

# A Survey on Automated Driving System Testing: Landscapes and Trends

SHUNCHENG TANG, ZHENYA ZHANG, YI ZHANG, JIXIANG ZHOU, YAN GUO, SHUANG LIU, SHENGJIAN GUO, YAN-FU LI, LEI MA, YINXING XUE, and YANG LIU

*Automated Driving Systems (ADS)* have made great achievements in recent years thanks to the efforts from both academia and industry. A typical ADS is composed of multiple modules, including sensing, perception, planning, and control, which brings together the latest advances in different domains. Despite these achievements, safety assurance of ADS is of great significance, since unsafe behavior of ADS can bring catastrophic consequences. Testing has been recognized as an important system validation approach that aims to expose unsafe system behavior; however, in the context of ADS, it is extremely challenging to devise effective testing techniques, due to the high complexity and multidisciplinary of the systems. There has been great much literature that focuses on the testing of ADS, and a number of surveys have also emerged to summarize the technical advances. Most of the surveys focus on the system-level testing performed within software simulators, and they thereby ignore the distinct features of different modules. In this paper, we provide a comprehensive survey on the existing ADS testing literature, which takes into account both module-level and system-level testing. Specifically, we make the following contributions: (1) we survey the module-level testing techniques for ADS and highlight the technical differences affected by the features of different modules; (2) we also survey the system-level testing techniques, with focuses on the empirical studies that summarize the issues occurring in system development or deployment, the problems due to the collaborations between different modules, and the gap between ADS testing in simulators and the real world; (3) we identify the challenges and opportunities in ADS testing, which pave the path to the future research in this field.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; • **Software and its engineering** → *Software verification and validation*; • **Security and privacy** → Systems security; • **Computing methodologies** → Artificial intelligence.

Additional Key Words and Phrases: ADS testing, module-level testing, system-level testing, system security

## ACM Reference Format:

Shuncheng Tang, Zhenya Zhang, Yi Zhang, Jixiang Zhou, Yan Guo, Shuang Liu, Shengjian Guo, Yan-Fu Li, Lei Ma, Yinxing Xue, and Yang Liu. 2023. A Survey on Automated Driving System Testing: Landscapes and Trends. *ACM Forthcoming* 1, 1 (January 2023), 61 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

## 1 INTRODUCTION

With the aim of bringing convenient driving experience, increasing driving safety and reducing traffic congestion, *automated driving systems (ADS)*, a.k.a. *self-driving cars* have attracted significant attention from both academia and industry. According to the statistics from a recent report [1],

---

Authors' addresses: S. Tang, Y. Zhang, J. Zhou, Y. Guo and Y. Xue are with University of Science and Technology of China, China, Z. Zhang is with Kyushu University, Japan, S. Liu is with College of Intelligence and Computing, Tianjin University, China, S. Guo is with Baidu Security, USA, Y. Li is with Department of Industrial Engineering, Tsinghua University, China, L. Ma is with University of Alberta, Alberta Machine Intelligence Institute, Canada, and Kyushu University, Japan and Y. Liu is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. Y. Xue is corresponding author, email: yxxue@ustc.edu.cn.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2023 Association for Computing Machinery.

XXXX-XXXX/2023/1-ART \$15.00

<https://doi.org/XXXXXXXX.XXXXXXX>

the autonomous car market was valued for more than 22 billion dollars in 2021. However, the state-of-the-practice ADS are still vulnerable to numerous safety and security threats, due to either complicated external environments or deliberate attacks from various sources. These threats may lead to system failure, which could bring catastrophic consequences and unacceptable losses [2]. Despite the rapid progress that has been made so far, safety assurance of ADS is still a major challenge to their full-scale industrialization. Some recent news, e.g., the report of Tesla's fatal accident [3], further highlights the importance of research in the safety assurance of automated driving.

In general, an ADS is composed of several modules for the functionalities of sensing, perception, planning and control. The sensing module collects and preprocesses the environmental data using a number of intelligent sensors, such as camera, radar, and LiDAR. The perception module extracts information from the sensors to understand the environmental conditions, including road conditions, obstacles, and traffic signs. Based on the output of the perception module, the planning module generates the optimal driving trajectories which are expected to be followed by the ADS. Lastly, the control module sends the lateral and longitudinal control signals to drive the ADS along the planned trajectories. In particular, some ADS adopt a special *end-to-end* design that integrates the perception, planning and control functionalities in a single module. These modules collaborate with each other and jointly decide the behavior of the ADS [4]; the abnormal function of any module can lead to system failures, which severely threatens the safety and security of the ADS.

Testing has been an effective approach to exposing potential problems and ensuring the safety of systems. However, the testing of ADS is known to be extremely challenging, due to the complexity and multidisciplinary nature of those systems. In recent years, there have been a surge of studies that focus on ADS testing. These published papers span over mainstream venues in various domains, such as transportation venues (e.g., ITSC, IV), software engineering venues (e.g., ICSE, ASE), artificial intelligence venues (e.g., CVPR, AAAI), and security venues (e.g., CCS, USENIX Security), which tackle the challenges in ADS testing from various perspectives (see the detailed statistics and analysis in §3.2). Numerous testing approaches have been proposed for solving different problems, and numerous bugs and vulnerabilities have been reported to facilitate the system reengineering that repairs the existing problems and ensures the system safety.

To better understand the landscape of ADS testing, there have been a number of surveys [5–9] that summarize the recent advances in this field. Grigorescu et al. [5] investigate the deep learning techniques for different modules of ADS and discuss the safety risks of these techniques. Rosique et al. [6] analyze the characteristics of the common sensors used for perception, as well as the performance of different simulators for the simulation of perception systems. Zhang et al. [7] present a literature review on the techniques for identification of critical scenarios, in which they point out the necessity of combining different scenario identification methods for safety assurance of ADS. Zhong et al. [8] review the works about scenario-based testing in high-fidelity simulators, and discuss the gap between the virtual environment and the real-world environment. Jahangirova et al. [9] propose a set of driving quality metrics and oracles for ADS testing, and demonstrate the effectiveness of combining the 26 best metrics as the functional oracles.

Most of the existing surveys view the ADS under study as a whole and investigate the methodologies of ADS testing from the perspective of the system level. In that case, as a typical problem setting, ADS testing consists in generating critical scenarios that can lead to system failures, such as collisions with obstacles. In addition, because of the high cost of testing ADS in the real world, most of the studies in these surveys adopt software simulators as the testing environments. While these surveys are useful, they are not sufficient to show the comprehensive landscape of ADS testing. Indeed, since ADS are complex and composed of multiple modules that differ from each other in technical design, their testing should capture the features of different modules and address

the challenges in different domains. Moreover, at the system level, the testing should concern the problems arising from the collaborations between different modules, and highlight the gap between simulation-based testing and real-world testing.

**Contributions.** To bridge this gap, we conduct a survey on ADS testing that focuses on both module-level testing and system-level testing. Specifically, at the module level, we reveal the distinction of the testing techniques for different modules due to their different features; at the system level, we focus on the challenges introduced by the cooperation between different modules and also discuss the different levels of realism of the testing environments. In particular, we answer the following research questions in this survey:

- **RQ1:** What are the techniques adopted for testing different modules of an ADS?
- **RQ2:** What are the techniques adopted for system-level testing of ADS?
- **RQ3:** What are the challenges and opportunities in the field of ADS testing?

In order to answer these questions, we make the following contributions in this paper:

- To answer **RQ1**, we survey the testing techniques for the different modules of ADS, and in particular, we highlight the technical differences in these testing techniques due to the characteristics of different modules;
- To answer **RQ2**, we survey the system-level testing techniques, with a focus on the following:
  - First, we review the existing empirical studies on the issues/bugs in public reports and repositories. These studies reveal the system issues occurring in their development or deployment, and show a bird’s-eye view on the potential system problems without running them;
  - Second, we study the existing investigations on the safety problems at the system level, when different modules collaborate and interact with each other during the running of the systems;
  - Third, we focus on the gap between simulation-based testing and real-world testing, which is an emerging topic of great importance, in order to understand the quality of testing.
- To answer **RQ3**, we identify the challenges and potential research opportunities for ADS testing, based on our survey results.

To the best of our knowledge, our work is the first one that unveils the intrinsic differences and challenges in ADS testing w.r.t. different modules; meanwhile, we give a specific emphasis on the comparison between the currently popular simulation-based testing and real-world testing. Moreover, our analysis and discussion on the challenges and opportunities exhibit the landscapes, and stimulate future research in this important field.

**Paper organization.** The rest of the paper is organized as follows: §2 overviews the background of the ADS; §3 describes the survey methodology, including the detailed scope, collection process, and collection results. The main results of this survey are in §4, §5 and §6. In §4, we survey the literature of empirical study on ADS testing; in §5, we survey the literature of techniques on module-level ADS testing and answer RQ1; in §6, we survey the literature of techniques on system-level ADS testing and answer RQ2. We then show the statistics and analysis of the works in §7. We summarize the challenges and potential research directions in §8 and answer RQ3. Lastly, we conclude this survey in §9.

## 2 PRELIMINARIES

Nowadays, autonomous systems have been deployed in various application domains, such as transportation, robotics, and healthcare, and they have made huge differences to our daily lives. In this work, we pay particular attention to *automated driving systems (ADS)*, i.e., *self-driving cars*, as a typical example to exemplify the concerns in the quality aspects of those systems. In this section, we first provide an overview of the categorization of ADS according to the levels of automation,

Table 1. Automation Levels and Definitions by SAE

Level	Name	Description	Example
0	No Driving Automation	Drivers perform all of the DDT	LDW
1	Driver Assistance	The system performs part of the DDT: either steering or acceleration/deceleration	ALC or ACC
2	Partial Driving Automation	The system performs part of the DDT: steering and acceleration/deceleration	ALC and ACC
3	Conditional Driving Automation	Drivers or fallback-ready users need to be receptive to ADS-issued requests	Traffic jam chauffeur
4	High Driving Automation	The system performs all of the DDT and DDT fallback within a specified ODD	Local driverless taxis
5	Full Driving Automation	The system performs all of the DDT and DDT fallback without ODD limitation	Full autonomous vehicles

from L0 to L5; then we show the general architecture of ADS; lastly, we showcase four open-source ADS and one simulation platform.

## 2.1 Overview of Automated Driving Systems

According to the complexity and variety of the ADS, the *society of automotive engineers (SAE)* proposed the taxonomy and definitions of driving automation systems, known as *SAE J3016*<sup>1</sup>, which has become a classification standard in recent years. This standard categorizes driving automation systems into six levels, ranging from *no driving automation* (Level 0) to *full driving automation* (Level 5). These levels are usually referred to as L0 to L5.

The definitions of the systems from L0 to L5 are as follows: (1) L0 systems only perform warnings and momentary interventions, such as *Lane Departure Warning (LDW)* and *Automated Emergency Braking (AEB)*, and the drivers need to perform all of the *dynamic driving tasks (DDT)*; (2) L1 systems support steering or acceleration/deceleration for drivers; example features include *Automated Lane Centering (ALC)* and *Adaptive Cruise Control (ACC)*; (3) L2 systems perform steering and acceleration/deceleration at the same time, and a typical L2 system should support both ALC and ACC; (4) L3 systems can execute responses to driving conditions within *Operational Design Domain (ODD)*, which is an operational restriction imposed to the ADS at the design stage, but these systems require fallback-ready people to handle system failures; an example is a traffic jam chauffeur; (5) L4 systems can further support the system fallback, and an example is a local driverless taxi; (6) L5 systems can handle all driving conditions. These are shown in Table 1.

A system in L0 to L2 is also known as an *advanced driver assistance system (ADAS)*, since it is only in charge of a part of the DDT, such as lateral control or longitudinal control, and the safety of the whole vehicle still relies on drivers. In contrast, a system in L3-L5 performs all of the DDT and drivers are not expected to interfere during the driving process, so it realizes the real *automated driving*.

Note that there exist other identified synonyms of *Automated Driving*, e.g., *autonomous driving*, *self-driving*, but in this paper, we follow the relevant terminology from *SAE J3016*, in which the term “ADS” refers to *Automated Driving System*. In literature [8], an ADAS is usually referred to as a system that belongs to L0-L2, while an ADS is referred to as a system that belongs to L3-L5. In

<sup>1</sup>[https://www.sae.org/standards/content/j3016\\_202104/](https://www.sae.org/standards/content/j3016_202104/)

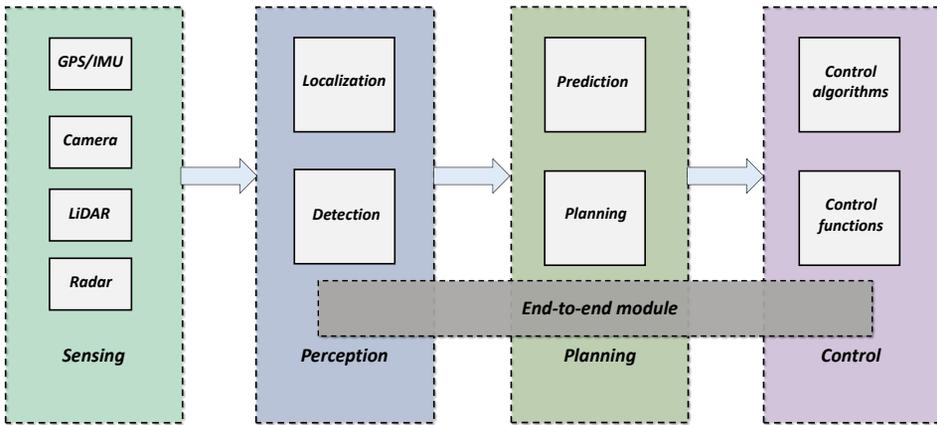


Fig. 1. The typical architecture of an ADS

this survey, since many testing techniques are independent of the automation levels of the systems under test, we sometimes mix the use of the terms and, by ADS, we refer to the systems over all of the levels of driving automation.

## 2.2 Architecture of ADS

A common ADS is composed of four functional modules, namely, *the sensing module*, *the perception module*, *the planning module* and *the control module*, as shown in Fig. 1. In the sensing module, intelligent sensors (e.g., camera, radar and LiDAR) are used to collect the driving context from the physical world. The perception module extracts useful environmental information from the sensor data, and sends it to the planning module for motion planning. Based on the information, the planning module generates the optimal driving trajectory. Lastly, the control module outputs the control commands to drive the vehicle along the trajectory. Moreover, some modern ADS adopt a special design named *end-to-end module*. In the remainder of this section, we elaborate on the functionalities of these modules in the typical architecture of an ADS.

**Sensing module.** By adopting various physical sensors, the sensing module takes charge of collecting and preprocessing driving environmental information from the physical world. The common sensors used by an ADS include *Global Positioning Systems (GPS)*, *inertial measurement units (IMU)*, cameras, *radio detection and ranging (radar)*, and *light detection and ranging (LiDAR)*. Specifically, GPS provides the absolute position data (e.g., latitude, longitude and heading angle) while IMU provides the temporal data (e.g., acceleration and angular velocity). The combination of these two sensors can provide more accurate real-time positioning of the autonomous vehicles. Cameras are used to record and capture visual information on the driving road for the perception module, and radar is used to detect obstacles by radio waves. Nowadays, LiDAR has become an indispensable component in many leading ADS (e.g., APOLLO and AUTOWARE), since it can collect 3D point cloud data and process it with higher measurement accuracy. In comparison with camera sensors that are sensitive to light conditions (e.g., shadows and bright sunlight), LiDAR sensors are more robust under these environments, and the generated 3D point cloud can be further utilized to build 3D models that better characterize the surrounding objects.

**Perception module.** With the help of deep learning techniques, the perception module processes sensor data (e.g., pictures and 3D point cloud) from the sensing module to accomplish a series of perception tasks, such as localization, detection, and prediction.

- *Localization* provides the real-time location of the ADS during the driving process. Furthermore, localization is mostly implemented by fusing the data from GPS, IMU and LiDAR. Specifically, the 3D point cloud data of LiDAR are used to match the features stored in a *High-Definition (HD) Map*, in order to determine the most likely location.
- *Detection* includes lane detection, traffic light detection, and object detection. The data of camera are often used for lane detection and traffic light detection, while the data of camera, radar and LiDAR are often fused by several algorithms (e.g., extended *Kalman filters* [10]) for object detection. These detection tasks are mostly implemented by using *deep neuron networks (DNNs)*, such as faster RCNN [11] and Yolov3 [12].

The prediction task also benefits from the perception module and is mainly used for trajectory planning. We leave the introduction of this task below in the planning module.

**Planning module.** By using DNNs and planning algorithms, the planning module takes perception data as input and makes decisions for the control module to control the vehicle. It has two submodules, namely, the *prediction submodule* and the *planning submodule*.

- *The prediction submodule* estimates the future trajectories of the moving objects (e.g., vehicles and pedestrians) detected by the perception module. For a given moving object, the possibility of its path is often evaluated by machine learning (ML) algorithms, e.g., LSTM, RNN.
- *The planning submodule* generates the optimal driving trajectory for ego vehicle based on the prediction results. Specifically, this module is responsible for three tasks, namely, *route planning*, *behavior planning* and *motion planning*.
  - Route planning selects the optimal path for the vehicle by using path algorithms, such as *Dijkstra* and *A\**;
  - Behavior planning makes decisions for the actions taken by the ADS, such as lane changing and car following, based on the system requirements and traffic rules;
  - Motion planning generates velocity and steering angle plans which are locally optimal, by considering several factors, including safety, efficiency, and comfort.

**Control module.** Based on the trajectories planned by the planning module, the control module finally takes charge of the longitudinal and lateral control of the vehicle. By using control algorithms (e.g., *proportional integral derivative (PID) control* [13] and *model predictive control (MPC)* [14]), this module generates appropriate control commands (e.g., steering and braking) and sends them to the related hardware, i.e., the *electronic control unit (ECU)*, via the protocol of *controller area network (CAN) bus*. Note that, this module is critical for several functionalities provided by the ADS, including ACC, AEB, and *Lane Keeping Assistance (LKA)*.

**End-to-end module.** As shown in Fig. 1, besides the common modules mentioned above, there exists another *end-to-end* design that combines the perception, planning, and control processes in one module. To be specific, this module mainly consists of special deep learning models, which are trained by labeled data that maps information from sensors directly to the corresponding control commands. Consequently, these models could output the control commands based on the current driving environment.

### 2.3 Open-Source Systems and Tool Stacks

In this section, we introduce four open-source ADS, namely, APOLLO, AUTOWARE, OPENPILOT and PYLOT, and one simulation platform called BEAMNG [15]. The first three ADS have been widely adopted for commercial usage in practice [16–18] and the last ADS is from academia.

**APOLLO.** APOLLO has been a popular open-source ADS developed by *Baidu* since 2017; as of Dec. 2021, it has been updated to version 7.0.0. The hardware platform of APOLLO includes camera,

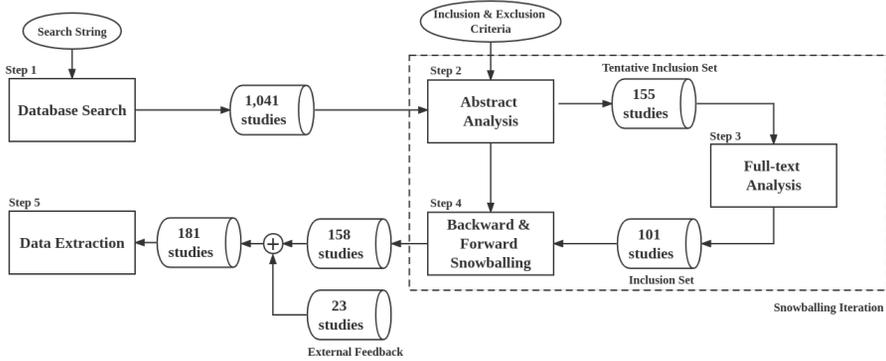


Fig. 2. Overview of the paper collection methodology

LiDAR, millimeter wave radar, and Human-Machine Interface (HMI) device, and currently the communications over different components are managed by CyberRT [19]. The functionalities of APOLLO include cruising, urban obstacle avoidance, and lane changing.

**AUTOWARE.** AUTOWARE is another open-source L4 ADS developed by the research group of Nagoya University in 2015. Though it is mainly applicable to urban roads, it also suits highways and other road conditions. By using the sensors introduced in §2.2, it supports a series of functionalities including connected navigation, traffic light recognition, and object tracking. Unlike APOLLO which uses CyberRT, AUTOWARE adopts ROS [20] for communications over different components.

**OPENPILOT.** OPENPILOT is a popular open-source L2 ADAS developed by Comma.ai and it has been updated to version 0.8.12 as of Dec. 2021. OPENPILOT supports common L2 features, such as ACC, ALC, and Forward Collision Warning (FCW). Unlike other L2 ADAS, OPENPILOT has a high degree of portability—it can be compatible with more than 120 types of vehicle models by using related hardware set (e.g., Car Harness [21] and Comma Two [22]).

**PYLOT.** PYLOT [23] is a modular and open-source autonomous driving platform developed by UC Berkeley in 2021. For achieving the trade-off between latency and accuracy, it is built on a deterministic dataflow system called ERDOS [24]. PYLOT also has other built-in features such as modularity, portability, and debuggability, which allow researchers to implement or test ADS functions with higher efficiency.

**BEAMNG.** BEAMNG [25] is a popular image-generating simulation platform, which has been widely used in the Search-Based Software Testing competition (SBST) [26]. Specifically, it is based on a physically-accurate engine that can support customized vehicle models and realistic damage. For example, different components of a vehicle can have different degrees of deformation after a collision. In addition, BEAMNG also contains a driving agent called BEAMNG.AI [27], which could take over one or more vehicles and drive in several different modes.

### 3 PAPER COLLECTION METHODOLOGY AND RESULT

In this section, we introduce our paper collection methodology in §3.1 and present the statistics and analysis of the results in §3.2.

### 3.1 Paper Collection Methodology

This section introduces the methodology adopted in our paper collection process, which is illustrated in Fig. 2. The overall process consists of five main steps, namely, *database search*, *abstract analysis*, *full-text analysis*, *backward & forward snowballing*, and *data extraction*. The intermediate results of each step during the process are reported in our supplementary website<sup>2</sup> and are also available in *Zenodo* [28]. We now describe the details of each step, as follows:

**3.1.1 Database Search.** This step aims to find the potentially relevant papers by searching in electronic databases. Specifically, we select *DBLP*<sup>3</sup> as our database, which is a popular bibliography database containing a comprehensive list of research venues in computer science. Moreover, our search targets the titles of the papers, since the title often conveys the theme of a paper. We optimize the search string in an iterative manner, in order to collect as many related papers as possible. The final search string used during our search process is shown as follows:

((“automated vehicle” OR “automated driving” OR “autonomous car” OR “autonomous vehicle” OR “autonomous driving” OR “self-driving” OR “driver assistance system” OR “intelligent system” OR “intelligent vehicle” OR “intelligent agent”)  
AND  
 (“test” OR “attack” OR “validation” OR “evaluation” OR “quality assurance” OR “quality assessment” OR “oracle” OR “mutation” OR “fuzzing”))

The first group of terms (above “AND”) represents the identified synonyms of automated driving, which contains the terms such as “autonomous driving” and “self-driving”; the second group of terms (below “AND”) contains the common phases in the process of quality assessment of software systems (e.g., “test” and “validation”), along with popular testing approaches (e.g., “mutation” and “fuzzing”) and a keyword “oracle”, which is a significant concept in software testing. The terms in each group are connected with *OR* operator, while the two groups are connected with *AND* operator, which means that a relevant paper should cover the characteristics of both groups. Overall, the application of the above search string on *DBLP* retrieves 1185 papers. After removing 144 duplicates, the final number of papers we collected is 1041.

**3.1.2 Abstract Analysis.** To determine whether each primary candidate paper is relevant to ADS testing, we perform a manual analysis on the abstracts of 1041 papers obtained from database search in §3.1.1. This process is conducted by two assessors, i.e., the first two authors, following the inclusion and exclusion criteria formulated as follows:

- Inclusion Criteria:
  - IC1 papers that propose a method for testing the modules of ADS or the whole system;
  - IC2 papers that introduce metrics as test oracles or adequacy criteria for testing the modules of ADS or the whole system;
  - IC3 papers published between January 2015 and June 2022.
- Exclusion Criteria:
  - EC1 preprint papers or non-peer-reviewed papers;
  - EC2 early results or preliminary studies;
  - EC3 papers that do not target ADS;
  - EC4 survey papers or summary papers;
  - EC5 papers that do not focus on assessing quality aspects of ADS or its components;

<sup>2</sup><https://sites.google.com/view/ads-testing-survey>

<sup>3</sup><https://dblp.org/>

**EC6** papers that focus on other quality aspects such as Human-Machine Interface (HMI), cyber security, and adversarial defense.

Specifically, for **IC2**, test oracle refers to the metrics that measure whether an ADS or its components misbehave, and test adequacy refers to the criteria that judge whether a test suite has been sufficient for testing; for **EC3**, papers related to other intelligent systems, e.g., *unmanned aerial vehicle (UAV)*, are excluded since we mainly focus on automated driving systems; for **EC4**, such relevant papers are discussed in §1 for a comparison with our survey; for **EC5**, papers that do not report novel techniques or metrics for ADS testing are excluded, e.g., the papers that focus on the engineering implementation of testbeds; for **EC6**, only the studies that assess quality aspects, e.g., safety and security of the ADS or its modules, are considered.

As a result of manual analysis of the abstracts, the two assessors fully agree on the inclusion of 101 papers, and have divergent opinions on 54 papers, i.e., those included by one assessor but excluded by the other. To cover more relevant studies, in this step, papers included by either one assessor or both assessors are all added into a *tentative inclusion set*, which contains 155 papers.

**3.1.3 Full-text Analysis.** In this step, we download the papers in the *tentative inclusion set* and conduct a full-text analysis. For those papers included by both assessors, we further analyze the introduction, conclusion, or other parts to determine whether a certain paper proposes an approach or a metric for ADS testing. If the assessors' decisions conflict, the two assessors will first review the inclusion and exclusion criteria defined in §3.1.2, and have a discussion. In cases where the conflict still exists, a senior researcher will join the discussion and resolve the dispute. After an agreement on removing 54 irrelevant papers, the number of the inclusion set is 101.

**3.1.4 Backward & Forward Snowballing.** In order to reduce the risk of missing relevant papers, we perform both backward snowballing and forward snowballing [29] on the 101 papers in the *inclusion set*, and the process is assigned to the two assessors. In backward snowballing, they check the reference list in the existing studies to obtain candidate papers, while in forward snowballing, they use *Google Scholar*<sup>4</sup> to access the papers that cite the existing studies. For those candidate papers produced by snowballing, the two assessors apply the inclusion criteria and exclusion criteria defined in §3.1.2, and conduct both *Abstract Analysis* in §3.1.2 and *Full-text Analysis* in §3.1.3 to identify the relevant papers that could be added into the *inclusion set*. As a result of performing snowballing for one iteration, we identify 57 new papers that are relevant to ADS testing. Hence, the number of papers in the inclusion set after snowballing is 158. To avoid missing relevant papers that may not be obtained through our collection process, we also ask for feedback from domain experts and collect 23 papers as a result. Finally, we collect 181 papers for data extraction.

**3.1.5 Data Extraction.** In this step, all the resulting 181 papers are thoroughly read by the authors. Specifically, the authors need to identify the testing target (ADS module or system) and the proposed method or metrics for ADS testing. The identified information is then extracted into a *data extraction form*. Since the data extraction process requires careful reading of each paper, this task is conducted by three authors as the assessors to share the overall workload. Each assessor is assigned more than 50 papers, and to ensure accuracy, the extracted information from the three assessors is all reviewed in parallel by another author. All conflicting decisions are resolved in the discussion at this stage.

## 3.2 Paper Collection Results

In this section, we analyze the collected papers from three perspectives, namely, the publication venues, the targeted system modules, and the publication years. We show the distribution of the

<sup>4</sup><https://scholar.google.com/>

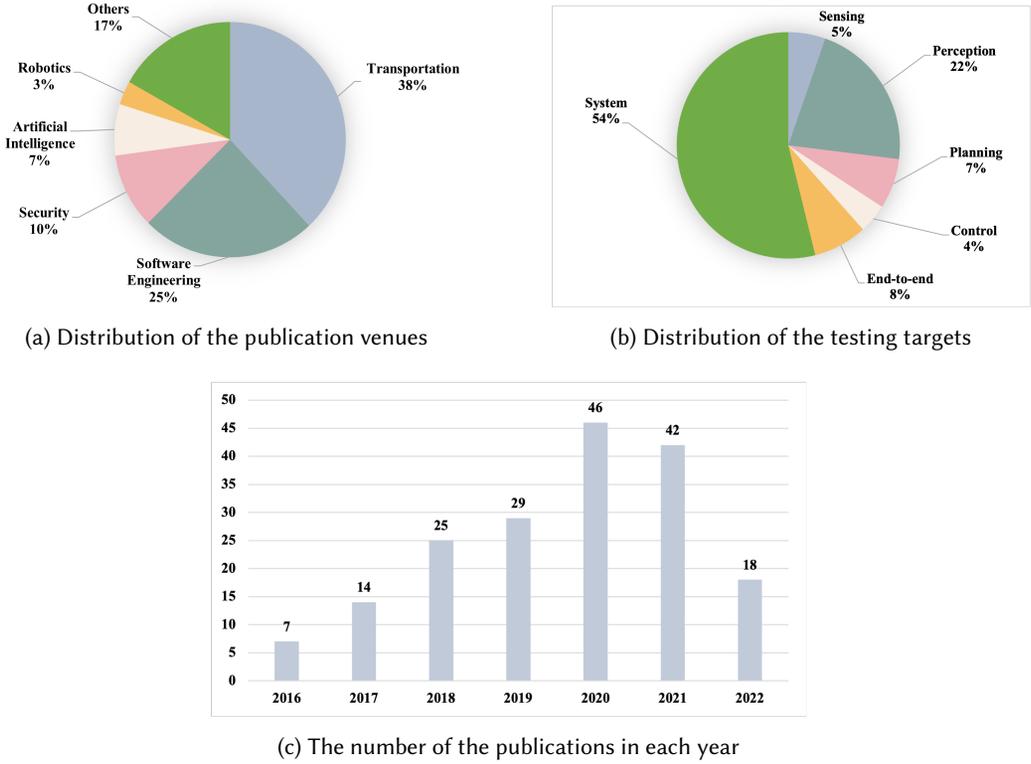


Fig. 3. The statistical information of the publications in ADS testing

publication venues of all the papers in Fig. 3a and the distribution of the targeted modules/system in Fig. 3b. Moreover, we present the number of papers published in different years in Fig. 3c.

**Publication venues.** In Fig. 3a, we can see that, (i) many of the papers, up to 38%, are published in transportation venues such as *International Conference on Intelligent Transportation Systems (ITSC)* and *IEEE Intelligent Vehicles (IV)*; (ii) 25% of the papers are published in software engineering venues such as *International Conference on Software Engineering (ICSE)* and *International Conference on Automated Software Engineering (ASE)*; (iii) the adversarial attack methods for vulnerability detection of ADS are related to the security of the systems, and hence 10% are published in the security venues, such as *USENIX Security Symposium* and *IEEE Symposium on Security and Privacy (S&P)*; (iv) since the ADS and artificial intelligence are closely related, 7% of the papers are published in artificial intelligence venues such as *Computer Vision and Pattern Recognition Conference (CVPR)* and *AAAI Conference on Artificial Intelligence (AAAI)*.

**Target modules.** In Fig. 3b, we can see that, obviously, the papers on system-level testing dominate the largest percentage, up to 54%. These papers involve testing techniques that span over both simulation-based testing and mixed-reality testing. Moreover, the number of the papers concerning with the perception module is the second largest, up to 22%. The perception module takes charge of object detection and image semantic segmentation using deep learning, which is important but vulnerable to safety and security threats, and thus becomes a popular research direction. Compared to the perception module, there are fewer papers concerning other modules, such as the planning module or the control module.

**Publication year.** In Fig. 3c, we can see that the number of the papers related to ADS testing shows a general ascending trend, from 2015 to 2021. This trend indicates that the safety and security of ADS are attracting more and more research attention from researchers. The reason for fewer relevant papers in 2022 is that only partial papers had been published by the time we collected the papers.

## 4 LITERATURE OF EMPIRICAL STUDY ON ADS TESTING

In this section, we provide an overview of the papers that perform empirical study in the field of ADS testing. By the term of empirical study, we mean that, instead of executing the systems in a simulated or real-world environment, these studies perform empirical analysis based on existing databases, such as project repositories and public crash reports. In general, empirical study is an essential step before the experimental ADS testing, since it provides experiences and insights in the distribution of potential safety risks.

We classify these studies into three categories, namely, *system study*, *bug/issue study* and *public report study*. System study, shown in §4.1, mainly analyzes the architectures of ADS and is thus beneficial for understanding the system behavior before running it. Bug/issue study, shown in §4.2, focuses on collecting and analyzing the bugs and issues of ADS, which are usually raised by users, developers, and researchers and published in project repositories. Public report study, shown in §4.3, refers to the analysis on those real-world disengagements and crashes reported in various databases (e.g., the crash reports released by *California Department of Motor Vehicles (CADMV)* [30]). These reports target at real-world system failures, and they provide important references for understanding system reliability in the real world.

### 4.1 System Study

Because of the high complexity of the system architectures of ADS, it is necessary to have a comprehensive understanding of the systems before performing their evaluation. The system studies, e.g., on APOLLO [31], build the logical architectures that disclose the connections over different modules in ADS. As a result, these lines of work can bring insights into the potential vulnerabilities and suggest useful metrics for system testing.

Peng et al. [31] investigate the collaboration between the code and the DNN models in APOLLO; specifically, they study which roles are played by the code and the underlying DNN models, respectively. They find that the 28 DNN models used in APOLLO interact with each other in diverse ways, e.g., the output of one DNN can be used as the input of another DNN, and the outputs of multiple DNNs can be combined as the input of another DNN. Moreover, the code also plays an important role in the system workflow, e.g., it can be used for filtering out invalid output of DNNs, and it can complement the imperfect outcome of DNNs.

### 4.2 Bug/Issue Study

For those open-source ADS, issues and bugs reported in their public repositories (e.g., GitHub) reflect real problems encountered by users and developers during the development and deployment. Therefore, systematic analysis on these issues [32, 33] can provide insights into the root causes of system failures. In this section, we review two studies [34, 35] in the field of ADS testing.

Garcia et al. [34] present a comprehensive study of bugs in two ADS, namely APOLLO and AUTOWARE. Specifically, they collect bugs from the commits across the APOLLO and AUTOWARE repositories in GitHub and perform a manual analysis on these bugs and commits. As a result, they obtain 13 root causes (e.g., algorithm, data, memory) for system crashes, 20 symptoms (e.g., speed and velocity control, vehicle trajectory) and 18 bug-related components (e.g., perception, planning, control), based on their analysis of 499 bugs in the two ADS.

Tang et al. [35] perform a study on issue analysis for OPENPILOT. They collect 235 bugs from 1293 pull requests and 694 issues of the OPENPILOT project in GitHub and Discord<sup>5</sup>. These bugs are then classified into 5 categories, including (DNN) model bugs, plan/control bugs, car bugs, hardware bugs, and UI bugs. Among these different types of bugs, they find that the car bugs related to the interface with different car models dominate 31.48%, and plan/control bugs related to the control of car behaviors account for 25.95%.

### 4.3 Public Report Study

The following works all perform analysis on public reports, i.e., CADMV [30], which is a database involving disengagement and crash records on public roads. Specifically, a *disengagement* refers to a failure that requires a human driver to take over control of the vehicle; a *crash* refers to a collision with other traffic participants. These empirical studies investigate the relevant factors, such as the causes, the correlations, and the impacts of these system failures, and they also shed light on future system developments.

**Analysis of disengagement reports.** In the works [36–38], the authors analyze the disengagements based on different metrics. Lv et al. [36] classify the disengagement events into two types, namely *active disengagement* and *passive disengagement*, and investigate the root causes of each group. Boggs et al. [37] apply the binary logistic regression [39] to categorize the cause of the disengagements in more details. The results show that the planning discrepancy (e.g., improper localization, motion planning) accounts for 41% of ADS disengagements. Khattak et al. [38] investigate the relationship between disengagements and crashes, and find relevant factors that could increase the likelihood of a disengagement without a crash.

**Analysis of crash reports.** The following works [40–46] analyze the crash reports and identify the contributing factors. Leilabadi et al. [40] apply text analysis to the crash reports, and they find that the crashes mostly occur when vehicles run in the automated mode, and the most frequent ADS crash type is the rear-end collision. Favaro et al. [41] focus on the dynamics aspect and present the speed distribution of those crash vehicles. Wang et al. [42] adopt regression and classification tree (CART) to investigate the types and severity of these crashes. They find that the severity increases significantly when an automated vehicle is responsible for the event. Das et al. [43] utilize Bayesian latent class model to perform the analysis and identify six collision patterns. Aziz et al. [44] investigate both crash data involving ADS and without ADS, and build a spatial-temporal mapping of the contributing factors between them. Song et al. [45] conclude that the most representative crash pattern is the “collision following ADS stop”, i.e., an automated vehicle stops suddenly and gets hit by other vehicles on the road. Besides CADMV, the crash data in other databases such as UK’s STATS19 [47] are also analyzed with statistical approaches [46, 48].

### 4.4 Discussion

Table 2 summarizes the collected papers that empirically study the issues in ADS testing. Several existing system studies focus on APOLLO, and there are also studies that cover other open-source ADS, such as AUTOWARE and OPENPILOT. Moreover, there are many works [36–38, 40, 41, 43–46, 49] that target the disengagement and crash reports for identifying the root causes or failure types, as these investigations are critical to understanding the ADS safety performances in the real world.

<sup>5</sup><https://discord.com/>

Table 2. Summary of the papers for empirical study on ADS testing

Category	Description	Literature
System study	Introducing the interaction between the code and the DNN models in APOLLO	[31]
Bug/issue study	Finding the root causes, symptoms, and bug-related components based on analysis on bugs of APOLLO and AUTOWARE	[34]
	Performing categorization and analysis on bugs of OPENPILOT	[35]
Public report study	The analysis and classification of the disengagements based on different perspectives (e.g., modules)	[36–38]
	The identification of the common crash types by different methods (e.g., text analysis)	[40–46, 48]

**Summary:** Many empirical studies focus on studying the systems themselves, e.g., APOLLO, to understand the characteristics of the systems or bugs/issues from their public project repositories. There are also many studies that analyze the public crash reports to understand the safety problems of ADS in the real world.

## 5 LITERATURE OF TECHNIQUES ON MODULE-LEVEL ADS TESTING

In this section, we introduce the works on module-specific testing for ADS with the goal of answering RQ1 in §1. These modules under test include the ones that have been introduced in §2.2, namely, the sensing module (in §5.1), the perception module (in §5.2), the planning module (in §5.3), the control module (in §5.4) and the end-to-end module (in §5.5).

We introduce these studies from three perspectives, namely, test methodology, test oracle and test adequacy. Concretely, (i) test methodology introduces various methods or technical innovations for testing; (ii) test oracle defines metrics that can be used to judge whether the module behaves correctly; (iii) test adequacy proposes coverage criteria that tell if the test cases in a test suite are sufficient. Note that, due to the different features of different modules, it can be the case that, for a specific module, not all of the three perspectives are identified as important scientific topics, so we may only introduce the related literature from only a part of the perspectives.

### 5.1 Sensing Module

The sensing module is the frontier module of an ADS and the performance of the physical sensors (e.g., camera, radar, LiDAR) in this module is critical to the safety and security of the whole ADS. Relevant studies on the test methodology of this module can be divided into *physical testing* (shown in §5.1.1) and *deliberate attack* (shown in §5.1.2). Physical testing aims to test the performance of the sensors under different physical conditions, while deliberate attack interferes with the input signals of the sensors to diminish the sensing quality.

**5.1.1 Physical Testing.** Physical testing [50, 51] aims to assess the sensors' capabilities of handling specific tasks under different physical environments, such as harsh weather conditions. Kutila et al. [50] perform a detection distance testing of LiDAR in the foggy and snowy conditions. The results show that the maximum measurable distance by the LiDAR decreases by 20 – 40m under harsh weather conditions. They also compare the detection capability of LiDAR with different wavelengths in their follow-up work [51]. Concretely, they test the detection accuracy of LiDAR at 905nm and 1550nm wavelengths in foggy and rainy weather, and the results indicate that the LiDAR with a larger wavelength can detect the environment more accurately when the visibility is low.

Table 3. Summary of the papers for the sensing module testing

Methodology	Description	Literature	Test Sensor	
Physical testing	Testing the detection distance of LiDAR under different weather conditions	[50, 51]	LiDAR	
Deliberate attack	Jamming attack	Using intense light to blind the sensors	[52]	LiDAR
		Utilizing a laser and a jammer to interfere with the sensors	[53]	Camera and ultrasonic sensors
		Placing an opposite ultrasonic sensor	[54]	Ultrasonic sensors
	Spoofing attack	Modifying the raw data	[55, 56]	GPS
		Utilizing a Software Defined Radio	[57]	Radar
	Creating invisible objects with simple LEDs	[58]	Camera	

**5.1.2 Deliberate Attack.** Unlike physical testing, deliberate attack refers to the intentional attacks launched by human attackers. This type of attack on the sensors of ADS can be classified into *jamming attack* and *spoofing attack*, which are introduced below.

**Jamming attack.** This is a basic type of attack on sensors by generating noises using specific tools to interfere with the sensors and damage their normal functionalities. Shin et al. [52] propose a blinding attack method against LiDAR by using intense light with the same wavelength as the target sensor. Yan et al. [53] utilize a laser to cause irreversible damage to cameras and an ultrasonic jammer to interfere with ultrasonic sensors. Another attack on ultrasonic sensors [54] works by placing an ultrasonic sensor opposite to the target sensor.

**Spoofing attack.** Spoofing attack is performed by injecting fake data to deceive sensors. Meng et al. [55] and Zeng et al. [56] spoof the GPS receivers to a wrong destination by modifying the raw signals of these sensors. Komissarov et al. [57] utilize a *Software Defined Radio* to fool the *mmWave radar*, e.g., they make it produce the wrong measurement of vehicle speed. Wang et al. [58] first utilize the features of *infrared lights* to perform spoofing attack. Specifically, the proposed approach could create invisible objects with simple LEDs to fool the camera sensors, and thus introduce localization errors to the vehicles.

**5.1.3 Discussion.** Table 3 shows the summary of the papers for the sensing module testing. It can be seen that the existing physical testing works [50, 51] mainly focus on testing LiDAR sensors under different weather conditions, e.g., the foggy weather, the snowy weather. This is because the LiDAR sensor has become a key component in ADS, and its robustness is of great significance to the vehicle's safety. Besides, we find more works that perform deliberate attack including *jamming attack* [52–54] and *spoofing attack* [55–58] on other physical sensors. With the usage of specific devices, e.g., lasers [53] and LEDs [58], these two types of attacks have been demonstrated to be effective for finding abnormal behaviors of the target sensors.

**Summary:** Physical testing focuses on testing sensors under different weather conditions. There are more works performing deliberate attack on the sensing module, e.g., jamming attack and spoofing attack, with the usage of specific hardware devices.

## 5.2 Perception Module

The perception module receives and processes sensor data; based on that, it perceives external environments. The literature we collected includes the test methodologies (shown in §5.2.1), the

test oracles (shown in §5.2.2), and the test adequacy criteria (shown in §5.2.3) for testing the DNN models used in the perception module of ADS.

**5.2.1 Testing Methodology.** Adversarial attack is the major approach for testing the DNN models used in the perception module, which attempts to generate *adversarial examples* to trigger wrong inference results of perception. Based on the attacker’s knowledge about the target model, adversarial attacks can be classified into *white-box attacks*, in which the attackers have access to the training parameters of the target model, and *black-box attacks*, in which the attackers have limited or no knowledge of the model. Based on the attackers’ desired outcomes, there exist *targeted attacks*, in which the prediction that the model makes is limited to specific classes, and *non-targeted attacks*, in which the model can predict an arbitrary wrong class [59]. In general, there are three basic methods for performing adversarial attack, namely, by solving an optimization problem, by leveraging the *generative adversarial networks (GAN)* [60] and by poisoning the training data. In the following, we introduce the literature that adopts these methods.

**Optimization-based attack.** We denote by  $F$  a DNN model, which takes as input a picture  $x$  and gives as output a label  $y$ . In general, an adversarial attack consists in solving the following optimization problem:

$$\min \delta \quad s.t. \quad F(x + \delta) = y^*, \quad y^* \neq y^o \quad (1)$$

where  $\delta$  is a perturbation added to the picture  $x$ , and  $y^*$  is a wrong label that is different from the correct label  $y^o$ . In other words, an adversarial attack involves finding the minimum perturbation that leads a DNN model to the wrong inference result. In most cases, the collected literature on adversarial attack follows this general framework; meanwhile, these papers also differ in their applications and motivations.

The following works [61–67] focus on performing adversarial attacks on camera-based perception tasks (e.g., object detection, traffic sign recognition, and semantic segmentation). Chen et al. [61] propose an attack method, called *ShapeShifter*, to generate perturbations against the object detector *Faster R-CNN* [11]. To make the perturbations more robust, they adopt the *Expectation over Transformation* technique [68] that adds random distortions iteratively, during the optimization process for generating perturbations. Zhao et al. [62] propose two approaches for generating adversarial perturbations: one is called *hiding attacks* that can make object detectors unable to recognize objects; and the other is *appearing attack* that can lead the object detectors to make incorrect recognition. Zhang et al. [63] propose an attack method for object detectors, which could generate camouflage on 3D objects, i.e., vehicles, and make it undetectable by target models. Unlike the classification loss adopted by most studies, Choi et al. [64] consider the object loss defined as the detector’s confidence on the existence of objects in an area. The adversarial perturbations generated by their approach could make the target object detector *YOLOv4* [69] produce numerous false positives, i.e., those objects that do not exist in the clean images are unexpectedly detected. Xu et al. [65] perform an adversarial attack on the popular segmentation model *DeepLab-V3+* [70]. The perturbations generated are quite small and can be stealthily projected to an unnoticed area in the original image. Li et al. [66] propose the first black-box attack on traffic sign recognition models, which could generate adversarial perturbations efficiently. Kumar et al. [67] present another black-box attack method on traffic sign recognition models. Instead of maximizing the loss of the correct class, they accelerate the convergence through minimizing the loss of the class which is incorrectly predicted by target models.

In addition to attacking the camera-based object detectors, there are also works [71–79] that focus on attacking the LiDAR-based 3D object detectors. Cao et al. [71] present a white-box attack method on a LiDAR-based perception module by adding the spoofed points into the original 3D

point clouds. In the later work [72], their generated adversarial perturbations could fool both the camera and the LiDAR-based perception algorithms. Black-box attacks on the LiDAR-based object detectors are performed in [73–76] and experimental results show that the target models are highly sensitive to those adversarial 3D perturbations. Yang et al. [77] consider both white-box and black-box scenarios and generate perturbations for roadside objects such that they can be misidentified as vehicles by the perception module. Zhu et al. [78] generate perturbations for roadside objects but they target LiDAR-based semantic segmentation tasks. Unlike existing works that generate perturbations for 3D objects, Li et al. [79] add perturbations to the vehicle trajectories and their method can result in a significant drop in the precision of the object detector, to nearly zero.

While the adversarial attack framework in Eq. 1 is effective in fooling DNN models, it does not consider the realism of the perturbed pictures. There is literature that considers the adversarial attack problem under physical conditions. Eykholt et al. [80, 81] propose an attacking method, called *Robust Physical Perturbations (RP<sub>2</sub>)*, that induces road sign classifiers to produce wrong classification results under real-world physical conditions, e.g., different viewpoint angles and different distances to the signs. Experimental results show that the attacked classifier misclassifies the traffic signs with a rate of 100% in the lab environment and 84.8% in the real world.

**GAN-based attack.** This type of attack [82] generates adversarial perturbations to fool a DNN model by training a GAN [60]. A GAN consists of two neural network models, namely, a generator  $G$  and a discriminator  $D$ ; specifically,  $G$  is used to generate perturbations and add them to an input image, and  $D$  is used to distinguish the generated image by  $G$  and the original image. The objective of training a generator  $G$  is to make the perturbed image of  $G$  indistinguishable by the discriminator  $D$ ; this can be implemented by optimizing a loss function  $L_G$ . For fooling the target DNN, another loss function  $L_D$  is needed to stimulate the adversarial images produced by the GAN to be misclassified. As a result, the final objective function is formalized as follows:

$$L = \gamma \cdot L_G + L_D$$

where  $\gamma$  is a parameter that controls the relative importance of  $L_G$  and  $L_D$ .

Liu et al. [83] propose a GAN-based attack framework called *perceptual-sensitive GAN (PS-GAN)*, which generates adversarial patches with high visual fidelity. Experimental results show that the adversarial patches can significantly reduce the classification accuracy of the target DNNs. Xiong et al. [84] propose a multi-source attack method based on GAN, which generates adversarial perturbations that can fool both camera-based and LiDAR-based perception models. Yu et al. [85] utilize the *cycle-consistent generative adversarial network (CycleGAN)* [86] to synthesize corner cases for testing traffic sign detection models.

**Trojan attack.** This type of attack [87] is also called *poisoning attack* or *backdoor attack*. Specifically, it works by injecting malicious samples with *trigger patterns* into the training data of the target DNN models. Then the models can learn the malicious behaviors and make incorrect predictions when the inputs contain such triggers. The following works [88, 89] are all based on this idea.

Jiang et al. [88] utilize *particle swarm optimization* [90] to perform this type of attack on traffic sign recognition models. Experimental results show that the classification accuracy could drop to 62% due to only 10% injected training data. Ding et al. [89] propose the Trojan attack for deep generative models such as DeRaindrop Net [91], which is a GAN-based network for raindrops removal. Experimental results show that the model could be triggered to misclassify the traffic light or the value on the speed limit sign when it normally removes the raindrops.

**5.2.2 Test Oracle.** A test oracle defines a metric used to distinguish between the expected and unexpected behavior of the system under test. Sometimes, an oracle is obviously identified; however,

that is not always the case. In the perception testing, due to the huge input space (that involves all the possible input images) of the DNN models, it is a great challenge to specify the oracles for all the input images. We collect several types of test oracles that have been adopted for perception testing, namely, ground-truth labeling [92, 93], metamorphic testing [94–99] and formal specifications [100, 101] to judge whether a bug exists in the perception module.

**Ground-truth labeling.** The general approach of testing a DNN in the perception module is to match the inferred label by the DNN with the ground-truth label, given an image. Usually, these ground-truth labels are obtained by manual labeling. For instance, the ground-truth labels in [102, 103] are produced in this way. However, manual labeling is notoriously expensive and laborious; to that end, automatically labeling methods are pursued by researchers. Zhou et al. [92] propose an automatic labeling method to detect the *road* component in the camera sensor images. Their method identifies the road component in the 3D point cloud captured by a LiDAR for the same scene, and projects the identified area onto the corresponding image. The projected area labels the *road* component in the camera sensor images, which can be used for the validation of semantic segmentation models. Philipp et al. [93] propose another approach for automatically generating dimension and classification references for object detection. The dimension references are calculated by considering the occurred situations of each object and measuring the related features, e.g., projection angle, based on a given HD map. The classification references are generated by a decision tree, which considers the features such as kinematic behavior and the interaction with infrastructure elements of each object.

**Metamorphic testing.** Metamorphic testing [104] was introduced by Chen et al. to tackle the problem when the test oracle is absent in traditional software testing. Consider the testing of a program  $f$  that implements the trigonometric function  $\sin$ . Normally, for any input  $x$ , given the ground-truth value  $\sin(x)$  as the oracle for  $f(x)$ , we can assess the correctness of  $f$  by checking if  $f(x) = \sin(x)$ . However, assume that the ground-truth value  $\sin(x)$  is unknown. In this case, testing  $f$  by checking if  $f(x) = \sin(x)$  is not possible; instead, we can use the *metamorphic testing* that tests the program based on a metamorphic relation. For instance, in this case, a metamorphic relation can be built as  $f(x) = f(\pi - x)$ , due to the property  $\sin(x) = \sin(\pi - x)$  held by  $\sin$ . Hence, the correctness of  $f$  can be assessed by metamorphic testing, which consists in checking if  $f(x) = f(\pi - x)$ , for any input  $x$ .

Metamorphic testing has been studied for testing the perception module of an ADS; various metamorphic relations have been proposed, over images [94–98] and frames in a scenario [99].

- *Metamorphic relations over images.* Shao et al. [94] introduce a metamorphic relation in object detection, that is, the detected object in the original images should also be detected in the synthetic images. For testing traffic light recognition models, Bai et al. [95] propose another metamorphic relation, which states that, when traffic lights change from one color to another, the recognition results of the target models should change correspondingly. Zhou et al. [96] propose a metamorphic relation for LiDAR-based object detection, that is, the noise points outside the *Region of Interest (ROI)* should not affect the detection of objects within the *ROI*. Woodlief et al. [97, 98] check the model inconsistencies between original images and mutated images, e.g., the images of a vehicle with changed color.
- *Metamorphic relations over frames in a scenario.* Ramanagopal et al. [99] propose two metamorphic relations, respectively for identifying temporal and stereo inconsistencies that exist in different frames of a scenario. The temporal metamorphic relation says that an object detected in a previous frame should also be detected in a later frame; the stereo metamorphic relation is defined in a similar way, for regulating the spatial consistency of the objects in different frames of a scenario.

**Formal specifications.** Recently, temporal logics-based formal specifications have been adopted in the monitoring of the perception module of ADS. In general, temporal logics are a family of formalism used to express temporal properties of systems, e.g., an event should *always* happen during a system execution; flagship temporal logics include *linear temporal logic (LTL)* [105] and *metric temporal logic (MTL)* [106]. Dokhanchi et al. [100] propose an adaptation of temporal logic to express desired properties of perception; the new formalism is called *Timed Quality Temporal Logic (TQTL)*. Specifically, TQTL can be used to express temporal properties that should be held by the perception module during object detection, e.g., “*whenever a lead car is detected at a frame, it should also be detected in the next frame*”. Conceptually, the properties expressed by TQTL are similar to the ones in [99]; however, by adopting such a formal specification to express these properties, one can synthesize a monitor that automatically checks the satisfiability of the system execution. TQTL is later extended to *Spatio-Temporal Quality Logic (STQL)* [101], which has enriched syntax to express more refined properties over the bounding boxes used in object detection. The authors also propose an online monitoring framework, named *PerceMon*, for monitoring the perception module at runtime of the ADS.

**5.2.3 Test Adequacy.** Measuring the adequacy of the testing for DNN models in the perception module is challenging, due to the complexity of DNN models. Compared to program execution, DNN inference involves a completely different logical process, which is deemed to be non-interpretable. In this domain, various metrics, analogous to the test adequacy criteria for programs, have been proposed; some of the metrics are for general DNN testing, while some are dedicated to ADS testing. Below, we introduce two typical lines of such adequacy criteria.

**Structural coverage.** Neuron coverage is proposed in [107], inspired by the structural coverage used in traditional software testing. Pei et al. [107] analogize DNN inference to program execution, and consider the *neuron activation* as a symbol that indicates whether a neuron is “covered”. Based on this analogy, they define *neuron coverage* by what percentage of the neurons that are activated, as the counterpart of structural coverage in DNN. Inspired by [107], a number of other neuron coverage criteria are proposed. For instance, *k-multisection neuron coverage* [108] is the refined version of neuron coverage that considers not only “activated” neurons but also “not activated” neurons; *surprise adequacy* [109] pursues the novelty of an individual test case based on whether it is out of the distribution of the training data.

**Combinatorial coverage.** Combinatorial testing [110] utilizes *combinatorial coverage* for test case generation, which measures the coverage of the combinations of different system parameters. The *t*-way combination coverage is a typical criterion, which is defined by the number of the *t*-wise combinations covered by the test suite, out of the total number of possible *t*-wise combinations. For instance, consider a system that has 3 binary parameters *a*, *b* and *c*. Given a test suite  $T = \{\langle 0, 0, 1 \rangle, \langle 0, 1, 0 \rangle, \langle 1, 0, 0 \rangle, \langle 1, 1, 0 \rangle\}$  that involves 4 test cases, the 2-way combination of *T* is computed by  $\frac{1}{3}$ , which indicates that one combination *ab* is covered by *T* (since *T* involves all the possible cases 00, 01, 10, 11 of *ab*), over all the three possible combinations *ab*, *ac*, and *bc*.

Combinatorial coverage has been used to solve the adequacy problem in the testing of the perception module. Gladisch et al. [111] characterize the scenarios by using multiple parameters concerning different features, such as lane types and road types. They then apply combinatorial coverage as a guidance to generate test cases that can reveal system failures and achieve high coverage. Cheng et al. [112] propose *k*-projection coverage that aims to reduce the combinatorial explosion during test case generation, by incorporating domain expertise. Xia et al. [113] utilize the *analytic hierarchy process* to identify the key factors and then generate test cases for a lane detection algorithm with combinatorial coverage guarantee.

Table 4. Summary of the papers for the perception module testing: Part I

Methodology	Description	Literature	Test Objective	Environment
Optimization-based attack	Replacing true traffic signs with generated adversarial traffic signs	[61]	Object detector: Faster-RCNN[11]	Real world
	Generating transferable adversarial traffic signs and stickers	[62]	Object detectors: Faster-RCNN[11] and YOLOv3 [12]	Real world
	Generating camouflage on 3D objects	[63]	Object detectors: Mask R-CNN [114] and YOLOv3-SPP [12]	Simulation
	Focusing on the objectness loss	[64]	Object detector: YOLOv4 [69]	Digital dataset
	Adding perturbations to the unnoticed area	[65]	Segmentation models: ResNet-101 [115] and MobileNet [116]	Digital dataset
	Performing black-box attacks on traffic sign recognition models	[66, 67]	Models from the Kaggle Competition [117]	Digital dataset
	Adding spoofed points into the original 3D point clouds	[71]	Perception module of APOLLO	Simulation
	Generating adversarial images against multi-sensor fusion based perception	[72]	Perception module of APOLLO	Simulation
	Performing the black-box attack by the occlusion information	[73–76]	Perception module of APOLLO and LiDAR-based object detectors [118–120]	Digital dataset
	Generating perturbations for roadside objects	[77, 78]	LiDAR-based object detectors [118, 119, 121] and a segmentation model [122]	Real world
	Adding perturbations to the trajectories of vehicles	[79]	Object detectors: PointRCNN [118] and PointPillar++ [123]	Digital dataset
	Pasting generated adversarial stickers on traffic signs	[80, 81]	Classifiers: LISA-CNN [124], GTSRB-CNN [125] and Inception-v3 [126]	Digital dataset
GAN-based attack	Generating adversarial patches with high visual fidelity	[83]	Classifiers: VGG16 [127], ResNet [115] and VY [128]	Digital dataset
	Proposing a multi-source attack method	[84]	Semantic segmentation model: VAE-GAN [129]	Digital dataset
	Synthesizing corner cases by utilizing CycleGANs	[85]	Object detector: PatchGAN [130]	Digital dataset
Trojan attack	Utilizing particle swarm optimization	[88]	Classifier: LeNet-5 [131]	Digital dataset
	Performing the Trojan attack for models used for raindrops removal	[89]	Classifiers: DeRaindrop Net [91] and RCAN [132]	Digital dataset

**5.2.4 Discussion.** Table 4 summarizes the collected papers for testing the perception module. This module includes a number of DNN models for understanding the environmental information. It is important, since many crashes are caused due to the vulnerabilities of this module [34]. It can be seen that a large number of papers perform adversarial attacks on this module. These methods include three categories, namely, optimization-based attack, GAN-based attack, and Trojan attack. The first two types of methods could generate *adversarial examples* to fool the DNN models, while the third type of method targets the training process of the models. The DNN models take charge of various tasks of perception, including object classification [80, 83], semantic segmentation [65, 78], and camera-/LiDAR-based object detection [61–67, 71–77, 79]. Since the generated adversarial perturbations may not be effective in a noisy physical environment [133], a number of methods (e.g., *Robust Physical Perturbations* [80]) are proposed to overcome this challenge.

As mentioned in §5.2.2, it is also a great challenge to judge the correctness of the output of the perception module. We collect three types of approaches, including ground-truth labeling [92, 93], metamorphic testing [94–99], and formal specifications [100, 101] for tackling this problem. In summary, the first approach focuses on automatically generating ground-truth labels for single

Table 4. Summary of the papers for the perception module testing: Part II

Oracle	Description	Literature
Ground-truth labeling	Generating road label automatically by LiDAR	[92]
	- Generating dimension and classification references for object detection	[93]
Metamorphic testing	The detected object in the original images should also be detected in the synthetic images	[94]
	The recognition results of target models should change when traffic lights change	[95]
	The detection of objects should be same with the affect of noise	[96]
	Check the model inconsistencies between the original images and the mutated images	[97, 98]
Formal specifications	The object detected in a previous frame should also be detected in a later frame	[99]
	Adapting TQTL to express desired properties of perception	[99, 100]
	Adapting STQL to express more refined properties	[101]
Adequacy	Description	Literature
Structural coverage	Neuron coverage: the percentage of the neurons that are activated	[107]
	K-multisection neuron coverage: considering activated neurons and not activated neurons	[108]
	Surprise adequacy: the novelty of a test case based on its range in the training data distribution	[109]
Combinatorial coverage	Characterizing the scenarios by multiple parameters	[111]
	Incorporating domain expertise to reduce the combinatorial explosion	[112]
	Utilizing the analytic hierarchy process to identify the key factors	[113]

images, while the other two methods tend to express the properties between continuous frames and are thus suitable for evaluating the perception module at runtime.

Traditional coverage metrics, e.g., code coverage, are typically not suitable for estimating the test adequacy of DNN-based models. Structural coverage metrics like neuron coverage [107] have become a mainstream substitute. Recently, there is a different voice [134–136] saying that neuron coverage and its extensions may lack effectiveness in guiding ML testing. In addition to neuron coverage, combinatorial testing [111–113] is another approach for tackling the test adequacy problem of the perception module.

**Summary:** A large number of papers perform adversarial attacks on the perception module, covering various perception tasks, e.g., camera-/LiDAR-based object detection and semantic segmentation. Testing oracle problem of this module has been studied through different approaches, e.g., metamorphic testing. Structural coverage metrics (e.g., neuron coverage) and combinatorial testing techniques are widely adopted for the guarantee of testing adequacy.

### 5.3 Planning Module

The planning module takes the information from the perception module as input and produces a suitable driving trajectory as a reference for the control module to make decisions. In the planning module, we introduce the studies on test methodology (shown in §5.3.1), test oracle (shown in §5.3.2) and test adequacy (shown in §5.3.3).

**5.3.1 Test Methodology.** Testing of the planning module consists in providing traffic scenarios for an ADS, and checking if the planner module generates trajectories that satisfy properties such as safety, comfort, and low cost. Note that, the path planning module is usually integrated into the whole ADS, and highly coherent with other modules: the input of the module comes from the

perception module, and the output trajectory is a reference for the control module, rather than the actual one observable from the system. Therefore, testing the planning module independently is a challenging task.

Due to the above reasons, there exist no large numbers of studies on the testing dedicated to the planning module. The studies we collected are based either on dedicated path planning systems [137–139], or on the assumption of the perfection of the perception and control modules [140]. In summary, *search-based testing* is the major technique for testing of the planning module, and it is adopted in most of the works [137–139, 141–145] for this module.

**Search-based testing.** According to [146], scenarios are defined on three abstraction levels, namely, *functional scenarios*, *logical scenarios*, and *concrete scenarios*. A functional scenario has the highest abstraction level and defines only the basic conditions and participants of a scenario; on top of a functional scenario, a logical scenario is defined by a set of parameters and their ranges; with the parameter values fixed in a logical scenario, a concrete scenario is generated. In the context of scenario generation, search-based testing usually consists in searching in the parameter space of a logical scenario for a concrete scenario, with specific objectives. Below are some examples of applying search-based testing to generate concrete scenarios for the testing of the planning module.

The works [137–139, 141] use a dedicated path planning system from their industry collaborator, which computes the trajectories of the ADS based on several constraints, e.g., safety and traffic regulations. The aggressiveness of the path planning strategy is decided by a system parameter, named *weight*. Laurent et al. [137] define a coverage criterion named *weight coverage*, which is used to characterize the testing adequacy of the weight parameter. Later in [138], they propose two search-based techniques, named *single-weight approach* and *multi-weight approach*, that automatically generate testing scenarios guided by weight coverage. Specifically, the single-weight approach searches for the scenarios that cover one specific weight of the path planner, while the multi-weight approach generates scenarios that cover different weights simultaneously using the multi-objective search. Arcaini et al. [139] consider searching for the *driving patterns* that are identified by the features appearing in the planned trajectory, such as longitudinal/lateral acceleration and curvature. The driving patterns that take place in a trajectory for a considerable duration are relevant to the characteristics of the path planner, and thus facilitate engineers in system assessment. Since the testing scenarios in their previous works contain numerous irrelevant elements and are thus hard to debug, in their latest work [141], they target the simplification of testing scenarios—they remove all the irrelevant traffic participants, but the failures can still be triggered.

Althoff et al. [142] propose the notion of *drivable area* for motion planning algorithms, which represents a safe solution space in which the ADS can avoid collision. Then they adopt a search method to generate scenarios that are highly critical in the sense that the drivable area is limited. In their follow-up work [143], the authors consider the interference of other traffic participants in the drivable area, in order to increase the complexity of the scenarios. In their experiment, the evolutionary algorithms [147] are demonstrated to be advantageous in finding a local optimum over these complex and diverse scenarios. Bak et al. [144] apply *random exploring trees (RRT)* to search for the *adversarial agent perturbations*, which indicate that the behaviors of other vehicles are only modified slightly. Kahn et al. [145] generate occlusion scenarios for testing the behavior planning module of an ADS. To be specific, they apply an occlusion-guided search method to inject vehicles into the scenarios extracted from naturalistic data. Experimental results show that the number of occlusion-caused collisions generated by their approach is 40 times higher than that from the naturalistic data.

**5.3.2 Test Oracle.** Assessing the correctness of the output of the planning module, i.e., a planned trajectory, is a challenging problem, due to the lack of an oracle that represents the “correct” trajectory. In this section, we introduce the studies [148, 149] that define different metrics as the oracles to evaluate the correct functionality of the planning module.

Calò et al. [148, 149] define the notion of *avoidable collisions*, to distinguish them from *unavoidable collisions*, in a dedicated path planner. By their definition, a collision is *avoidable* if it can be avoided from happening in the same scenario by using a different system configuration of the ADS. Compared to the unavoidable ones, the avoidable collisions are considered critical, since these collisions require system reengineering to rectify the unsafe behavior.

**5.3.3 Test Adequacy.** As mentioned in §5.3.1, the inputs to the planning module involve both external parameters that identify a scenario and internal parameters of the ADS. Because there are infinitely many possible combinations of these parameters, generating test cases that are sufficiently diverse remains a great challenge. In this section, we collect the studies [137, 140] that propose coverage measurements on the space of the parameters. Namely, the *weight coverage* criterion [137] refers to the coverage of the possible configurations of the path planner under test; the *route coverage* criterion [140] is proposed to measure whether different features of a map have been explored by the test suite.

Laurent et al. [137] propose a coverage criterion, named *weight coverage*, to test a dedicated path planning system. In their path planner, there is a *weight* function that consists of six weight parameters, which affect the path planning decisions from different aspects, such as safety and comfort. In order to cover diverse planning decisions made by the system, the authors use the weight coverage to guide the exploration of the weight parameter space. Thereby, they manage to generate scenarios that cover more diverse combinations of the weight parameters.

Tang et al. [140] propose another coverage criterion called *route coverage* for testing the route planning functionality of APOLLO. Based on a *Petri net* abstracted from the map, they quantify the route diversity based on the *junction topology feature* and *route feature*. The junction topology feature describes the relative position and connection relationship of the roads at a junction, while the route feature describes the action of APOLLO to track a selected road. By mutating the test cases, they achieve a high route coverage ratio and thus obtain a diverse test suite that covers various features of the map.

**5.3.4 Discussion.** The summary of the collected papers for testing the planning module is shown in Table 5. We find that search-based testing is a dominant technique that has been demonstrated to be effective in revealing faults in the planning module [137–139, 141–145]. In addition, several metrics are proposed for facilitating the testing on the planning module, e.g., *avoidable collision* [148] for tackling the oracle problem, *weight coverage* [137] and *route coverage* [140] for evaluating the sufficiency of the test suite. However, most of these metrics are dedicated to specific path planning systems, and it needs to be further explored whether they could be generalized to the planning modules of other systems.

**Summary:** Search-based testing is an effective technique for revealing faults in the planning module. Several metrics have been proposed for testing particular path planning systems, and it needs further exploration on how to generalize these metrics to other systems.

## 5.4 Control Module

Based on the trajectories produced by the planning module, the control module takes charge of the lateral and longitudinal control of the ADS. By using various control algorithms, such as *model*

Table 5. Summary of the papers for the planning module testing

Methodology	Description	Literature	Test Objective	Environment
Search-based testing	Searching for testing scenarios with weight coverage guarantee	[137, 138]	A dedicated path planner	Simulation
	Searching for the specific driving patterns	[139, 141]		
	Searching for the scenarios in which the drivable area is limited	[142, 143]	Motion planners	Digital dataset
	Applying RRT to search for adversarial agent perturbations	[144]	Five path planners, e.g., Frenet Planner [150]	Simulation
	Injecting vehicles into the scenarios extracted from naturalistic data	[145]	Strategic planners	Digital dataset

Oracle	Description	Literature
Avoidable collisions	A collision can be avoided from happening in the same scenario by using a different system configuration of the ADS.	[148, 149]

Adequacy	Description	Literature
Weight coverage	Based on a weight function consisting of weight parameters which affect planning decisions	[137]
Route coverage	Based on the junction topology feature and route feature	[140]

*predictive control (MPC)* and *proportional integral derivative (PID) control*, it generates control signals, e.g., acceleration, deceleration, and steering angle, to the CAN bus for the control of the whole system. In this module, we introduce the works on the testing of the control module, from the perspectives of test methodology (shown in §5.4.1) and test oracle (shown in §5.4.2).

**5.4.1 Test Methodology.** The control module takes charge of multiple functionalities, such as the longitudinal control and the lateral control of the ADS. Hence, the testing of this module focuses on detecting vulnerabilities in the control mechanisms.

**Fault injection.** Fault injection is a method that deliberately introduces faults into a system, in order to assess the fault tolerance of the system. Uriagereka et al. [151] adopt this technique for testing the fault tolerance ability of the control module of an ADS. Specifically, they inject faulty GPS signals into the lateral control function of the ADS, which makes it produce wrong steering commands. By calculating the *fault tolerant time interval*, which denotes the duration from the activation of the fault to the occurrence of unsafe behavior, they find the lateral control system can tolerate this type of fault for as long as 177ms. Zhou et al. [152] inject faulty control signals through CAN bus to cause collisions without being detected by ADS safety mechanisms, e.g., *forward collision warning*. They evaluate the method with OPENPILOT and find that the lateral control of the system is the typically vulnerable part with a high attack success rate.

**Sampling.** We refer to *sampling* as the statistical method that samples values from a probability distribution. Wang et al. [153] sample the relevant parameters, e.g., speeds of the *non-player characters (NPCs)* and the ego vehicle, for generating scenarios of different challenge levels. The method is evaluated on the unprotected left-turn scenario and experimental results demonstrate the robustness of the MPC controller. In their later work [154], *game theory* is applied to characterize the interactive behaviors of NPCs in highway merging scenarios.

**Falsification.** Temporal logic-based falsification [155–161] is applied to ADS testing in [162, 163]. Originally, falsification refers to a technique for testing of the general *cyber-physical systems*, guided by the quantitative semantics of temporal logic specifications, which indicates how far is the system from being unsafe. Tuncali et al. [162] propose a falsification-based automatic test

Table 6. Summary of the papers for the control module testing

Methodology	Description	Literature	Test Objective	Environment
Fault injection	Injecting faulty GPS signals	[151]	Lateral control module of an urban vehicle	Simulation
	Injecting faulty control signals through the CAN bus	[152]	The control module of OPENPILOT	Simulation
Sampling	Sampling the relevant parameters for generating different challenge level scenarios	[153, 154]	MPC controller [14]	Simulation
Falsification	Utilizing temporal logic-based falsification to search for critical scenarios	[162, 163]	Collision avoidance controller	Simulation

Oracle	Description	Literature
Optimization-based oracle model	The model can generate an oracle area in a given scenario	[164]

generation framework for testing collision avoidance controllers. They utilize a cost function, i.e., the quantitative semantics of the temporal logic specification, as a guidance in searching for the critical scenarios in which the relative speed of the two vehicles in the collision is minimal. The obtained scenarios can be taken as the behavioral boundary that divides the safe and unsafe behaviors. As a follow-up work, Tuncali et al. [163] utilize the *Rapidly-exploring Random Trees (RRT)* algorithm for ADS falsification. They incorporate a new cost function that applies *time-to-collision* to measure the seriousness of the collision. As a result, the new method achieves better effectiveness in searching for safety-critical scenarios, thanks to the exploration brought by the RRT algorithm.

**5.4.2 Test oracle.** Like the planning module, the control module also faces the oracle problem in its testing—indeed, it is usually not straightforward to determine whether a control decision is “correct”. In [164], Djoudi et al. propose a framework to determine whether the control module makes “the correct decision”. They design a model to generate an oracle area in the given scenario, which is the closest safe position ahead of the vehicle. A control decision is then considered as “the correct decision”, if it could drive the vehicle close to the oracle area.

**5.4.3 Discussion.** Table 6 summarizes the collected papers for the testing of the control module, where the number of studies is not too large. Note that currently most of the control modules of ADS adopt mature control techniques directly, such as *PID* [13] and *MPC* [14], which partially explains why this module is not extensively studied. The collected studies adopt three major techniques for testing the control module, including fault injection [151, 152], sampling [153, 154], and falsification [162, 163]. To tackle the oracle problem in control module testing, the framework proposed in [164] could generate an oracle area for judging whether the control decision is “correct”. However, we do not find much work that handles the test adequacy problem for this module. In general, since the control module deals with continuous dynamics, it is challenging and thus requires further exploration to define adequacy criteria for test cases in the future.

**Summary:** Since most of the control modules of the ADS adopt those mature control techniques, e.g., *MPC* and *PID*, there are not many works studying the testing of the control module. Existing techniques mainly include fault injection, sampling, and falsification. There are some works that study the oracle problem of control modules. There is few work that studies the adequacy criteria for testing control modules.

## 5.5 End-to-End Module

The end-to-end (e2e) module is a special design adopted by many modern ADS, which integrates the functionalities of perception, planning and control in a single DNN-based model. The DNN model is often developed by *supervised learning*, which is trained by using a training dataset consisting of realistic driving data. Each element of the dataset is a pair  $\langle I, c \rangle$ , which maps the information  $I$  at the end of the sensor to a label  $c$  that indicates the desired control decision at the end of the controller. After training, a model can infer control decisions based on the driving environment at runtime in order to drive the ADS properly. For instance, in some modern ADS that perform steering angle control, the end-to-end DNN model takes as input the sensing information including road conditions and the status of other cars, and outputs a series of predicted steering angles for controlling the ADS. In this section, we introduce the collected studies on the testing of the end-to-end module from the perspectives of test methodology (shown in §5.5.1), test oracle (shown in §5.5.2), and test adequacy (shown in §5.5.3).

*5.5.1 Test Methodology.* As mentioned before, an end-to-end DNN model integrates three functionalities, namely perception, planning and control, in a single module. Among these three functionalities, perception is the most vital part as it provides input information to other modules; meanwhile, it is also the most vulnerable to external environments, as it essentially involves image recognition tasks that rely on deep learning. Compared to a DNN just for perception, although an end-to-end DNN does not directly output the perception information, the control decisions it makes still depend on the perception information. Therefore, like the case in the perception module, generating adversarial images or scenarios that fool the end-to-end DNN is still the major testing methodology for testing the end-to-end modules.

We introduce three approaches, namely, search-based testing, optimization-based adversarial attack, and GAN-based attack. The first approach has been introduced in §5.3.1; the last two approaches have been introduced in §5.2.1.

**Search-based testing.** Search-based testing searches for a target test case in the input space, guided by certain objectives. One commonly used objective is the coverage of the test suite—maximizing the cumulative coverage of a test suite can expose more diverse behavior of the system, and thus allow a better chance of detecting the target test case. In the context of DNN testing, neuron coverage is proposed by Pei et al. [107] to analogize the structural coverage in traditional programs. In their follow-up work, Tian et al. [165] propose a coverage-guided testing framework called *DeepTest* for DNN testing. They propose various image operations, e.g., scaling, shearing, and rotating, as the test input (image) mutation methods; then they generate test cases by applying these operations to seed images, and keep only those mutants that enlarge the cumulative neuron coverage of a test suite. Experiments are conducted on three end-to-end models, and the results show the effectiveness of their method in test case generation.

In addition to coverage, the seriousness of the unsafe behavior is another factor that can be used as the search objective, and this has been considered by Li et al. [166]. In their work, the seriousness of the unsafe behavior of the end-to-end module is formulated as the deviation of the actual steering angle made in the test scenario from the expected steering angle. The authors design an objective function that takes into account both the coverage and the seriousness, such that they can detect not only diverse but also serious unsafe test cases.

**Optimization-based attack.** The optimization-based adversarial attacking framework has been introduced in §5.2.1. Zhou et al. [167] introduce a framework called *DeepBillboard* that can generate adversarial perturbations which are added to billboard. The perturbations they generate can mislead the steering angles in a series of frames captured by camera sensors during the driving process, in

spite of the physical conditions, such as different distances and different angles to the billboard. Later Pavlitskaya et al. [168] extend *DeepBillboard* with the *projected gradient sign* method [169], and experimental results show that the curved and rainy scenes are more vulnerable to these adversarial attacks. In another line of work, adversarial black lines are utilized to attack the end-to-end driving models [170, 171]. These black lines are easy to paint on the public road and can lead to a deviation of an ADS from the original path.

**GAN-based attack.** GAN has been introduced in §5.2.1, and it has been considered as a major approach for adversarial attacking. Kong et al. [172] propose a GAN-based approach called *PhysGAN* which utilizes 3D tensors, i.e., a slice of video containing hundreds of frames, to generate adversarial roadside signs that can continuously mislead the end-to-end driving models with high efficacy and robustness. In another work, to generate realistic adversarial images, Zhang et al. [173] propose a GAN-based approach called *DeepRoad*. They demonstrate that their generated adversarial images are realistic under various weather conditions, and effective in detecting unsafe system behaviors.

**5.5.2 Test Oracle.** An oracle of the end-to-end module indicates which is the correct control decision at each moment of a scenario. Although this can be done with the help of human drivers, it is too expensive and prone to errors. Existing works propose various automatic methods to solve the oracle problem of the end-to-end module, including metamorphic testing [165, 173, 174], differential testing [107], and model-based oracle [175, 176].

**Metamorphic testing.** As introduced in §5.2.2, metamorphic testing is a viable way to solve the oracle problem. In the testing of end-to-end models, there are a few works that leverage metamorphic relations to define the test oracles, e.g., *DeepTest* [165] and *DeepRoad* [173]. The metamorphic relation introduced by *DeepTest* [165] is that, the steering angle should not change significantly for the same scenes under different weather and lighting conditions. Similarly, *DeepRoad* [173] aims to detect *model consistency*, which means, for a synthetic image and the original image, the difference between two predicted steering angles is smaller than a threshold. Pan et al. [174] introduce a metamorphic relation for testing end-to-end models in a foggy environment; the relation requires that the density and direction of fog not affect the output steering angle of the target models.

**Differential testing.** Pei et al. [107] apply differential testing to generate scenarios which reveal the inconsistencies between different DNN models. For the same scenario, they expect that the DNNs under test should give the same inference result. The violation of this property is considered as an unexpected behavior.

**Model-based oracle.** Stocco et al. [175] propose a so-called *self-assessment oracle* for the potential risk prediction of ADS. The self-assessment oracle involves training a probabilistic model that characterizes the distribution of the potential risks under various real scenarios. This model can be used to monitor the real environment during the execution of the ADS and predict situations that are probably not handled by the ADS. This novel idea is also studied by Hussain et al. [176].

**5.5.3 Test Adequacy.** Combinatorial coverage is also adopted in end-to-end module testing, e.g., the 2-way combinatorial testing based on image transformations [177]. In §5.2.3, we introduce the structural coverage for DNN testing, which analogizes the structural coverage in traditional program testing. Since the end-to-end module also relies on DNN models, these structural coverage criteria are also used in the testing of the end-to-end module. Neuron coverage, which has been introduced in §5.2.3, is used by its authors for a coverage-guided testing [165], as mentioned in §5.5.1. The refined structural coverage criteria for DNNs, such as *k-multisection neuron coverage (KMNC)* and *neuron boundary coverage (NBC)* [108, 166], which is also elaborated on in §5.5.1.

Table 7. Summary of the papers for the end-to-end module testing

Methodology	Description	Literature	Test Objective	Environment
Search-based testing	Generating transformed images with high neuron coverage	[165]	Three DNN models: Rambo [178], Chauffeur [179] and Epoch [180]	Digital dataset
	Designing an objective function to search for the diverse and serious unsafe test cases	[166]	Three DNN models: Dave-1 [181], Dave-3 [182] and Chauffeur	Digital dataset
Optimization-based attack	Replacing the original billboard with an adversarial billboard	[167]	Four DNN models: Dave-1, Dave-2 [183], Dave-3 and Epoch	Digital dataset
	Extending attack methods in [167] to generate adversarial patches	[168]	An end-to-end driving model called DriveNet [184]	Simulation
	Generating adversarial black lines on the road	[170, 171]	Two end-to-end driving models in CARLA	Simulation
GAN-based attack	Generating adversarial roadside sign	[172]	Three DNN models: Dave-2, Epoch and Rambo	Digital dataset
	Generating realistic adversarial images	[173]	Three DNN models: Autumn [185], Chauffeur and Rwrightman [186]	Digital dataset

Oracle	Description	Literature
Metamorphic testing	The steering angle should not change significantly under different conditions	[165, 173]
	The density and direction of fog should not affect the output steering angle of the target models	[174]
Differential testing	The DNNs under test should give the same inference result for the same scenario	[107]
Model-based oracle	Predicting the situation that the ADS is probably not able to handle	[175, 176]

Adequacy	Description	Literature
Combinatorial coverage	2-way combinatorial testing based on image transformations	[177]

**5.5.4 Discussion.** As with the perception module, the end-to-end module also contains many DNN-based models; however, these models are not only used for perception, but also for the control of the vehicles. Consequently, adversarial attack methods used in the perception module testing, including optimization-based method [167, 168, 170, 171] and GAN-based method [172, 173], are also adopted as the testing methodologies for this module. One observation is that, compared to perception module testing that tests DNN models using single images, the work [167, 172] for end-to-end module testing often use a series of images, i.e., the frames captured by cameras in a system execution. Another major testing approach is the coverage-based testing [107, 165, 166, 173], in which the testing is guided by coverage criteria proposed for measuring whether the system behavior has been sufficiently explored.

Since it is hard to evaluate the correctness of the output steering angle for an input image, metamorphic testing [165, 173, 174] and differential testing [107] are adopted for tackling this problem. In addition, we find that other oracle techniques, e.g., model-based oracles [175, 176], can be used to solve the oracle problem for this module.

**Summary:** Many of the testing techniques used for testing the perception module can also be used for testing the end-to-end module, such as adversarial attack. One notable difference from the perception module is that, in the end-to-end module, these techniques are applied in a driving context involving a series of continuously-changing images, rather than a single image. Besides those metrics that have been used in the perception module, e.g., metamorphic testing, new techniques, e.g., differential testing, are employed to solve the oracle problem. In terms of test adequacy metrics, this module is very similar to the perception module.

## 5.6 Answer to RQ1

In total, we survey over 80 papers that study the testing of different modules of ADS. Various testing techniques have been proposed for testing different modules. Based on our survey, we can draw the following conclusions: (1) for the sensing module, physical testing and deliberate attack on the sensors could effectively find their abnormal behaviors; (2) for the perception module and the end-to-end module, adversarial attack is the most widely-used approach, since the two modules mainly rely on the use of DNN-based models; (3) for the planning module, though the relevant studies are not so many, search-based testing has been extensively adopted; (4) for the control module, main testing techniques include fault injection, sampling, and falsification.

Despite the numerous techniques dedicated to different modules, we also find some open challenges for the testing of these modules. For example, the neuron coverage in [107] may not be effective for testing the perception module and the end-to-end module. More details about these open challenges are also discussed in §8.

## 6 LITERATURE OF TECHNIQUES ON SYSTEM-LEVEL ADS TESTING

In this section, we introduce the research works on system-level ADS testing with the goal of answering RQ2 in §1. Different from module-level testing, system-level testing focuses on the failures that threaten the safety of the whole vehicle due to the collaborations between modules. In §5, most of the testing works are done in simulated environments, implemented by various software simulators. In this section, we introduce not only simulation-based testing in §6.1, but also introduce the *mixed-reality testing* in §6.2 that introduces real hardware in the testing loop.

### 6.1 System-Level Testing with Simulators

We first introduce system-level testing conducted with the help of software simulators. Similar to the module-level works, we also present these studies from three perspectives, namely, test methodology (shown in §6.1.1), test oracle (shown in §6.1.2) and test adequacy (shown in §6.1.3).

**6.1.1 Test Methodology.** In the literature, we find various testing methods for the system-level testing of ADS, including search-based testing, adaptive stress testing, sampling-based methods, and adversarial attack. In this section, we introduce these testing methods.

**Search-based testing.** Search-based testing (or a similar concept named *fuzzing*<sup>6</sup>), is one of the most widely-adopted methodologies in ADS testing. As introduced in §5.3.1, it consists in searching in the parameter space for specific parameter values that achieve a testing objective. In this section, we introduce the works [187–205] to illustrate the ideas.

Dreossi et al. [187] propose a compositional search-based testing framework, and apply it for the testing of ADS with machine learning components (i.e., mostly perception). The basic idea in their work is the cooperative use of the perception input space and the whole system input space: the constraints on one space can reduce the search efforts in the other space. In this way, they improve the efficiency of searching for counterexamples. Abdesslem et al. [188] propose a multi-objective search algorithm for detecting errors caused by *feature interaction*. A *feature interaction* describes the interaction between different ADS functionalities, e.g., an AEB command could be overridden by an ACC command since the two functionalities both control the braking actuator. In practice, search-based testing has also proved to be effective for industry-level ADS. Li et al. [196] propose

<sup>6</sup>Search-based testing and Fuzzing are similar concepts coming from different communities. The former emphasizes on the testing methodology via search, which relies on well-defined fitness functions and applies search heuristics, e.g., evolutionary algorithms, to find the target test cases. Fuzzing comes from the security community and its methodological essence lies at its randomness. Similar to search-based testing, fuzzing also comes with an objective function as a guidance that helps it achieve the target more efficiently.

*AV-Fuzzer* used for testing of APOLLO, and they show the effectiveness of this framework in finding dangerous scenarios.

There is a line of work [190–192] that studies the relationship between test input and system behavior. Riccio et al. [190] propose the notion of *frontier of behaviors*, which represents the boundary of inputs where the system starts to behave abnormally. In their later work [191], they firstly provide an *interpretable feature map* that explains the correlations between test inputs and system behaviors, and leverage *Illumination Search* [206] to explore the feature space. This approach is enhanced in their follow-up work [192] for finding those test inputs that contribute to the exploration of the feature map.

Lane keeping system is an important target in ADS testing. When search-based testing is applied, different road representations can affect the effectiveness of the approach. Castellano et al. [193] compare six road representations for testing lane keeping systems, and found that curvatures and orientation are essential factors which affect the behaviors of these systems. Gambi et al. [189] propose a novel approach called *ASFAULT* to generate virtual roads for testing lane keeping systems. Experiments on BEAMNG.AI [27] and DEEPDRIVING [207] demonstrate that the proposed approach could generate effective testing roads that cause vehicles to deviate from the correct lane. Open-source search-based tools, e.g., *Frenetic* [208], are also developed and the comparison of these tools are reported in [26, 209].

There are other works that aim to improve the search efficiency by designing better search algorithm. Abdessalem et al. [188, 210] combine multi-objective search with decision tree classification for test generation of ADS. In their framework, the classification checks whether the scenario is a critical one, and accelerates the search process. Goss et al. [194] apply *Rapidly-exploring Random Trees (RRT)* based on an *Eagle Strategy* to estimate the critical scenario boundaries. Zheng et al. [195] propose a quantum genetic algorithm that allows lower population size. Luo et al. [199] study the test case prioritization techniques and employ multi-objective search algorithms to find violations with a higher probability of occurrence. Test case prioritization techniques are also utilized to accelerate the regression testing of ADS [200, 201] and achieve remarkable results.

Search-based testing is usually based on system simulations; however, even with software simulators, the simulations of ADS can still be expensive and slow. There is another line of work [202–205] that trains surrogate models as the substitute for testing acceleration. Abdessalem et al. [202] train a surrogate model that maps the scenario parameters to fitness functions, and use the surrogate to detect the non-critical parameters for search space reduction. *Gaussian Process* is also leveraged for training a surrogate model in [203]. To search for more collision scenarios, Beglerovic et al. [204] train a surrogate model by utilizing the critical scenarios that already exist. For finding an optimal surrogate model for ADS testing, Sun et al. [205] compare six types of surrogate models, e.g., *Extreme Gradient Boosting (EGB)* and *Kriging (KRG)* surrogates, in two logical scenarios.

**Adaptive stress testing.** Stress testing has been widely adopted in various domains of the industry, which performs testing by providing test cases beyond the capability of the system under test. *Adaptive stress testing*, literally, performs stress testing in an adaptive manner; namely, it prioritizes the test cases and allocates different testing resources to them accordingly. Therefore, specifying the policy of priority assignment is the key to adaptive stress testing. Koren et al. [211] apply adaptive stress testing for ADS, and design a priority assignment policy based on the difference between the expected behavior and the actual behavior. In a later work [212], they propose a new priority assignment policy based on *Responsibility-Sensitive Safety (RSS)* [213], which defines the utopian behavior of the cars by which no collision will happen in a scenario. The new policy is thus defined according to the distance of the ADS behavior compared to the utopian cases in the RSS rules [212]. Baumann et al. [214] adopt reinforcement learning, namely, *Q-learning* [215], for exposing more

critical scenarios in the overtaking scenario. Reinforcement learning is also combined with RSS rules for generating edge cases in [216].

**Sampling.** One use case in ADS testing is to generate scenarios by sampling from a natural scenario distribution, in order to make the generated scenario realistic. This has been studied in [217]. Nitsche et al. [218] propose a sampling-based framework for validating ADS at road junctions. Specifically, they first cluster the junction scenarios along with the representative variations from the real-world accident data, and then these relevant parameters are sampled by the *Latin Hypercube Sampling (LHS)* method and used to compose concrete scenarios for simulation testing.

Sampling is also used to help the identification of the failure features. Corso et al. [219] combine *signal temporal logic (STL)* with sampling method to generate disturbance trajectories for testing. Those trajectories are interpretable and easier for debugging due to the features of STL, i.e., the description of logical relationships over time. In another work [220] of them, dynamic programming is applied during the sampling process to discover more failure scenarios.

Batsch et al. [221] sample the simulation data in a traffic jam scenario with the CARMAKER [222] simulation platform. The obtained data sets are then used to train a *Gaussian Process Classification* model, which could probabilistically estimate the boundary between safe scenarios and unsafe scenarios. Schütt et al. [223] utilize Bayesian optimization and Gaussian process to identify the relevant parameters of a logical scenario, i.e., they find the vehicle speed has no influence in one *vulnerable road user (VRU)* testing scenario. Birkemeyer et al. [224] leverage a *Feature Model*, i.e., features are organized as nodes in a tree structure, to represent a scenario space for sampling. Experimental results show that the FM-based sampling method is suitable for scenario selection for ADS testing.

Moreover, advanced sampling techniques can be applied to achieve specific goals; for example, *importance sampling* [225] is a technique used to sample rare events. In normal occasions, unsafe scenarios are indeed rare to happen, so detecting those scenarios is hard and costly. In that case, importance sampling can be applied to accelerate the testing [226, 227]. Zhao et al. [228–234] work extensively in this direction. The main aim of their work is to spend less simulations to detect more system failures, under various scenarios. Specifically, in [228–230, 232, 233], they investigate the cut-in/lane change scenarios; in [231] and [234], they focus on the car-following scenario and the unprotected pedestrian crossing scenario, respectively.

**Adversarial attack.** Adversarial attack has been introduced in §5.2.1, in which it is used for testing the perception module. Here, we introduce several works [235–238] that also attack the perception module, but they assess the influence of the attack on the whole system. Sato et al. [235] generate attack patches, as a camouflage for dirty roads, that mislead the lateral control functionality of the victim ADS to deviate from the lane. Rubaiyat et al. [236] generate perturbations to camera-captured images, based on a system-level safety risk analysis, to assess the reliability of OPENPILOT under real-world environmental conditions. Nassi et al. [237] leverage the print advertisement to perform the attack, e.g., they embed an adversarial traffic sign on the back of other vehicles, and mislead the system to wrong behaviors. Wang et al. [238] perform an attack that adds perturbations to the trajectories of *NPCs*, and modifies the corresponding LiDAR sensor data.

**6.1.2 Test Oracle.** The oracles of the system-level testing of ADS are usually defined by safety metrics, such as *time-to-collision*, which measures how far the ADS under test is from dangerous situations. These metrics can be directly computed by monitoring the system behavior in the simulator, or expressed as formal specifications, such as *signal temporal logic (STL)*, which can automatically monitor the system behavior and compute the metric values. Besides, metamorphic relations are also used in some works for defining the oracle of ADS.

Table 8. Commonly-used Safety Metrics

Category	Name	Description
Temporal metrics	TTC [163]	The time until two objects collide with the current speed and path
	WTTC [241]	The time of the collision in the most likely accident scenario
	MprISM [242]	Estimating the TTC with the consideration of game interaction between vehicles
	THW [243]	The time between two objects reaching the same location
	TTR [243]	The remaining time until the start of the last driving maneuver that can avoid collisions with all objects in the scenario
Non-Temporal metrics	SD [196]	The stopping distance of the vehicle at the maximum comfortable deceleration
	LP [191]	The distance between vehicle center and lane center
	DRAC [244]	The minimum deceleration rate required by a vehicle to avoid a crash
	SARR [245]	The number of steering angle reversals larger than a certain value

**Safety metrics.** In system-level testing, a suitable safety metric, or called criticality metric, can be leveraged to find more system violations. There have been studies [9, 239, 240] that comprehensively investigate these safety metrics and here part of commonly-used metrics are listed in Table 8. These safety metrics can be categorized into temporal metrics and non-temporal metrics. Temporal metrics describe the temporal requirements to moving objects, and the most popular ones are *Time-to-Collision (TTC)* [163] and its extensions, e.g., *Worst-Time-to-Collision (WTTC)* [241], that measure the closeness of the ego car to collision in the scenario. Weng et al. [242] propose the *Model Predictive Instantaneous Safety Metric (MPrISM)*, which considers the interaction between moving vehicles. Another metrics include *Time Headway (THW)* and *Time-to-React (TTR)* [243]. The former calculates the time of the ego vehicle to reach the position of the lead vehicle, and the latter estimates the remaining time for a required reaction, e.g., a braking action.

Non-temporal metrics concern different aspects, such as distance, deceleration, and steering. One distance metric called *Stop Distance (SD)* [196] calculates the distance for a vehicle to stop with a maximum comfortable deceleration. Another distance metric is called *Lateral Position (LP)* [191], which defines the distance between the center of the vehicle and the center of the driving lane. Deceleration metrics, such as *Deceleration Rate to Avoid a Crash (DRAC)* [244], consider the deceleration rate during emergency. Steering metrics, such as *Steering Angle Reversal Rate (SARR)* [245], focus on the steering angle of a vehicle during the driving process.

There are works [197, 198] that propose to organize and utilize these safety metrics in an elegant manner. Also, Li et al. [246] propose to design metrics that involve more factors such as the relationship between scenarios, tasks, and functionalities of an ADS.

**Formal specifications.** As introduced in §5.2.2, formal specification uses temporal logic languages to express the properties which the system should hold during the running; then by specification-based monitoring, the satisfaction of the system behavior can be automatically decided. On the system-level testing of ADS, *signal temporal logic (STL)*, which can express the properties over real-time continuous variables, is the proper selection of specification language. There are a few works that adopt STL as the specification language [187, 247, 248], in which STL monitors are synthesized to decide whether the behavior of the ADS satisfy the desired safety properties. Zhang et al. [249] utilize formal specifications to represent driving rules and ADS behaviors to check the consistency between them.

**Metamorphic testing.** Metamorphic testing has been discussed for the module-level testing in §5.2.2 and §5.5.2. On the system-level testing, Han et al. [250] utilize metamorphic relations to

distinguish between real failures and false alarms. The metamorphic relation regulates that the behavior of the ADS should be similar in slightly different scenarios; otherwise, the collision in one of such scenarios is considered *avoidable*, and thus a real failure.

**6.1.3 Test Adequacy.** In system-level testing, the adequacy of testing is embodied by the diversity of the testing scenarios for the ADS. In this section, we introduce two lines of work that define various metrics to characterize the diversity of scenarios.

**Scenario coverage.** There is a line of work that defines *coverage* for scenarios. The intuition is that the testing is sufficient if all different types of scenarios are covered [251]. Tang et al. [252] classify the scenarios based on the topological structure of the map. Kerber et al. [253] define a distance measure over scenarios based on their spatiotemporal features, which enable scenario clustering. Besides, the temporal, spatial, and causal information of the simulation data can be further abstracted into *situations* for covering more test scenarios [254, 255].

**Combinatorial coverage.** Combinatorial coverage has been introduced in §5.2.3. Unlike the above coverage criteria defined directly on the features of the scenarios, combinatorial coverage considers the coverage of the combinations of different parameters that identify different scenarios. Tuncali et al. [247, 248] propose to use *covering array* for scenario generation in ADS testing. Covering array is a specific mechanism in software testing that guarantees the satisfaction of the *t*-way combination coverage of the parameters. See §5.2.3 for more details about *t*-way combination coverage. Guo et al. [256] propose the definition of *scenario complexity* and apply combinatorial testing techniques to generate more complex testing scenarios. Shu et al. [257] adopt the three-way combinatorial testing method on lane-changing scenarios, which ensures a high coverage of the generated critical scenarios. Li et al. [258] utilize the ontology concept, i.e., formulations of entities and their relationships, to describe the driving environment of an ADS. Then the constructed ontologies are combined with combinatorial testing techniques for generating concrete scenarios with coverage guarantee. Another work [259] proposes a scenario generation framework called *ComOpT*, based on *t*-way combinatorial testing, and finds numerous system failures of APOLLO. Moreover, combinatorial testing is also used in [260] to tackle the regression testing problem of ADS.

**6.1.4 Discussion.** As shown by Table 9, search-based testing [187–205] is the most widely-used technique for testing the whole ADS, with different focuses, e.g., studying the relations between test input and system behavior [190–192], testing lane keeping systems [26, 189, 193, 209] and test case prioritization [199–201]. Although simulation-based testing aims to solve the high cost problem of real-world testing, it may repeatedly simulate the same type of scenarios, which is also a time-consuming process. Consequently, adaptive stress testing [211, 212, 214, 216] and sampling-based techniques [217–219, 221, 223, 224, 228–230] are applied for accelerating the testing process. As in the cases of the perception and end-to-end modules, adversarial attack [235–238] has also been adopted for system-level testing, which aims to detect the vulnerabilities of the perception that affect the safety of the whole system. Note that among these testing techniques, adaptive stress testing has not been studied extensively, but it has a high potential for future ADS testing since it is effective in various domains of the industry [269].

System-level testing usually relies on safety metrics, e.g., temporal and non-temporal metrics (as shown in Table 8) and metamorphic relations [250], as the oracles that measure the occurrences of safety violations during the testing process. For ensuring the adequacy of system-level testing, there are two lines of work, namely, scenario coverage [252–254] and combinatorial testing [247, 248, 256–260], which propose metrics to characterize the diversity of testing scenarios.

Table 9. Summary of the papers for simulation-based system-level testing: Part I

Methodology	Description	Literature	Test Objective	Environment
Search-based testing	Incorporating the perception input space and the whole system input space to accelerate search	[187]	AEB systems [261, 262]	Simulation
	Searching for unsafe feature interactions with decision trees	[188, 210]	Systems from IEE [263]	Simulation
	Generating virtual roads for testing lane keeping systems	[189–193]	Lane keeping system in BEAMNG [15]	Simulation
	Searching for critical scenario boundaries	[194, 195]	Systems built based on simulators	Simulation
	Finding safety violations of an ADS in the dynamic environment	[196]	APOLLO	Simulation
	Finding violations with a higher probability of occurrence	[199–201]	Systems such as BEAMNG.AI [27]	Simulation
	Training surrogate models to accelerate testing	[202–205]	Systems like AEB system	Simulation
Adaptive stress testing	Assigning different priorities to the test cases	[211, 212, 214, 216]	Systems like the Intelligent Driver Model [264]	Simulation
Sampling	Sampling junction scenarios	[218]	Collision avoidance system	Simulation
	Sampling the simulation data in a traffic jam scenario	[221]	System in CARMAKER	Simulation
	Combining STL with the sampling method	[219, 220]	Intelligent Driver Model	Simulation
	Identifying the relevant parameters of a logical scenario	[223]	Intelligent Driver Model	Simulation
	Sampling scenario space represented by feature model	[224]	AEB system	Simulation
	Adopt Importance Sampling to sample rare events	[226–234]	Systems like the ACC system [265]	Simulation
Adversarial attack	Generating attack patches as a camouflage of dirty roads	[235]	OPENPILOT	Simulation and real world
	Generating adversarial perturbations under different weather	[236]	OPENPILOT	Simulation
	Adding perturbations on the trajectories of NPCs	[238]	Driving models [266–268]	Digital dataset

Overall, there exist more works in system-level testing than that in module-level testing. Moreover, there are many other works that study the differences between simulation-based testing and real-world testing. More details can be found in §6.3.

**Summary:** There are a large number of studies that leverage different techniques, e.g., search-based testing, adaptive stress testing, and sampling-based techniques, for testing the ADS at the system level. Besides, numerous metrics have been proposed for different usages in the testing process, e.g., they are used to measure the occurrences of safety violations, and they are used to characterize the diversity of testing scenarios.

## 6.2 Mixed-Reality Testing

Due to the expensiveness of the real-world testing of ADS, most approaches in §5 and in §6.1 test ADS in software simulators. Although modern simulators can be powerful and high-fidelity, simulation-based testing is not sufficient to reveal all the problems of ADS, due to the gap between simulators and the real world. As a trade-off, mixed-reality testing combines simulation-based testing with real-world testing. In this section, we introduce several special testing schemes, which replace certain parts of the components in the testing loop, with physical components. Specifically,

Table 9. Summary of the papers for simulation-based system-level testing: Part II

Oracle	Description	Literature
Safety metrics	Temporal metrics and non-temporal metrics (more details in table 8).	-
	Providing a set of fitness function templates for different testing goals	[197, 198]
Formal specifications	Adopting STL as the specification language to express the properties over real-time continuous variables	[187, 247, 248]
	Utilizing formal specifications to represent driving rules and ADS behaviors	[249]
Metamorphic testing	The behavior of the ADS should be similar in slightly different scenarios	[250]

Adequacy	Description	Literature
Scenario coverage	Based on the topological structure of the map	[252]
	Based on spatiotemporal features	[253]
	Based on the temporal, spatial, and causal information of the simulation data	[254, 255]
Combinatorial coverage	Applying t-way combinatorial testing techniques for scenario generation	[247, 248, 256, 257, 259, 260]
	Utilizing the ontology concept to describe the driving environment	[258]

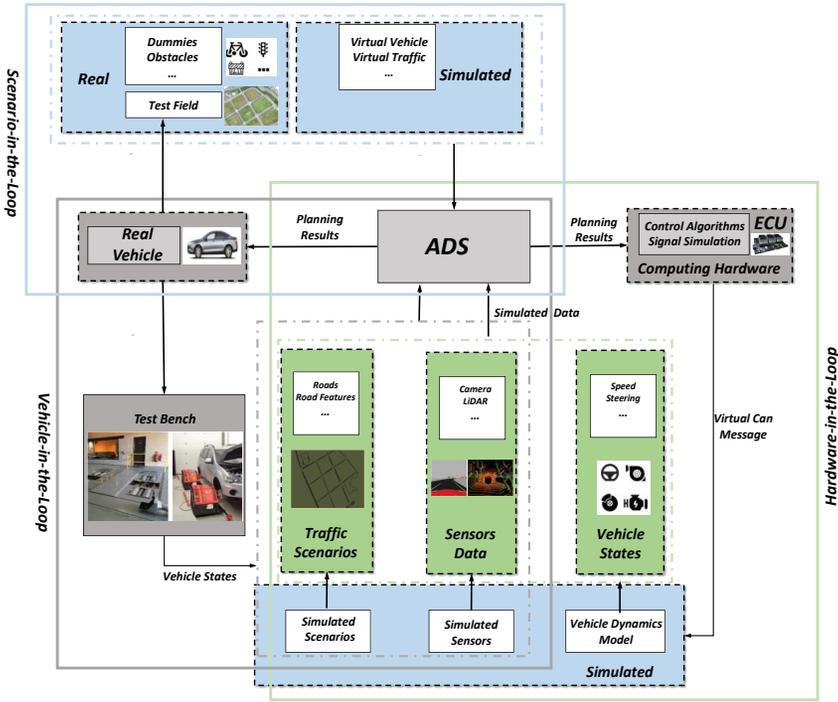


Fig. 4. Illustration of the HiL, ViL, and ScIL approaches

these schemes include *hardware-in-the-loop (HiL)*, *vehicle-in-the-loop (ViL)*, and *Scenario-in-the-Loop (ScIL)*; their mechanisms are illustrated in Fig. 4.

**6.2.1 Hardware-in-the-Loop.** HiL testing usually introduces the real ECU hardware into the testing loop, as shown in the green box in Fig. 4. There is a line of work [270–273] that adopts this testing method. Chen et al. [270, 271] propose an HIL testing platform that could simulate multi-agent

interaction on large-scaled scenarios with the usage of *OpenStreetMap* [274]. Brogle et al. [272] build their HiL platform based on CARLA and *robot operating system (ROS)*, which achieves high fidelity in vehicle dynamics and sensor data output. Gao et al. [273] design another HiL platform for AEB testing and find that the performance of the AEB functions in HiL tests is close to that in real road tests.

**6.2.2 Vehicle-in-the-Loop.** Different from HiL testing, ViL testing works by integrating a synchronized virtual scenario into a real vehicle, as shown in the gray box in Fig. 4. The following works [275–279] we collected are all based on this idea. Chen et al. [275] propose a ViL testing platform which can reconstruct scenarios based on the corresponding HD map. For simulating more realistic scenarios, these works [276–278] integrate popular traffic simulators, such as SUMO [280] and VISSIM [281], into the ViL testing loop. Stocco et al. [279] utilize the Donkey Car platform [282] to build a 1:16 scale car which is controlled by end-to-end driving models. They test these driving models in a closed-track environment and study the transferability of failures between simulation and the real world.

**6.2.3 Scenario-in-the-Loop.** SciL testing narrows the gap between simulator and the real world by integrating more real components like the pedestrian dummies into the loop, as shown in the blue box in Fig. 4. Szalay et al. first propose the concept of SciL testing in [283] and they develop a SciL testing platform based on SUMO and UNITY [284] in a later work [285]. Horvath et al. [286] study the SciL testing by comparing the implementation process of this method with that of ViL testing. The authors find that the two testing methods have the same basis, but SciL testing is still at an early stage.

### 6.3 Simulation-Based Testing vs. Real-World Testing

Regarding the efforts in simulation-based testing, a natural question arises that, how far is the simulation-based testing still from the real-world testing. Moreover, Kalra et al. [287] find that the ADS should be driven hundreds millions of miles to demonstrate their reliability. As an emerging issue, this topic has attracted increasing research attention; here, we introduce the latest progress from two perspectives, namely, the realism of test cases and the realism of simulators.

**Realism of test cases.** One question arises in the simulation-based testing that the virtual scenarios generated by testing algorithms that lead to system failures may never happen in the real world. Indeed, simulators give a high liberty to create traffic participants, of which, nevertheless, only a subset can really happen.

There is a line of work that aims to bridge this gap and thus generate natural scenarios for ADS testing. Nalic et al. [288] propose a co-simulation framework using two simulation tools CARMAKER (for vehicle dynamics) and VISSIM (for traffic simulation); their framework can generate scenarios based on calibrated traffic models derived from real-world data. In their later work [289], stress testing method, which has been introduced in §6.1.1, is applied for increasing the number of detected critical scenarios under the co-simulation environment. Klischat et al. [290] utilize *OpenStreetMap* to extract real-world road intersections, and combine with SUMO to generate realistic traffic scenarios. Wen et al. [291] focus on triggering the events in a specific area near the ego vehicle, and a CNN-based selector is utilized to choose those scenario agents which could achieve more realistic results.

The following works [292–296] focus on reconstructing scenarios from public crash reports. Mostadi et al. [295] utilize a distance metric, i.e., *Manhattan distance*, to align the virtual scenarios to real-world scenarios. Computer vision algorithms, i.e., object detection and tracking, are adopted in [292, 296] to extract the trajectories of the vehicles from the crash videos. Gambi et al. [189, 293,

294] utilize *natural language processing* (NLP) techniques to extract the relevant information and then calculate the abstract trajectories for recreating the crash. Experimental results show that the method could accurately reconstruct the crashes in public reports, and the generated test cases are able to expose faults in open-source ADS, i.e., DEEPDRIVING [207].

There is also a line of work [297–300] that focuses on narrowing the reality gap in the training process. By including components such as augmented data, small scale cars, and real-world tracks, they could generate more realistic cases to train the perception model or reinforcement learning algorithms for automated driving.

**Realism of simulators.** A comparative study on the assessment of testing in different levels of simulation is performed by Antkiewicz et al. [301]. In their work, the authors study simulation-based testing, mixed-reality testing, and real-world testing on two scenarios, i.e., car following and surrogate actor pedestrian crossing. They propose various metrics, e.g., realism, costs, agility, scalability, and controllability, and based on these metrics, they compare the different testing schemes under evaluation. As their conclusion, they quantitatively show the performance difference among the testing schemes: although real-world testing is better in terms of realism, it is more costly, and less agile, scalable, and controllable, compared to simulation-based testing; the performance of mixed-reality testing is in the middle of them. Testing ADS in different simulators is studied by Borg et al. [302], in which they utilize search-based testing techniques to generate scenarios in two simulators, i.e., PRESCAN [303] and PRO-SIVIC [304]. They find notable differences of the test outputs, e.g., they detect different safety violations. Consequently, they recommend involving multiple simulators for more robust simulation-based testing in the future.

Although simulation-based testing cannot achieve the same realism as real-world testing, to what extent can the results of simulation-based testing benefit real-world testing? This question is investigated in [305], where the authors perform simulation-based testing to identify critical scenarios, and map them to a real-world environment. Their key insights involve that, 62.5% of the unsafe scenarios detected by the simulators translate to real collisions; and 93.3% of the safe scenarios with the simulators are also safe in the real world. Another question is whether the simulator-generated dataset can substitute real-world dataset for DNN-based ADS testing, which have been studied in [306, 307]. Moreover, they also compare offline testing, e.g., module-level testing, and online testing, e.g., system-level testing, in terms of their pros and cons. Experiments on DNN-based ADS show that: the average prediction error difference on two datasets is less than 0.1, which means the simulator-generated dataset can serve as an alternative to the real-world dataset; online testing is more suitable than offline testing for DNN-based ADS testing, since online testing could detect more errors, i.e., those errors caused by accumulation over time, than offline testing. Reway et al. [308] evaluate the simulation-to-reality gap by testing an object detection algorithm under three different environments, namely, a real proving ground and two simulation software, considering four weather conditions. The gap is quantitatively calculated by considering metrics such as *precision* and *recall* on each platform. One of their experimental results is that the gap between real and simulation domains under nighttime and rainy conditions is larger than that under daytime conditions.

#### 6.4 Answer to RQ2

Overall, we have surveyed more than 90 papers dedicated to the system-level testing of the ADS. We find that those module-level testing techniques, such as search-based testing, sampling, and adversarial attack, are also widely adopted for finding failures arising from collaborations over different modules at the system level. Besides, more metrics, which can be found in §6.1.2 and §6.1.3, are proposed or utilized for facilitating the testing process. Another observation is that more

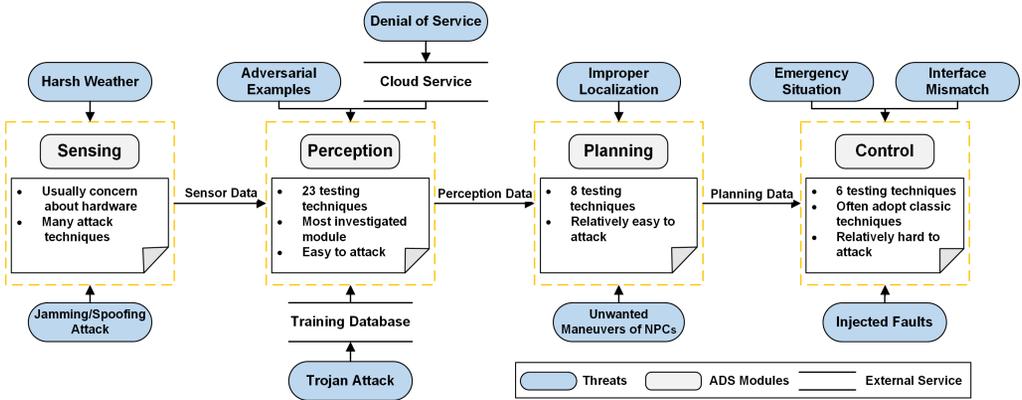


Fig. 5. Threat model of ADS

than 30 papers focus on bridging the gap between the simulation and the real-world environments, e.g., by introducing real components into the testing loop or by making a comparison between the simulation-based testing and the real-world testing.

Similar to module-level testing, there still remain several open challenges for system-level testing of the ADS. For example, since the system executions during testing are expensive and time-consuming, it needs future exploration on how to accelerate the testing process. More discussions about the challenges and future research directions can be found in §8.

## 7 STATISTICS AND ANALYSIS OF LITERATURE

In this section, based on the survey results in §4, §5 and §6, we perform a statistical analysis. Specifically, we provide the threat model for general ADS in §7.1, and we collect popular datasets, tool stacks, and programming languages for ADS testing in §7.2.

### 7.1 The Threat Model for General ADS

In this section, we construct a threat model in order to summarize the safety and security threats that each module may confront, based on our survey results. To build the threat model, we first summarize the threats discovered in the papers which we survey in the module-level testing in §5; then, as a complement, we review the bugs shown in the empirical studies [34, 35] on open-source ADS, to understand the concrete issues encountered in each module during system development. Our threat model is shown in Fig. 5.

**Threats to sensing.** In this module, existing studies mainly concern about the hardware aspect, e.g., those physical sensors which are critical hardware used in an ADS for collecting the information of the external environments. A common threat such as harsh weather conditions could reduce the capabilities of the intelligent sensors. There are also many deliberate attack techniques, such as *jamming attack* [52–54] and *spoofing attack* [55–58] (see details in §5.1.2) that target this module, and could interfere with these sensors and harm their normal functionalities.

**Threats to perception.** The perception module is the most investigated and we collect 23 testing techniques dedicated to this module. Common threat comes from *adversarial examples* that are generated by adding perturbations to normal images, which can fool the deep learning models in the perception module to make incorrect predictions, as shown by [61–65, 65–67, 71–80, 83]. Another type of threat is called *Trojan attack* [88, 89], in which malicious data are injected into the

training data of the deep learning models. Moreover, in the case when the ADS requires an HD map from the cloud service, *Denial of service (DoS)* [309] or fake HD map data [310] can interfere with perception tasks such as localization.

**Threats to planning.** With the data produced by the perception module, the planning module takes charge of several tasks, e.g., object trajectory prediction, path planning, and we collect 8 testing techniques for this module. A common threat comes from the unwanted maneuvers of *non-player characters (NPCs)* [137–139, 141–145], which can interfere with the prediction for moving objects and thus lead to an unsafe trajectory plan. Moreover, improper localization from the perception module can also threaten the accuracy of output trajectories.

**Threats to control.** This module mostly adopts those mature control techniques, e.g., *MPC* and *PID*, and thus are relatively hard to attack. One threat comes from the injected faults [151, 152], which could affect the longitudinal/lateral control of the vehicle. Another threat emerges when an emergency situation is encountered, e.g., when an emergency braking is needed, and the control module may fail to handle these cases. Moreover, the output signals are sent via the CAN bus to the ECU for controlling the vehicle. Since this process involves a data transmission between software and hardware, a potential threat is the interface mismatch [311], e.g., an inappropriate steering angle rate, in practical usage.

## 7.2 Datasets and Tool Stacks for ADS Testing

**Datasets.** In the context of ADS, deep learning components handle safety-critical tasks, e.g., perception and end-to-end control, so it is necessary to validate their robustness under various scenarios. This process typically relies on data from the real world, which is, however, hard to obtain in general. Fortunately, there are a collection of publicly available datasets to solve the problem, which involve large quantities of real-world pictures and videos recorded by onboard sensors. For example, the KITTI dataset [313] contains over 10,000 images of traffic scenarios, collected by a variety of sensors including high-resolution *RGB/grayscale stereo cameras* and a *3D laser scanner*.

In this section, we summarize the scenario-driven datasets for ADS testing in Table 10. The first column shows the time when each dataset was released. The next three columns give the name, brief description, and the size of each dataset, and the last column indicates the related works that adopt these datasets. Note that the datasets for other machine learning testing tasks [337] that have nothing to do with ADS testing are not listed here; in other words, all the datasets listed here are dedicated to ADS testing.

As shown in Table 10, we collect 27 datasets released from 2004 to 2022, including popular ones like the KITTI dataset [313] and emerging ones like the CrashD [336] dataset. One observation is that these datasets span over various physical conditions, e.g., different times of the day [317], different weather conditions [329, 332] and different traffic density [327]. They also span over various application scenarios, such as urban street [316, 320, 322], highway [325, 326, 338], and intersection [331]. In addition, we find that some of these datasets are specific to a certain task, e.g., pedestrian detection [319, 321], and traffic sign detection [125, 314, 315].

As the column of reference in Table 10 shows, several datasets such as the KITTI dataset [313] and the Udacity dataset [317] are frequently used in ADS testing due to the diverse tasks they support, such as object detection and semantic segmentation. However, we also find that a number of datasets have not been widely used, due to their own limitations. For example, the round [333] and openDD [334] can only be used for validating the behavior planning of ADS in the scenario of

Table 10. Scenario driven datasets for ADS testing

Time	Dataset	Description	Size	Reference
2004	NGSIM [312]	Vehicle trajectory data at four different locations	-	[143]
2012	KITTI [313]	Driving scenes captured by a standard station wagon	12,919 images	[58, 72, 73, 84, 99, 100, 167, 172, 247, 292, 292]
2013	GTSRB [125]	Containing 43 classes of traffic signs in Germany	50,000+ images	[66, 80, 83, 88]
2014	BelgiumTS [314]	A large dataset with traffic sign annotations	10,000+ images	[88]
2015	LISA [315]	A traffic sign dataset containing US traffic signs	43,000+ images	[80]
2016	Cityscapes [316]	A diverse set of stereo video sequences recorded in street scenes	25,000 images	[65, 92]
2016	Udacity [317]	Video frames taken from urban road	410,530 images	[107, 166, 167, 172-174, 177, 306]
2016	SYNTHIA [318]	Multiple categories of virtual city rendering pictures	220,000+ images	-
2016	Stanford Drone[319]	The movement and dynamics of pedestrians across the university campus	69GB videos and images	-
2017	RobotCar [320]	Various combinations of weather, traffic and pedestrians, as well as long-term changes such as road engineering	-	-
2017	CityPersons [321]	A dataset with a large proportion of blocked pedestrians images	5,050 images	-
2017	Mapillary Vistas [322]	Street view of multiple cities under multiple seasons and weather conditions	25,000 images	-
2018	GTA5 [323]	Synthetic images of urban traffic scenes collected using the game engine	24,966 images	[99]
2018	BDD100K [324]	Various scene types and weather conditions at different times of the day	100,000 videos	-
2018	comma2k19 [325]	Over 33 hours of commute in California's 280 highway	33h videos	[235]
2018	highD [326]	Traffic conditions of six different locations obtained by drone	147h videos	-
2018	ApolloScape [327]	Images under different conditions and traffic density	146,997 images	[94]
2019	ACFR [328]	Vehicle traces at 5 Roundabouts	23,000 images	-
2019	nuScenes [329]	Images under different times of day and weather conditions	1,400,000 images	-
2019	INTERACTION [330]	A dataset collected under interactive driving scenes with semantic maps	-	-
2019	Waymo [103]	Including a perception dataset with high-resolution sensor data and labels, and a motion dataset with object trajectory and corresponding 3D map	493,354 images	-
2020	inD [331]	Naturalistic trajectories of vehicles and vulnerable road users recorded at German intersections	10h videos	-
2020	Ford [332]	Multiple seasons, traffic conditions, and driving environments	-	-
2020	round [333]	Naturalistic trajectories of vehicles and vulnerable road users recorded at German roundabouts	-	-
2020	openDD [334]	A trajectory dataset covering seven roundabouts	62h videos	-
2021	Bosch Small Traffic Light [335]	An accurate dataset for vision-based traffic light detection	13,427 images	[58, 95]
2022	CrashD [336]	A synthetic LiDAR dataset to quantify the generality of 3D object detectors on out-of-domain samples	-	[300]

roundabouts; SYNTHIA [318] and GTA5 [339] contain synthetic images from virtual environments, which may be not realistic enough for ADS testing.

**Tool stacks.** As mentioned before, simulation-based testing has become an important alternative approach for real-world testing. Simulators usually provide vehicle dynamics, e.g., longitudinal and lateral motion of the vehicle, and virtual traffic scenarios. Moreover, simulators can help generate those extreme scenarios for testing, e.g., harsh weather, which are rarely encountered in the real world. There have been many advanced simulation platforms developed for ADS testing in recent years. For example, CARLA [345] is an open-source simulator for ADS training and testing, which supports various sensor models and environmental conditions.

In this section, we summarize the simulation platforms, namely, the simulators usually used for ADS testing in Table 11. As shown in the table, we collect 20 simulation platforms including classical platforms such as MATLAB/SIMULINK [340] and CARSIM [341], and emerging popular simulators such as CARLA and LGSVL [351]. Since these simulators have their own pros and cons, we compare them in the table and focus on several aspects of interest, e.g., their gap from real environment.

Table 11. Simulation platforms for ADS testing

Simulator	Open-source	Vehicle dynamic		X-in-the-loop				Interface to other simulators	Reference
		customization	soft/rigid	MiL	SiL	HiL	ViL		
MATLAB/SIMULINK [340]	×	✓	-	✓	✓	✓	-	CARSim, CARMAKER, PRESCAN, GAZEBO, CARLA, rFPRO, VTD, COGNATA, ADAMS PRO-SiVIC	[162, 163, 248] [188, 204, 277] [153, 218, 257] [195, 198]
CARSim [341]	×	✓	rigid	✓	✓	✓	-	MATLAB/SIMULINK, rFPRO, NVIDIA DRIVE SIM, VTD, PRO-SiVIC, DONKEY CAR	[195, 257]
VISSIM [281]	×	×	-	-	✓	✓	✓	CARLA, VTD, PRESCAN, CARMAKER, rFPRO, SUMO	[288, 289]
SUMO [280]	✓	×	-	-	✓	✓	✓	CARLA, VISSIM, COGNATA, rFPRO	[194, 242, 277] [283]
WEBOTS [342]	✓	-	rigid	-	✓	-	-	-	[247, 292]
VTD [343]	×	✓	-	✓	✓	✓	✓	CARSim, MATLAB/SIMULINK, ADAMS, VISSIM, rFPRO	[260, 308]
GAZEBO [344]	✓	-	rigid	-	✓	✓	-	MATLAB/SIMULINK, ADAMS	[254, 271, 275]
PRESCAN [303]	×	-	rigid	✓	✓	✓	✓	MATLAB/SIMULINK, VISSIM, PRO-SiVIC	[166, 188, 306] [195, 302]
BEAMNG [25]	✓	✓	soft	✓	✓	✓	✓	-	[189, 193, 293] [190, 200, 293] [191, 192]
CARLA [345]	✓	×	rigid	✓	✓	✓	✓	CARSim, VISSIM, SUMO, MATLAB/SIMULINK	[101, 171, 346]
AirSIM [347]	✓	×	rigid	-	✓	✓	-	-	-
rFPRO [348]	×	✓	rigid	-	✓	✓	-	CARSim, MATLAB/SIMULINK, CARMAKER, VISSIM, VTD, SUMO	-
COGNATA [349]	×	✓	-	✓	✓	✓	-	MATLAB/SIMULINK, SUMO	-
NVIDIA DRIVE SIM [350]	✓	✓	-	-	✓	✓	✓	CARMAKER, CARSim	-
LGSVL [351]	✓	×	-	✓	✓	✓	-	-	[72, 140, 196] [235, 252, 305]
SCANE STUDIO [352]	×	✓	soft/rigid	✓	✓	✓	✓	-	[164]
ADAMS [353]	×	✓	rigid	-	✓	✓	-	GAZEBO, MATLAB/SIMULINK, VTD	-
CARMAKER [222]	×	✓	rigid	✓	✓	✓	✓	MATLAB/SIMULINK, VISSIM, rFPRO, NVIDIA DRIVE SIM	[214, 277, 288] [218, 221, 289] [198, 221, 308]
PRO-SiVIC [354]	×	✓	-	✓	✓	✓	✓	MATLAB/SIMULINK, CARSim, PRESCAN	[302]
DONKEY CAR [282]	✓	×	-	-	✓	✓	✓	MATLAB/SIMULINK	[279]

Specifically, the first column lists the name and the second column shows the accessibility of each simulator. The third column is relevant to physical aspects, that is, whether the simulator allows for customizing a dynamic model and whether it is a soft-body or rigid-body based simulator. The fourth column indicates the level of support for mixed-reality testing, including model-in-the-loop (MiL), software-in-the-loop (SiL), hardware-in-the-loop (HiL), and vehicle-in-the-loop (ViL). The fifth column presents the capability of these simulators to complement each other, e.g., whether they support co-simulation with other simulators. The last column indicates the related works that adopt these simulators in their research.

Based on the table, we can draw the following conclusions:

Table 12. Open-source ADS

System Category	Name	Description
Modular	APOLLO	A commercial-grade ADS developed by Baidu
	AUTOWARE [355]	A L4 ADS developed by Nagoya University
	OPENPILOT	A commercial-grade L2 ADAS developed by Comma.ai
	PYLOT [23]	A modular ADS with low-latency dataflow from academic
End-to-end	LBC [356]	An end-to-end controller based on imitation learning
	DEEPDRIVING [357]	A CNN based end-to-end system that provides ACC and ALC
	NVIDIA CNN LANE FOLLOWER [358]	An end-to-end lane following system based on CNN
	UDACITY DNN MODELS [359]	DNN-based steering prediction models from the Udacity challenge, e.g., CHAUFFEUR [179] and EPOCH [180]
Other	BEAMNG.AI [15]	An AI agent in BEAMNG, which can realize simple control of vehicles
	CARLA PID [345]	A specific controller built in CARLA

- First, there exist many commercial simulators which are not open-source, e.g., CARMAKER [222] and PRESCAN [303]. These simulators could be expensive and difficult for researchers to satisfy their research goals. In comparison, open-source simulators like CARLA could have broader prospects for future research;
- Second, accurate physical dynamic models are needed to bridge the gap between simulation-based testing and real-world testing, and satisfy different testing requirements, e.g., smooth road needs a lower friction coefficient. We find that there have been simulators, i.e., BEAMNG [15] and CARSIM, dedicated to this aspect and allowing for dynamic model customization. In particular, BEAMNG is also a soft-body based simulator which supports more realistic collision effects (see more details in §2.3);
- Third, we find that most simulators support software-in-the-loop and hardware-in-the-loop testing. Several simulators, e.g., CARMAKER and VTD [343], support vehicle-in-the-loop testing, which closes the gap between hardware-in-the-loop testing and real-world testing.
- Lastly, we find that a number of simulators have built-in interfaces to other simulators. This is essential to perform co-simulation for ADS testing since these simulators have their own pros and cons, and co-simulation could complement each other for a more realistic testing environment. For example, CARMAKER (accurate vehicle dynamics) and VISSIM (representational traffic flow) are combined into a co-simulation framework [288] for generating more realistic testing scenarios.

Overall, it can be seen from the reference column that MATLAB/SIMULINK and BEAMNG have been widely used for ADS testing. Simulators like NVIDIA DRIVE SIM and GAZEBO also have great potential for future research since they cover multiple features we list in the table, e.g., whether they could perform ViL testing or co-simulation with other simulators.

Moreover, we also introduce several publicly available systems under test in Table 12. OPENPILOT, APOLLO, AUTOWARE and PYLOT are all modular systems and have already been described in §2.3, so we will not repeat them here. In addition to modular ADS systems, there also exist open-source end-to-end based systems: LBC [356] is an imitation learning controller, which uses camera images and direction commands as input to control the direction of the vehicle in the lane and intersection; DEEPDRIVING [357] and NVIDIA CNN LANE FOLLOWER [358] are also widely used CNN-based end-to-end controllers; Open-source DNN models from the Udacity self-driving challenge, such as CHAUFFEUR [179] and EPOCH [180], are another line of end-to-end driving controllers; Besides, there are also driving agents and controllers from simulators, such as BEAMNG and CARLA. BEAMNG.AI,

Table 13. Programming languages for scenario generation

Language	Dependencies	Supported simulators	Other features	Reference
OpenScenario	Unified Modeling Language (UML), XML	CARLA, Matlab, PRESCAN	A scenario is described in a “Storyboard” tag in XML, which includes a series of events	[101]
GeoScenario [360]	XML	An Unreal-based driving simulator	The language is based on open street map standard. Users can either program by dragging icons, or code in an XML editor.	-
Scenic [361]	Imperative, object oriented (Python-like)	CARLA, LGSVL, WEBOTS	It is a probabilistic programming language that can specify the input distributions of machine learning components, and use that information for testing and analysis.	[305]
stiEF [362]	Domain specific language	VTD	It supports multilingual representations for scenario description.	-
ScenML [363]	Graph-based modeling framework	CARLA	It allows information modeling at different depths, to support scenarios at different abstraction levels.	-
CommonRoad [364]	XML	SUMO	It provides a benchmark set that contains scenarios for the study of motion planning.	[143]
ScenGene [365]	A hierarchical representation model	-	It supports scenario generation via bio-inspired operations, such as crossover and mutation.	-
paracosm [366]	Reactive programming model	Udacity’s self-driving simulator	It adopts reactive objects that allow to describe temporal reactive behavior of entities. It also defines coverage criteria for test case generation.	-

which has been mentioned in §2.3, is an AI agent in BEAMNG simulator and could accept virtual images in the simulator as input for path planning and trajectory tracking. CARLA PID is a specific module in CARLA that performs calculations at the motion planning stage, and estimates the acceleration, braking, and steering inputs required to reach target positions.

**Programming languages.** In order to systematically generate test cases, it has become a trend to propose new programming languages for testing scenario description. In this way, the generation of a new test case boils down to writing a program that describes the scenario. Also, researchers can make use of existing coverage criteria for programs, such as the code coverage criteria, to assess the adequacy of the generated tests.

To define such a programming language, researchers need to formally express the basic elements in an ADS scenario, e.g., the ego car, other cars, pedestrians, and static objects. Since these languages are usually dependent on existing formats, they vary in their ways of expressing those elements, based on their dependent formats. For instance, *Scenic* [361], a python-like language, requires users to define those objects as variables; in contrast, *GeoScenario* [360] provides users with a graphical interface where users can drag the icons to describe a scenario. Moreover, these languages usually do not emerge independently; instead, they come with specific simulators, or even specific ADS.

In this section, we summarize the state-of-the-art programming languages for test case generation in Table 13, and introduce their dependent formalism, their bonded simulator, other features, and their adoption in ADS testing. There exists literature, e.g., [367], that surveys programming languages for the test generation of ADS. Compared to [367], our main aim is to show the ecosystems and the landscape of the use of these languages in ADS testing, as a reference for the readers to better understand the testing techniques in §5 and §6. Also, our study includes some latest achievements, e.g., *paracosm* [366] and *ScenGene* [365], in this direction.

As shown by Table 13, we collect 8 representative programming languages, including the classic ones, such as *OpenScenario*, that have been widely used in different stages of the development of ADS, and emerging ones, such as *paracosm* [366]. As our findings, first, different languages are

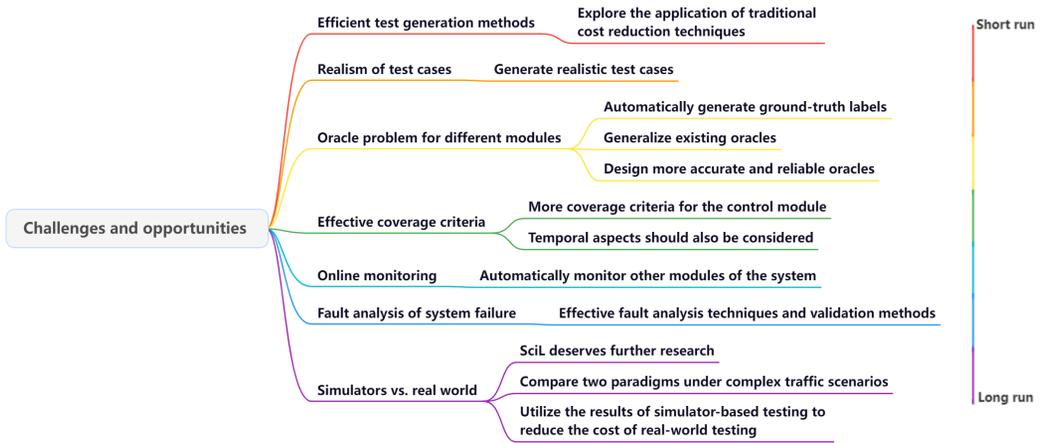


Fig. 6. Illustration of challenges and opportunities

designed for different purposes and attached with different features, e.g., Scenic [361] allows probabilistic sampling for testing driving systems with machine learning components; SceGene [365] designs bio-inspired operations, such as crossover and mutation, for scenario generation. Second, some of these languages provide more user-friendly features; for instance, some of the languages, e.g., SceML [363] provide a GUI for users to define their scenarios. However, as the column of reference shows, most of these languages have not been widely adopted in practice. This can be due to several reasons: one possibility is that some languages are still too specialized for practitioners to adopt them in their work; also, since many of the languages, such as GeoScenario [360], are designed for specific systems, they are still ad hoc and not easily extensible to be adopted in other systems.

In conclusion, programming languages are increasingly deemed as powerful weapons for test case generation in ADS testing, but currently they have not been widely adopted in practice yet.

## 8 CHALLENGES AND OPPORTUNITIES

As this survey reveals, ADS testing has experienced rapid growth in recent years. Nevertheless, there are still many challenges and open questions in its development and deployment. Based on our analysis of the collected literature and our discussions in each section, we answer RQ3 by listing the challenges and opportunities in this direction, as shown in Fig. 6. To account for it, there exist several solutions to the first four challenges that could be improved, while the last three challenges still need further exploration and require a long period of research.

**Efficient test generation methods.** Efficiency is one of the most important objectives in ADS testing, since system executions, whether in simulator environments or the real world, are too expensive. There have been many methods that aim to reduce the number of system executions, e.g., training surrogate models [202–205], or adopting sampling-based methods [217–219, 221, 223, 224, 228–230], as discussed in §6.1.1. However, there are several limitations to these methods; for example, the process of preparing training data in [202] for surrogate models is time-consuming. One potential future direction is to explore the application of traditional cost reduction techniques, such as test selection and test prioritization, to further accelerate the testing process.

**Realism of test cases.** Generating realistic test cases that can really threaten the safety of ADS in the real world should be another important goal of test case generation. Unrealistic test cases that

cannot happen in the real world are meaningless and not worthy of being taken care of. However, compared to efficiency, this aspect is usually ignored. Generating realistic test cases is a demand over different modules, and some existing works have paid attention to this problem. For example, in the perception module testing,  $RP_2$  [80] is proposed to generate test cases under real physical conditions; in the planning module testing, *avoidable collision* [148] is proposed to filter out useless test cases; moreover, this is also a major issue in system-level testing, as discussed in §6.3. In addition to these efforts, the problem is worthy of more attention, in order to find out those really useful test cases.

**Oracle problem for different modules.** Although there have been many works that try to design suitable oracles for different modules of ADS, there still remain many open challenges in defining oracles regarding different characteristics of different modules. For the perception module, as discussed in §5.2.4, the automatically labeling method in [92] targets only at semantic segmentation models, so one future direction is to explore how to automatically generate high-fidelity ground-truth labels for other types of models in the perception module. For the planning module, as discussed in §5.3.4, the criteria such as *avoidable collision* [148] are ad hoc and may not be generalized to other systems. *Metamorphic relations* are widely adopted by works [94–99, 165, 173] for different modules, but they may lack sufficiently accuracy and so lead to false positives. Hence, one potential future direction is to design more accurate and reliable oracles for the testing of different modules in ADS.

**Effective coverage criteria.** Coverage criteria are used as guidance to generate diverse test cases for testing. As discussed in §5 and §6, various coverage criteria have been proposed for testing different modules of the ADS, e.g., *neuron coverage* [107] for perception and end-to-end modules, *weight coverage* [137] and *route coverage* [140] for the planning module. Notably, few coverage criteria have been proposed for the control module, which indicates a future research direction. Moreover, one problem in the existing studies is that they mainly consider covering the spatial aspects of the test cases; for instance, *neuron coverage* [107] is computed based on the activated neurons in a DNN model and used as a guidance to trigger diverse behavior of single DNNs. However, in the testing of ADS which run over a time period, even though a strange behavior for a moment is triggered to happen, if it is immediately corrected, it may not affect the system behavior over the period. Therefore, in the testing of ADS, we need to trigger the diverse behavior of the DNNs over time. For instance, if a DNN keeps making wrong predictions for a period, it is likely to lead to a collision. Besides, several studies [134–136] have demonstrated that neuron coverage may not be suitable for guiding the testing process. Whether these findings will effect the ADS testing or there exists more effective criteria dedicated to perception testing needs further exploration. To sum up, coverage criteria dedicated to the control module are expected to be proposed in the future, and another research direction is to consider incorporating the temporal aspects into the existing coverage criteria.

**Online monitoring.** In this work, we mainly see testing techniques for ADS based on the posterior checking of the system execution; another effective quality assurance scheme is *online monitoring* [368, 369] that monitors the system behavior at runtime. As an advantage, online monitoring can detect unsafe behavior during the system execution, and thus warn drivers to take actions to avoid the safety risk. As discussed in §5.2.4, there have been some works, e.g., [101], that rely on formal temporal specifications to monitor the perception module at runtime. Besides, the model-based oracle proposed by Stocco et al. [175] is also a system-level online monitoring approach, as it predicts the misbehaviors of the system at runtime. However, how to automatically monitor other modules of the system remains a great challenge. One potential future direction is to develop

monitoring techniques for other modules. Meanwhile, more expressive specification languages should be provided to handle real-world system requirements.

**Fault analysis of system failure.** As this survey shows, the function of an ADS relies on the collaborative work of different modules; indeed, the wrong function of any module can cause a failure at the system level. Therefore, one question arises that which module should be deemed as the main cause of the system failure. Currently, as discussed in §6.1.4, the research attention is mostly focused on failure detection, rather than fault analysis. Moreover, fault analysis of ADS is challenging in nature, because it requires defining the boundaries of each module properly and making the oracles of each module clear. Sometimes, the failure of the system is not due to single modules but to the interactions between different modules. Therefore, one future direction is to propose effective fault analysis techniques as well as their validation methods.

**Simulators vs. real world.** Because of the high cost of real-world testing, simulation-based testing is the most commonly used testing paradigm; however, even with modern high-fidelity simulators (e.g., CARLA and LGSVL), there is still a gap from real-world testing. Recently, lightweight mixed-reality testing schemes, including *hardware-in-the-loop (HiL)*, *vehicle-in-the-loop (ViL)*, and *scenario-in-the-loop (SciL)* (more detail in §6.2), that mix the simulation-based testing and the real-world testing, also emerge to achieve a trade-off between the two. While HiL and ViL testing have developed quickly over the years, SciL testing, which is closest to the real world, is still at a theoretical stage and has not yet been widely adopted. As discussed in §7.2, existing simulators all have their pros and cons, and one future direction is to combine their distinguishing features, e.g., co-simulation, to enhance the realism of the simulation environment. Moreover, there have been several works in §6.3 that try to estimate how far the simulation-based testing is from the real-world testing. Nevertheless, in the case of handling complex traffic scenarios in testing, there are still open questions, such as the selection between simulation-based testing and real-world testing, and how to mitigate the weaknesses of the selected testing paradigm, that are seeking for better answers. To sum up, the gap between simulation-based testing and real-world testing still exists, and one research direction is to explore how to utilize the results of simulation-based testing to reduce the cost of real-world testing.

**Answer to RQ3.** Based on our survey results, we identify 7 major challenges for ADS testing and discuss the corresponding potential research opportunities. Moreover, as shown in Fig. 6, we find that several challenges such as the efficiency of test generation could be improved in the short run; by contrast, some other challenges (for example, how to mitigate the gap between simulation and real-world environments) may require a long period of research.

## 9 CONCLUSION

This survey provides a comprehensive overview and analysis of the relevant studies on ADS testing. These testing works cover both module-level testing and system-level testing of ADS, and we also include the works on empirical study w.r.t. system testing, mixed-reality testing, and real-world testing. In the introduction to the testing of each module, we respectively unfold the landscape of the literature from three perspectives, namely, test methodology, test oracle and test adequacy. Based on the literature review, we also perform analysis on the landscape of ADS testing, and propose a number of challenges and research opportunities in this direction.

Our work gives a specific emphasis on the technical differences in the testing of different modules, and also reveals the gap between simulation-based testing and real-world testing. Moreover, our analysis and discussion on the challenges and opportunities based on the literature review point out the future direction of research in this field. We hope that this work can inspire and motivate

more contributions to the safety assurance of ADS, and we also hope that ADS can be sufficiently reliable to be adopted by more people as early as possible.

## ACKNOWLEDGMENTS

This work was supported in part by the Anhui Provincial Department of Science and Technology under Grant 202103a05020009, in part by National Natural Science Foundation of China under Grant 61972373, the Basic Research Program of Jiangsu Province under Grant BK20201192 and the National Research Foundation Singapore under its NSoE Programme (Award Number: NSOE-TSS2019-03). The research of Dr. Xue is also supported by CAS Pioneer Hundred Talents Program of China. Lei Ma's research was supported in part by Canada CIFAR AI Chairs Program, NSERC (No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JSPS KAKENHI Grant No.JP20H04168, No.JP21H04877, JST-Mirai Program Grant No.JPMJMI20B8.

## REFERENCES

- [1] Mordor Intelligence Inc. Autonomous (driverless) car market - growth, trends, covid-19 impact, and forecast (2022 - 2027), 2022. URL <https://www.mordorintelligence.com/industry-reports/autonomous-driverless-cars-market-potential-estimation>.
- [2] Jin Cui, Lin Shen Liew, Giedre Sabaliauskaite, and Fengjun Zhou. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90:101823, 2019.
- [3] Bay City News. Fatal crash on sb i-680 onramp in san jose, 2022. URL <https://www.kron4.com/news/bay-area/fatal-crash-on-sb-i-680-onramp-in-san-jose/>.
- [4] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.
- [5] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.
- [6] Francisca Rosique, Pedro J Navarro, Carlos Fernández, and Antonio Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 19(3):648, 2019.
- [7] Xinhai Zhang, Jianbo Tao, Kaige Tan, Martin Torngren, Jose Manuel Gaspar Sanchez, Muhammad Rusyadi Ramli, Xin Tao, Magnus Gyllenhammar, Franz Wotawa, Naveen Mohan, et al. Finding critical scenarios for automated driving systems: A systematic mapping study. *IEEE Transactions on Software Engineering*, 2022.
- [8] Ziyuan Zhong, Yun Tang, Yuan Zhou, Vania de Oliveira Neves, Yang Liu, and Baishakhi Ray. A survey on scenario-based testing for automated driving systems in high-fidelity simulation. *arXiv preprint arXiv:2112.00964*, 2021.
- [9] Gunel Jahangirova, Andrea Stocco, and Paolo Tonella. Quality metrics and oracles for autonomous vehicles testing. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 194–204. IEEE, 2021.
- [10] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.
- [11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement.
- [13] Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.
- [14] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.
- [15] BeamNG. [beamng.tech](https://beamng.tech/). <https://beamng.tech/>, .
- [16] Eric Walz. <https://m.futurecar.com/5209/Baidus-Apollo-Go-Robotaxis-Are-Now-Deployed-in-All-of-Chinas-Top-Tier-Cities-After-Launching-in-Shenzhen>.
- [17] TIER IV. <https://www.mih-ev.org/en/featuring-info/?id=1140>.
- [18] R.Baldwin. <https://www.engadget.com/2020-01-13-comma-two-openpilot-hands-on.html>.
- [19] Baidu. <https://cyber-rt.readthedocs.io/en/latest/>.
- [20] ROS. <https://www.ros.org/>.
- [21] comma.ai. <https://comma.ai/shop/products/comma-car-harness>, .
- [22] comma.ai. <https://comma.ai/shop/products/two>, .
- [23] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A Wright, Joseph E Gonzalez, and Ion Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.

- [24] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. D3: a dynamic deadline-driven approach for building autonomous vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 453–471, 2022.
- [25] BeamNG GmbH. Beamng.tech technical paper. <https://www.beamng.gmbh/research>.
- [26] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. Sbst tool competition 2021. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*, pages 20–27. IEEE, 2021.
- [27] BeamNG. Beamng.ai. <https://documentation.beamng.com/tutorials/ai/>.
- [28] Shuncheng Tang, Zhenya Zhang, Yi Zhang, Jixiang Zhou, Yan Guo, Shuang Liu, Shengjian Guo, Yan-Fu Li, Lei Ma, Yinxing Xue, and Yang Liu. A Survey on Automated Driving System Testing: Landscapes and Trends, November 2022. URL <https://doi.org/10.5281/zenodo.7304210>.
- [29] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Martin J. Shepperd, Tracy Hall, and Ingunn Myrtveit, editors, *18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, London, England, United Kingdom, May 13-14, 2014*, pages 38:1–38:10. ACM, 2014. doi: 10.1145/2601248.2601268. URL <https://doi.org/10.1145/2601248.2601268>.
- [30] Testing of Autonomous Vehicles with a Driver. State of california department of motor vehicles. <https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing>, 2019.
- [31] Zi Peng, Jinqiu Yang, Tse-Hsun Chen, and Lei Ma. A first look at the integration of machine learning models in complex autonomous driving systems: a case study on apollo. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1240–1250, 2020.
- [32] Dinghua Wang, Shuqing Li, Guanping Xiao, Yepang Liu, and Yulei Sui. An exploratory study of autopilot software bugs in unmanned aerial vehicles. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 20–31, 2021.
- [33] Fiorella Zampetti, Ritu Kapur, Massimiliano Di Penta, and Sebastiano Panichella. An empirical characterization of software bugs in open-source cyber-physical systems. *Journal of Systems and Software*, 192:11425, 2022.
- [34] Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, and Qi Alfred Chen. A comprehensive study of autonomous vehicle bugs. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 385–396, 2020.
- [35] Shuncheng Tang, Zhenya Zhang, Jia Tang, Lei Ma, and Yinxing Xue. Issue categorization and analysis of an open-source driving assistant system. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 148–153. IEEE, 2021.
- [36] Chen Lv, Dongpu Cao, Yifan Zhao, Daniel J Auger, Mark Sullman, Huaji Wang, Laura Millen Dutka, Lee Skrypchuk, and Alexandros Mouzakitis. Analysis of autopilot disengagements occurring during autonomous vehicle testing. *IEEE/CAA Journal of Automatica Sinica*, 5(1):58–68, 2017.
- [37] Alexandra M Boggs, Ramin Arvin, and Asad J Khattak. Exploring the who, what, when, where, and why of automated vehicle disengagements. *Accident Analysis & Prevention*, 136:105406, 2020.
- [38] Zulqarnain H Khattak, Michael D Fontaine, and Brian L Smith. Exploratory investigation of disengagements and crashes in autonomous vehicles under mixed traffic: An endogenous switching regime framework. *IEEE Transactions on intelligent transportation systems*, 22(12):7485–7495, 2020.
- [39] Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.
- [40] Shervin Hajinia Leilabadi and Stephan Schmidt. In-depth analysis of autonomous vehicle collisions in california. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 889–893. IEEE, 2019.
- [41] Francesca M Favarò, Nazanin Nader, Sky O Eurich, Michelle Tripp, and Naresh Varadaraju. Examining accident reports involving autonomous vehicles in california. *PLoS one*, 12(9):e0184952, 2017.
- [42] Song Wang and Zhixia Li. Exploring the mechanism of crashes with automated vehicles using statistical modeling approaches. *PLoS one*, 14(3):e0214550, 2019.
- [43] Subasish Das, Anandi Dutta, and Ioannis Tspakis. Automated vehicle collisions in california: Applying bayesian latent class model. *IATSS research*, 44(4):300–308, 2020.
- [44] HM Abdul Aziz and AM Hasibul Islam. A data-driven framework to identify human-critical autonomous vehicle testing and deployment zones. In *Proceedings of the 14th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 1–8, 2021.
- [45] Yu Song, Madhav V Chitturi, and David A Noyce. Automated vehicle crash sequences: Patterns and potential uses in safety testing. *Accident Analysis & Prevention*, 153:106017, 2021.
- [46] E Esenturk, Siddhartha Khastgir, A Wallace, and P Jennings. Analyzing real-world accidents for test scenario generation for automated vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 288–295. IEEE, 2021.
- [47] Department for Transport. Road safety data - stats19. <https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-927747e5ce24a11f/road-safety-data>, 2019.

- [48] Xinpeng Wang, Ding Zhao, Hui Peng, and David J LeBlanc. Analysis of unprotected intersection left-turn conflicts based on naturalistic driving data. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 218–223. IEEE, 2017.
- [49] D LeBlanc, James R Sayer, Shan Bao, Scott Bogard, Mary Lynn Buonarosa, Adam Blankespoor, and Dillon Funkhouser. Driver acceptance and behavioral changes with an integrated warning system: Key findings from the ivbss fot. *22nd ESV*, pages 1–10, 2011.
- [50] Matti Kutila, Pasi Pyykönen, Werner Ritter, Oliver Sawade, and Bernd Schäufler. Automotive lidar sensor development scenarios for harsh weather conditions. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 265–270. IEEE, 2016.
- [51] Matti Kutila, Pasi Pyykönen, Hanno Holzhüter, Michele Colomb, and Pierre Duthon. Automotive lidar performance verification in fog and rain. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1695–1701. IEEE, 2018.
- [52] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 445–467. Springer, 2017.
- [53] Chen Yan, Wenyuan Xu, and Jianhao Liu. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. *Def Con*, 24(8):109, 2016.
- [54] Bing Shun Lim, Sye Loong Keoh, and Vrizlynn LL Thing. Autonomous vehicle ultrasonic sensor vulnerability and impact assessment. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 231–236. IEEE, 2018.
- [55] Qian Meng, Li-Ta Hsu, Bing Xu, Xiapu Luo, and Ahmed El-Mowafy. A gps spoofing generator using an open sourced vector tracking-based receiver. *Sensors*, 19(18):3993, 2019.
- [56] Kexiong Curtis Zeng, Shinan Liu, Yuanchao Shu, Dong Wang, Haoyu Li, Yanzhi Dou, Gang Wang, and Yaling Yang. All your {GPS} are belong to us: Towards stealthy manipulation of road navigation systems. In *27th {USENIX} security symposium ({USENIX} security 18)*, pages 1527–1544, 2018.
- [57] Rony Komissarov and Avishai Wool. Spoofing attacks against vehicular fmcw radar. In *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, pages 91–97, 2021.
- [58] Wei Wang, Yao Yao, Xin Liu, Xiang Li, Pei Hao, and Ting Zhu. I can see the light: Attacks on autonomous vehicles using invisible lights. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1930–1944, 2021.
- [59] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS’17 Competition: Building Intelligent Systems*, pages 195–231. Springer, 2018.
- [60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [61] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018.
- [62] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn’t believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1989–2004, 2019.
- [63] Yang Zhang, PD Hassan Foroosh, and Boqing Gong. Camou: Learning a vehicle camouflage for physical adversarial attack on object detections in the wild. *ICLR*, 2019.
- [64] Jung Im Choi and Qing Tian. Adversarial attack and defense of yolo detectors in autonomous driving scenarios. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1011–1017. IEEE, 2022.
- [65] Xing Xu, Jingran Zhang, Yujie Li, Yichuan Wang, Yang Yang, and Heng Tao Shen. Adversarial attack against urban scene segmentation for autonomous vehicles. *IEEE Transactions on Industrial Informatics*, 17(6):4117–4126, 2020.
- [66] Yujie Li, Xing Xu, Jinhui Xiao, Siyuan Li, and Heng Tao Shen. Adaptive square attack: Fooling autonomous cars with adversarial traffic signs. *IEEE Internet of Things Journal*, 8(8):6337–6347, 2020.
- [67] K Naveen Kumar, C Vishnu, Reshmi Mitra, and C Krishna Mohan. Black-box adversarial attacks in autonomous vehicle technology. In *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7. IEEE, 2020.
- [68] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.
- [69] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [70] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

- [71] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.
- [72] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021.
- [73] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 877–894, 2020.
- [74] Xupeng Wang, Mumuxin Cai, Ferdous Sohel, Nan Sang, and Zhengwei Chang. Adversarial point cloud perturbations against 3d object detection in autonomous driving systems. *Neurocomputing*, 466:27–36, 2021.
- [75] Chang Chen and Teng Huang. Camdar-adv: generating adversarial patches on 3d object. *International Journal of Intelligent Systems*, 36(3):1441–1453, 2021.
- [76] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Can we use arbitrary objects to attack lidar perception in autonomous driving? In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1945–1960, 2021.
- [77] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. Robust roadside physical adversarial attack against deep learning in lidar perception modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 349–362, 2021.
- [78] Yi Zhu, Chenglin Miao, Foad Hajiaghajani, Mengdi Huai, Lu Su, and Chunming Qiao. Adversarial attacks against lidar semantic segmentation in autonomous driving. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 329–342, 2021.
- [79] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021.
- [80] Kevin Eykholt, Ivan Evtimov, Earlece Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.
- [81] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlece Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18)*, 2018.
- [82] Chaowei Xiao, Bo Li, Jun Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 3905–3911. International Joint Conferences on Artificial Intelligence, 2018.
- [83] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1028–1035, 2019.
- [84] Zuobin Xiong, Honghui Xu, Wei Li, and Zhipeng Cai. Multi-source adversarial sample attack on autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 70(3):2822–2835, 2021.
- [85] Handi Yu and Xin Li. Intelligent corner synthesis via cycle-consistent generative adversarial networks for efficient validation of autonomous driving systems. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 9–15. IEEE, 2018.
- [86] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
- [87] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.
- [88] Wenbo Jiang, Hongwei Li, Sen Liu, Xizhao Luo, and Rongxing Lu. Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles. *IEEE transactions on vehicular technology*, 69(4):4439–4449, 2020.
- [89] Shaohua Ding, Yulong Tian, Fengyuan Xu, Qun Li, and Sheng Zhong. Trojan attack on deep generative models in autonomous driving. In *International Conference on Security and Privacy in Communication Systems*, pages 299–318. Springer, 2019.
- [90] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS’95. Proceedings of the sixth international symposium on micro machine and human science*, pages 39–43. Ieee, 1995.
- [91] Rui Qian, Robby T Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2482–2491, 2018.

- [92] Wei Zhou, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1951–1963, 2019.
- [93] Robin Philipp, Zhijing Zhu, Julian Fuchs, Lukas Hartjen, Fabian Schuldt, and Falk Howar. Automated 3d object reference generation for the evaluation of autonomous vehicle perception. In *2021 5th International Conference on System Reliability and Safety (ICSRS)*, pages 312–321. IEEE, 2021.
- [94] Jinyang Shao. Testing object detection for autonomous driving systems via 3d reconstruction. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 117–119. IEEE, 2021.
- [95] Tongtong Bai, Yong Fan, Ya Pan, and Mingshuang Qing. Metamorphic testing for traffic light recognition in autonomous driving systems. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 38–44. IEEE, 2021.
- [96] Zhi Quan Zhou and Liqun Sun. Metamorphic testing of driverless cars. *Communications of the ACM*, 62(3):61–67, 2019.
- [97] Trey Woodlief, Sebastian Elbaum, and Kevin Sullivan. Semantic image fuzzing of ai perception systems. In *Proceedings of the 44th International Conference on Software Engineering*, pages 1958–1969, 2022.
- [98] Xiangling Wang, Siqi Yang, Jinyang Shao, Jun Chang, Ge Gao, Ming Li, and Jifeng Xuan. Object removal for testing object detection in autonomous vehicle systems. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 543–549. IEEE, 2021.
- [99] Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan, and Matthew Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018.
- [100] Adel Dokhanchi, Heni Ben Amor, Jyotirmoy V Deshmukh, and Georgios Fainekos. Evaluating perception systems for autonomous vehicles using quality temporal logic. In *International Conference on Runtime Verification*, pages 409–416. Springer, 2018.
- [101] Anand Balakrishnan, Jyotirmoy Deshmukh, Bardh Hoxha, Tomoya Yamaguchi, and Georgios Fainekos. Percemon: Online monitoring for perception systems. In *International Conference on Runtime Verification*, pages 297–308. Springer, 2021.
- [102] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrusis, Alexander Brock, Burkhard Gusefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28, 2016.
- [103] Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.
- [104] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. Metamorphic testing: a new approach for generating next test cases. *arXiv preprint arXiv:2002.12543*, 2020.
- [105] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. iee, 1977.
- [106] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.
- [107] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.
- [108] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.
- [109] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.
- [110] Changhai Nie and Hareton Leung. A survey of combinatorial testing. *ACM Computing Surveys (CSUR)*, 43(2):1–29, 2011.
- [111] Christoph Gladisch, Christian Heinzemann, Martin Herrmann, and Matthias Woehle. Leveraging combinatorial testing for safety-critical computer vision datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 324–325, 2020.
- [112] Chih-Hong Cheng, Chung-Hao Huang, and Hirotoishi Yasuoka. Quantitative projection coverage for testing ml-enabled autonomous systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 126–142. Springer, 2018.
- [113] Qin Xia, Jianli Duan, Feng Gao, Qiuxia Hu, and Yingdong He. Test scenario design for intelligent driving system ensuring coverage and effectiveness. *International Journal of Automotive Technology*, 19(4):751–758, 2018.

- [114] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [116] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [117] Kaggle InClass challenge. <https://www.kaggle.com/c/nyu-cv-fall-2018/data>.
- [118] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointnet: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.
- [119] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.
- [120] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.
- [121] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.
- [122] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [123] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020.
- [124] Andreas Mogelmoose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.
- [125] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.
- [126] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [127] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [128] V Yadav. p2-traffic signs. <https://github.com/vxy10/p2-TrafficSigns>.
- [129] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.
- [130] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.
- [131] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [132] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018.
- [133] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.
- [134] Shenao Yan, Guan hong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 775–787, 2020.
- [135] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. Is neuron coverage a meaningful measure for testing deep neural networks? In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 851–862, 2020.
- [136] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(2):1–22, 2021.
- [137] Thomas Laurent, Paolo Arcaini, Fuyuki Ishikawa, and Anthony Ventresque. A mutation-based approach for assessing weight coverage of a path planner. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 94–101. IEEE, 2019.

- [138] Thomas Laurent, Paolo Arcaini, Fuyuki Ishikawa, and Anthony Ventresque. Achieving weight coverage for an autonomous driving system with search-based test generation. In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 93–102. IEEE, 2020.
- [139] Paolo Arcaini, Xiao-Yi Zhang, and Fuyuki Ishikawa. Targeting patterns of driving characteristics in testing autonomous driving systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 295–305. IEEE, 2021.
- [140] Yun Tang, Yuan Zhou, Fenghua Wu, Yang Liu, Jun Sun, Wuling Huang, and Gang Wang. Route coverage testing for autonomous vehicles via map modeling. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11450–11456. IEEE, 2021.
- [141] Paolo Arcaini, Xiao-Yi Zhang, and Fuyuki Ishikawa. Less is more: Simplification of test scenarios for autonomous driving system testing. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 279–290. IEEE, 2022.
- [142] Matthias Althoff and Sebastian Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.
- [143] Moritz Klischat and Matthias Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2352–2358. IEEE, 2019.
- [144] Stanley Bak, Johannes Betz, Abhinav Chawla, Hongrui Zheng, and Rahul Mangharam. Stress testing autonomous racing overtake maneuvers with rrt. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 806–812. IEEE, 2022.
- [145] Maximilian Kahn, Atrisha Sarkar, and Krzysztof Czarnecki. I know you can’t see me: Dynamic occlusion-aware safety validation of strategic planners for autonomous vehicles using hypergames. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11202–11208. IEEE, 2022.
- [146] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1821–1827. IEEE, 2018.
- [147] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.
- [148] Alessandro Calò, Paolo Arcaini, Shaukat Ali, Florian Hauer, and Fuyuki Ishikawa. Generating avoidable collision scenarios for testing autonomous driving systems. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 375–386. IEEE, 2020.
- [149] Alessandro Calò, Paolo Arcaini, Shaukat Ali, Florian Hauer, and Fuyuki Ishikawa. Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 1055–1063, 2020.
- [150] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010.
- [151] Garazi Juez Uriagereka, Ray Lattarulo, Joshue Pérez Rastelli, Estibaliz Amparan Calonge, Alejandra Ruiz Lopez, and Huascar Espinoza Ortiz. Fault injection method for safety and controllability evaluation of automated driving. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1867–1872. IEEE, 2017.
- [152] Xugui Zhou, Anna Schmedding, Haotian Ren, Lishan Yang, Philip Schowitz, Evgenia Smirni, and Homa Alemzadeh. Strategic safety-critical attacks against an advanced driver assistance system. *arXiv preprint arXiv:2204.06768*, 2022.
- [153] Xinpeng Wang, Yiqun Dong, Shaobing Xu, Hui Peng, Fucong Wang, and Zhenghao Liu. Behavioral competence tests for highly automated vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1323–1329. IEEE, 2020.
- [154] Xinpeng Wang, Hui Peng, Songan Zhang, and Kuan-Hui Lee. An interaction-aware evaluation method for highly automated vehicles. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 394–401. IEEE, 2021.
- [155] Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, and Ichiro Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2894–2905, 2018.
- [156] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. Effective hybrid system falsification using monte carlo tree search guided by qb-robustness. In *International Conference on Computer Aided Verification*, pages 595–618. Springer, 2021.
- [157] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.
- [158] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.
- [159] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. Falsifai: Falsification of ai-enabled hybrid control systems guided by time-aware coverage criteria. *IEEE Transactions on Software Engineering*, 2022.

- [160] Jiayang Song, Deyun Lyu, Zhenya Zhang, Zhijie Wang, Tianyi Zhang, and Lei Ma. When cyber-physical systems meet ai: a benchmark, an evaluation, and a way forward. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, pages 343–352, 2022.
- [161] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Aniruddh Chandratre, Alexandre Donzé, Georgios Fainekos, Goran Frehse, Khouloud Gaaloul, Jun Inoue, Tanmay Khandait, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Masaki Waga, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2021 category report: Falsification with validation of results. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21), Brussels, Belgium, July 9, 2021*, volume 80 of *EPiC Series in Computing*, pages 133–152. EasyChair, 2021.
- [162] Cumhur Erkan Tuncali, Theodore P Pavlic, and Georgios Fainekos. Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In *2016 IEEE 19th international conference on intelligent transportation systems (itsc)*, pages 1470–1475. IEEE, 2016.
- [163] Cumhur Erkan Tuncali and Georgios Fainekos. Rapidly-exploring random trees for testing automated vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 661–666. IEEE, 2019.
- [164] Adel Djoudi, Loic Coquelin, and Rémi Regnier. A simulation-based framework for functional testing of automated driving controllers. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [165] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.
- [166] Zhong Li, Minxue Pan, Tian Zhang, and Xuandong Li. Testing dnn-based autonomous driving systems under critical environmental conditions. In *International Conference on Machine Learning*, pages 6471–6482. PMLR, 2021.
- [167] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358. IEEE, 2020.
- [168] Svetlana Pavlitskaya, Sefa Ünver, and J Marius Zöllner. Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.
- [169] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [170] Adith Bloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Simple physical adversarial examples against end-to-end autonomous driving models. In *2019 IEEE International Conference on Embedded Software and Systems (ICESSE)*, pages 1–7. IEEE, 2019.
- [171] Adith Bloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*, 110:101766, 2020.
- [172] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14254–14263, 2020.
- [173] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142. IEEE, 2018.
- [174] Ya Pan, Haiyang Ao, and Yong Fan. Metamorphic testing for autonomous driving systems in fog based on quantitative measurement. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 30–37. IEEE, 2021.
- [175] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 359–371, 2020.
- [176] Manzoor Hussain, Nazakat Ali, and Jang-Eui Hong. Deepguard: a framework for safeguarding autonomous driving systems from inconsistent behaviour. *Automated Software Engineering*, 29(1):1–32, 2022.
- [177] Jaganmohan Chandrasekaran, Yu Lei, Raghu Kacker, and D Richard Kuhn. A combinatorial approach to testing deep neural network-based autonomous driving systems. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 57–66. IEEE, 2021.
- [178] Team Rambo. Steering angle model: Rambo. <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo>.
- [179] Team Chauffeur. Steering angle model: Chauffeur. <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur>.
- [180] Team Epoch. Steering angle model: Epoch. <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/cg23>.

- [181] NVIDIA. Nvidia-autopilot-keras: End to end learning for self-driving cars. <https://github.com/0bserver07/Nvidia-Autopilot-Keras>, .
- [182] Alex Starovoitov. Behavioral cloning: end-to-end learning for self-driving cars. <https://github.com/navoshta/behavioral-cloning>.
- [183] Jacob Gildenblat. Visualizations for understanding the regressed wheel steering angle for self driving cars. <https://github.com/jacobgil/keras-steering-angle-visualizations>, 2016.
- [184] Christian Hubschneider, Andre Bauer, Michael Weber, and J Marius Zöllner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.
- [185] Team Autumn. Steering angle model: Autumn. <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/autumn>.
- [186] Team Rwrightman. Steering angle model: Rwrightman. <https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation>, 2016.
- [187] Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning*, 63(4):1031–1053, 2019.
- [188] Raja Ben Abdesslem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 143–154. IEEE, 2018.
- [189] Alessio Gambi, Marc Mueller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328, 2019.
- [190] Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 876–888, 2020.
- [191] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. Deephyperion: exploring the feature space of deep learning-based systems through illumination search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 79–90, 2021.
- [192] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. Efficient and effective feature space exploration for testing deep learning systems. *ACM Trans. Softw. Eng. Methodol.*, may 2022. ISSN 1049-331X. doi: 10.1145/3544792. URL <https://doi.org/10.1145/3544792>. Just Accepted.
- [193] Ezequiel Castellano, Ahmet Cetinkaya, and Paolo Arcaini. Analysis of road representations in search-based testing of autonomous driving systems. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pages 167–178. IEEE, 2021.
- [194] Quentin Goss and Mustafa İlhan Akbaş. Eagle strategy with local search for scenario based validation of autonomous vehicles. In *2022 International Conference on Connected Vehicle and Expo (ICCVE)*, pages 1–6. IEEE, 2022.
- [195] Xiaokun Zheng, Huawei Liang, Biao Yu, Bichun Li, Shaobo Wang, and Zhilei Chen. Rapid generation of challenging simulation scenarios for autonomous vehicles based on adversarial test. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1166–1172. IEEE, 2020.
- [196] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–36. IEEE, 2020.
- [197] Florian Hauer, Alexander Pretschner, and Bernd Holzmüller. Fitness functions for testing automated and autonomous driving systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 69–84. Springer, 2019.
- [198] Nicola Kolb, Florian Hauer, and Alexander Pretschner. Fitness function templates for testing automated and autonomous driving systems in intersection scenarios. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 217–222. IEEE, 2021.
- [199] Yixing Luo, Xiao-Yi Zhang, Paolo Arcaini, Zhi Jin, Haiyan Zhao, Fuyuki Ishikawa, Rongxin Wu, and Tao Xie. Targeting requirements violations of autonomous driving systems by dynamic evolutionary search. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 279–291. IEEE, 2021.
- [200] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. Single and multi-objective test cases prioritization for self-driving cars in virtual environments. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2022.
- [201] Christian Birchler, Nicolas Ganz, Sajad Khatiri, Alessio Gambi, and Sebastiano Panichella. Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor. In *29th IEEE International Conference on Software Analysis, Evolution, and Reengineering, Honolulu, USA (online), 15-18 March 2022*. ZHAW Zürcher Hochschule für Angewandte Wissenschaften, 2022.

- [202] Raja Ben Abdesslem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, pages 63–74, 2016.
- [203] Felix Batsch, Alireza Daneshkhan, Vasile Palade, and Madeline Cheah. Scenario optimisation and sensitivity analysis for safe automated driving using gaussian processes. *Applied Sciences*, 11(2):775, 2021.
- [204] Halil Beglerovic, Michael Stolz, and Martin Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.
- [205] Jian Sun, Huajun Zhou, He Zhang, Ye Tian, and Qinghui Ji. Adaptive design of experiments for accelerated safety evaluation of automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2020.
- [206] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.
- [207] Voyage. Deepdrive. <https://deepdrive.io/>.
- [208] Ezequiel Castellano, Ahmet Cetinkaya, Cédric Ho Thanh, Stefan Klikovits, Xiaoyi Zhang, and Paolo Arcaini. Frenetic at the sbst 2021 tool competition. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*, pages 36–37. IEEE, 2021.
- [209] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. Sbst tool competition 2022. In *2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)*, pages 25–32. IEEE, 2022.
- [210] Raja Ben Abdesslem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 1016–1026. IEEE, 2018.
- [211] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.
- [212] Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019.
- [213] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.
- [214] Daniel Baumann, Raphael Pfeffer, and Eric Sax. Automatic generation of critical test cases for the development of highly automated driving functions. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5. IEEE, 2021.
- [215] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.
- [216] Dhanoop Karunakaran, Stewart Worrall, and Eduardo Nebot. Efficient statistical validation with edge cases to evaluate highly automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.
- [217] Yasuhiro Akagi, Ryosuke Kato, Sou Kitajima, Jacobo Antona-Makoshi, and Nobuyuki Uchida. A risk-index based sampling method to generate scenarios for the evaluation of automated driving vehicle safety. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 667–672. IEEE, 2019.
- [218] Philippe Nitsche, Ruth Helen Welsh, Alexander Genser, and Pete D Thomas. A novel, modular validation framework for collision avoidance of automated vehicles at road junctions. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 90–97. IEEE, 2018.
- [219] Anthony Corso and Mykel J Kochenderfer. Interpretable safety validation for autonomous vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [220] Anthony Corso, Ritchie Lee, and Mykel J Kochenderfer. Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [221] Felix Batsch, Alireza Daneshkhan, Madeline Cheah, Stratis Kanarachos, and Anthony Baxendale. Performance boundary identification for the evaluation of automated vehicles using gaussian process classification. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 419–424. IEEE, 2019.
- [222] IPG. Carmaker. <https://ipg-automotive.com/en/products-solutions/software/carmaker/>.
- [223] Barbara Schütt, Marc Heinrich, Sonja Marahrens, J. Marius Zöllner, and Eric Sax. An application of scenario exploration to find new scenarios for the development and testing of automated driving systems in urban scenarios. In Jeroen Ploeg, Markus Helfert, Karsten Berns, and Oleg Gusikhin, editors, *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2022, Online Streaming, April 27-29, 2022*, pages 338–345. SCITEPRESS, 2022. doi: 10.5220/0011064600003191. URL <https://doi.org/10.5220/0011064600003191>.

- [224] Lukas Birkemeyer, Tobias Pett, Andreas Vogelsang, Christoph Seidl, and Ina Schaefer. Feature-interaction sampling for scenario-based testing of advanced driver assistance systems\*. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 1–10, 2022.
- [225] Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.
- [226] Matthew O’Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. *Advances in neural information processing systems*, 31, 2018.
- [227] Justin Norden, Matthew O’Kelly, and Aman Sinha. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.
- [228] Ding Zhao, Henry Lam, Huei Peng, Shan Bao, David J LeBlanc, Kazutoshi Nobukawa, and Christopher S Pan. 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, 2016.
- [229] Zhiyuan Huang, Henry Lam, David J LeBlanc, and Ding Zhao. Accelerated evaluation of automated vehicles using piecewise mixture models. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2845–2855, 2017.
- [230] Zhiyuan Huang, Henry Lam, and Ding Zhao. An accelerated testing approach for automated vehicles with background traffic described by joint distributions. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 933–938. IEEE, 2017.
- [231] Ding Zhao, Xianan Huang, Huei Peng, Henry Lam, and David J LeBlanc. Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):733–744, 2017.
- [232] Zhiyuan Huang, Yaohui Guo, Mansur Arief, Henry Lam, and Ding Zhao. A versatile approach to evaluating and testing automated vehicles based on kernel methods. In *2018 Annual American Control Conference (ACC)*, pages 4796–4802. IEEE, 2018.
- [233] Zhiyuan Huang, Ding Zhao, Henry Lam, David J LeBlanc, and Huei Peng. Evaluation of automated vehicles in the frontal cut-in scenario—an enhanced approach using piecewise mixture models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 197–202. IEEE, 2017.
- [234] Xinpeng Wang, Huei Peng, and Ding Zhao. Combining reachability analysis and importance sampling for accelerated evaluation of highway automated vehicles at pedestrian crossing. *ASME Letters in Dynamic Systems and Control*, 1(1), 2021.
- [235] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under {Physical-World} attack. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3309–3326, 2021.
- [236] Abu Hasnat Mohammad Rubaiyat, Yongming Qin, and Homa Alemzadeh. Experimental resilience assessment of an open-source driving agent. In *2018 IEEE 23rd Pacific rim international symposium on dependable computing (PRDC)*, pages 54–63. IEEE, 2018.
- [237] Ben Nassi, Jacob Shams, Raz Ben-Netanel, and Yuval Elovici. badvertisement: Attacking advanced driver-assistance systems using print advertisements. In *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*, pages 376–383. IEEE, 2022. doi: 10.1109/EuroSPW55150.2022.00045. URL <https://doi.org/10.1109/EuroSPW55150.2022.00045>.
- [238] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021.
- [239] SM Sohel Mahmud, Luis Ferreira, Md Shamsul Hoque, and Ahmad Tavassoli. Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS research*, 41(4):153–163, 2017.
- [240] Lukas Westhofen, Christian Neurohr, Tjark Koopmann, Martin Butz, Barbara Schütt, Fabian Utesch, Birte Neurohr, Christian Gutenkunst, and Eckard Böde. Criticality metrics for automated driving: A review and suitability analysis of the state of the art. *Archives of Computational Methods in Engineering*, pages 1–35, 2022.
- [241] Hiroshi WAKABAYASHI, Yoshihiko TAKAHASHI, Shigehiro NIIMI, and Kazumi RENGE. Traffic conflict analysis using vehicle tracking system/digital vcr and proposal of a new conflict indicator. *Infrastructure Planning Review*, 20: 949–956, 2003.
- [242] Bowen Weng, Sugghosh J Rao, Eeshan Deosthale, Scott Schnelle, and Frank Barickman. Model predictive instantaneous safety metric for evaluation of automated driving systems. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1899–1906. IEEE, 2020.
- [243] Andreas Tamke, Thao Dang, and Gabi Breuel. A flexible method for criticality assessment in driver assistance systems. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 697–702. IEEE, 2011.
- [244] Sverker Almqvist, Christer Hyden, and Ralf Risser. Use of speed limiters in cars for increased safety and a better environment. *Transportation Research Record*, (1318), 1991.

- [245] John R McLean and Errol R Hoffmann. Analysis of drivers' control movements. *Human Factors*, 13(5):407–418, 1971.
- [246] Li Li, Wu-Ling Huang, Yuehu Liu, Nan-Ning Zheng, and Fei-Yue Wang. Intelligence testing for autonomous vehicles: A new approach. *IEEE Transactions on Intelligent Vehicles*, 1(2):158–166, 2016.
- [247] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1555–1562. IEEE, 2018.
- [248] Cumhur Erkan Tuncali, Georgios Fainekos, Danil Prokhorov, Hisahiro Ito, and James Kapinski. Requirements-driven test generation for autonomous vehicles with machine learning components. *IEEE Transactions on Intelligent Vehicles*, 5(2):265–280, 2019.
- [249] Qingzhao Zhang, David Ke Hong, Ze Zhang, Qi Alfred Chen, Scott Mahlke, and Z Morley Mao. A systematic framework to identify violations of scenario-dependent driving rules in autonomous vehicle software. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(2):1–25, 2021.
- [250] Jia Cheng Han and Zhi Quan Zhou. Metamorphic fuzz testing of autonomous vehicles. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 380–385, 2020.
- [251] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955. IEEE, 2019.
- [252] Yun Tang, Yuan Zhou, Yang Liu, Jun Sun, and Gang Wang. Collision avoidance testing for autonomous driving systems on complete maps. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 179–185. IEEE, 2021.
- [253] Jonas Kerber, Sebastian Wagner, Korbinian Groh, Dominik Notz, Thomas Kühbeck, Daniel Watzenig, and Alois Knoll. Clustering of the scenario space for the assessment of automated driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 578–583. IEEE, 2020.
- [254] István Majzik, Oszkár Semeráth, Csaba Hajdu, Kristóf Marussy, Zoltán Szatmári, Zoltán Micskei, András Vörös, Aren A Babikian, and Dániel Varró. Towards system-level testing with coverage guarantees for autonomous vehicles. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 89–94. IEEE, 2019.
- [255] Zaid Tahir and Rob Alexander. Intersection focused situation coverage-based verification and validation framework for autonomous vehicles implemented in carla. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 191–212. Springer, 2022.
- [256] Peng Guo and Feng Gao. Automated scenario generation and evaluation strategy for automatic driving system. In *2020 7th International conference on information science and control engineering (ICISCE)*, pages 1722–1733. IEEE, 2020.
- [257] Hong Shu, Haoran Lv, Kang Liu, Kang Yuan, and Xiaolin Tang. Test scenarios construction based on combinatorial testing strategy for automated vehicles. *IEEE Access*, 9:115019–115029, 2021.
- [258] Yihao Li, Jianbo Tao, and Franz Wotawa. Ontology-based test generation for automated and autonomous driving functions. *Information and software technology*, 117:106200, 2020.
- [259] Changwen Li, Chih-Hong Cheng, Tiantian Sun, Yuhang Chen, and Rongjie Yan. Comopt: Combination and optimization for testing autonomous driving systems. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7738–7744. IEEE, 2022.
- [260] Elias Rocklage, Heiko Kraft, Abdullah Karatas, and Jörg Seewig. Automated scenario generation for regression testing of autonomous vehicles. In *2017 IEEE 20th international conference on intelligent transportation systems (itsc)*, pages 476–483. IEEE, 2017.
- [261] Taeyoung Lee, Kyongsu Yi, Jangseop Kim, and Jaewan Lee. Development and evaluations of advanced emergency braking system algorithm for the commercial vehicle. In *22nd International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2011.
- [262] Udacity. Udacity's self-driving car simulator. <https://github.com/udacity/self-driving-car-sim>.
- [263] International electronics & engineering. <https://www.iee.lu/>.(2018).
- [264] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.
- [265] Uwe Kiencke and Lars Nielsen. *Automotive Control Systems: For Engine, Driveline and Vehicle*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2000. ISBN 3540669221.
- [266] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956. IEEE, 2019.
- [267] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

- [268] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.
- [269] Myrvin H Ellestad. *Stress testing: principles and practice*. Oxford University Press, 2003.
- [270] Yu Chen, Shitao Chen, Tangyike Zhang, Songyi Zhang, and Nanning Zheng. Autonomous vehicle testing and validation platform: Integrated simulation system with hardware in the loop. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 949–956. IEEE, 2018.
- [271] Shitao Chen, Yu Chen, Songyi Zhang, and Nanning Zheng. A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. *IEEE Transactions on Intelligent Vehicles*, 4(3):425–436, 2019.
- [272] Craig Brogle, Chao Zhang, Kai Li Lim, and Thomas Bräunl. Hardware-in-the-loop autonomous driving simulation without real-time constraints. *IEEE Transactions on Intelligent Vehicles*, 4(3):375–384, 2019.
- [273] Ying Gao, Zhigang Xu, Xiangmo Zhao, Guiping Wang, and Quan Yuan. Hardware-in-the-loop simulation platform for autonomous vehicle aeb prototyping and validation. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [274] Microsoft. Openstreetmap. <http://www.openstreetmap.org/>.
- [275] Yu Chen, Shitao Chen, Tong Xiao, Songyi Zhang, Qian Hou, and Nanning Zheng. Mixed test environment-based vehicle-in-the-loop validation—a new testing approach for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1283–1289. IEEE, 2020.
- [276] Tamás Tettamanti, Mátyás Szalai, Sándor Vass, and Viktor Tihanyi. Vehicle-in-the-loop test environment for autonomous driving with microscopic traffic simulation. In *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–6. IEEE, 2018.
- [277] Selim Solmaz, Martin Rudigier, and Marlies Mischinger. A vehicle-in-the-loop methodology for evaluating automated driving functions in virtual traffic. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1465–1471. IEEE, 2020.
- [278] Hexuan Li, Demin Nalic, Vamsi Makkapati, Arno Eichberger, Xuan Fang, and Tamás Tettamanti. A real-time co-simulation framework for virtual test and validation on a high dynamic vehicle test bed. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1132–1137. IEEE, 2021.
- [279] Andrea Stocco, Brian Pulfer, and Paolo Tonella. Mind the gap! a study on the transferability of virtual vs physical-world testing of autonomous driving systems. *IEEE Transactions on Software Engineering*, 2022.
- [280] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. Sumo (simulation of urban mobility)—an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187, 2002.
- [281] PTV. Vissim. <https://www.ptvgroup.com/en/solutions/products/ptv-vissim/>.
- [282] Donkey Car. <https://www.donkeycar.com/>, 2021.
- [283] Zsolt Szalay, Mátyás Szalai, Bálint Tóth, Tamás Tettamanti, and Viktor Tihanyi. Proof of concept for scenario-in-the-loop (scil) testing for autonomous vehicle technology. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVEx)*, pages 1–5. IEEE, 2019.
- [284] Unity. Unity. <https://unity.com/>.
- [285] Mátyás Szalai, Balázs Varga, Tamás Tettamanti, and Viktor Tihanyi. Mixed reality test environment for autonomous cars using unity 3d and sumo. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMII)*, pages 73–78. IEEE, 2020.
- [286] Márton Tamás Horváth, Qiong Lu, Tamás Tettamanti, Árpád Török, and Zsolt Szalay. Vehicle-in-the-loop (vil) and scenario-in-the-loop (scil) automotive simulation concepts from the perspectives of traffic simulation and traffic control. *Transport and Telecommunication*, 20(2):153–161, 2019.
- [287] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.
- [288] Demin Nalic, Arno Eichberger, Georg Hanzl, Martin Fellendorf, and Branko Rogic. Development of a co-simulation framework for systematic generation of scenarios for testing and validation of automated driving systems. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1895–1901. IEEE, 2019.
- [289] Demin Nalic, Hexuan Li, Arno Eichberger, Christoph Wellershaus, Aleksa Pandurevic, and Branko Rogic. Stress testing method for scenario-based testing of automated driving systems. *IEEE Access*, 8:224974–224984, 2020.
- [290] Moritz Klischat, Edmond Irani Liu, Fabian Holtke, and Matthias Althoff. Scenario factory: Creating safety-critical traffic scenarios for automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2020.
- [291] Mingyun Wen, Jisun Park, and Kyungeun Cho. A scenario generation pipeline for autonomous vehicle simulators. *Human-centric Computing and Information Sciences*, 10(1):1–15, 2020.
- [292] Sai Krishna Bashetty, Heni Ben Amor, and Georgios Fainekos. Deepcrashtest: Turning dashcam videos into virtual crash tests for automated driving systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*,

- pages 11353–11360. IEEE, 2020.
- [293] Alessio Gambi, Tri Huynh, and Gordon Fraser. Generating effective test cases for self-driving cars from police reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 257–267, 2019.
- [294] Tri Huynh, Alessio Gambi, and Gordon Fraser. Ac3r: automatically reconstructing car crashes from police reports. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 31–34. IEEE, 2019.
- [295] Mohamed El Mostadi, H el ene Waeselyncq, and Jean-Marc Gabriel. Virtual test scenarios for adas: Distance to real scenarios matters! In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 836–841. IEEE, 2022.
- [296] Zhang Xinxin, Li Fei, and Wu Xiangbin. Csg: Critical scenario generation from real traffic accidents. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1330–1336. IEEE, 2020.
- [297] Hongli Zhou, Xiaolei Chen, Guanwen Zhang, and Wei Zhou. Deep reinforcement learning for autonomous driving by transferring visual features. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4436–4441. IEEE, 2021.
- [298] Bharathan Balaji, Sunil Mallya, Sahika Genc, Saurabh Gupta, Leo Dirac, Vineet Khare, Gourav Roy, Tao Sun, Yunzhe Tao, Brian Townsend, et al. Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562*, 2019.
- [299] Matthew O’Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. *Proceedings of Machine Learning Research*, 123, 2020.
- [300] Alexander Lehner, Stefano Gasperini, Alvaro Marcos-Ramiro, Michael Schmidt, Mohammad-Ali Nikouei Mahani, Nassir Navab, Benjamin Busam, and Federico Tombari. 3d-vfield: Adversarial augmentation of point clouds for domain generalization in 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17295–17304, 2022.
- [301] Michał Antkiewicz, Maximilian Kahn, Michael Ala, Krzysztof Czarnecki, Paul Wells, Atul Acharya, and Sven Beiker. Modes of automated driving system scenario testing: Experience report and recommendations. *SAE International Journal of Advances and Current Practices in Mobility*, 2(2020-01-1204):2248–2266, 2020.
- [302] Markus Borg, Raja Ben Abdesslem, Shiva Nejati, Franois-Xavier Jegeden, and Donghwan Shin. Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 383–393. IEEE, 2021.
- [303] Siemens. Prescan. <https://m.tass.plm.automation.siemens.com/cn/prescan-2>.
- [304] Assia Belbachir, Jean-Christophe Smal, Jean-Marc Blosseville, and Dominique Gruyer. Simulation-driven validation of advanced driving-assistance systems. *Procedia-Social and Behavioral Sciences*, 48:1205–1214, 2012.
- [305] Daniel J Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.
- [306] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel C Briand. Comparing offline and online testing of deep neural networks: An autonomous car case study. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 85–95. IEEE, 2020.
- [307] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand. Can offline testing of deep neural networks replace their online testing? *Empirical Software Engineering*, 26(5):1–30, 2021.
- [308] Fabio Reway, Abdul Hoffmann, Diogo Wachtel, Werner Huber, Alois Knoll, and Eduardo Ribeiro. Test method for measuring the simulation-to-reality gap of camera-based object detection algorithms for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1249–1256. IEEE, 2020.
- [309] Men Long, Chwan-Hwa Wu, and John Y Hung. Denial of service attacks on network-based control systems: impact and mitigation. *IEEE Transactions on Industrial Informatics*, 1(2):85–96, 2005.
- [310] Meital Ben Sinai, Nimrod Partush, Shir Yadid, and Eran Yahav. Exploiting social navigation. *arXiv preprint arXiv:1410.0151*, 2014.
- [311] Verylukygyu. Openpilot issue. <https://github.com/commaai/openpilot/issues/21557>, 2021.
- [312] Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.
- [313] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [314] Belgium dataset. <https://hci.iwr.uni-heidelberg.de/content/bosch-small-traffic-lightsdataset>, .
- [315] Mark Philip Philipsen, Morten Born  Jensen, Andreas M ogelmoose, Thomas B Moeslund, and Mohan M Trivedi. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In *intelligent transportation systems (ITSC), 2015 IEEE 18th international conference on*, pages 2341–2345. IEEE, 2015.

- [316] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [317] Udacity. Udacity. <https://fcav.engin.umich.edu/research/failing-to-learn>.
- [318] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.
- [319] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.
- [320] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.
- [321] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3221, 2017.
- [322] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.
- [323] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.
- [324] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2(5):6, 2018.
- [325] Harald Schafer, Eder Santana, Andrew Haden, and Riccardo Biasini. A commute in data: The comma2k19 dataset. *arXiv preprint arXiv:1812.05752*, 2018.
- [326] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.
- [327] Baidu. Apolloscape. <https://github.com/ApolloScapeAuto/dataset-api>.
- [328] Alex Zyner, Stewart Worrall, and Eduardo M Nebot. Acfr five roundabouts dataset: Naturalistic driving at unsignalized intersections. *IEEE Intelligent Transportation Systems Magazine*, 11(4):8–18, 2019.
- [329] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.
- [330] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.
- [331] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934. IEEE, 2020.
- [332] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James McBride. Ford multi-av seasonal dataset. *The International Journal of Robotics Research*, 39(12):1367–1376, 2020.
- [333] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. The round dataset: A drone dataset of road user trajectories at roundabouts in germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020. doi: 10.1109/ITSC45102.2020.9294728.
- [334] Antonia Breuer, Jan-Aike Termöhlen, Silviu Homoceanu, and Tim Fingscheidt. opendd: A large-scale roundabout drone dataset. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [335] Bosch Night dataset. <https://hci.iwr.uni-heidelberg.de/content/bosch-small-traffic-lightsdataset>, .
- [336] Crashd dataset. <https://crashd-cars.github.io/>, 2022.
- [337] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.
- [338] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.
- [339] Manikandasriram. Gta. <https://fcav.engin.umich.edu/research/failing-to-learn>.
- [340] MathWorks. Matlab simulink. [https://ww2.mathworks.cn/products/simulink.html?s\\_tid=hp\\_products\\_simulink](https://ww2.mathworks.cn/products/simulink.html?s_tid=hp_products_simulink).
- [341] Mechanical Simulation. Carsim. <https://www.carsim.com/products/carsim/index.php>.
- [342] Cyberbotics Ltd. Webots. <http://www.cyberbotics.com/#cyberbotics>.
- [343] Vires. Vtd. <https://vires.mscsoftware.com/vtd-vires-virtual-test-drive/>.
- [344] OSRF. Gazebo. <http://gazebosim.org/>.

- [345] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [346] Ziyuan Zhong, Zhisheng Hu, Shengjian Guo, Xinyang Zhang, Zhenyu Zhong, and Baishakhi Ray. Detecting safety problems of multi-sensor fusion in autonomous driving. *arXiv preprint arXiv:2109.06404*, 2021.
- [347] Shital Shah, Debadepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.
- [348] rFpro. rFpro. <https://www.rfpro.com/>.
- [349] Cognata. Cognata. <https://www.cognata.com/cn/>.
- [350] NVIDIA. Nvidia drive constellation. <https://developer.nvidia.com/zh-cn/drive/drive-constellation>.
- [351] Guodong Rong, Byung Hyun Shin, Hadi Tabatabae, Qiang Lu, Steve Lemke, Märtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.
- [352] AVSIMULATION. Scanner studio. <https://www.avsimulation.com/scaner-studio/>.
- [353] MSC software. Adams. <https://www.mscsoftware.com/product/adams>.
- [354] Nicolas Hiblel, Dominic Gruyer, Jean-Sébastien Barreiro, and Bertrand Monnier. Pro-sivic and roads, a software suite for sensors simulation and virtual prototyping of adas. In *Proceedings of DSC*, pages 277–288, 2010.
- [355] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPs)*, pages 287–296. IEEE, 2018.
- [356] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.
- [357] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiang Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.
- [358] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [359] Udacity self-driving challenge. <https://github.com/udacity/self-driving-car/tree/master/challenges>.
- [360] Rodrigo Queiroz, Thorsten Berger, and Krzysztof Czarnecki. Geoscenario: An open dsl for autonomous driving scenario representation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 287–294. IEEE, 2019.
- [361] Daniel J Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 63–78, 2019.
- [362] Florian Bock, Christoph Sippl, Aaron Heinz, Christoph Lauer, and Reinhard German. Advantageous usage of textual domain-specific languages for scenario-driven development of automated driving functions. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2019.
- [363] Barbara Schütt, Thilo Braun, Stefan Otten, and Eric Sax. Sceml: A graphical modeling framework for scenario-based testing of autonomous vehicles. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 114–120, 2020.
- [364] Matthias Althoff, Markus Koschi, and Stefanie Manzingler. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.
- [365] Ao Li, Shitao Chen, Liting Sun, Nanning Zheng, Masayoshi Tomizuka, and Wei Zhan. Scogene: Bio-inspired traffic scenario generation for autonomous driving testing. *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [366] Rupak Majumdar, Aman Mathur, Marcus Pirron, Laura Stegner, and Damien Zufferey. Paracosm: A test framework for autonomous driving simulations. *Fundamental Approaches to Software Engineering*, 12649:172, 2021.
- [367] Jing Ma, Xiaobo Che, Yanqiang Li, and Edmund M-K Lai. Traffic scenarios for automated vehicle testing: A review of description languages and systems. *Machines*, 9(12):342, 2021.
- [368] Ezio Bartocci and Yliès Falcone, editors. *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*. Springer, 2018. ISBN 978-3-319-75631-8. doi: 10.1007/978-3-319-75632-5. URL <https://doi.org/10.1007/978-3-319-75632-5>.
- [369] Zhenya Zhang, Paolo Arcaini, and Xuan Xie. Online reset for signal temporal logic monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4421–4432, 2022.