proposed to generate scenarios to populate a database of test scenarios, which can then be used in physical tests (Alvarez et al., 2017; Zlocki et al., 2015). To prevent the inclusion of non-relevant scenarios, the scenarios generated through Monte Carlo simulation have to be evaluated regarding their relevance as described in Section 3.2.

A three level approach to random testing is presented by Khastgir et al. (2017). Firstly the use case is varied, for example if a highway cut-in situation or an urban intersection is of interest. Then the test scenario is randomized, which consists of trajectories of the other vehicles and environment conditions. Lastly the initial parameters of the ego vehicle are randomized. To still achieve sufficient functional coverage of the parameter space, Khastgir et al. (2017) propose to divide the parameter ranges and carry out a fixed amount of random simulations in each sub-range instead of randomizing over the entire parameter range. Although this might help to achieve better coverage of the parameter space, it also increases the amount of necessary simulations and runs the risk of executing numerous insignificant scenarios.

### *3.5.2. Guided Random Search*

In recent years, crude Monte Carlo simulation has shown to be outperformed by a number of more intelligent algorithms (Betts & Petty, 2016; Gietelink, 2007; Zhao, 2016). The general idea is to guide the Monte Carlo simulation in its exploration of the search space with the use of a priori information. The generation of inputs and initial states is still randomized, but the sample size is algorithmically reduced. A number of reduction methods are explained by Gietelink (2007).
A way to increase the efficiency of Monte Carlo simulations is through variance reduction of the random seed generator. The most widespread method for variance reduction is *importance sampling*:

If the probability density function from where the random seeds are sampled is adjusted, it shifts the simulation to draw more samples from the interesting regions of the search space (Shortle & L'Ecuyer, 2011). If a uniform distribution is used, this degenerates to a generic Monte Carlo simulation.

Gietelink et al. (2005, 2006) showed that importance sampling efficiently reduces the amount of necessary tests compared to a Monte Carlo simulation for an adaptive cruise control system (ACC).

Other situations that have been evaluated using importance sampling are lane-change maneuvers (Zhao et al., 2017), and car-following maneuvers (de Gelder & Paardekooper, 2017; Zhao et al., 2018). The work of Zhao et al. (2018) and de Gelder and Paardekooper (2017) differs from that of Gietelink et al. (2005, 2006), as they reconstructed scenarios from big-data N-FOT studies (SPMD; see Table 1), while Gietelink et al. (2005, 2006) evaluated a universal car-following situation.

Both situations are relatively simple however, as they only include the ego vehicle and one other vehicle, the lead-vehicle. Although highly critical scenarios can emerge from the setting with only two actors, it would be interesting how the emergence of further actors would be handled by the automated system.

White and Cohen (1980) found that failure regions in software tests are contiguous, which represents essentially *a priori* knowledge about the SUT. Thus Chen, Kuo, and Liu (2009); Chen, Kuo, Merkel, and Tse (2010); Chen, Leung, and Mak (2004) introduced *adaptive random testing (ART)*, to more effectively find faults in software. In essence *ART* aims at generating new test cases far away from previous, successful test cases, in order to reach those failure regions. The generalized approach includes:

(1) Random generation of first input set from uniform distribution
(2) Run tests with this first input set
(3) Random generation of new input set from uniform distribution
(4) Choose input with largest distance from input of last successful test
(5) Run test with new input

This methodology has been adapted to the testing of unmanned underwater vehicles by Mullins et al. (2018): They propose an algorithm which uses adaptive sampling to find performance boundary areas, where small changes in initial parameters result in a transition between performance modes, e.g. from success to failure. Comparable to the previously described adaptive random testing, this method is founded on the assumption, that successful and failed scenarios are grouped into contiguous regions. To find these performance boundaries they sample from a meta-model generated from the SUT through surrogate optimization. They describe two metrics for generating the meta-model: Gaussian process regression and k-nearest neighbor density and variance estimation. Sampling from a surrogate model speeds up the search significantly, as it

can be executed much quicker than the original SUT. Subsequently they cluster the cases around the performance boundaries and thus obtain an acceptable number of interesting boundary cases.

Similarly to Mullins et al. (2018), Huang et al. (2017) use a Kriging-model (a different designation for a Gaussian process regression model, stemming from geostatistics) as a surrogate model for predict the outcomes of a test.

Mullins et al. (2018) use a universal approach to a wide range of automated vehicles and thus leave the definition of performance modes vague. For automated road vehicles the performance modes are relatively clear, however, the search could be improved if these performance modes are estimated beforehand and incorporated in the subsequent clustering. Furthermore, the method needs to be extended to include additional actors.

A method for testing a collision warning/avoidance system based on an errable driver model is described by Yang and Peng (2010). The errable driver model was based on the probability distribution of an undisclosed FOT. Although this results in a close representation of real driving scenarios, corner cases which are not included in the underlying naturalistic FOT data, are not accounted for. It is questionable if the data from human-centered FOTs are valid to serve as underlying distribution to test the behavior of automated systems.

#### *3.5.3. Evolutionary Algorithms*

Evolutionary algorithms are a group of search algorithms inspired by naturalistic evolution in biology and are generally applied to optimization problems. In the domain of software testing, evolutionary algorithms have widely been used to find software bugs (Alsmadi, 2010; Anand et al., 2013; Bao et al., 2017; Briand, Labiche, & Shousha, 2005; Mahmoud & Ahmed, 2015; Saeed et al., 2016; Shiba et al., 2004). A subset of evolutionary algorithms are genetic algorithms which can be used to search for critical scenarios. To this end, the search is formulated as an optimization problem, with an appropriate definition of the fitness function of the genetic algorithm. The fitness function evaluates the test results and thus needs to be defined so that the worst test results, i.e. the most challenging scenarios, are assigned the highest fitness value.

Genetic algorithms operate according to the following procedure: Firstly, an initial population of test cases is generated randomly and its members are evaluated by means of the fitness function. The fittest members are then selected and used to parent a new generation, which is bred through crossing over the properties of the parents and subsequent mutation, i.e. randomly changing properties. The operators for selection, crossover and mutation effectively steer the procedure to find optimal solutions. The algorithm terminates after reaching a predefined number of generations or after reaching a specified fitness goal.

The challenge in genetic algorithms is to maintain adequate diversity throughout the search space. On the one hand, an increased mutation rate would result in a higher coverage of the search space, but also drastically raises computational effort. If the mutation rate is too low on the other hand, the chances are higher that the algorithm converges on a local optimum.

An advantage of genetic algorithms is the possibility to include a-priori knowledge in the initial population. Previously conducted tests and known failure regions can thus be incorporated and reduce the necessary simulation effort (Bühler & Wegener, 2004).

Additionally, it was found that genetic algorithms outperform crude Monte Carlo random search for simple function tests (Betts & Petty, 2016). The study does not

indicate however, if that holds true for more complex problems like high fidelity automated vehicle simulations, where, in addition to the high mutation rate, an increase in independent variables exponentially increases the amount of necessary executions.

Previous work in the aviation industry made use of genetic algorithms for systems testing. Briand et al. (2005) use them to find cases where the execution of aperiodic tasks becomes time critical, due to the scheduling algorithm of the function under test.

A search method to find critical scenarios for a pedestrian collision avoidance system using a genetic algorithm is presented by Abdessalem, Nejati, Briand, and Stifter (2016). The use of a Neural Network trained surrogate model circumvents the necessity to execute the simulation in every iteration and thus speeds up the overall process. This speed-up was benchmarked to a genetic algorithm without the surrogate model and a Monte Carlo random search, showing significant improvements over both.

An extension was developed by Abdessalem, Nejati, Briand, and Stifter (2018) in form of a learnable evolutionary algorithm for the evaluation of vision-based systems. Here decision tree classification models are used to segment the search space into critical and non-critical regions and apply the evolutionary steps of the genetic algorithms in these critical regions. This allows for evaluation of a larger search space, but can be at risk of neglecting small critical regions, not covered in the initial population and subsequent classification.

Evolutionary algorithms where also used by Bühler and Wegener (2004, 2008) to evaluate an automated parking system and a braking assistance system, showing a distinct performance boundary.

#### *3.5.4. Conclusions of the Search-based Testing Strategies*

The largest proportion of found literature applied a search-based strategy to finding scenarios for the validation of AVs (24 methods, c.f. Table 2). While the simplest approach is the random search (Section 3.5.1), which is often used as baseline comparison, more sophisticated algorithms have been proposed that were classified as guided random search (Section 3.5.2) and the search with evolutionary algorithms (Section 3.5.3).

The search-based methods usually provide a general framework for evaluating AVs (23 out of 24 methods, cf. Column 20 of Table 2) and are able to model sequential scenarios (21 out of 24 methods, cf. Column 22 of Table 2).

While most search-based methods cover the ego vehicle (17 out of 24 methods) and dynamic objects (15 out of 24 methods) of the scenario, only few model static objects (3 out of 24 methods) and the environment (6 out of 24 methods) as well (cf. Columns 7-10 of Table 2).

The majority of the search-based testing methods (17 out of 24 methods), aimed at validating the decision making system of AVs, with none of the methods examining the actuation and sensor fusion systems (cf. Columns 14-17 of Table 2).

## 4. Discussion

It was found that none of the methods are able to model all aspects of a scenario (i.e. Ego Vehicle, Static Objects, Dynamic Objects and Environment; Columns 7 - 10 in Table 2), while at the same time validating all the subsystems (i.e. Perception, Sensor Fusion, Decision Making and Actuation; Columns 14 - 17 in Table 2). The validation is therefore typically targeting one driving task level (i.e. Navigation, Guidance or

Stabilization; Columns 11 - 13 in Table 2) at a time, with only a few methods testing on multiple levels.

This reveals a gap in the modeling capabilities of the current virtual validation methods. In order for a scenario-based simulation to work alongside physical tests such as XiL and FOTs, more aspects of a scenario need to be modeled. This is necessary to have information on the influence of parameters on the scenarios and inform subsequent physical tests on which parameters need to be more closely controlled.

The gaps could be addressed by investigating the following research questions:

(1) How could the number of considered scenario parameters be increased, while keeping the simulation computationally viable?
(2) Is it possible to identify failure regions in the scenario parameter space which allow us to define corner case scenarios and thus target subsequent testing efforts?
(3) How is it possible to include a systems engineering approach where knowledge of the components limitations is systematically utilized to accelerate the identification of critical scenarios?

Furthermore we describe that the current criticality assessment of scenarios has several shortcomings and define the following gaps:

Current criticality metrics fail to deal with scenarios, where each dynamic object can alter its path at any point. Most TTC and GT based metrics assume that all the dynamic objects of a situation continue on their trajectory without alteration and are therefore only reasonable on a short time frame before a collision.

Another shortcoming is the detection of near miss scenarios. These are crucial to be detected in a validation as a type of corner case scenario and cannot be detected by current criticality metrics.

Finally it might be necessary to include a subjective criticality assessment in the evaluation of AV scenarios, as humans have a subjective feeling of safety, which might differ from how the current objective measurements are judged.

We define three research questions on the criticality assessment based on these observations:

(1) How could current criticality metrics be improved to deal with the behavior of dynamic elements in a scenario?
(2) How could near miss scenarios be included in the detection of critical scenarios?
(3) How could a subjective criticality assessment be related to current objective criticality metrics?

Future work could therefore be carried out on validation methods that include a larger number of modeled parameters; searching the scenario space with the help of statistical methods in order to deal with the combinatorial explosion, while utilizing a more comprehensive criticality assessment. An examination on where failure regions in the scenario space are located could inform on what corner case scenarios are and indicate the influence of different scenario parameters.

## 5. Conclusions

The contribution of this paper is a survey of the state-of-the-art of the virtual, scenario-based validation of highly automated vehicles. The validation methods commonly used during development processes following the V-Model, XiL and FOTs, are detailed and recent advances for ADAS and AV testing are pointed out.

Furthermore we present a methodical classification of published validation methods in three overarching strategies: the combinatorial testing strategy, the robustness testing strategy and three search-based testing strategies. Table 2 gives a tabular classification of the covered research in the corresponding strategy classes of this taxonomy.

The taxonomy helps to coordinate the virtual validation effort with XiL and FOTs in a holistic, scenario-based strategy. To this end we analyzed the modeling capabilities of the methods, the targeted AV systems (perception, sensor fusion, decision making and actuation) and the metrics used to evaluate the criticality of a scenario (cf. Section 3.2). Additionally we looked at how the individual methods measure coverage of the scenario space, which shows the effectiveness of the methods in dealing with the combinatorial explosion and allows for comparison between different methods.

We defined a number of research questions to address the gaps we found through the analysis of the literature.

Table 2.: Categorisation of literature in regard to the proposed taxonomy for scenario-based testing strategies

| Paper | Allocated Strategy | | | | | Modeling Capabilities | | | | Driving Task Level | | | Examined System | | | | Features | | | | | | | Novel | Review |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Combinatorial Testing | Robustness Testing | Random Testing | GR Testing | Evolutionary Testing | Ego | Static Objects | Dynamic Objects | Environment | Navigation | Guidance | Stabilization | Perception | Sensor Fusion | Decision Making | Actuation | Case Study + Simulation Environment (SE) | Specific Parameters | General Framework | Static | Sequential | Criticality Metric | Effectiveness Measure | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| (D. R. Kuhn et al., 2010) | x | | | | | | | | | | | | | | | | | | x | x | | | Statement coverage, decision coverage, condition coverage, MCDC, t-way coverage, (t+k)-way coverage | | x |
| (Nie & Leung, 2011) | x | | | | | | | | | | | | | | | | | | x | x | | | | | x |
| (J. Zhang et al., 2014) | x | | | | | | | | | | | | | | | | | | x | x | | | | | x |
| (Cohen et al., 1997) | x | | | | | | | | | | | | | | | | Computer system | | x | x | | | Code coverage | x | |
| (D. Kuhn et al., 2004) | x | | | | | | | | | | | | | | | | | | x | x | | | Exhaustive testing | | x |
| (Rocklage et al., 2017) | x | | | | | x | | x | x | | x | | | | x | | Cut-in (merge) + Virtual Test Drive | | x | | x | | | x | |
| (Mahmoud & Ahmed, 2015) | x | | | | | | | | | | | | | | | | Software program | | x | x | | Known faults | Comparison to other CA generation methods | x | |
| (Colbourn, 2004) | x | | | | | | | | | | | | | | | | | | x | x | | | | | x |
| (Tuncali et al., 2018) | x | x | | | | x | | x | x | | x | | x | | x | | Jaywalking Pedestrian + STALIRO, Webots | | x | | x | Collision | Comparison to random simulation | x | |
| (Shiba et al., 2004) | x | | | | x | | | | | | | | | | | | | | x | x | | | Compared to AETG | x | |
| (Betts & Petty, 2016) | x | x | x | | x | x | | | x | | x | | | | x | | UAC control | | x | | x | Deviation from optimal path | Comparison to complete set of test cases | x | |
| (Feilhauer et al., 2016) | | x | | | | x | | x | x | | x | | x | | | | proprietary SE | x | | | x | | | x | |
| (Galko et al., 2014) | | x | | | | x | | x | x | | x | | x | | | | LDW + Pro-Sivic | x | | | x | | | x | |

Table 2.: Categorisation of literature in regard to the proposed taxonomy for scenario-based testing strategies (continued)

| Paper | Allocated Strategy | | | | | Modeling Capabilities | | | | Driving Task Level | | | Examined System | | | | Features | | | | | | | Novel | Review |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Combinatorial Testing | Robustness Testing | Random Testing | GR Testing | Evolutionary Testing | Ego | Static Objects | Dynamic Objects | Environment | Navigation | Guidance | Stabilization | Perception | Sensor Fusion | Decision Making | Actuation | Case Study + Simulation Environment (SE) | Specific Parameters | General Framework | Static | Sequential | Criticality Metric | Effectiveness Measure | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| (Kropp et al., 1998) | | x | | | | | | | | | | | | | | | | x | | x | | | | x | |
| (Koopman & Wagner, 2016) | | x | | | | | | | | | | x | | | | x | | x | | x | | | | | x |
| (Hospach et al., 2016) | | x | | | | | | | x | | x | | x | | | | simulated rain affecting camera vision systems + OpenCV | x | | | x | | | x | |
| (Eykholt et al., 2018) | | x | | | | | | | x | | x | | x | | | | traffic sign recognition | x | | | x | | Comparison of classification accuracy between real and perturbed images | x | |
| (Goodfellow et al., 2015) | | x | | | | | | | x | | x | | x | | | | image recognition | x | | | x | | | x | |
| (Petit et al., 2015) | | x | | | | x | | | | | x | | x | | | | "Blinding" camera and LIDAR + HiL | x | | x | | | | x | |
| (Shin et al., 2017) | | x | | | | x | | | | | x | | x | | | | LIDAR spoofing and "blinding" + HiL | x | | x | | | | x | |
| (Hutchison et al., 2018) | | x | | | | x | | | | x | x | x | | x | | | | | x | | x | | | x | |
| (Tuncali et al., 2016) | | x | | | | x | | x | x | | x | | x | | x | | Cut-in scenario + STALIRO | | x | | x | Collision, TTC | | x | |
| (Lei, Li, Liu, Morisset, & Stolz, 2010) | | x | | | | | | | | | | | | | | | | x | | x | | | All states and transitions reached | x | |
| (Zlocki et al., 2015) | | | x | | | x | | x | x | | | | | | | | | | x | | x | | | x | |
| (Alvarez et al., 2017) | | | x | | | | | | | | | | | | | | Rear-end collision + SIMPATO, PRAEDICO | | x | | x | | | | x |

Table 2.: Categorisation of literature in regard to the proposed taxonomy for scenario-based testing strategies (continued)

| Paper | Allocated Strategy | | | | | Modeling Capabilities | | | | Driving Task Level | | | Examined System | | | | Features | | | | | | | Novel | Review |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Combinatorial Testing | Robustness Testing | Random Testing | GR Testing | Evolutionary Testing | Ego | Static Objects | Dynamic Objects | Environment | Navigation | Guidance | Stabilization | Perception | Sensor Fusion | Decision Making | Actuation | Case Study + Simulation Environment (SE) | Specific Parameters | General Framework | Static | Sequential | Criticality Metric | Effectiveness Measure | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| (Althoff & Mergel, 2011) | | | x | | | x | x | x | | x | x | | | | x | | Car following + Matlab | x | | | x | Collision | Comparison to Monte Carlo random simulation | x | |
| (Khastgir et al., 2017) | | | x | | | x | | x | x | | x | | | | x | | Cut-in scenario, AEB + Vitaq | | x | | x | | Functional coverage | x | |
| (Zhao, 2016) | | | x | x | | x | | x | | | x | | | | x | | Car following, lane change + proprietary SE | | x | | x | TTC, Maximum Abbreviated Injury Score | Comparison to Monte Carlo random simulation | x | |
| (de Gelder & Paardekooper, 2017) | | | x | x | | x | | x | | | x | | | | x | | Car following + Matlab/Simulink | | x | | x | TTC | | x | |
| (Gietelink, 2007) | | | x | x | | x | | x | x | | x | | x | | x | | Cooperative ACC system, driver information and warning system, pre-crash warning system + PreScan | | x | | x | TTC, THW, required deceleration, critical braking distance | Comparison to VeHIL tests | x | |
| (Gietelink et al., 2005) | | | x | x | | x | | x | | | x | | | | x | | Traffic jam approach with ACC | | x | | x | Collision | | x | |
| (Gietelink et al., 2006) | | | x | x | | x | | x | | | x | | | | x | | Car following for ACC | | x | | x | TTC | | x | |
| (Zhao et al., 2017) | | | x | x | | x | | x | | | x | | | | x | | Cut-in scenario | | x | | x | TTC, distance between lead and ego, MAIS2+ | Comparison to Monte Carlo random simulation | x | |
| (Mullins et al., 2018) | | | | x | | x | | | x | x | x | | | | x | | Path planning of UUV + proprietary SE | | x | | x | Collision | Sobol and lola-voronoi space filling methods for test generation | x | |
| (Shortle & L'Ecuyer, 2011) | | | | x | | | | | | | | | | | | | | | x | | x | | | | x |
| (Zhao et al., 2018) | | | | x | | x | | x | | | x | | | | x | | Car following | | x | | x | Distance between lead vehicle and ego, MAIS2+ | Comparison to Monte Carlo random simulation | x | |

Table 2.: Categorisation of literature in regard to the proposed taxonomy for scenario-based testing strategies (continued)

| Paper | Allocated Strategy | | | | | Modeling Capabilities | | | | Driving Task Level | | | Examined System | | | | Features | | | | | | | Novel | Review |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Combinatorial Testing | Robustness Testing | Random Testing | GR Testing | Evolutionary Testing | Ego | Static Objects | Dynamic Objects | Environment | Navigation | Guidance | Stabilization | Perception | Sensor Fusion | Decision Making | Actuation | Case Study + Simulation Environment (SE) | Specific Parameters | General Framework | Static | Sequential | Criticality Metric | Effectiveness Measure | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| (Chen et al., 2004) | | | | x | | | | | | | | | | | | | | | x | x | | | Comparison to Monte Carlo random simulation | x | |
| (Chen et al., 2009) | | | | x | | | | | | | | | | | | | | | x | x | | | Comparison to original algorithm | x | |
| (Chen et al., 2010) | | | | x | | | | | | | | | | | | | | | x | x | | | Comparison to Monte Carlo random simulation | x | |
| (Huang et al., 2017) | | | | x | | x | | x | | | x | | | | x | | Cut-in scenario | | x | | x | TTC, speed lead vehicle, distance ego-obj | Comparison to random simulation | x | |
| (Yang & Peng, 2010) | | | | x | | x | | x | | | x | | | | x | | Car following, CAS, collision warning + Matlab | | x | | x | TTC, distance criteria, acceleration criteria | | x | |
| (Bühler & Wegener, 2004) | | | | | x | x | x | | | x | x | | | | x | | parallel parking + Matlab | | x | | x | distance ego-obj, area between ego and obj | | x | |
| (Briand et al., 2005) | | | | | x | | | | | | | | | | | | | | x | | x | Time-to-deadline (computer system) | | x | |
| (Abdessalem et al., 2016) | | | | | x | x | | x | x | | x | | x | | x | | Pedestrian Detection System + PreScan | | x | | x | Distance, TTC | Comparison to random simulation | x | |
| (Abdessalem et al., 2018) | | | | | x | x | | x | x | | x | | x | | x | | AEB + PreScan | | x | | x | TTC | Comparison to NSGA without decision trees | x | |
| (Bühler & Wegener, 2008) | | | | | x | x | x | x | | x | x | | | | x | | Automatic parking system, Brake assistance system | | x | | x | Brake Assist: TTC; Parking system: distance criteria | Comparison to random simulation, manual test case selection | x | |
| (Samatha, Chokkadi, & Yogananda, 2012) | | | | | x | | | | | | x | x | | | x | | Flight control system + Matlab/Simulink | | x | | x | | Block coverage (Simulink), results compared to Taguchi DoE | x | |

**Acknowledgements**

The first author would like to express their thanks to Prof. Andrew Parkes as well as the Reviewers and Editors for their valuable comments on this work.

## References

Abdessalem, R. B., Nejati, S., Briand, L. C., & Stifter, T. (2016). Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st ieee/acm international conference on automated software engineering - ase 2016* (pp. 63–74). New York: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=2970276.2970311

Abdessalem, R. B., Nejati, S., Briand, L. C., & Stifter, T. (2018). Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th international conference on software engineering - icse '18* (pp. 1016–1026). Gothenburg: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=3180155.3180160

ADAC. (2016). *Zahlen, Fakten, Wissen. Aktuelles aus dem Verkehr.* (Tech. Rep.). München: ADAC e.V. Retrieved from https://www.adac.de/{_}mmm/pdf/statistik{_}zahlen{_}fakten{_}wissen{_}1016{_}208844.pdf

Alfonso, J., Sánchez, N., Menéndez, J. M., & Cacheiro, E. (2015, aug). Cooperative ITS communications architecture: the FOTsis project approach and beyond. *IET Intelligent Transport Systems*, *9*(6), 591–598. Retrieved from http://digital-library.theiet.org/content/journals/10.1049/iet-its.2014.0205

Alkim, T., Bootsma, G., & Looman, P. (2007, apr). *The Assisted Driver - Systems that support driving* (Report). Delft: Ministry of Transport, Public Works and Water Management. Retrieved from http://www.fot-net.eu/download/Wiki/the{_}assisted{_}driver.pdf

Alsmadi, I. (2010, may). Using genetic algorithms for test case generation and selection optimization. In *Ccece 2010* (pp. 1–4). Calgary: IEEE. Retrieved from http://ieeexplore.ieee.org/document/5575262/

Althoff, M., & Mergel, A. (2011, dec). Comparison of Markov Chain Abstraction and Monte Carlo Simulation for the Safety Assessment of Autonomous Cars. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1237–1247. Retrieved from http://ieeexplore.ieee.org/document/5875884/

Alvarez, S., Page, Y., Sander, U., Fahrenkrog, F., Helmer, T., Jung, O., ... Op den Camp, O. (2017). Prospective Effectiveness Assessment of ADAS and Active Safety Systems via Virtual Simulation: A Review of the Current Practices. In *25th international technical conference on the enhanced safety of vehicles (esv)* (pp. 17–0346). Detroit: NHTSA. Retrieved from http://www-esv.nhtsa.dot.gov/Proceedings/25/25ESV-000346.pdf

Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., ... Zhu, H. (2013, aug). An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software*, *86*(8), 1978–2001. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0164121213000563

Bao, X., Xiong, Z., Zhang, N., Qian, J., Wu, B., & Zhang, W. (2017, nov). Path-oriented test cases generation based adaptive genetic algorithm. *PLOS ONE*, *12*(11), e0187471. Retrieved from https://dx.plos.org/10.1371/journal.pone.0187471

Barnard, Y., Innamaa, S., Koskinen, S., Gellerman, H., Svanberg, E., & Chen, H. (2016). Methodology for Field Operational Tests of Automated Vehicles. *Transportation Research Procedia*, *14*, 2188–2196. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S235214651630237X

Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2015, may). The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*, *41*(5), 507–525. Retrieved from http://ieeexplore.ieee.org/document/6963470/

Battelle Memorial Institute. (2007, jan). *Evaluation of the Volvo Intelligent Vehicle Initiative Field Operational Test* (Vol. 422637210; Technical report No. FHWA-RD-98-114). Washington D.C.: U.S. Department of Transportation. Retrieved from https://rosap.ntl.bts.gov/view/dot/3687

Betts, K. M., & Petty, M. D. (2016). Automated Search-Based Robustness Testing for Autonomous Vehicle Software. *Modelling and Simulation in Engineering*, *2016*, 1–15. Retrieved from http://www.hindawi.com/journals/mse/2016/5309348/

Bezzina, D., & Sayer, J. (2015, jun). *Safety Pilot Model Deployment* (Vol. DOT HS 812; Research Report). Washington D.C.: NHTSA. Retrieved from https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/812171-safetypilotmodeldeploydeltestcondrtmrep.pdf

Bock, T. (2008). *Vehicle in the Loop - Test- und Simulationsumgebung für Fahrerassistenzsysteme* (1st ed. ed.). Goettingen: Cuvillier. Retrieved from https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=5023312

Bock, T., Maurer, M., & Farber, G. (2007). Validation of the Vehicle in the Loop (VIL); A milestone for the simulation of driver assistance systems. In *2007 ieee intelligent vehicles symposium* (pp. 612–617). Istanbul: IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4290183

Briand, L. C., Labiche, Y., & Shousha, M. (2005). Stress testing real-time systems with genetic algorithms. In *Proceedings of the 2005 conference on genetic and evolutionary computation - gecco '05* (p. 1021). New York: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1068009.1068183

Brown, A., Gonder, J., & Repac, B. (2014). An Analysis of Possible Energy Impacts of Automated Vehicles. In G. Meyer & S. Beiker (Eds.), *Road vehicle automation* (1st ed., pp. 137–153). Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-05990-7{_}13

Bühler, O., & Wegener, J. (2004, mar). Automatic Testing of an Autonomous Parking System Using Evolutionary Computation. *SAE Technical Paper*, *01*(0459), 1 – 8. Retrieved from http://papers.sae.org/2004-01-0459/

Bühler, O., & Wegener, J. (2008, oct). Evolutionary functional testing. *Computers & Operations Research*, *35*(10), 3144–3160. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0305054807000329

Calefato, C., Ferrarini, C., Kutila, M., & Landini, E. (2015). Development of cost efficient ADAS tool platform for automotive industry. In *22nd its world congress* (pp. 5–9). Bordeaux. Retrieved from https://trid.trb.org/view/1411976

Chen, T. Y., Kuo, F.-C., & Liu, H. (2009, sep). Adaptive random testing based on distribution metrics. *Journal of Systems and Software*, *82*(9), 1419–1433. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0164121209001101

Chen, T. Y., Kuo, F.-C., Merkel, R. G., & Tse, T. (2010, jan). Adaptive Random Testing: The ART of test case diversity. *Journal of Systems and Software*, *83*(1), 60–66. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0164121209000405

Chen, T. Y., Leung, H., & Mak, I. K. (2004). Adaptive Random Testing. In M. J. Maher (Ed.), *Advances in computer science - asian 2004. higher-level decision making. asian 2004. lecture notes in computer science, vol 3321* (pp. 320–329). Berlin, Heidelberg: Springer. Retrieved from http://link.springer.com/10.1007/978-3-540-30502-6{_}23

Cohen, D., Dalal, S., Fredman, M., & Patton, G. (1997, jul). The AETG system: an approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, *23*(7), 437–444. Retrieved from http://ieeexplore.ieee.org/document/605761/

Colbourn, C. J. (2004). Combinatorial aspects of covering arrays. *Le Matematiche (Catania)*, *58*(1-2), 121 – 167.

Cunto, F., & Saccomanno, F. F. (2008, may). Calibration and validation of simulated vehicle safety performance at signalized intersections. *Accident Analysis & Prevention*, *40*(3), 1171–1179. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0001457508000055

de Gelder, E., & Paardekooper, J.-P. (2017, jun). Assessment of Automated Driving Systems using real-life scenarios. In *2017 ieee intelligent vehicles symposium (iv)* (pp. 589–594). Los Angeles: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7995782/

Department for Transport. (2019). *Code of Practice: Automated vehicle trialling.* London: Department of Transport. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment{_}data/file/776511/code-of-practice-automated-vehicle-trialling.pdf

Department of Motor Vehicles. (2018). *California Code of Regulations, Title 13, Division 1, Chapter 1, Article 3.7 and Article 3.8.* Department of Motor Vehicles. Retrieved from https://www.dmv.ca.gov/portal/wcm/connect/a6ea01e0-072f-4f93-aa6c-e12b844443cc/DriverlessAV{_}Adopted{_}Regulatory{_}Text.pdf?MOD=AJPERES

Di Fabio, U., Broy, M., Brüngger, R. J., Eichhorn, U., Grunwald, A., Heckmann, D., ... Nehm, K. (2017). *Ethik-Kommission Automatisiertes und Vernetztes Fahren* (Tech. Rep.). Berlin: Bundesministerium für Verkehr und digitale Infrastruktur. Retrieved from https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/bericht-der-ethik-kommission.pdf?{_}{_}blob=publicationFile

Duarte, F., & Ratti, C. (2018, oct). The Impact of Autonomous Vehicles on Cities: A Review. *Journal of Urban Technology*, *25*(4), 3–18. Retrieved from https://www.tandfonline.com/doi/full/10.1080/10630732.2018.1493883

Ervin, R. D. (2005, aug). *Automotive Collision Avoidance System Field Operational Test Report - Methodology and Results* (Vol. DOT HS 809; Final research report No. 99798). Washington D.C.: National Highway Traffic Safety Administration. Retrieved from http://hdl.handle.net/2027.42/49539

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., ... Song, D. (2018, jun). Robust Physical-World Attacks on Deep Learning Models. In *Cvpr 2018.* Salt Lake City. Retrieved from http://arxiv.org/abs/1707.08945

Feilhauer, M., Haering, J., & Wyatt, S. (2016, sep). Current Approaches in HiL-Based ADAS Testing. *SAE International Journal of Commercial Vehicles*, *9*(2), 2016–01–8013. Retrieved from http://papers.sae.org/2016-01-8013/

FESTA. (2016). *FESTA Handbook* (6th ed.). FOT-Net. Retrieved from http://fot-net.eu/wp-content/uploads/sites/7/2017/01/FESTA-Handbook-Version-6.pdf

Galko, C., Rossi, R., & Savatier, X. (2014, jul). Vehicle-Hardware-In-The-Loop system for ADAS prototyping and validation. In *2014 international conference on embedded computer systems: Architectures, modeling, and simulation (samos xiv)* (pp. 329–334). Agios Konstantinos: IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6893229

Geyer, S., Baltzer, M., Franz, B., Hakuli, S., Kauer, M., Kienle, M., ... Winner, H. (2014, may). Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems*, *8*(3), 183–189. Retrieved from http://digital-library.theiet.org/content/journals/10.1049/iet-its.2012.0188

Gietelink, O. (2007). *Design and Validation of Advanced Driver Assistance Systems* (PhD Dissertation). Delft University of Technology.

Gietelink, O., De Schutter, B., & Verhaegen, M. (2005). PROBABILISTIC VALIDATION OF ADVANCED DRIVER ASSISTANCE SYSTEMS. *IFAC Proceedings Volumes*, *38*(1), 97–102. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S1474667016380806

Gietelink, O., De Schutter, B., & Verhaegen, M. (2006). Adaptive importance sampling for probabilistic validation of advanced driver assistance systems. In *2006 american control conference* (p. 6 pp.). Minneapolis: IEEE. Retrieved from http://ieeexplore.ieee.org/document/1657344/

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015, may). Explaining and Harnessing Adversarial Examples. In *Iclr 2015* (pp. 1 – 11). San Diego. Retrieved from http://arxiv.org/abs/1412.6572

Greenblatt, J. B., & Saxena, S. (2015, sep). Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles. *Nature Climate Change*, *5*(9), 860–863. Retrieved from http://www.nature.com/articles/nclimate2685

Hallerbach, S., Xia, Y., Eberle, U., & Koester, F. (2018, apr). Simulation-based Identification of Critical Scenarios for Cooperative and Automated Vehicles. *SAE Technical Paper*, *2018-01*(1066), 1 – 12. Retrieved from http://www.sae.org/content/2018-01-1066/

Hartman, A., & Raskin, L. (2004, jul). Problems and algorithms for covering arrays. *Discrete Mathematics*, *284*(1-3), 149–156. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0012365X0400130X

Hospach, D., Mueller, S., Rosenstiel, W., & Bringmann, O. (2016). Simulation of Falling Rain for Robustness Testing of Video-Based Surround Sensing Systems. In *Proceedings of the 2016 design, automation & test in europe conference & exhibition (date)* (pp. 233–236). Dresden: IEEE. Retrieved from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7459310

Huang, Z., Lam, H., & Zhao, D. (2017, oct). Towards affordable on-track testing for autonomous vehicle — A Kriging-based statistical approach. In *2017 ieee 20th international conference on intelligent transportation systems (itsc)* (pp. 1–6). Yokohama: IEEE. Retrieved from http://arxiv.org/abs/1707.04897http://ieeexplore.ieee.org/document/8317656/

Hutchison, C., Zizyte, M., Lanigan, P. E., Guttendorf, D., Wagner, M., Goues, C. L., & Koopman, P. (2018). Robustness testing of autonomy software. In *Proceedings of the 40th international conference on software engineering: Software engineering in practice - icse-seip '18* (pp. 276–285). New York, New York: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=3183519.3183534

ISO/IEC/IEEE 24765:2010(E). (2010). *Systems and software engineering – Vocabulary* (Vol. 2010; Standard). Geneva: International Organization for Standardization. Retrieved from https://ieeexplore.ieee.org/document/5733835/

Kalra, N., & Paddock, S. M. (2016, dec). 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. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0965856416302129

Kessler, C., Etemad, A., Alessandretti, G., Heinig, K., Selpi, Brouwer, R., ... Benmimoun, M. (2012). *euroFOT Deliverable D11.3* (Tech. Rep. No. Final Report). Aachen: euroFOT Consortium. Retrieved from http://www.eurofot-ip.eu/en/library/deliverables/sp1{_}d113{_}final{_}report.htm

Khastgir, S., Dhadyalla, G., Birrell, S., Redmond, S., Addinall, R., & Jennings, P. (2017, mar). Test Scenario Generation for Driving Simulators Using Constrained Randomization Technique. *SAE Technical Paper*, *2017-01*(1672), 1 – 7. Retrieved from http://papers.sae.org/2017-01-1672/

Koenig, A., Gutbrod, M., Hohmann, S., & Ludwig, J. (2017, mar). Bridging the Gap between Open Loop Tests and Statistical Validation for Highly Automated Driving. *SAE International Journal of Transportation Safety*, *5*(1), 81–87. Retrieved from http://papers.sae.org/2017-01-1403/

Koopman, P., & Wagner, M. (2016, apr). Challenges in Autonomous Vehicle Testing and Validation. *SAE International Journal of Transportation Safety*, *4*(1), 15–24. Retrieved from http://papers.sae.org/2016-01-0128/

Kropp, N., Koopman, P., & Siewiorek, D. (1998). Automated robustness testing of off-the-shelf software components. In *Digest of papers. twenty-eighth annual international symposium on fault-tolerant computing (cat. no.98cb36224)* (pp. 230–239). München: IEEE. Retrieved from http://ieeexplore.ieee.org/document/689474/

Kuhn, D., Wallace, D., & Gallo, A. (2004, jun). Software fault interactions and implications for software testing. *IEEE Transactions on Software Engineering*, *30*(6), 418–421. Retrieved from http://ieeexplore.ieee.org/document/1321063/

Kuhn, D. R., Kacker, R. N., & Lei, Y. (2010). *Practical combinatorial testing* (Tech. Rep.).

Gaithersburg, MD: National Institute of Standards and Technology. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-142.pdf

Lagnoux, A. (2006, jan). RARE EVENT SIMULATION. *Probability in the Engineering and Informational Sciences*, *20*(01), 44 – 66. Retrieved from http://www.journals.cambridge.org/abstract{_}S0269964806060025

LeBlanc, D., Sayer, J., Winkler, C., Ervin, R., Bogard, S., Devonshire, J., ... Gordon, T. (2006, jun). *Road Departure Crash Warning System Field Operational Test: Methodology and Results* (Vol. UMTRI-2006; Final research report). Washington D.C.: National Highway Traffic Safety Administration. Retrieved from https://deepblue.lib.umich.edu/bitstream/handle/2027.42/49242/99788.pdf?sequence=1

Lei, B., Li, X., Liu, Z., Morisset, C., & Stolz, V. (2010, oct). Robustness testing for software components. *Science of Computer Programming*, *75*(10), 879–897. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0167642310000328

Mahmoud, T., & Ahmed, B. S. (2015, dec). An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. *Expert Systems with Applications*, *42*(22), 8753–8765. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0957417415004893

Menzel, T., Bagschik, G., & Maurer, M. (2018, jun). Scenarios for Development, Test and Validation of Automated Vehicles. In *2018 ieee intelligent vehicles symposium (iv)* (pp. 1821–1827). IEEE. Retrieved from http://arxiv.org/abs/1801.08598https://ieeexplore.ieee.org/document/8500406/

Mononen, P., Franzen, S., Pagle, K., Morris, A., Innamaa, S., Karlsson, M., ... Fruttaldo, S. (2013). *TeleFOT Deliverable D1.15* (Tech. Rep. No. Final Report). TeleFOT Consortium.

Mullins, G. E., Stankiewicz, P. G., Hawthorne, R. C., & Gupta, S. K. (2018, mar). Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software*, *137*, 197–215. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0164121217302546

Nagel, H.-H., Enkelmann, W., & Struck, G. (1995, aug). FhG-Co-driver: From map-guided automatic driving by machine vision to a cooperative driver support. *Mathematical and Computer Modelling*, *22*(4-7), 185–212. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/089571779500133M

Najm, W. G., Smith, J. D., & Yanagisawa, M. (2007). *Pre-Crash Scenario Typology for Crash Avoidance Research* (Vol. DOT HS 810; Tech. Rep.). Washington D.C.: National Highway Traffic Safety Administration. Retrieved from https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/pre-crash{_}scenario{_}typology-final{_}pdf{_}version{_}5-2-07.pdf

NHTSA. (2007). *Electronic Stability Control (ESC).* Retrieved from https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/fmvss/ESC{_}FR{_}03{_}2007.pdf

Nie, C., & Leung, H. (2011, jan). A survey of combinatorial testing. *ACM Computing Surveys*, *43*(2), 1–29. Retrieved from http://portal.acm.org/citation.cfm?doid=1883612.1883618

Nieto, J. L., Pou, A., Cacheiro, E., & Diamandouros, K. (2015, oct). *Deliverable D5.8 M48 Final Dissemination Report* (Final report). Retrieved from http://www.fotsis.com/index.php/library/deliverables-old?download=181:d58-m48-final-dissemination-report

Petit, J., Stottelaar, B., Feiri, M., & Kargl, F. (2015). Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. In *Black hat europe* (pp. 1–13). Retrieved from https://www.blackhat.com/docs/eu-15/materials/eu-15-Petit-Self-Driving-And-Connected-Cars-Fooling-Sensors-And-Tracking-Drivers-wp1.pdf

Potters, P. (2009). *Accident Prevention Systems.* Stockholm: Connekt/ITS Netherlands. Retrieved from http://2doubmisw11am9rk1h2g49gq.wpengine.netdna-cdn.com/wp-content/uploads/sites/7/2009/09/04-AOS-PPotters-EN1.pdf

Rocklage, E., Kraft, H., Karatas, A., & Seewig, J. (2017, oct). Automated scenario generation

for regression testing of autonomous vehicles. In *2017 ieee 20th international conference on intelligent transportation systems (itsc)* (pp. 476–483). Yokohama: IEEE. Retrieved from http://ieeexplore.ieee.org/document/8317919/

Roesener, C., Sauerbier, J., Zlocki, A., Fahrenkrog, F., Wang, L., Varhelyi, A., & de Gelder, E. (2017). A Comprehensive Evaluation Approach for Highly Automated Driving. In *25th enhanced safety of vehicles 2017.* Detroit: NHTSA. Retrieved from https://www-esv.nhtsa.dot.gov/Proceedings/25/25ESV-000259.pdf

Ross, C., & Guhathakurta, S. (2017, dec). Autonomous Vehicles and Energy Impacts: A Scenario Analysis. *Energy Procedia*, *143*, 47–52. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S187661021736410X

Rueger, F., Sieber, M., Siegel, A., Siedersberger, K.-H., & Faerber, B. (2015). Kontrollierbarkeitsbewertung von FAS der aktiven Sicherheit in frühen Phasen des Entwicklungsprozesses mit dem Vehicle in the Loop (VIL). In C. Stiller (Ed.), *10. workshop fahrerassistenzsysteme 2015* (pp. 149–159). Darmstadt: Uni-DAS e.V.

SAE. (2018). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* (Tech. Rep.). Author. Retrieved from https://doi.org/10.4271/J3016{_}201806

Saeed, A., Ab Hamid, S. H., & Mustafa, M. B. (2016, dec). The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review. *Applied Soft Computing*, *49*, 1094–1117. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S1568494616304240

Samatha, K., Chokkadi, S., & Yogananda, J. (2012). A Genetic Algorithm Approach for Test Case Optimization of Safety Critical Control. *Procedia Engineering*, *38*, 647–654. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S1877705812019935

Sayer, J., LeBlanc, D., Bogard, S., Funkhouser, D., Bao, S., Buonarosa, M. L., & Blankespoor, A. (2011, jun). *Integrated Vehicle-Based Safety Systems Field Operational Test Final Program Report* (Vol. UMTRI-2010; Tech. Rep.). Washington D.C.: U.S. Department of Transportation. Retrieved from http://www.nhtsa.gov/DOT/NHTSA/NVS/CrashAvoidance/TechnicalPublications/2011/811482.pdf

Schuldt, F., Menzel, T., & Maurer, M. (2015). Eine Methode für die Zuordnung von Testfällen für automatisierte Fahrfunktionen auf X-in-the-Loop Verfahren im modularen virtuellen Testbaukasten. In C. Stiller (Ed.), *10. workshop fahrerassistenzsysteme 2015* (pp. 171–182). Darmstadt: Uni-DAS e.V.

Schuldt, F., Saust, F., Lichte, B., Maurer, M., & Scholz, S. (2013). Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen. *AAET2013 - Automatisierungssysteme, Assistenzsysteme Und Eingebettete Systeme Für Transportmittel*, 114–134. Retrieved from http://www.digibib.tu-bs.de/?docid=00052570

Schulze, M., Mäkinen, T., Kessler, T., Metzner, S., & Stoyanov, H. (2014, jul). *DRIVE C2X Deliverable D11.6* (Vol. D11.6; Project Deliverable No. Final Report). Sindelfingen: DRIVE C2X Consortium. Retrieved from https://www.eict.de/fileadmin/redakteure/Projekte/DriveC2X/Deliverables/DRIVE{_}C2X{_}D11{_}6{_}Final{_}report{_}{_}full{_}version{_}.pdf

Shiba, T., Tsuchiya, T., & Kikuno, T. (2004). Using artificial life techniques to generate test cases for combinatorial testing. In *Proceedings of the 28th annual international computer software and applications conference, 2004. compsac 2004.* (pp. 72–77). Hong Kong: IEEE. Retrieved from http://ieeexplore.ieee.org/document/1342808/

Shin, H., Kim, D., Kwon, Y., & Kim, Y. (2017). Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications. In W. Fischer & N. Homma (Eds.), *Cryptographic hardware and embedded systems - ches 2017* (pp. 445–467). Taipei: Springer, Cham. Retrieved from http://link.springer.com/10.1007/978-3-319-66787-4{_}22

Shortle, J. F., & L'Ecuyer, P. (2011, jan). Introduction to Rare-Event Simulation. In J. J. Cochran (Ed.), *Wiley encyclopedia of operations research and management science* (1st ed.). Hoboken, NJ, USA: John Wiley & Sons, Inc. Retrieved from http://doi.wiley.com/

10.1002/9780470400531.eorms0006
Sippl, C., Bock, F., Wittmann, D., Altinger, H., & German, R. (2016). From Simulation Data to Test Cases for Fully Automated Driving and ADAS. *Testing Software and Systems. ICTSS 2016. Lecture Notes in Computer Science*, *9976*, 191–206. Retrieved from http://link.springer.com/10.1007/978-3-319-47443-4{_}12
Sobhani, A., Young, W., Sarvi, M., & Bahrololoom, S. (2013). Using a simulation based road safety index to assess risk of turning behaviour at signalised intersections. In *Proceedings of world conference in transport research* (pp. 1 – 15).
Statistisches Bundesamt (Destatis). (2018). *Verkehrsunfälle* (Vol. 7; Tech. Rep.). Statistisches Bundesamt (Destatis). Retrieved from https://www.destatis.de/DE/Publikationen/Thematisch/TransportVerkehr/Verkehrsunfaelle/VerkehrsunfaelleMonat/VerkehrsunfaelleM2080700181014.pdf?{_}{_}blob=publicationFile
Stellet, J. E., Vogt, P., Schumacher, J., Branz, W., & Zollner, J. M. (2016, jun). Analytical derivation of performance bounds of autonomous emergency brake systems. In *2016 ieee intelligent vehicles symposium (iv)* (pp. 220–226). Gothenburg: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7535389/
Stellet, J. E., Zofka, M. R., Schumacher, J., Schamm, T., Niewels, F., & Zollner, J. M. (2015, sep). Testing of Advanced Driver Assistance Towards Automated Driving: A Survey and Taxonomy on Existing Approaches and Open Questions. In *2015 ieee 18th international conference on intelligent transportation systems* (pp. 1455–1462). Las Palmas: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7313330/
Tsugawa, S., & Kato, S. (2010, nov). Energy ITS: another application of vehicular communications. *IEEE Communications Magazine*, *48*(11), 120–126. Retrieved from http://ieeexplore.ieee.org/document/5621978/
Tuncali, C. E., Fainekos, G., Ito, H., & Kapinski, J. (2018, jun). Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components. In *2018 ieee intelligent vehicles symposium (iv)* (pp. 1555–1562). Changshu: IEEE. Retrieved from http://arxiv.org/abs/1804.06760https://ieeexplore.ieee.org/document/8500421/
Tuncali, C. E., Pavlic, T. P., & Fainekos, G. (2016, nov). Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles. In *2016 ieee 19th international conference on intelligent transportation systems (itsc)* (pp. 1470–1475). Rio de Janeiro: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7795751/
Uberbacher, M., Wolze, P., & Burtsche, T. (2017). Sicherheitsfunktionen erlebbar testen. *ATZ Automobiltech Z (ATX - Automobiltechnische Zeitschrift)*, *119*(7-8), 60–63.
Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., & Maurer, M. (2015, sep). Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In *2015 ieee 18th international conference on intelligent transportation systems* (pp. 982–988). Las Palmas: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7313256/
van der Horst, R., & Hogema, J. (1993). Time-to-Collision and Collision Avoidance Systems. In *6th ictct workshop* (pp. 1 – 12). Salzburg: ICTCT. Retrieved from http://www.ictct.org/migrated{_}2014/ictct{_}document{_}nr{_}365{_}Horst.pdf
Wachenfeld, W. (2017). *How Stochastic can Help to Introduce Automated Driving* (PhD Dissertation, Technische Universität Darmstadt). Retrieved from http://tuprints.ulb.tu-darmstadt.de/5949/
Watzenig, D., & Horn, M. (Eds.). (2017). *Automated Driving*. Cham: Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-31895-0
Waymo. (2017). *On the Road to Fully Self-Driving* (Tech. Rep.). Retrieved from https://waymo.com/safetyreport/
White, L., & Cohen, E. (1980, may). A Domain Strategy for Computer Program Testing. *IEEE Transactions on Software Engineering*, *SE-6*(3), 247–257. Retrieved from http://ieeexplore.ieee.org/document/1702726/
Winner, H., Hakuli, S., Lotz, F., & Singer, C. (Eds.). (2016). *Handbook of Driver Assistance Systems*. Cham: Springer International Publishing. Retrieved from http://link.springer

.com/10.1007/978-3-319-12352-3

Yang, H.-H., & Peng, H. (2010, dec). Development and evaluation of collision warning/collision avoidance algorithms using an errable driver model. *Vehicle System Dynamics*, *48*(sup1), 525–535. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/00423114.2010.515745

Young, W., Sobhani, A., Lenné, M. G., & Sarvi, M. (2014, may). Simulation of safety: A review of the state of the art in road safety simulation modelling. *Accident Analysis & Prevention*, *66*, 89–103. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0001457514000128

Zhang, J., Zhang, Z., & Ma, F. (2014). *Automatic Generation of Combinatorial Test Data*. Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-662-43429-1

Zhang, T., Tao, D., Qu, X., Zhang, X., Lin, R., & Zhang, W. (2019, jan). The roles of initial trust and perceived risk in public's acceptance of automated vehicles. *Transportation Research Part C: Emerging Technologies*, *98*, 207–220. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S0968090X18308398

Zhao, D. (2016). *Accelerated Evaluation of Automated Vehicles* (PhD Dissertation, University of Michigan). Retrieved from https://deepblue.lib.umich.edu/handle/2027.42/120657

Zhao, D., Huang, X., Peng, H., Lam, H., & LeBlanc, D. J. (2018, mar). Accelerated Evaluation of Automated Vehicles in Car-Following Maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, *19*(3), 733–744. Retrieved from http://ieeexplore.ieee.org/document/7933977/

Zhao, D., Lam, H., Peng, H., Bao, S., LeBlanc, D. J., Nobukawa, K., & Pan, C. S. (2017, mar). Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques. *IEEE Transactions on Intelligent Transportation Systems*, *18*(3), 595–607. Retrieved from http://ieeexplore.ieee.org/document/7534875/

Zhou, J., Schmied, R., Sandalek, A., Kokal, H., & del Re, L. (2016, apr). A Framework for Virtual Testing of ADAS. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, *9*(1), 2016–01–0049. Retrieved from http://papers.sae.org/2016-01-0049/

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., ... Zeeb, E. (2014). Making Bertha Drive—An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, *6*(2), 8–20. Retrieved from http://ieeexplore.ieee.org/document/6803933/

Zlocki, A., Eckstein, L., & Fahrenkrog, F. (2015, jan). Evaluation and Sign-Off Methodology for Automated Vehicle Systems Based on Relevant Driving Situations. *Transportation Research Record: Journal of the Transportation Research Board*, *2489*, 123–129. Retrieved from http://trrjournalonline.trb.org/doi/10.3141/2489-14

Zofka, M. R., Kohlhaas, R., Schamm, T., & Zollner, J. M. (2014, jun). Semivirtual simulations for the evaluation of vision-based ADAS. In *2014 ieee intelligent vehicles symposium proceedings* (pp. 121–126). Dearborn: IEEE. Retrieved from http://ieeexplore.ieee.org/document/6856593/

Zofka, M. R., Kuhnt, F., Kohlhaas, R., Rist, C., Schamm, T., & Zöllner, J. M. (2015). Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems. In *18th international conference on information fusion (fusion)* (pp. 1422–1428). Washington D.C.: IEEE. Retrieved from https://ieeexplore.ieee.org/document/7266724/