ADEPT: A Testing Platform for Simulated Autonomous Driving

Sen Wang[†], Zhuheng Sheng[†], Jingwei Xu[*], Taolue Chen, Junjun Zhu, Shuhui Zhang, Yuan Yao, Xiaoxing Ma

State Key Lab of Novel Software Technology, Nanjing University, China

Department of Computer Science, Birkbeck, University of London, UK

ABSTRACT

Effective quality assurance methods for autonomous driving systems ADS have attracted growing interests recently. In this paper, we report a new testing platform ADEPT, aiming to provide practically realistic and comprehensive testing facilities for DNN-based ADS. ADEPT is based on the virtual simulator CARLA and provides numerous testing facilities such as scene construction, ADS importation, test execution and recording, etc. In particular, ADEPT features two distinguished test scenario generation strategies designed for autonomous driving. First, we make use of real-life accident reports from which we leverage natural language processing to fabricate abundant driving scenarios. Second, we synthesize physically-robust adversarial attacks by taking the feedback of ADS into consideration and thus are able to generate closed-loop test scenarios. The experiments confirm the efficacy of the platform.

KEYWORDS

Software testing, Deep Neural networks, Autonomous driving, Test case generation, Testing platform

ACM Reference Format:

Sen Wang[†], Zhuheng Sheng[†], Jingwei Xu[*], Taolue Chen, Junjun Zhu, Shuhui Zhang, Yuan Yao, Xiaoxing Ma. 2022. ADEPT: A Testing Platform for Simulated Autonomous Driving. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3551349.3559528

1 INTRODUCTION

There has been an increasing concern about the reliability and safety of autonomous driving systems (ADS), especially with the use of deep neural networks (DNN) as their core components. Testing, as one of the most effective quality assurance methods of ADS, is expected to play an indispensable role in ADS development. Testing with physical autonomous vehicles is natural, but clearly is neither cost-effective nor safe. In particular, it is usually challenging to reproduce various types of accidents (e.g., those involving switching between severe weather conditions) and its cost can be financially unaffordable. A realistic alternative and arguably more effective strategy is to test ADS under a simulated environment, where both driving scenes and physical logic (i.e., car dynamics, traffic regulations, and weather conditions) are built in a virtual world.

In this paper, we present a platform ADEPT (**A**utonomous **D**riving t**E**sting **P**la**T**form) to provide comprehensive testing for ADS which are akin to real-world scenarios. Several virtual simulators including CARLA [9] and AirSim [25] are available. We base our tool on CARLA in light of its abundant APIs and community support. ADEPT supports various functionalities, including scene construction, ADS importation, test execution and recording, etc. Of paramount importance for testing frameworks is test case generation, which, in our setting, exhibits in the form of building up a specific scene for ADS to react. The inherent complexity of ADS gives rise to a tremendous search space of scenes for which exhaustive search is clearly prohibitive. Moreover, ad-hoc testing case generation methods are prone to yielding impractical and less meaningful scenarios from the physical-world perspective. For example, the scenario that an obstacle suddenly appears in the middle of the road would be meaningless as it rarely happens.

In ADEPT, we introduce two strategies to enable effective generation of more realistic test cases. First, we observe a connection between harmful circumstances and security breaches which we use to direct the search toward a more security-sensitive area. To this end, we extract potential ADS infractions from real-life accident reports by leveraging natural language processing (NLP) techniques, which are then used to fabricate testing scenarios. Second, we adapt adversarial examples of DNNs to generate testing scenario. In view of the central role of DNNs in ADS (e.g., in the decision-making module) and their fragility to imperceptible malicious perturbations known as adversarial examples, it is not difficult to generate test scenarios based on these examples. However, the effectiveness of adversarial examples are usually dependent on physical conditions (e.g., illumination, blur, and angle of view), making the test scenarios unrealistic. Moreover, an individual attack that tricks ADS into making mistakes is usually insufficient as ADS can recover from these one-time, sporadic mistakes. In ADEPT, we instead generate a sequence of robust adversarial examples that takes the feedback of ADS into account and thus form a closed-loop test scenario.

The code is available at https://github.com/shengzh-oOoO/ADEPT and a video demo can be found at https://youtu.be/evMorf0uR_s.

2 THE TOOL ADEPT

Figure 1 depicts the three-layer architecture of ADEPT, i.e., *Simulator Kernel*, *ADS Testing Libraries*, and *Scenario Engine.*

The Simulator Kernel layer includes several key components directly built upon Unreal Engine and Carla to provide specific operations for ADS testing, e.g., interactive camera. Note that all

[†]These authors contributed equally to this work.
[*]Corresponding author.

ASE '22, October 10–14, 2022, Rochester, MI, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3559528

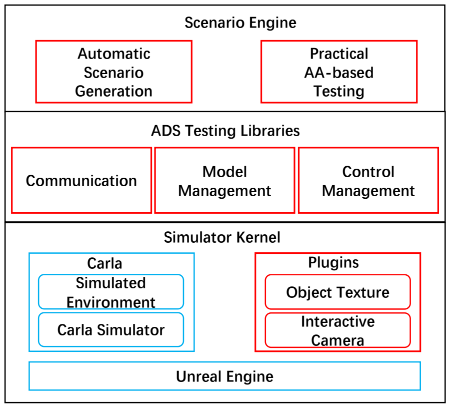


Figure 1: An overview of ADEPT.

modules in red boxes are developed by us, while those in blue boxes are essentially from either CARLA or Unreal Engine.

The ADS Testing Libraries layer includes three key modules.

- the *communication* module, which is responsible for the interactions between subjects (e.g., driving model or attack algorithm) and objects in simulators (e.g., cars, humans, and electronic billboards). This module is designed to encapsulate the basic CARLA APIs for developing and recording specialized ADS testing scenarios, as data access and transmission between the subjects and the simulator are inefficient via CARLA's native APIs.
- the *model management* module, which is responsible for the management of ADS and cars, e.g., connection of ADS to ADEPT and to place the required sensors on the victim car.
- the *control management* module, which deploys the generated scenario online, e.g., it utilizes virtual camera redirected to platform objects to adjust textures in real-time.

The Scenario Engine layer implements various methods for effectively generating real-world scenarios that may trigger ADS violations. Currently, there are two such methods, as detailed below.

Generation of Test Scenarios from Accident Reports. This engine takes a traffic accident report and translates the accident description (in natural language) to an intermediate representation (in a scenario-description language) for which we use Scenic [11]. The Scenic-supplied tools can load the scene and connect to the ADS for testing at the same time.

We leverage natural language processing aiming for automated translation of the accident report. Technically, we utilise a question-and-answer technique based on the question-answering system GPT-3 [5]. We define a set of questions as well as Scenic code templates. The report text and predefined questions are sent to GPT-3. Upon the response, we choose the template that corresponds to the accident report's description from the predefined templates and fill in the template's important missing information. Specifically, we inquire about the location of the collision (including whether it occurred at a three-way, four-way, or non-intersection), the time, the weather, and the relative position relationship between the two cars involved (including the left-right relationship of the lane, the front-to-back relationship of the distance, etc.).

Closed-loop Testing Based on Adversarial Attack. It is well-known adversarial examples (AEs) reveal the weakness of DNNs

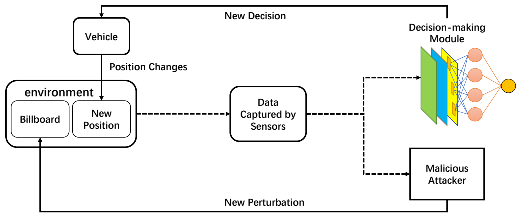


Figure 2: The closed-loop practical hijacking.

which are widely adopted in contemporary ADS, so one can easily utilize them to test ADS. In real-world conditions, however, several physical conditions (such as illumination, blur, angle of vision, etc.) may severely diminish the effectiveness of AEs, especially when the car is in motion. Moreover, sophisticated ADS is able to recover from the past deception, further undermining the effectiveness of a single attack. We propose two strategies to address these issues.

(1) We generate AEs more robust in the physical sense. We find, based on the observation of pixel shifts, that the color between the real pixel and the illustrated pixel of an RGB-camera-captured image can be captured by a nonlinear transformation. On the basis of pixel data collected from the testing platform, we employ a lightweight neural network to fit such a nonlinear transformation. Since the car's perspective does not change significantly during the victim vehicle's decision-making interval, factors caused by the moving state on our testing platform consist primarily of blur and minor changes in angle of view. Here, we resort to Expectation Over Transformation (EOT) [3], which abstracts various transformations into distributions and embeds the transformations as the parallel preprocess of the input, then regards the expectation over all transformations as the ultimate optimized object, with each step of iteration accepting the input updated in the previous step.

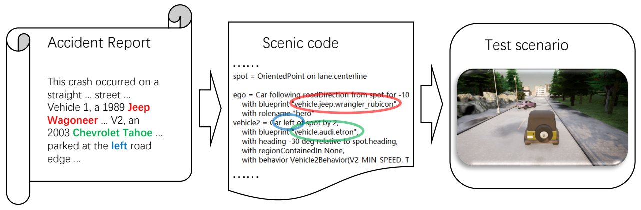
(2) We design a sequence of AEs to achieve continuous deception for the vehicle by incorporating the feedback into successive attacks. Figure 2 depicts the "practical AA-based testing" engine of Figure 1 where the feedback refers to the positional and postural changes of the victim vehicle as observed by the Malicious Attacker Module (MAM). MAM leverages the feedback to anchor itself from the view of the victim vehicle in order to conduct tailored attacks. With adversarial attack algorithms, MAM determines what to show in the next frame attempting to fool the victim vehicle.

To this end, MAM considers the dynamic model of the vehicle, i.e., the actual steering radian and traveling distance at one frame. We implement the Pure Pursuit (PP) [8] algorithm to force the victim vehicle to pursue the goal moving point, thereby following the prescribed trajectory. The target curve comprises multiple points, and the PP algorithm calculates the curvature that will move a vehicle from its current position to the goal position specified by the closest point and the lookahead distance. Eventually, the estimated curvature will serve as the attack target.

3 EVALUATION

In this section, we evaluate ADEPT with the two scenario engines.

Generation of Test Scenarios based on Accident Reports. To select the adequate subject for evaluation, we choose the end-to-end ADS, which is the winner of the Camera-Only track of the CARLA



This crash occurred on a straight ... street ... Vehicle 1, a 1989 **Jeep Wagoneer** ... V2, an 2003 **Chevrolet Tahoe** ... parked at the **left** road edge ...

```
......
spot = OrientedPoint on lane.centerline

ego = Car following roadDirection from spot for -10
    with blueprint "vehicle.jeep.wrangler_rubicon",
    with rolename "hero"
vehicle2 = Car left of spot by 2,
    with blueprint "vehicle.audi.etron",
    with heading -30 deg relative to spot.heading,
    with regionContainedIn None,
    with behavior Vehicle2Behavior(V2_MIN_SPEED, T
......
```

Figure 3: An example of ADS test case generation.

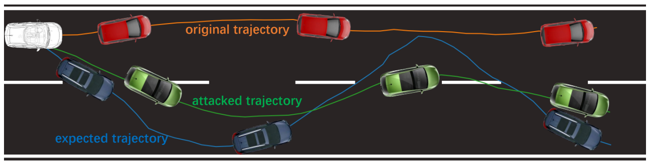


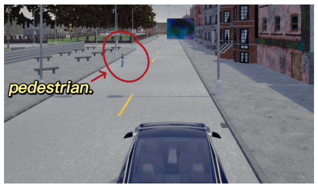
Figure 4: The scenario of trajectory hijacking.

challenge 2020 [26], to be the subject, and the accident reports from NHTSA Crash Viewer [22] for ADEPT.

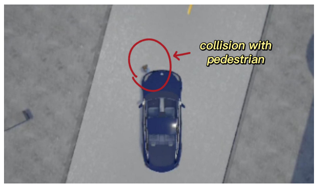
We use 20 traffic accident reports to generate 365 test cases. ADEPT first translates each accident report to the intermediate Scenic code several times. Then, ADEPT generates multiple test scenarios from each Scenic code with slight mutation in the unmentioned description of the code. The test cases rely on the predefined environment templates in Scenic. In the evaluation, we select two typical accident locations, e.g., at crossroads or not, with different vehicles' directions. As shown in Table 1, 73 pieces of valid Scenic code are used. Based on the criteria of risk levels for vehicle accidents, the 365 generated test cases trigger 133 accidents in the simulated environment, of which 54 cases involve serious risks.

Figure 3 shows an example. A customized GPT-3 extracts valuable information (e.g., number of cars, the Vehicle brand, the position, speed, and direction) from the accident description in natural language to generate a Scenic code. Then, the ADS test case is generated based on the Scenic code. Since Carla and Scenic do not support Chevrolet Tahoe, the tool randomly selects an Audi Etron as the vehicle 2 at this time.

Closed-loop Testing based on Adversarial Attacks. To evaluate the testing for the closed-loop adversarial attack, we choose NVIDIA CNN-based steering model [4] as the core decision-making module of the victim vehicle, which receives a front-view RGB image as input and outputs a floating number between -1.0 and 1.0 indicating the steering angle ($-90°$ to $90°$). The NVIDIA model is designed to drive straight along the lane without committing violations such as crossing the yellow line or entering the sidewalk. During the experiment, the victim vehicle is set to maintain a constant speed on a straight road. To exhibit the capability of ADEPT, we evaluate two ADS testing scenarios, i.e., trajectory tracking and pedestrian collision. Firstly, the control management module places a billboard alongside the road. Multiple distributed locations in the form of Cartesian coordinates are provided to predetermine a purported trajectory. As shown in Figure 4, the red line represents the original route, and the blue line represents the preset course, while the green line represents the realistic trajectory the victim car follows during the experiment. Our realistic hijacking successfully misleads the victim vehicle to take a risky S-bend, and we notice that the green



(a) Manipulating the ADS via the billboard to hit the pedestrian.



(b) Collision occurred after the closed-loop manipulation.

Figure 5: The scenario of pedestrian collision.

line in the figure is a little later than anticipated, which is caused by the fact that AA only operates on the current frame while the actual attacked frame occurs later.

Secondly, as depicted in Figure 5, we install a billboard with a pedestrian walking in front of the victim vehicle alongside the road. We want to continuously deceive the automobile into colliding with the pedestrian (note that the testing personnel manipulates the pedestrian during the experiment). As shown in the accompanied video, the victim vehicle is effectively hijacked to dynamically misdirected to follow and finally hit the moving pedestrian.

4 RELATED WORK

Abrecht et al. [1] classify all ADSs testing into four levels, i.e., directly test individual deep learning models (L1); hardware sensors to the scope of co-testing (L2); continue to consider external environmental factors on the basis of L2 (L3); and test ADS in the closed-loop environment (L4). Our testing platform achieves L4 testing capabilities.

AC3R [12] is the first approach for testing ADS via automatically reconstructed car crashes from accident reports. AC3R builds the scene from an empty world and ignores the construction of background entities, such as buildings, greenbelt, etc. It may not be realistic for testing. Moreover, it cannot process long reports.

Besides generating test scenarios from accident report text, Nguyen et al.[20] generated test scenarios from accident sketches, and Xinxin et al.[28] generated test scenarios from accident videos. In addition to generating 3D test scenarios, Holland et al.[15] and Goss et al.[14] also generated 2D plane scenarios from accident reports or data for testing the decision-and-control module of ADS.

Due to the complexity and unpredictability of autonomous driving, defining and finding critical scenarios becomes the key challenge [29]. Gladisch et al. [13] formulate the problem as a Search-based Testing problem. To optimize the search process, Klischat et al. [16] use evolutionary algorithms; Althoff et al. [2] reduce search space by analyzing the reachability of scenarios; SAMOTA [27] combines surrogate-assisted optimization and many-objective search to generate test scenarios effectively and efficiently; SALVO [21] reduces search space by generating test scenarios from existing maps. In SBST Tool Competition 2021 [23], competitors competed on how to search for road layouts that could lead to a vehicle accident. Gambi et al. [12] try to generate scenarios based on accident databases. Scenic [11] and Paracosm [19] provide languages and tools to define and build critical scenarios manually.

Adversarial examples of DNNs have been extensively studied in literature. Typical generation algorithms include CW [6], PGD [18], etc. Simple application of AEs for testing can be classified as L1

Type of accident location	Reports	Scenic codes	Scenarios	Safety	Risk	
					Light Risk	Serious Risk
Non-Crossroads	**11**	**46**	**230**	**153/66.52%**	**48/20.87%**	**29/12.61%**
V1&V2 in same direction	8	40	200	130/65.00%	43/21.50%	27/13.50%
V1&V2 in opposite directions	3	6	30	23/76.67%	5/16.67%	2/6.67%
Crossroads	**9**	**27**	**135**	**79/58.52%**	**31/22.96%**	**25/18.52%**
Total	**20**	**73**	**365**	**232/63.56%**	**79/21.64%**	**54/14.79%**

Table 1: Results of ADS in test scenarios generated based on accident reports.

testing. RP2 [10] and Shapeshifter [7] implement adversarial attacks by modifying the surface texture of objects instead of image pixels. Deepbillboard [30] and PhysGAN [17] shoot videos with billboards in the real world, which can be classified as L3 testing. Patel et al. [24] implement pseudo-L4 testing of ADS in a close-loop environment, but the AEs are crafted by modifying image pixels.

5 CONCLUSION

We present a new testing platform ADEPT for ADS, which conducts testing based on a virtual environment and provides numerous facilities. Distinguished features of ADEPT include deriving realistic scenarios from accident reports and crafting physically-meaningful closed-loop testing scenarios via adversarial examples of DNNs.

ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China (62025202, 62172199, 61872340, 62072309), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Fundamental Research Funds for the Central Universities. T. Chen is partially supported by an overseas grant from the State Key Laboratory of Novel Software Technology, Nanjing University (KFKT2022A03), and Birkbeck BEI School Project (EFFECT).

REFERENCES

[1] Stephanie Abrecht, Lydia Gauerhof, Christoph Gladisch, Konrad Groh, Christian Heinzemann, and Matthias Woehrle. 2021. Testing deep learning-based visual perception for automated driving. *TCPS* 5, 4 (2021), 1–28.

[2] Matthias Althoff and Sebastian Lutz. 2018. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium*. 1326–1333.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing Robust Adversarial Examples. In *ICML*.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy*. 39–57.

[7] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. 2018. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *ECML PKDD*. 52–68.

[8] R. Craig Coulter. 1992. Implementation of the Pure Pursuit Path Tracking Algorithm.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. 1–16.

[10] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *CVPR*. 1625–1634.

[11] Daniel J Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. 2020. Scenic: A language for scenario specification and data generation. (2020).

[12] Alessio Gambi, Tri Huynh, and Gordon Fraser. 2019. 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*. 257–267.

[13] Christoph Gladisch, Thomas Heinz, Christian Heinzemann, Jens Oehlerking, Anne von Vietinghoff, and Tim Pfitzer. 2019. Search-based testing in automated driving control applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 26–37.

[14] Quentin Goss, Yara AlRashidi, and Mustafa İlhan Akbaş. 2021. Generation of modular and measurable validation scenarios for autonomous vehicles using accident data. In *2021 IEEE Intelligent Vehicles Symposium*. 251–257.

[15] James C Holland and Arman Sargolzaei. 2020. Verification of autonomous vehicles: Scenario generation based on real world accidents. In *2020 SoutheastCon*, Vol. 2. 1–7.

[16] Moritz Klischat and Matthias Althoff. 2019. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium*. 2352–2358.

[17] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. 2020. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *CVPR*. 14254–14263.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[19] Rupak Majumdar, Aman Mathur, Marcus Pirron, Laura Stegner, and Damien Zufferey. 2019. Paracosm: A language and tool for testing autonomous driving systems. (2019).

[20] Vuong Nguyen, Alessio Gambi, Jasim Ahmed, and Gordon Fraser. 2022. CRISCE: Towards Generating Test Cases from Accident Sketches. In *2022 IEEE/ACM 44th ICSE-Companion*. 339–340.

[21] Vuong Nguyen, Stefan Huber, and Alessio Gambi. 2021. SALVO: Automated Generation of Diversified Tests for Self-driving Cars from Existing Maps. In *2021 IEEE International Conference on Artificial Intelligence Testing*. 128–135.

[22] NHTSA. 2016. NHTSA Crash Viewer. http://crashviewer.nhtsa.dot.gov/, Last accessed on 2022-5-24.

[23] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. Sbst tool competition 2021. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*. 20–27.

[24] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. 2019. Adaptive adversarial videos on roadside billboards: Dynamically modifying trajectories of autonomous vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5916–5921.

[25] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*. Springer, 621–635.

[26] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. 2020. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *CVPR*. 7153–7162.

[27] Fitash Ul Haq, Donghwan Shin, and Lionel Briand. 2022. Efficient Online Testing for DNN-Enabled Systems using Surrogate-Assisted and Many-Objective Optimization. In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*.

[28] Zhang Xinxin, Li Fei, and Wu Xiangbin. 2020. CSG: Critical scenario generation from real traffic accidents. In *2020 IEEE Intelligent Vehicles Symposium*. 1330–1336.

[29] 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. 2022. Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping Study. *IEEE TSE* (2022).

[30] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2020. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 347–358.