Scenario-Based Virtual Testing for Autonomous Vehicles: Methods and Considerations

Huijia Sun
ShanghaiTech University
Shanghai, China
sunhj2022@shanghaitech.edu.cn

Hongchen Cao
ShanghaiTech University
Shanghai, China
caohch1@shanghaitech.edu.cn

***Abstract*—In the development and deployment of Autonomous Vehicles (AVs), testing AV's driving intelligence is definitely a critical step, which is not only restricted to safety but also includes security. For reason that real-world testing is inefficient and dangerous, research has focused on scenario-based virtual testing techniques to obtain safety-critical test cases automatically. This paper provides a survey of existing works on scenario-based AV virtual testing. We collect a list of publications and discuss them based on the testing workflow and considerations for scenario-based AV virtual testing. We also distill the research focus, trends, and prevailing open-source autonomous driving systems and tools.**

***Index Terms*—Autonomous Vehicle, autonomous driving system, virtual testing, scenario-based testing**

I. Introduction

Recently, fueled by the increasingly rapid progress in Autonomous Vehicles (AVs), a driverless future seems just around the corner. Whereas a set of fatalities of AVs with low-level autonomy have increased awareness of AV safety and security, which has raised general concerns about them [1]–[4].

AV safety refers to the condition of a "steady state" of a self-driving car doing what it is supposed to do. While AV security refers to necessary measures taken to an AV to make it free from external or internal intentional or malicious harm to its system resources, and protection afforded to prevent such detriment from happening to ensure the normal functioning of the AV. Besides, as a kind of Cyber-Physical System (CPS), AV integrates the dynamics of the physical processes with the software and networking, interacting with humans almost all the time. Thus, AV is highly security-critical as any slight mistakes made inside it can directly lead to undesirable or fatal outcomes, such as permanent stops, or collisions. For example, a wrong trajectory prediction to a background vehicle in front can lead to an erroneous driving decision, such as overtaking instead of giving way, then a car crash will happen within the foreseeable future.

The Society of Automotive Engineers (SAE) defines six levels of vehicle autonomy [5]. AVs are rapidly developed from level 0, i.e., no autonomy, to level 4, i.e., fully controlling but sometimes driver controlling the vehicle. As the autonomy level goes up, AVs face more and more uncertainties/disturbances in the real world, which is closely related to AV safety and security. Fortunately but deservedly, the growing importance of AV safety and security has prompted the continued expansion of the related research community. A series of conferences are now focused on this topic, including FSE [6], ASE [7], and ISSRE [8]. Through these avenues, researchers and AV manufacturers together explore the fundamental question: How to guarantee AV safety and security? This has greatly stimulated the growth of related publications.

The topic of AV safety and security is relevant to the field of Cyber Security, but also applicable more generally in the field of Software Engineering (SE). From the Software Engineering (SE) perspective, AV's development and deployment require extensive testing of AVs' driving intelligence, to ensure AV's functionality in an unforeseeable number of situations in the real world. There have already been many AV testing techniques in academia and industry, while some issues still need to be handled.

For AV testing, in the area of industry, a common practice is naturalistic on-road testing, to study the interactions between AVs and human-controlled vehicles. Every year companies licensed test their self-driving cars on public roads in California and reel off more and more miles [9]–[11]. The California Department of Motor Vehicles (DMV) collects these associated data and then publishes Disengagement Report annually [12]. However, this approach suffers from several limitations. Besides its expensiveness, one major problem is inefficiency because of the rarity of dangerous events. As the accident rate is very low for human drivers (e.g., 1.09 fatalities per 100 miles), it requires about 8.8 billion miles to encounter so many accidents, which would take hundreds of years under even aggressive testing assumptions [13]. Thus, virtual testing with more critical scenarios becomes an alternative practice.

In this paper, we focus on the fundamental branch of AV testing research: scenario-based AV virtual testing. Just like traditional software testing, it is closely associated with other activities in the software engineering process. Using various scenarios, it aims at revealing safety and security issues in the Autonomous Driving System (ADS), one of the core modules of an AV, testing whether the ADS satisfies safety requirements, i.e., test oracles, both before and after AV deployment, and then guiding software repair to fix these issues.

Being a kind of software testing activity, some ideas and techniques of scenario-based AV testing are shared with well-known solutions already widely adopted in traditional software

testing. However, compared to traditional software, ADS utilizes both logic-based and non-deterministic Machine Learning based controllers to process multiple sensor data, plan driving trajectories and activate the vehicle, which integrates the dynamics of the physical processes with the software. This difference brings several unique challenges, such as the test oracle problem [14], [15]. In addition, considering the complexity of an ADS, the computing load of testing is also a crucial issue to be settled.

This paper consolidates and distills the progress in the field of scenario-based AV virtual testing. It starts by providing background on scenario-based AV virtual testing as the first part. In the second part, we collect a lot of papers from different SE conferences and discuss these papers from the perspective of testing workflow, from which we summarize a set of considerations in scenario-based AV virtual testing. Then, we outline research trends and suggest research opportunities in AV testing in the third part. Furthermore, we also introduce two open-source ADSs and related simulators in the final part. Figure 1 shows the structure of this paper. Section 3 described the detailed organization of the survey.

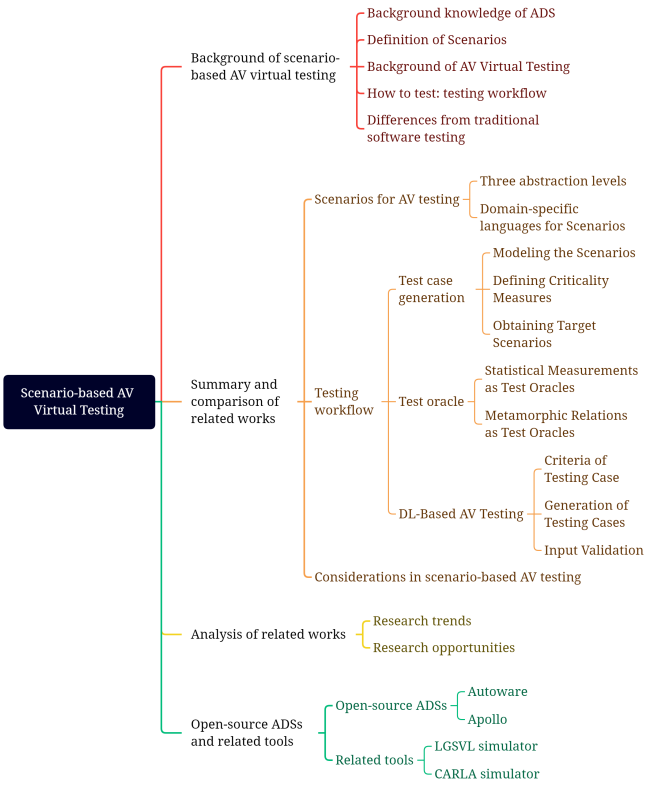

Fig. 1. Structure of this paper

In our survey project, we make the following contributions:

- We provide a fundamental survey of 35 scenario-based AV virtual testing papers published in recent years.
- We discuss the existing works of AV testing from two perspectives, detailing these works from a more comprehensive viewpoint.
- We analyze the research trends and suggest research opportunities in AV testing.
- We introduce two of the most prevailing open-source ADSs and related simulators, providing navigation for researchers and partitioners interested in this field.

II. Background of Scenario-based AV Virtual Testing

In this section, we first provide background knowledge of ADS and the definitions of AV safety and Security and scenario-based AV virtual testing. After introducing the basic concepts, we discuss the testing workflow and the testing objects of AV testing. Besides, we describe the differences between AV testing and traditional software testing to help clarify its unique characteristics, and function of AV testing in the software engineering process.

A. Autonomous Driving System

Instead of human drivers, AV uses ADS technology to control its steering and acceleration, and perceive its surrounding environment (e.g., lanes, pedestrians, and other vehicles) [16], [17]. Consequently, ADS is a decisive factor in determining the AV's intelligent driving functionality. In general AV software system architecture, shown in Figure 2, the decision-making step almost takes charge of the whole system, which contains modules of perception, prediction, and planning. The detailed functionality of each module in an ADS is as follows:

- **Sensors**: An ADS supports a wide range of sensors, such as cameras, inertial measurement units (IMU), Global Positioning Systems (GPS), sonar, RADAR, and LiDAR. Raw data will be preprocessed, filtering sensor noise [18].
- **Perception Module**: By using computer vision and deep-learning techniques, the perception module processes multiple sensor inputs to detect obstacles (e.g., vehicles, pedestrians, traffic signs, or lanes) in the surrounding environment. To improve the accuracy of detection, perception module often adopts multiple object detection pipelines and then merged them into a final track list of objects, using fusion algorithms such as Kalman filters [19]. For example, Baidu Apollo incorporates the capability of a camera-based and a LiDAR-based object-detection pipeline to recognize obstacles and fuse their individual tracks to obtain a final track list [20].
- **Prediction Module**: After recognizing obstacles in the surrounding environment, the prediction module estimates their future trajectories, including their positions, headings, velocities, accelerations, etc. The prediction module also provides the probabilities of these predicted trajectories using neural networks (e.g., MLPs and RNNs) [19].
- **Planning Module**: As a critical module in high-level (e.g., Level 4 [5]) ADSs, the planning module calculates driving trajectories, considering driving safety, efficiency and comfortability, by solving a Linear Programming (LP) or Quadratic Programming (QP) problem using

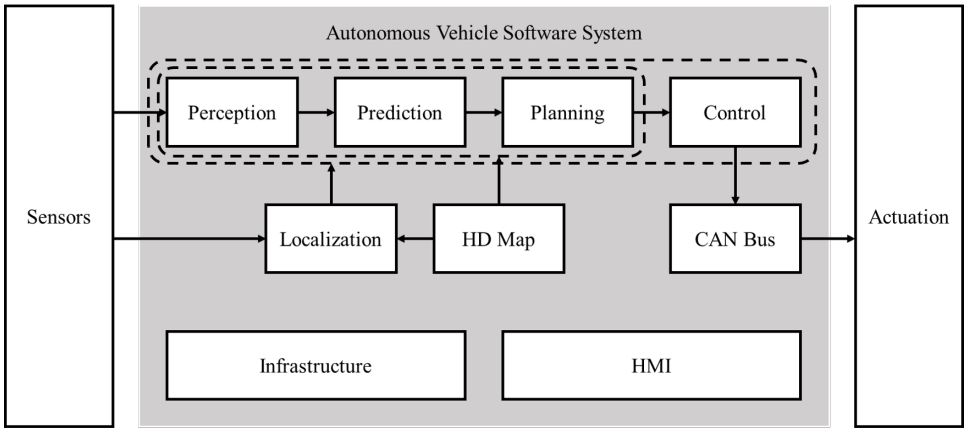

Fig. 2. General Autonomous Vehicle software architecture

output data from the localization and prediction modules [19]. More concretely, such a module adopts a 3-layer design, including route planning, behavioral planning, and local planning [21]. Routing planning selects a route based on the given origin and destination. Then behavioral planning makes high-level driving decisions, such as cruising, stopping, and lane changing, according to the real-time driving environment to follow the selected route. Finally, local planning processes the high-level decision as concrete low-level driving trajectories (e.g., waypoints), which will then be passed to the control module to set in motion.

- **Localization Module**: Similarly, using multi-sensor fusion algorithms (e.g., Error State Kalmen filter) to fuse multiple sets of input data from GPS, IMU, and LiDAR [20]. The localization module provides the real-time localization of the AV [19].
- **Control Module**: The control module enforces the planned trajectory using control algorithms such as MPC and PID to calculate the required steering and throttling [19]. The planned trajectories are translated into control commands to pass to the CAN bus.
- **High-Definition Map (HD Map) Module** provides information, such as lane boundaries, traffic sign localizations, stationary objects, and routing, during runtime [19]. This information is queried by multiple modules, such as the localization module, planning module, and control module.
- **Controller Area Network (CAN) Bus** delivers information between ADS and the AV's mechanical system [22]. Using CAN Bus, the vehicle can send control commands and receive chassis information [19].
- **Infrastructure** includes a robotics middleware (e.g., ROS [23], CyberRT [20]) to supports the communication among components [19].
- **Human Machine Interface (HMI)** visualizes system status and interfaces with developers. Though not required for the ADS, using HMI will improve the usability of an ADS, thus Apollo and Autoware [24] both have it.

B. What Are Scenarios

Generally, scenarios are configurations of objects on a map (e.g., obstacles, vehicles, and pedestrians) as well as their dynamic behavior. In black box testing, scenarios are used to evaluate the ADS systematically. To accomplish the target, scenarios need to focus on various aspects like human understandable notation or a description via state variables. To handle the problem of scenario representation, [39] suggests three abstraction levels for scenarios. The three levels of abstraction for scenarios are: *functional scenarios* (Section 4.1), *logical scenarios* (Section 4.2), and *concrete scenarios* (Section 4.3), as shown in Figure 3.

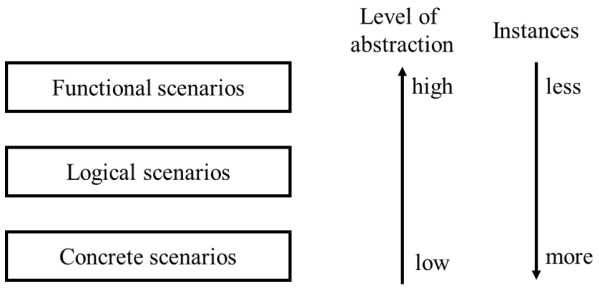

Fig. 3. Three levels of abstraction for scenarios

C. AV Virtual Testing

Testing is an effective procedure to ensure the safety of AV. Because of the inefficiency and expensiveness of on-road testing, virtual testing becomes the main direction taken by industry and academia.

ISO 26262 standard [37] is an international standard for the functional safety of electrical and electronic systems installed in serial production road vehicles. According to ISO 26262, scenario-based approaches are useful in deriving requirements and testing components and procedures in the system. In the field of AVs, some publications suggest that scenario-based methods are also functional in the development, testing, and validation of AVs [38], [39]. During the testing, the

implemented system is verified whether its requirements are all fulfilled. Thus, the tests should be systematically planned, specified, and executed according to the requirements. Additionally, testing results should be thoroughly evaluated according to the test oracle. Well-designed scenarios fulfill these conditions.

Extracting and classifying relevant scenarios from data sets of recorded traffic data is a straightforward approach for getting critical test cases to increase the efficiency of virtual testing [31], [32]. Another method is to group scenarios by random sampling methods or rely on training machine-learning algorithms, such as unsupervised learning and reinforcement learning [33]–[35]. However, one of the main disadvantages of the above two approaches is that it is challenging to collect data at this scale. Another is that obtained scenarios are restricted to observed ones, which can be not critical [36]. These downsides motivate a reasonable practice, generating critical scenarios for virtual testing automatically.

D. How to Test: Testing Workflow

The scenario-based virtual testing workflow refers to how to conduct scenario-based virtual testing with different testing activities. We introduce the testing workflow and its main procedures.

As presented in Figure 4, the workflow begins with scenario elicitation, in which scenarios are described in the form of logical scenarios. Through the requirement elicitation, the safety requirements are determined and specified for the ADS. Test cases are generated automatically. Meanwhile, test oracles are defined based on safety requirements. Then, test cases are executed on the ADS under test to examine whether the test oracles are violated, which is considered in the evaluation of the performance of the ADS. Additionally, the test execution results with a safety violation are used in locating and fixing the ADS. The entire process can be repeated to evaluate whether the repaired ADS works well concerning safety.

Section 5 described the detailed testing workflow of scenario-based virtual testing.

E. Differences from Traditional Software Testing

As a kind of CPS, AV integrates the dynamics of the physical processes with the software and networking, interacting with humans almost all the time. Compared to traditional software testing, AV testing has some unique characteristics. In this section, we compare traditional software testing and AV testing from different perspectives.

- **Test input.** Test inputs in scenario-based AV virtual testing are scenarios with objects on a map and their dynamic behavior. At first, these scenarios are perceived by the sensors, transformed into time-series sensor data, and then processed by decision-making modules, the central part of the ADS. While in traditional software testing, test inputs are often a set of discrete program inputs. These test inputs are received and used by the software directly.
- **Test oracle.** Testing is a valid method for software quality assurance. Oracles are criteria that testers use to examine whether the execution results of test cases are expected [40]. Traditional software testing often assumes an oracle exists. However, considering the complexity of the ADS, this assumption does not hold, because an oracle is unavailable and hard to be applied [41]. The intuition, ”the closer to the human drivers' behavior, the better”, is quite difficult or even impossible to be applied. Even for highly qualified human testers, it is still hard to determine the validity of every single behavior of an AV.
- **Test adequacy.** It is a common view that deploying inadequately tested software is inadvisable [42]. In traditional software testing, the solution is to measure the adequacy of the software being used. Based on this, various criteria have been proposed and adopted, such as line coverage, branch coverage, and dataflow coverage [43]. However, research on the test adequacy of ADSs is lacking.

III. Survey Methodology

We describe the survey scope, the paper collection approach, and the organization of our survey.

A. Survey Scope

We aim to define, collect, and analyze various papers to argue that there does exist a field of research that can be termed ”scenario-based AV virtual testing”.

We apply the following criteria to collect papers:

- The paper introduces the general idea of scenario-based AV virtual testing or one of the related aspects of scenario-based AV virtual testing.
- The paper proposes a method, framework, or study that targets scenario-based AV virtual testing.

Our survey focuses on the fundamental branch of AV testing research: scenario-based AV virtual testing. Therefore, we do not include papers about functionality-based AV testing or scenario-based on-road AV testing. We also do not include papers about safety definitions.

B. Paper Collection

The research focuses on a systematic keyword search in the topic section of literature databases, including ACM Digital Library [74], IEEE Xplore Digital Library [75], and Springer Lecture Notes in Computer Science [76].

The search conducted included the specific terms, “Autonomous driving testing”, “Self-driving car testing”, “Autonomous vehicles testing”, “Driverless car testing”, ”crash avoidance”, ”advanced driver assistance system”, ”automated vehicles testing”, and ”intelligence testing” either in the title, keywords or abstract and included academic journals and conferences listed in the mentioned databases.

We do not aim to investigate all papers related to autonomous driving safety testing, it is too large and we focus on scenario-based virtual testing. Therefore, we further filter papers related to this topic. Constrained by time and effort, we will choose 35 high-quality papers that have a strong

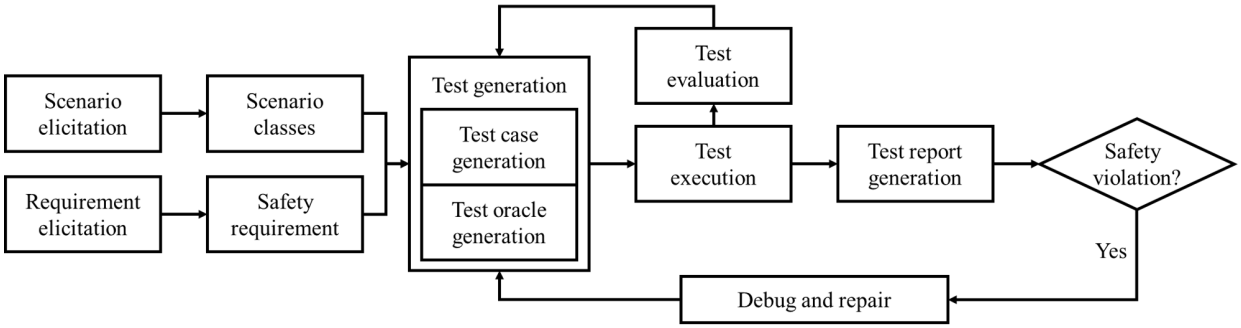

Fig. 4. Workflow of scenario-based AV virtual testing

inter-relationship with each other for the final study, thus summarising the progress and basic considerations in the field of scenario-based AV virtual testing.

C. Paper Organization

We review the collected papers in the following way:

Section 4 (Scenarios for AV Testing): Scenarios are essential components in scenario-based AV virtual testing. We first present three abstraction levels of scenarios to explain what a scenario is. We further introduce two Domain-Specific Languages from the abstraction level angle to explain the generation of scenarios.

Section 5 (Testing Workflow): We review the collected papers from the testing workflow angle and summarize these works according to test input generation and test oracle identification. For each part, we further categorize related work into several procedures. For example, we divided test case generation into four categories: modeling the scenarios, defining criticality measures, obtaining target scenarios, and improving testing coverage.

Section 6 (Considerations in Scenario-Based AV Virtual Testing): After discussing the methods in each procedure of scenario-based AV virtual testing workflow, we review these methods and distill considerations for scenario-based AV virtual testing.

Section 7 (Research Trends): We analyze the research trends of collected papers and compare the numbers of publications in different research venues and years.

Section 8 (Research Opportunities): We discuss open problems and distill research opportunities in scenario-based AV virtual testing.

IV. Scenarios for AV Testing

We describe three levels of abstraction for scenarios: functional scenarios, logical scenarios, and concrete scenarios. Additionally, we introduce two domain-specific languages for scenarios.

A. Functional Scenarios

Functional scenarios illustrate the scenarios from the most abstract level. On a semantic level, functional scenarios include the description of entities in the scenario and the relations/interactions of those entities. Mainly used by human experts, they are represented via a linguistic notation with well-defined vocabulary. Similarly, words in this vocabulary can be divided into three categories: environmental conditions (weather conditions, road geometry, and topology), entities (vehicles, obstacles, and pedestrians), and relations of those entities (overtaking and following). Sources for words in the vocabulary can be traffic laws or other actual standards [44].

As shown in Figure 5 (a), the road, the environmental condition, is depicted with its layout and geometry. The two entities, the car and the truck, and their relations are also described in the functional scenario.

B. Logical Scenarios

Based on functional scenarios, Logical scenarios provide a more detailed representation. Using state space variables, logical scenarios depict the parameter ranges of the state space variables. Probability distributions specify the parameter ranges, e.g., Gaussian distribution, Uniform distribution. Besides, the relations of the parameter ranges can be specified by correlations or numeric conditions. For example, when overtaking a vehicle, the speed of the overtaking vehicle is always greater than the speed of the overtaken one.

As shown in Figure 5 (b), the functional scenario depicted in (a) is converted to a logical scenario by transforming the linguistic representation into state space and specification of the parameters. The curvilinear shape of the road is presented by its radius, and the positions of the vehicles are presented by their longitudinal positions along the lane.

C. Concrete Scenarios

Based on logical scenarios, concrete scenarios determine distinct scenarios by selecting a concrete value from the parameter range. Any number of concrete scenarios can be derived because the value range of parameters is often continuous. Concrete scenarios can be applied as templates for test case generation. Figure 5 (c) shows how to derive a concrete scenario from the logical scenario.

It should be noted that the use of concrete scenarios in test case generation also needs the help of Domain-Specific Languages (DSLs), which describe the concrete scenarios to different degrees.

Natural Language Description:
On a two-lane motorway in a curve, a car and a truck are driving on the right lane of the road, whereby the car follows the truck.

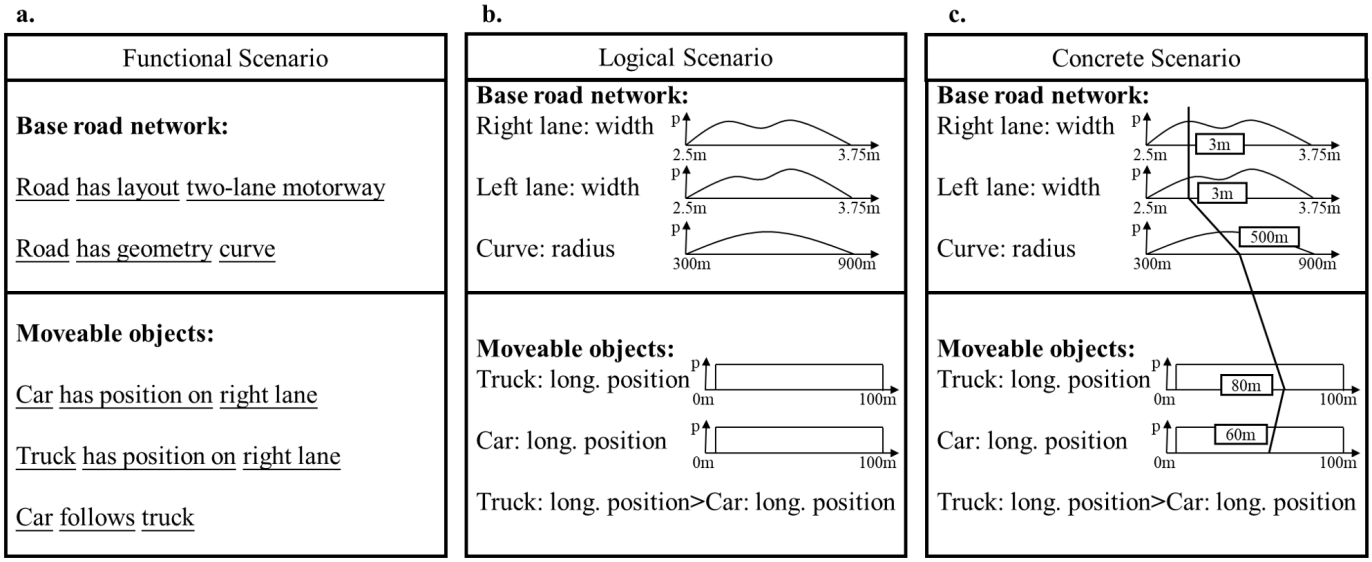

Fig. 5. Example of scenarios in 3 abstraction levels

D. DSLs for Scenarios

Being independent of AV simulators, SCENIC [45] and AVUnit [46] are two DSLs for CPSs and Robotics to write environment models, and evaluate the correctness of traces according to the specification the user-defined. This is an essential prerequisite to any formal analysis.

The scenario generation based on SCENIC presents the above three abstraction levels. For functional scenarios, SCENIC provides Classes, Objects, and Geometry to depict entities [45]. Object orientation in SCENIC provides a natural organizational principle for scenarios involving different types of entities and also improves compositionality, which leads to a language style close to natural languages. SCENIC also provides a list of specifiers to depict the properties of entities [45], such as their headings and positions. For logical scenarios, SCENIC provides several elementary distributions to model real-world stochasticity. Subsequently, the positional relationship between entities can be denoted as a certain distribution in the space. For concrete scenarios, scalars in SCENIC can be sampled from their distributions [45], thus properties, such as positions of entities, can be determined. A concrete scenario is generated finally. Additionally, SCENIC enforces several default requirements, which ensures the validity of the synthetic scenario, i.e., all objects do not intersect each other.

The basic functions of AVUnit and SCENIC are similar. Compared with SCENIC, AVUnit also support adding specification to a scenario [46], based on the Single Temporal Logic (STL) technique. Thus, the output test cases will not violate these specifications. For example, users can add specifications such as non-collision or traffic regulations to the synthetic scenarios.

V. Testing Workflow

In this section, we distill existing works and discuss the key activities involved in scenario-based AV virtual testing, i.e., test case generation and testing oracle identification.

A. Test Case Generation

We will first introduce the common procedures in test case generation, i.e., scenario modeling, criticality measures definition, and critical subspace exploring. Then, we discuss some differences between existing methods.

1) Modeling the Scenarios: Modeling Vehicle Behaviors is often the basis of defining criticality measures. The following scenario-based AV virtual testing works [47], [48], [51] model the scenarios from a global and objective perspective. [49], [52], [53] model the scenarios from Ego Vehicles' (EVs') perspective, according to the state of the AV. [54]–[57] consider the vehicle behaviors following the probabilistic statistics in Natural Driving Environment. [50], [58]–[60] model the scenarios with Signal Temporal Logic (STL) techniques.

Modeling from a global perspective. In the following paragraphs, we discuss [47], [48], [51], which model the scenarios from a global and objective perspective.

[47] is the previous work of [48]. The key step of the two works is modeling the motion planning of EVs. Thus, the properties of every traffic participant in every state need to be considered, such as trajectories, occupied space, and lanes of the road network, to provide a basis for describing the motion planning of the EV. Subsequently, the behaviors of each vehicle can be presented by the occupied area of the vehicle on the road. Finally, the two works denote a scenario by a tuple, including the initial state, the occupancy set of each

traffic participant at any time, and the allowed space on the road surface at any time. Besides, necessary constraints are also required in the model for optimizing and getting a more realistic model, for example, speed limits or other traffic rules and vehicle dynamics.

Compared to [47], [48] introduces the curvilinear coordinate system to present the trajectory of each traffic participant, which is closer to the actual driving state of vehicles. Additionally, [48] also introduces a constraint, the intersection of the occupied areas of any two vehicles is empty at any time, to ensure that no traffic participants collide with each other.

The modeling method adopted by [51] is much plainer than those in [47] and [48]. [51] only consider states (including the position of a vehicle's geometry center, velocity, and acceleration) of all vehicles. Subsequently, [51] focuses on the speed of each vehicle at each time step, especially the EV, instead of emphasis on the geometry feature of the occupied/empty area on the road in [47] and [48].

Modeling from EV's perspective. The above works model the scenarios from a global perspective. In the next paragraphs, we will list a set of works modeling scenarios from the EV's perspective.

Considering an AV as a CPS, [49] focuses on the representation of EV's trace vectors. The Operation Design Domain (ODD) is restricted to the pedestrian-involved vehicle following scenarios, as a result, the trace vectors of EV include the initial position and speed of the EV and the leading car, the initial velocity and orientation of the pedestrian, as well as the weather condition of the driving environment.

[53] models the scenarios by describing the maneuver chosen by the planning module of EV. Meanwhile, [53] considers the continuity of EV's maneuver, i.e., the current maneuver of EV is dependent on the current state of EV and the previous maneuver at the last time step, instead of simple discretization. The state of a vehicle consists of its position, velocity, and acceleration at that time step. In this way, after defining the vehicle kinetic model, it is uncomplicated to obtain the distance between EV and other vehicles.

[52] also describes the maneuver of EV. However, different from [53], [52] semantically divides the maneuver of EV into a set of categories and defines them as tasks. Thus, the maneuver of EV in a period of time is depicted as a temporal-spatial sequence of a list of tasks, as shown in Figure 6.

Modeling using statistical models. The works introduced above are all based on expert knowledge, focusing on the interactions between EV and certain NPCs. However, considering the traffic network in the real-world driving environment, the behaviors of vehicles satisfy certain probabilistic distribution models. In the following paragraphs, we discuss a set of works that model scenarios using probabilistic and statistical techniques.

[56] focuses on the cut-in scenarios, a vehicle moving from one lane to another in the same direction of travel, and generates lane change models based on naturalistic driving. The identification of the lane change events is as follows: When EV crosses the lane markers, both EV and the NPC behind it hold a velocity between 2m/s and 40m/s. At the same time, R_L, the distance between the rear edge of EV and the front edge of the NPC is between 0.1m and 75m. The key phrase of a lane change is captured by three variables: EV's velocity v_L, R_L, and Time To Collision (TTC) TTC_L. R_L and TTC_L can be modeled independently given the same v_L. [56] finds a Pareto distribution is a proper distribution to fit R_L^{-1}, and TTC_L^{-1} can be approximated by both Pareto distribution and exponential distribution. Then, the velocity of the rear NPC can be calculated by these variables. Subsequently, the model of cut-in scenarios is generated by describing the velocity of each vehicle and the distance between the two of them.

[57] is the following work of [54] and [55]. These three works divide scenarios in Naturalistic Driving Environment (NDE) into different categories according to their ODDs, extending the sampling method in [56]. Then, they analyze the probabilistic distributions of naturalistic driving data to obtain the exposure frequency of each scenario category. Maneuver challenge, i.e., the probability of EV's motion decision making, is also considered in the three works. Therefore, they generate models for scenarios of various categories, by depicting the joint probability of a list of driving events.

Compared to [54] and [55], [57] additionally introduces Naturalistic and Adversarial Driving Environment (NADE). NADE is created by training the NPCs in the NDE to learn when to execute what adversarial maneuver while ensuring unbiasedness and improving efficiency. The method proposed by [57] further paves the way for depicting more complex scenarios.

Modeling using STL technique. Although the works discussed above obtain delicate models from various perspectives, a common issue of them is that it is difficult to measure the situation coverage. In the next paragraphs, we introduce a set of works using the STL technique to model scenarios.

In the process, the dynamic process that operates in the driving environment is abstracted into a finite state transition system, a discrete representation of the dynamic process. This discrete representation and the dynamic process hold some similar properties. Then, the specification, now expressed as STL formula, is translated into an equivalent finite-state automation. Finally, the discrete representation and the automation are merged into a product automation, describing their conjoint behavior. This product automation is used to check whether or not the specification is satisfied.

In general, STL can be used for runtime monitoring of EV, which can be adopted to enhance the test coverage, besides detecting violations of certain specifications. [58] considers the driving performance under various scenarios, e.g., driving comfort. It uses STL to describe the status of EV, e.g., the vehicle should not do excessive braking unnecessarily or too often. [50] and [59] adopted STL to describe EV's motion planning, or interactions between EV and other traffic participants (NPCs, or pedestrians), for example, EV should stop within 3 seconds when a pedestrian is on the crossing, and EV remains stopped until the crossroad is free. [60] also

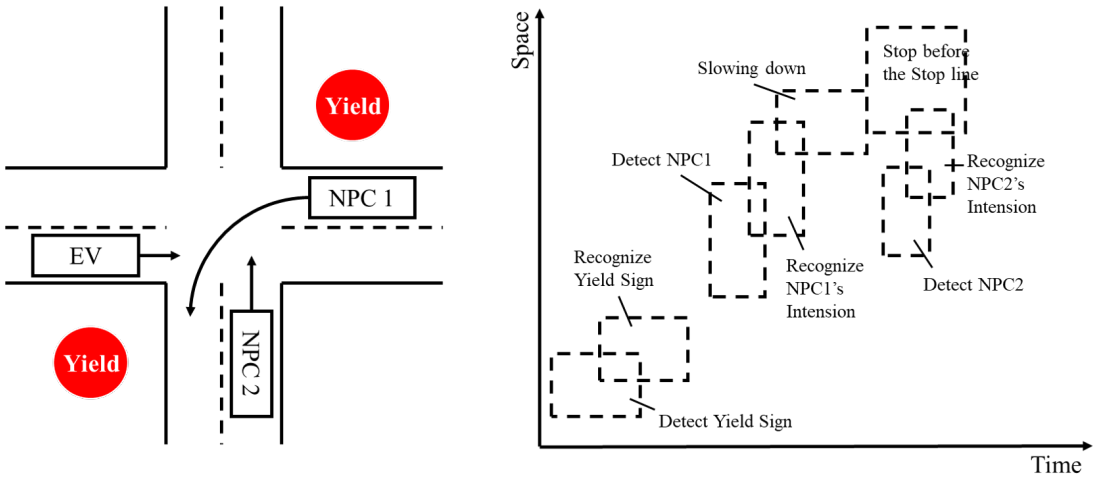

Fig. 6. Temporal-spatial sequence of assigned tasks

uses STL to model the status of EV under various scenarios. However, the difference from before is that [60] model the status from the perspective of driving legality, which is one of the pioneering works in test oracle generation and will be explained in the next section.

To briefly summarize, though approaches of modeling scenarios using statistical models can describe complex scenarios with various and comprehensive categories, these approaches suffer from the complexity of computation. The STL-based method is efficient in the runtime monitoring of EV. However, this method requires knowledge of the formal method. Yet being plain and straightforward, the two approaches left are often practicable in the application. Besides, the modeling idea of the two approaches can also be adopted in other ones.

After modeling the scenarios, a discretization step is usually performed to reduce the test effort to a finite number of concretizations.

2) Defining Criticality Measures: A common target of scenario-based AV virtual testing is to obtain more critical scenarios. To this end, defining proper criticality measures is a crucial step in test case generation. The following scenario-based AV virtual testing works [47], [48] define the drivable area as their criticality measure. [53] adopts distance as their criticality measure. [51], [52], [56] use velocity, acceleration, or jerk. [54], [55], [57] use joint probability to measure criticality. [49], [60] propose a score measuring how 'close' we are to violating certain specifications.

Drivable Area. This is the criticality measure adopted by [47], [48]. The drivable area reflects the properties of traffic participants. It considers not only the optimal solution of the motion planning problem but the space of all solutions as well. Nevertheless, the complexity of the model determines its performance because a complex model may be closer to the fact than a simple one, while it would be difficult to compute.

Distance. Intuitively, distance indicates the safety degree of a scenario, i.e., a larger distance suggests a safer status. After defining a vehicle kinetic model, [53] defines its criticality measure as $d_{safe}-d_{stop}$. d_{safe} presents the maximum distance the vehicle can travel without colliding with any static or dynamic object. d_{stop} describes the distance the vehicle will travel before coming to a complete stop while the maximum comfortable deceleration is applied. Besides, this criticality measure monitors globally in a scenario for every traffic participant concerning the conditions of safety violations.

Velocity, Acceleration, or Jerk. Jerk is the derivative of the acceleration or deceleration rates of a vehicle. Although not as intuitive as distance, velocity, acceleration, and jerk also indirectly reflects the safety degree of a scenario, i.e., a lower value of these variables indicates a lower accident risk of a scenario. [56] simply use the deformation of the velocity as its criticality measure. [51] uses the relative speed between EV and the NPC interacting with EV to compute the degree of danger in a scenario. Beside considering two events, i.e., collision and otherwise, [51] identify the most unsafe situation in a scenario as that having the maximum danger. [52] defines a score, which considers the minimum distance between EV and the NPC interacting with EV, and the maximum jerk of EV during the test. These two components of the score aim to measure the safety and comfortability of driving.

Joint Probability. In the previous section, we introduce a modeling method using probabilistic and statistical models. Being different from other approaches, [54], [55], [57] also adopt a unique way to define the criticality measure. In the modeling phase, these works obtain the exposure frequency, i.e., the probability of a class of events, and maneuver challenge, i.e., the probability of EV's motion decision making. Inspired by the concepts of the risk assessment, [54], [55] define the criticality of scenarios as the multiplication of exposure frequency and maneuver challenge. The criticality measure adopted by [57] is similar to those of the two works above. However, [57] defines maneuver challenge as the accident probability of the given state-action pair, which can describe a more adversarial driving environment.

Degree of Violation. [49] defines hybrid test objects, one

of which is failure distance. Failure distance evaluates how close the system is to violating its safety requirements at each time step. [49] defines five self-driving features: autonomous cruise control, traffic sign recognition of stop signs, traffic sign recognition of speed limit, pedestrian protection, and automated emergency braking. To compute failure distance, [49] further defines five failure distance functions of each self-driving feature. Modeling the driving status from the perspective of driving legality, [60] computes how 'close' EV is to violating the specifications, i.e., traffic rules, by defining a set of quantitative semantics and using the tool RTAMT [67].

To briefly summarize, some criticality measures, such as drivable area, suffer from the complexity of computation. Criticality measures using intuitional variables, such as distance and velocity, are widely adopted and achieve ideal results. The degree of violation offers a new idea to measure the criticality of scenarios, while efforts are required to define a concrete computing method for it. The criticality measure can be used as an optimization target in the next steps.

3) Obtaining Target Scenarios: Scenario generation often uses various algorithms to handle related issues. [47]–[49], [56], [61], [62] use objective-based optimization search algorithms to adequately search the solution space. [21], [53], [60] use Genetic Algorithm (GA) and Fuzzing technique to obtain test cases. [66] uses Importance Sampling techniques and Deep Learning techniques to accelerate the rare-event testing process. [54], [55], [57] design a novel algorithm to search the solution space and accelerate the testing process.

Using Search-Based Algorithms. [47], [48] search for more critical scenarios under the guidance of their criticality measure. [47] rewrite the discrete-time approximation of the optimization problem as a quadratic programming problem. Moreover, a binary search algorithm is applied to avoid getting stuck in local optimums.

After pruning the parameter space of a scenario by adding constraints, [48] adopts two Evolutionary Algorithms (EAs), differential evolution (DE) and particle swarm optimization (PSO), to find better local minima over a larger range of scenarios. [48] finds EAs are efficient in prompting solution diversity, and especially suited for global optimization problems for which no analytic gradient can be formulated. Moreover, a repair algorithm based on a linear program is also formulated, which enables that there are no collisions among other traffic participants in generated scenarios.

The basic idea of [54], [55], [57] is to find local critical scenarios by optimization methods and then search their neighborhood scenarios. However, most scenarios are uncritical with zero criticality and zero gradient of criticality. As a result, the optimization process degrades to a random sampling process, due to inadequate information on searching direction. To address this problem, these three works design an auxiliary objective function to guide searching directions, which aims to explore approximate searching directions. Under the guidance of the auxiliary objective function, these three works applied a multi-start optimization method to search the local critical scenarios, and the seed-fill method to search neighborhood critical scenarios.

[61] proposed the improved particle swarm optimization (IPSO) to improve the testing coverage, i.e., the optimization target is coverage of the search space. Compared with PSO, IPSO improves search performance from three aspects. First, IPSO introduces the Latin hypercube sampling method to make the distribution of the initial particle group uniform in the search space. Second, IPSO adopts the neighborhood operator to explore more modals, thus finding the current global optimum of the entire particle group. Finally, IPSO redesigns the convergence judgment and restart strategy to explore the search space by avoiding local convergence.

The optimization objective of [62] is the same as that of [61]. [62] proposes LAMBDA (Latent-Action Monte-Carlo Beam Search with Density Adaption). LAMBDA recursively divides the search space into accepted and rejected subspaces to localize the solution set. The density information makes LAMBDA handle the sampling bias of optimization, improving the exploration of the search space.

Both [49] and [56] use search-based algorithms for multi-objective optimization. The optimization objectives of [49] include three factors, which quantitate the violation degree of safety requirements. [56] searches for the collision and for the alternative configuration which avoids it. The search algorithm in [49] extends MOSA [68]. [56] transforms its optimization problem into a Pareto Front solving problem.

Using GA-based Fuzzing. [21], [53], [60] use the GA-based Fuzzing technique to obtain test cases. All three works share a similar fuzzing algorithm: First, the fitness function is set as the criticality measure defined above to drive the GA. Second, GA introduces randomness and diversity via two types of recombination operations, i.e., crossover and mutation, such that the GA can explore different descent directions and avoid local optimums. Finally, the fuzzing process is under the guidance of the intuition that there might exist a target case around a near-miss case.

Using DL techniques. [66] uses a DL-based method to accelerate the rare-event testing process. The first stage of the method is to learn the rare-event set by considering set learning as a classification task. The second stage is to apply an efficiency-certified IS on the rare-event probability over the learned set.

4) Improving Testing Coverage: Another prevailing target of scenario-based AV virtual testing is to improve testing coverage. For searching critical scenarios, most of the research aims to improve search efficiency by searching critical scenarios under the guidance of the criticality measure. Therefore, there are few methods to measure the coverage of the search space. To fill the research gap, [61], [62] adopt a confusion matrix to measure the variety of scenarios. [50], [58]–[60] introduce STL models to ensure testing coverage.

Confusion Matrix. The concept of the Confusion Matrix is illustrated in Figure 7. This method aims to reduce the cases of known unsafe scenarios (area 2) and unknown unsafe scenarios (area 3), to improve testing coverage and ensure the reliability of ADSs. [61] suppose the ratio of TP to T, the metric Recall,

is the emphasis for the critical scenarios search problem. However, if the samples are few and critical, the Recall can be held to 100%, which is meaningless. Consequently, [61] adopts the metric F1 score to quantify the coverage of the search space. [62] refers to F2 score to balance recall and precision, with recall being more emphasized. The selected metric can be used as the optimizing target. Therefore, the testing coverage can be improved by using optimization algorithms.

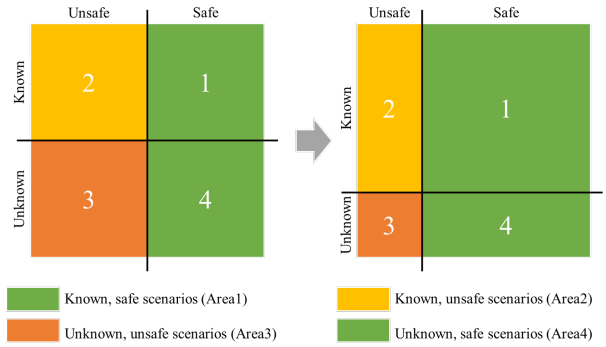

Fig. 7. The Confusion Matrix of Scenarios

STL Model. [50], [58]–[60] introduce STL models to ensure testing coverage, and we describe this method by [60]. As mentioned earlier, [60] uses STL to model scenarios. The STL model based on the original specification describes how EV does not violate the original specification. Thus, by negation operations, the STL model can guide the detection of different ways of violating the original specification, which paves the way for improving testing coverage.

B. Test Oracle Identification

In software engineering, oracles are criteria that testers use to judge whether a bug exists. As discussed in Section 2, the test oracle problem for scenario-based AV virtual testing is more challenging. In this section, we introduce weak oracles, strong oracles, and metamorphic relations in scenario-based AV virtual testing.

1) Statistical Measurements as Test Oracles: Various measurements have already been proposed, such as the criticality measures discussed above. Most of the criticality measures introduced above reflect the safety requirements of ADSs. These measures provide a quantitative way for testers to evaluate the safety of the ADS under test.

Drivable area in [47], [48], safety potential in [53], and criticality measures in [49], [51], [52], [54]–[57] consider whether a collision occurs in the testing scenario. They can be used as a threshold, by which testers can determine whether a fault is in the ADS under test. However, oracles based on these measures are all weak oracles. For AVs, arriving at the destination without collisions is only one of the requirements. The driving legality and comfortability are also worth considering.

The traffic laws provide rich sets of criteria for how a car should behave. Besides ensuring driving safety, an ADS should also obey traffic laws when driving in the real world. [63] and [60] adopt oracles based on traffic laws. The approach of test oracle generation in [60] expressed the traffic rules concerning the driver's perspective and is decoupled from concrete testing scenarios.

2) Metamorphic Relations as Test Oracles: A metamorphic relation is a relationship between the input and the output holding among multiple executions, which can be applied to check the functionality of the software under test [40].

[64], [65] execute metamorphic transformation on the perception modules of the ADSs, by publishing synthetic perception messages to perception modules. Then they analyze whether these transformations lead to changes or expected changes in the prediction. Finally, genuine failures and false alarms can be distinguished by this method.

C. DL-Based AV Testing

Fault injection is a testing technique for understanding how the modules in a complex system behave when stressed in unusual ways. Fault injection can be seen as the generation of inputs that cause erroneous model predictions in DL system testing. As modern AV systems are largely image-based, recent research has focused more on how to generate images that can cause erroneous behavior and result in drastic consequences, thus improving the robustness of deep learning models used in different parts of AV systems. Specifically, the images used for testing are generated by transforming images captured in the real world. New images are retrofitted with features that reflect extreme conditions (e.g., raindrops, fog, bright light) to test the model's performance on corner cases that are not included in the regular dataset.

1) Criteria of Testing Cases: We first introduce the criteria for generating test image cases, the appropriate criteria can help to test a deep learning model more thoroughly and identify potential flaws. For deep learning models in AV systems, the criteria are designed to cover more corner cases and thus test the performance of the model in extreme environments.

Neuron coverage. DeepXplore [69] introduces neuron coverage as the first white-box testing metric for DL systems. Neuron coverage is defined as the ratio of the number of unique activated neurons for all test inputs and the total number of neurons in the DNN. A neuron is considered activated if its output is higher than a threshold value (e.g., 0.5). However, Ma et al. [70] show that neuron coverage should be calculated with a higher degree of granularity. They propose DeepGauge which is a multi-granularity criterion for testing the DL system. It improves the calculation of neuron coverage by splitting it into two parts, neuron-level coverage, and layer-level coverage. The neuron-level coverage is further divided into three parts, k-multisection neuron coverage, neuron boundary coverage, and strong neuron activation coverage. The layer-level coverage includes Top-k Neuron Coverage and Top-k Neuron Patterns. These carefully designed computational criteria allow DeepGuage to better detect flaws in DL systems. The researchers used neuron coverage as a criterion to generate images that could find flaws in the AV system. For a set of input images, if they have a higher neuron

(a) Patch (b) Holes (c) Translation (d) Fog (e) Rain

Fig. 8. Driving scenes synthesized by DeepXplore (a)(b) and DeepTest (c)(d)(e)

coverage for the model in the AV system, this means that it is possible to test the model in more different scenarios and thus find potential flaws.

Differential Behaviours. DeepXplore [69] also introduces another test criterion, which is behavioral differences. Intuitively, a set of DL models with the same function should have the same output for the same input. Therefore, if there is an image whose output behavior varies significantly across a set of models, then it is likely to represent a corner case that could lead to incorrect model predictions. Thus, maximizing behavioral differences contributes to the generation of input images that are more representative of extreme cases.

2) Generation of Testing Cases: Unlike tests performed on deep learning models alone, testing images for models in AV systems need to conform to the laws of the physical world. Test cases generated by adding perturbations to the images, such as adversarial samples, can deceive the deep learning model, but do not reflect the extreme environments and conditions in the real physical world. Recent work yields high-quality test cases by transforming real images or using GAN for the image-to-image generation.

Image transformation. DeepXplore provides three kinds of transformation including changing brightness, adding occlusion, and adding dirt(i.e., a patch of perturbation). Compared to DeepXplore, DeepTest [71] focuses on generating more realistic synthetic images by applying image transformations on seed images. It generates testing input images that maximize the number of activated neurons. The transformation rules are manually selected to mimic different real-world conditions like camera lens distortions, object movements, different weather conditions(e.g., rainy and foggy), etc. It explores three groups of nine image transformation methods including changing brightness, changing contrast, translation, scaling, horizontal shearing, rotation, blurring, fog effect, and rain effect. Among them, translation, scaling, horizontal shearing, and rotation can be seen as an affine transformation, which is a linear mapping between two images that preserves points, straight lines, and planes. Adjusting brightness and contrast are both linear transformations since both can be implemented by adding/subtracting/multiplying a constant parameter to each pixel's current value. Blurring and adding fog/rain effects are all convolutional transformations that are implemented by performing the convolution operation on the input pixels with different kernels.

However, the transformations used in DeepXplore and DeepTest to generate testing images cannot reflect real-world driving scenes. As shown in Figure 8, strange patches or black holes are rare in reality and the blurring/fog/rain effects made by convolution operation are also unrealistic.

Image Generation. DeepRoad [72] is an unsupervised DNN-based framework for automatically testing the consistency of DNN-based AV systems. DeepRoad automatically synthesizes large amounts of diverse driving scenes without using manually defined image transformation rules (e.g. scale, shear, and rotation). In particular, DeepRoad leverages UNIT, an unsupervised image-to-image transformation based on Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). UNIT [73] supposes that two images contain the same contents but lie in different domains, they should have the same representations in a shared-latent space. Accordingly, given a new image from one domain (e.g., the sunny driving scene), UNIT can automatically generate its corresponding image in the other domain (e.g., the snowy driving scene).

3) Input Validation: Though testing cases are useful to expose the system's vulnerability, it is not sufficient for online testing. For instance, a DNN-based AV system can perform well in sunny environments, yet it might perform incorrectly at night since the feature it relies on for guiding disappear in such scenes. Thus, the AV system should validate input images online, and actively advise drivers to control the car when it cannot handle the invalid inputs. DeepRoad applies the Probably Approximately Correct (PAC) Learning theory to distinguish between input images belonging to different scenes for input validation.

VI. Considerations in Scenario-Based AV Virtual Testing

In the previous section, we discussed the basic procedures and methods in scenario-based AV virtual testing. In this section, we distill seven primary considerations in scenario-based AV virtual testing.

A. Safety Requirement

SR1: Safety requirements are quantitative and measurable in the testing processes. After identifying and specifying safety requirements correctly and completely, the feasibility of these requirements also needs consideration. For virtual testing, it is essential to ensure the measurability and computability of the requirements, as variables defined in the environment are high-dimonsional and the number of executed test cases is often quite large. Consequently, safety

requirements need to hold quantitative and measurable in the testing processes.

B. Scenario Model

SM1: In data-driven approaches, the datasets used should be representative, and the probability distributions need to be valid. In data-driven approaches, the datasets used need to be able to reflect the validity of measured distributions in the real world and their generalizability. In [54], [55], the datasets also need to be representative of the targeted ODD in some form. Thus, the data collected must record the entities and their relations observed in the real world, for evaluating the real-world driving environment.

SM2: Discretization suffers from the risk of missing valuable test cases. After modeling the scenarios, a discretization step is usually performed to reduce the test effort to a finite number of concretizations. Though excessively fine discretizations can lead to a large number of redundant test cases, any discretization procedure suffers from bringing about gaps in the original state space, resulting in missing potential test cases of interest.

C. Criticality Measure

CM1: Adopted criticality measures should be valid. Ideally, the applied criticality measure should capture as many critical scenarios as possible. Unfortunately, a single metric is sometimes not sufficient to meet this demand. Accordingly, some works adopt several metrics in conjunction, expecting each to capture different aspects of criticality respectively. Yet regardless of using a single criticality measure or a composite one, even simple metrics incorporate a group of parameters, which have a significant impact on the computational cost. Additionally, for composite metrics, their mathematical properties, e.g., being continuous, differentiable, or convex, determined the complexity of the optimization problem.

CM2: Proper thresholds for criticality measures are applied. Although the common practice is to specify the threshold empirically, it is tough to determine the appreciated threshold for criticality measures to detect safety faults. In addition, the threshold can be adopted to define the critical subspace, which is essential in optimization. Hence, a suitable threshold also affects the efficiency of the optimization.

D. Test Case Generation

TCG1: The critical subspace needs to be explored systematically. To prevent the potential that a critical subspace is missed entirely by the adopted optimization algorithms, optimization algorithms have better be executed repeatedly, with changing step lengths or starting points.

TCG2: The synthetic images for different scenarios should be realistic. While existing work can generate a large number of test images for different scenarios, a large proportion of these are not realistic, suggesting that the models cannot be tested effectively. Even generative methods based on deep neural networks such as GAN may generate test images that do not conform to the rules of the physical world or are unlikely to be present in realistic driving scenarios.

VII. Research Trends

We analyze the 35 papers used in this survey from three dimensions, including publication year, venue, and high-frequency vocabulary. Through these three perspectives, we summarise the research trend of scenario-based virtual testing for AV systems in recent years.

A. Publication year

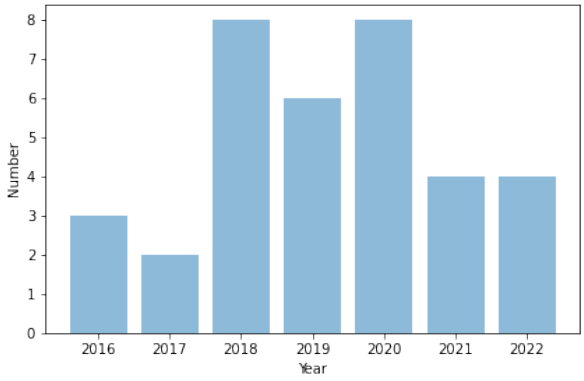

Fig. 9. Distribution among years of 35 papers

The rapid advances in deep learning technology and the reduction in the cost of vehicle sensors enable autonomous driving technology to develop dramatically in recent years, which greatly facilitate research into the automated testing of AV systems. As an effective testing method, scenario-based virtual testing attracts significant research interest. Constraints with time and effort prevent us from collecting and studying all relevant work, which leads to a relatively low number of papers published in 2021 and 2022 in our paper dataset. However, as shown in Fig. 9, 30 of 35 papers in our survey are published in the past 5 years. This demonstrates the success of scenario-based virtual testing in AV systems testing and is still an active field with plenty of topics to explore. We discuss the potential research opportunities in Sec. VIII in detail. We also find that the latest researches are more focused on testing the performance of AV systems in extreme scenarios and tends to use frontier techniques like those based on deep learning, multimodality, etc.

B. Venue

We label the venue of each paper by the field of the journal or conference where it is published. As shown in Fig. 10, two of the most active communities for scenario-based virtual testing are artificial intelligence (AI) and software engineering (SE). This makes perfect sense as the design of AV systems relies on advances in AI and system/software testing is an enduring topic in SE. Cross-collaboration between different domains is a trend in research on scenario-based virtual testing. We also notice that some researchers in the security community are focusing on this topic. Compared to SE and AI researchers, security researchers are focusing more

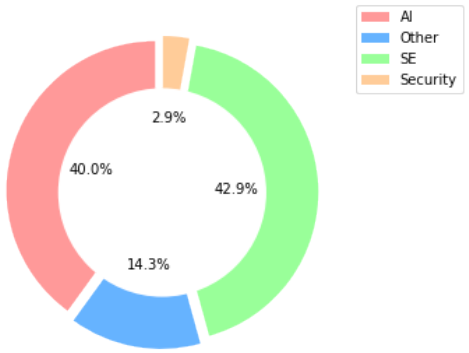

Fig. 10. Distribution among venues of 35 papers

on exploitable vulnerabilities in AV systems, which opens up new ideas in this direction.

C. High-frequency vocabulary

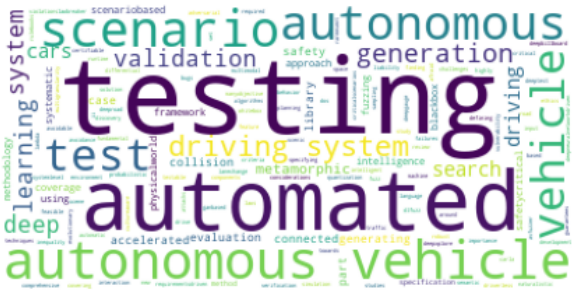

Fig. 11. Wordcloud of 35 papers' titles

As some of the papers in our paper dataset are not marked with keywords, we count the word frequencies of the 35 paper titles to show the issues and techniques of greatest interest in recent research. As the word cloud in Fig. 11 illustrates, in addition to words such as 'testing', 'autonomous', etc., which are used to indicate the research objective, words such as 'generation', 'deep', and 'learning' indicate that more and more research is attempting to use DL-based techniques to assist in the generation of test scenarios. The emergence of words such as "validation" also indicates that researchers are beginning to focus on extension issues such as validation beyond the generation of scenarios.

VIII. Research Opportunities

Though research in AV virtual testing is experiencing rapid growth, there are still plenty of research opportunities. In this section, we discuss potential research directions according to the testing workflow introduced above.

A. Test Oracle for Scenario-Based AV Testing

On the one hand, various manually-defined oracles bring problems for test oracle selection. In the previous sections, we present a list of statistical measurements for test oracle selection, with the research community still introducing further novel measurements. As each of these oracles has its own applicable conditions, there may not be a single oracle suitable for extensive contexts. Consequently, a direction for the research community is selecting suitable test oracles according to the intended contexts.

On the other hand, existing works apply statistical measurements as indirect oracles or employ metamorphic relations as pseudo oracles. Test oracle identification for AV testing is challenging compared to traditional software testing. It is demanding to construct a fully automated test oracle, as it essentially involves recreating the logic of a human driver. Another research direction, therefore, is designing automatic techniques for constructing oracles for AV testing.

B. Test Adequacy for Scenario-Based AV Testing

Test adequacy has already been widely studied in traditional software testing, while research on the test adequacy of ADSs is lacking. Besides checking the coverage of the testing, adequacy criteria also can be applied to guide test generation. For test adequacy of ADS, it remains an open problem.

For this question, an intuitive idea is to employ traditional software test adequacy metrics for AV testing. However, the complexity of the ADS makes it tough to apply these methods. Additionally, for DL-based modules in ADS, although there are several test adequacy metrics proposed for DL software, such as neuron coverage, layer coverage, and surprise adequacy, there is no empirical evidence that these metrics are applicable and effective for revealing safety faults and sufficiency of AV safety testing. Thus, a direction for the research community is checking the test adequacy for ADS.

C. Fault Localization

During extensive testing for the ADSs, it is usually accumulated a considerable amount of test records, which include violations of certain test oracles, such as collisions. If the causes of these faults can be found and applied as guidance for updating the ADSs, the performance of the ADSs will be improved in a more targeted manner. Subsequently, using these records for fault localization is also a promising research direction.

IX. Open-Source ADSs and Related Tools

This section introduces two prevailing open-source ADSs and the two most widely-adopted simulators for AV virtual testing, to provide quick navigation for researchers and practitioners.

A. Open-Source ADSs

Autoware [24] and Apollo [20] are both open-source projects for autonomous driving.

Autoware is available under Apache 2.0 license. It contains five main modules: localization, detection, prediction, planning, and control. A noteworthy part of Autoware is Autoware Core and Autoware Universe. The former includes all necessary functionality to support the Operation Design Domains targeted by the Autoware project. The latter enables additional packages to be built on top of Autoware Core, extending the functionality of Autoware [24]. ROS 1 [25] functions as the robotics middleware of Autoware, which supports the communication among components in the entire ADS. As one of the most popular frameworks for robot software development, ROS has accumulated a lot of experience, which avoids repetitive development work and improves development efficiency. However, ROS helps a little in resource scheduling, which impacts the performance of the ADS. Since task scheduling occurs in the kernel area, while the Linux kernel scheduler can only guarantee the fair allocation of resources.

Apollo considers these issues. Apollo is based on CyberRT [20], an open-source robotics middleware designed for autonomous driving scenarios specifically. CyberRT is hight optimized for performance, latency, and data throughput, based on a centralized computing model. CyberRT uses "co-routines" to move scheduling and tasks from kernel space to user space so that scheduling can be closely combined with algorithm business logic [20]. Compared with ROS, CyberRT also adds the Components concept, which provides the flexibility of modules. Thus, CyberRT supports asynchronous computing tasks, optimizes thread usage and system resource allocation, and also supports the definition of the module topology [20].

B. Simulators

High-fidelity simulators play an essential role in AV virtual testing. LGSVL [26] and CARLA [27] are two prevailing simulators used in virtual testing, suitable for end-to-end testing of the unique features of AVs.

LGSVL, or SVL Simulator, is a simulation platform used for autonomous vehicle and robotic system development [26]. Besides integration testing in scenario-based testing, LGSVL also enables users to perform modular algorithm testing and system verification. Based on Unity's game engine, LGSVL generates photo-realistic virtual environments using High Definition Rendering Pipeline (HDRP) technology [26]. LGSVL supports ROS 1, ROS 2, and CyberRT messages, which helps to connect the simulator with Autoware and Baidu Apollo [26]. Besides, the simulator provides a Python API for testing, by which users can set up supported sensors, weather conditions, traffic agents, etc. In addition to cameras, LiDAR, IMU, GPS, and radar, LGSVL also allows users to define custom sensors [26]. LG had decided to suspend the development of LGSVL, as of January 1, 2022.

CARLA is an open-source simulator for autonomous vehicles. Similar to LGSVL, CARLA supports the training and validation of ADSs. CARLA uses Unreal Engine 4 (UE4) [28], enabling realistic physics and basic NPC logic. CARLA is provided with integration with ROS via its ROS-bridge. Therefore, direct testing with CARLA and Apollo was not supported before. However, a recent project, [29], makes it possible to connect CARLA with Apollo. CARLA also provides a user-defined API to control the simulation [30]. However, compared to LGSVL, CARLA provides a smart method to control the behavior of other traffic participants by its Traffic Manager [30].

X. Conclusion

We have presented a survey of 35 papers on scenario-based AV virtual testing. We introduced the general architecture of modern ADS and compared AV testing with traditional software testing. We first introduced the three abstraction levels of scenarios for AV testing. From two perspectives, i.e., test case generation, and test oracle generation, we then summarized the current research status in scenario-based AV virtual testing workflow and distilled a list of fundamental considerations in this field. Subsequently, we analyzed research trends and opportunities for scenario-based AV virtual testing. Finally, we introduced two open-source ADSs and two simulators for AV virtual testing, which researchers and practitioners can access. We hope this survey will help fresh researchers from this field obtain some rudimental knowledge of scenario-based AV virtual testing.

Acknowledgment

We want to express our special thanks of gratitude to Professor Chen who gave us the golden opportunity to do this wonderful project on Scenario-Based Virtual Testing for Autonomous Vehicles, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to him.

References

[1] August 2019. Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says. https://www.nytimes.com/2016/07/01/business/self-driving-tesla-fatal-crash-investigation.html.
[2] August 2019. Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam. https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html.
[3] August 2019. Tesla: Autopilot was on during deadly Mountain View crash. https://www.mercurynews.com/2018/03/30/tesla-autopilot-was-on-during-deadly-mountain-view-crash/.
[4] August 2019. Fatal Tesla Crash Exposes Gap In Automaker's Use Of Car Data. https://www.forbes.com/sites/alanohnsman/2018/04/16/tesla-autopilot-fatal-crash-waze-hazard-alerts/#7bb735fb5572.
[5] SAE On-Road Automated Vehicle Standards Committee et al. 2014. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. SAE Standard J 3016 (2014), 1–16.
[6] "ESEC/FSE," https://www.esec-fse.org/, 2022, retrieved on Nov 21, 2022.
[7] "ASE," https://ase-conferences.org/, 2022, retrieved on Nov 21, 2022.
[8] "ISSRE," https://issre2022.github.io/, 2022, retrieved on Nov 21, 2022.
[9] February 2022. 2021 Disengagement Report from California, Mario Herger. https://thelastdriverlicenseholder.com/2022/02/09/2021-disengagement-report-from-california/
[10] February 2021. 2020 Disengagement Reports from California, Mario Herger. https://thelastdriverlicenseholder.com/2021/02/09/2020-disengagement-reports-from-california/
[11] February 2020. Disengagement Report 2019, Mario Herger. https://thelastdriverlicenseholder.com/2020/02/26/disengagement-report-2019/
[12] California Department of Motor Vehicles, https://www.dmv.ca.gov/portal/, 2022, retrieved on Nov 21, 2022.

[13] Kalra, N. and S. 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, 2016.

[14] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," IEEE Transactions on Software Engineering, vol. 48, no. 2, pp. 1–36, 2022.

[15] Han, J. C. , and Zhou, Z. Q. . (2020). Metamorphic Fuzz Testing of Autonomous Vehicles. IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). ACM.

[16] F. Zhu, L. Ma, X. Xu, D. Guo, X. Cui, and Q. Kong, "Baidu apollo auto-calibration system-an industry-level data-driven and learning based vehicle longitude dynamic calibrating algorithm," arXiv preprint arXiv:1808.10134, 2018.

[17] A. C. Madrigal, "Inside waymo's secret world for training self-driving cars," The Atlantic, vol. 23, 2017.

[18] Li, G. , et al. "AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems." 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) IEEE, 2020.

[19] Garcia, J. , et al. "A comprehensive study of autonomous vehicle bugs." ICSE '20: 42nd International Conference on Software Engineering 2020.

[20] 2020. Apollo 6.0. https://github.com/ApolloAuto/apollo/releases/tag/v6.0.0. retrieved on Nov 21, 2022.

[21] Wan, Z. , et al. "Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical World Attacks." 2022

[22] CAN Bus Explained – A Simple Intro [2022] https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial .retrieved on Nov 21, 2022.

[23] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In ICRA workshop on open source software, Vol. 3. Kobe, Japan, 5.

[24] 2022. Autoware.AI. www.autoware.ai/. retrieved on Nov 21, 2022.

[25] 2022. ROS 1. http://wiki.ros.org/ . retrieved on Nov 25, 2022.

[26] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Martins Mozeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In ITSC. IEEE, 1–6.

[27] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In CoRL (Proceedings of Machine Learning Research, Vol. 78). PMLR, 1–16.

[28] Epic Games. Unreal Engine 4. https://www.unrealengine.com. retrieved on Nov 25, 2022.

[29] Carla Apollo Bridge. https://github.com/guardstrikelab/carla_apollo_bridge . retrieved on Nov 25, 2022.

[30] CARLA Documentation. https://carla.readthedocs.io/en/latest/ . retrieved on Nov 25, 2022.

[31] A. Putz, A. Zlocki, J. K ¨ ufen, J. Bock, and L. Eckstein, "Database approach for the sign-off process of highly automated vehicles," in 25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration, 2017.

[32] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in Proc. of the Congress on Evolutionary Computation, vol. 2, 2004.

[33] R. Lee, O. J. Mengshoel, A. Saksena, R. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, "Adaptive stress testing: Finding failure events with reinforcement learning," arXiv preprint arXiv:1811.02188, 2018.

[34] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," in Proc. of the IEEE International Conference on Robotics and Automation, 2017, pp. 1443–1450.

[35] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 163–168.

[36] Althoff, M. , and S. Lutz . "Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles." 2018 IEEE Intelligent Vehicles Symposium (IV) IEEE, 2018.

[37] ISO, 26262 – Road vehicles – Functional Safety, 2016

[38] G. Bagschik, A. Reschka, T. Stolte, and M. Maurer, "Identification of Potential Hazardous Events for an Unmanned Protective Vehicle," in 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 2016, pp. 691–697.

[39] Menzel, T. , Bagschik, G. , & Maurer, M. . (2018). Scenarios for Development, Test and Validation of Automated Vehicles. IEEE, 10.1109/IVS.2018.8500406.

[40] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering 41, 5 (2015), 507–525.

[41] Zhou, Z. Q. , and L. Sun . "Metamorphic testing of driverless cars." Communications of the ACM 62.3(2019):61-67.

[42] Apostol Vassilev and Christopher Celi. 2014. Avoiding cyberspace catastrophes through smarter testing. Computer 47, 10 (October 2014), 102–106.

[43] J. Zhang, L. Zhang, M. Harman, D. Hao, Y. Jia, and L. Zhang, "Predictive mutation testing,"" IEEE Transactions on Software Engineering, vol. 45, no. 9, pp. 898–918, 2019.

[44] Richtlinie für die Anlage von Autobahnen - English title: Guidelines for Constructing Motorways, Forschungsgesellschaft für Straßen und Verkehrswesen Std., 2009

[45] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2019. Scenic: a language for scenario specification and scene generation. In PLDI. ACM, 63–78.

[46] 2021. AVUnit. https://avunit.readthedocs.io/en/latest/. Online; accessed August 2022.

[47] Althoff, M. , and S. Lutz . "Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles." 2018 IEEE Intelligent Vehicles Symposium (IV) IEEE, 2018.

[48] Klischat, M. , and M. Althoff . "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms." 2019 IEEE Intelligent Vehicles Symposium (IV) IEEE, 2019.

[49] Abdessalem, R. B. , et al. "Testing Autonomous Cars for Feature Interaction Failures using Many-Objective Search." 2018:143-154.

[50] Tuncali, C. E. , et al. "Requirements-Driven Test Generation for Autonomous Vehicles With Machine Learning Components." IEEE Transactions on Intelligent Vehicles 5.2(2020):265-280.

[51] A Calò, et al. "Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems." 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST) IEEE, 2020.

[52] Li, L. , et al. "Intelligence Testing for Autonomous Vehicles: A New Approach." IEEE Transactions on Intelligent Vehicles 1.2(2017):158-166.

[53] Li, G. , et al. "AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems." 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) IEEE, 2020.

[54] Feng, S. , et al. "Testing Scenario Library Generation for Connected and Automated Vehicles, Part I: Methodology." IEEE, 10.1109/TITS.2020.2972211. 2020.

[55] Feng, S. , et al. "Testing Scenario Library Generation for Connected and Automated Vehicles, Part II: Case Studies." IEEE 9(2021).

[56] Zhao, D. , et al. "Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques." IEEE Trans Intell Transp Syst (2017):1-13.

[57] Feng S, Yan X, Sun H, et al. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment[J]. Nature Communications, 2021, 12(1): 1-14. doi: 10.1038/s41467-020-20314-w

[58] Majzik, István, et al. "Towards system-level testing with coverage guarantees for autonomous vehicles." 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE, 2019.

[59] Zapridou, Eleni, Ezio Bartocci, and Panagiotis Katsaros. "Runtime verification of autonomous driving systems in CARLA." International Conference on Runtime Verification. Springer, Cham, 2020.

[60] Yang Sun, Christopher M. Poskitt, Jun Sun, Yuqi Chen, and Zijiang Yang. 2022. LawBreaker: An Approach for Specifying Traffic Laws and Fuzzing Autonomous Vehicles. In 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3551349.3556897

[61] Feng, Tianyue, et al. "Multimodal critical-scenarios search method for test of autonomous vehicles." Journal of Intelligent and Connected Vehicles ahead-of-print (2022).
[62] Liu, Lihao, et al. "LAMBDA: Covering the Solution Set of Black-Box Inequality by Search Space Quantization." arXiv preprint arXiv:2203.13708 (2022).
[63] Censi, Andrea, et al. "Liability, ethics, and culture-aware behavior specification using rulebooks." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.
[64] Zhou, Z. Q. , and L. Sun . "Metamorphic testing of driverless cars." Communications of the ACM 62.3(2019):61-67.
[65] Han, J. C. , and Z. Q. Zhou . "Metamorphic Fuzz Testing of Autonomous Vehicles." IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20) ACM, 2020.
[66] Arief, M. , et al. "Deep Probabilistic Accelerated Evaluation: A Robust Certifiable Rare-Event Simulation Methodology for Black-Box Safety-Critical Systems." International Conference on Artificial Intelligence and Statistics PMLR, 2021.
[67] Dejan Nickovic and Tomoya Yamaguchi. 2020. RTAMT: Online Robustness Monitors from STL. In ATVA (LNCS, Vol. 12302). Springer, 564–571.
[68] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2015. Reformulating Branch Coverage as a Many-Objective Optimization Problem. In Proceedings of the International Conference on Software Testing, Verification and Validation, (ICST'15). Graz, Austria, 1–10.
[69] Kexin Pei, Yinzhi Cao, Junfeng Yang and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In Proceedings of the 26th Symposium on Operating Systems Principles, (SOSP'17), Shanghai, China, October 28-31, 2017. 1-18
[70] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE'18, Montpellier, France, September 3-7, 2018. 120-131
[71] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th International Conference on Software Engineering, ICSE'18, Gothenburg, Sweden, May 27 - June 03, 2018, 303-314
[72] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE'18, Montpellier, France, September 3-7, 2018, 132-142
[73] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In Advances in Neural Information Processing Systems, NeurIPS'17, December 4-9, 2017, Long Beach, CA, USA, 700–708
[74] Association for Computing Machinery. The ACM Digital Library. https://dl.acm.org/
[75] Institute of Electrical and Electronics Engineers. IEEE Xplore Digital Library. https://ieeexplore.ieee.org/Xplore/home.jsp
[76] Springer Nature. Springer Lecture Notes in Computer Science. https://link.springer.com/search?facet-discipline="Computer+Science"