Diversity-Guided Genetic Algorithm for Safety-Critical Scenario Generation in Autonomous Driving Testing

Hua Qi
Kyushu University, The University of Tokyo
qi-hua@g.ecc.u-tokyo.ac.jp

Siyuan Chen
The Univeristy of Tokyo
alsachai1234@outlook.com

Abstract

As Autonomous Driving Systems (ADSs) rapidly advance and deploy in real-world settings, ensuring their safety and reliability has become paramount. While real-world testing provides comprehensive evaluation, it is prohibitively expensive and time-consuming. Simulation-based testing offers a practical alternative, with search-based methods demonstrating significant potential for efficiently identifying safety-critical scenarios. However, existing search-based approaches often prioritize efficiency over diversity, resulting in limited scenario coverage that may miss critical edge cases. To address this limitation, we propose a diversity-guided genetic algorithm for critical scenario generation. Our method introduces a diversity score to quantify scenario dissimilarity and computes the diversity rate between current scenarios and recent populations to guide the search process. We implement and evaluate our approach on the CARLA simulator across three distinct driving scenarios, comparing it against two baseline methods. Experimental results demonstrate that our method effectively and efficiently generates diverse safety-critical scenarios.

1 Introduction

With the rapid advancement of artificial intelligence in recent years, numerous companies, particularly automotive manufacturers, have begun developing autonomous driving technology and deploying their own self-driving vehicles. This has led to the development of diverse Autonomous Driving Systems (ADSs), ranging from traditional rule-based systems to cutting-edge end-to-end architectures. More recently, the emergence of Large Language Models (LLMs) has inspired a new generation of ADSs that integrate LLMs into the decision-making process. Notable examples include Vision-Language-Action-based (VLA-based) ADSs, which leverage LLMs to enhance autonomous vehicle control.

Conversely, the rapid deployment of ADSs has coincided with an increasing number of autonomous vehicle accidents spanning multiple platforms and manufacturers. Despite substantial technological progress, autonomous driving remains constrained to Level 3, even though the industry recognizes six SAE Levels of Automation. Advancing to Level 4 presents substantial challenges, with numerous critical issues requiring resolution. Among these, ensuring ADS reliability and safety stands out as a paramount concern that cannot be overlooked.

Given that ADSs are safety-critical systems, comprehensive testing and evaluation of their safety and reliability are essential. Since ADSs must operate on physical vehicles, many companies conduct real-world testing by deploying systems on actual vehicles driven in real-world conditions. While this approach provides comprehensive evaluation, it is both time-consuming and costly. Moreover,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Embodied and Safe-Assured Robotic Systems.

safety considerations typically require a human safety driver to monitor operations and intervene in emergency situations.

To address these limitations, simulation-based testing has gained significant traction. Simulation platforms allow engineers to construct controlled environments, generate a wide range of test scenarios, and assess ADS performance under high-risk conditions without the need for physical vehicles or human operators. Recent research has yielded numerous simulation-based testing methodologies aimed at improving ADS reliability and safety [1, 9, 5].

Among simulation-based testing methods, some approaches attempt to replicate real-world driving environments in simulators. For instance, Yan *et al.* [18] transform naturalistic driving environments into virtual simulation scenarios. While this approach maintains scenario realism, collecting naturalistic driving data is resource-intensive due to its limited availability. Other approaches construct pre-crash scenarios from crash databases [13, 14]. However, these pre-defined scenarios remain insufficient for comprehensive ADS testing.

To efficiently and effectively generate test scenarios for ADSs, researchers have proposed search-based approaches. Search-based methods demonstrate significant potential for identifying critical scenarios within vast input spaces. These approaches employ search algorithms, such as genetic algorithms, to explore diverse critical scenarios. Some methods [11, 20] also incorporate fuzzing techniques to refine specific hazardous scenarios, enabling more precise detection of collision scenarios that expose safety violations in ADSs. In search-based methods, the search strategy plays a crucial role, as it directly impacts both the efficiency and effectiveness of critical scenario discovery. However, while these methods prioritize efficiency and effectiveness, they often overlook the diversity of critical scenarios.

In this paper, we propose a novel critical scenario generation method based on a diversity-guided genetic algorithm. Our method quantifies scenario dissimilarity through a diversity score and computes the diversity rate between the current scenario and recent populations to guide the genetic algorithm. By maximizing the diversity rate, our method can effectively and efficiently identify critical scenarios.

We implement our approach on the CARLA simulator [3], a widely recognized platform for ADS testing, using its built-in Autopilot system as the ADS under test to control the ego vehicle. To evaluate the effectiveness of our method, we conduct experiments across three distinct driving scenarios and perform comparative analysis against two baseline approaches. The experimental results confirm that our method achieves both effectiveness and efficiency in generating diverse safety-critical scenarios capable of inducing ADS collisions.

To summarize, this paper makes the following contributions:

- We propose a diversity metric and integrate it into the genetic algorithm to guide scenario generation. This approach effectively addresses the common limitation where generated scenarios lack sufficient diversity.
- We develop a diversity-based scenario generation method that efficiently and effectively evolves existing scenarios into diverse safety-critical scenarios.

2 Method

2.1 Overview

We design an efficient method that generates diverse, critical test scenarios to expose safety violations in ADSs. Fig. 1 illustrates the workflow of our method.

A scenario comprises an ego vehicle, controlled by the target ADS, and additional agents (non-player characters, NPCs) whose presence can affect the ego vehicle, such as other vehicles and pedestrians. In this work, we focus on scenarios containing one NPC vehicle. We formally define a simulation scenario as $S = (P_{\text{ego}}, P_{\text{npc}})$, where P_{ego} represents the configuration parameters of the ego vehicle and P_{npc} represents the configuration parameters of NPC vehicles. The configuration of the ego vehicle is specified as $P_{\text{ego}} = (s_{\text{ego}}, t_{\text{ego}}, v_{\text{ego}})$, where s_{ego} denotes the initial position and t_{ego} denotes the target destination of the ego vehicle. v_{ego} denotes the initial velocity of the ego vehicle. The NPC vehicle configuration is defined as $P_{\text{npc}} = (r_{\text{npc}}, v_{\text{npc}}, \mathcal{A})$, where r_{npc} represents the prescribed

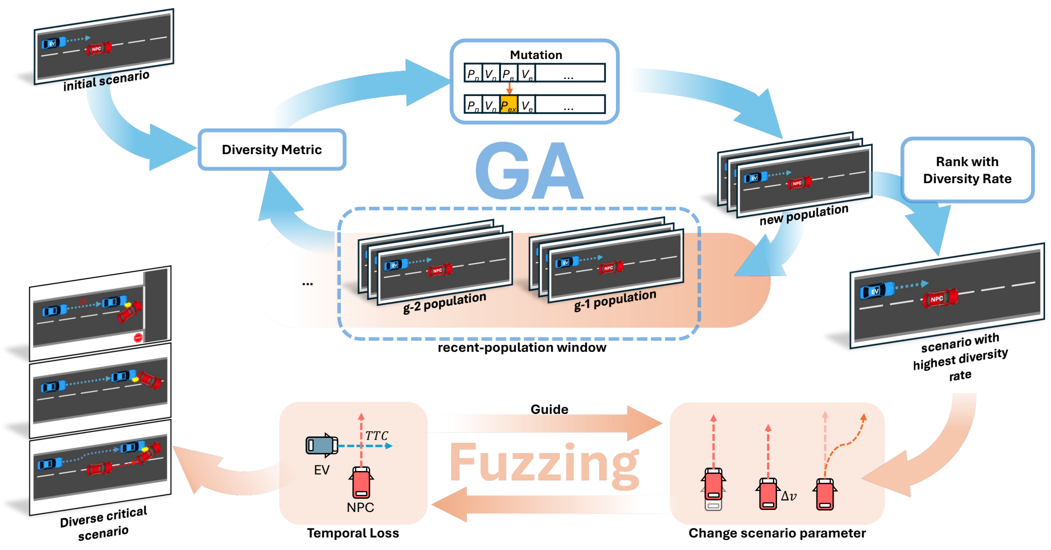

Figure 1: The workflow of our method.

route trajectory that the NPC vehicle must traverse, v_{npc} represents the initial velocity of NPC vehicle, and $\mathcal{A}$ denotes a temporally-ordered sequence of control actions to be executed by the NPC vehicle. Formally, the action sequence is expressed as: $\mathcal{A} = \{(a_{t_1}, t_1), (a_{t_2}, t_2), \ldots, (a_{t_m}, t_m)\}$ where $a_{t_i} \in \mathcal{O}$ represents the action to be executed at timestamp t_i. Each action a is selected from a discrete action set $\mathcal{O}$. In this work, we define $\mathcal{O} = \{\texttt{None}, \texttt{ACC}, \texttt{DEC}, \texttt{LANE}\}$, which comprises four primitive maneuvers: `None` (maintain current state), `ACC` (longitudinal acceleration), `DEC` (longitudinal deceleration) and `LANE` (lateral lane change). Detailed specifications and parameterizations for these actions are provided in Table 1.

Table 1: The details of the actions set $\mathcal{O}$.

Action	**Description**
`None`	The NPC vehicle remains driving states and does nothing.
`ACC`	The NPC vehicle accelerates for a short duration of 10 simulation ticks.
`DEC`	The NPC vehicle decelerates for a short duration of 10 simulation ticks.
`LANE`	The NPC vehicle switches to a different available lane.

We initialize the scenario as $S_0 = (P^0_{\text{ego}}, P^0_{\text{npc}})$, $P^0_{\text{ego}} = (s^0_{\text{ego}}, t^0_{\text{ego}}, v^0_{\text{ego}})$, and $P^0_{\text{npc}} = (r^0_{\text{npc}}, v^0_{\text{npc}}, \varnothing)$ where the initial action sequence of NPC vehicle $\mathcal{A}^0 = \varnothing$ as the NPC vehicle does not apply any actions in the initial run and the simulator is executed using S_0. Next, we employ a genetic algorithm to evolve a new generation of scenarios and to compute a diversity metric among them, which is used to rank the scenarios. A fuzzing module then explores the high-ranking, diverse scenarios to identify critical scenarios under the guidance of a temporal loss function.

2.2 Diversity metric

We define a scenario S as $S = (P_{\text{ego}}, P_{\text{npc}})$, which can be decomposed into $S = (\mathbb{P}, \mathbb{S})$, where $\mathbb{P} = (s_{\text{ego}}, t_{\text{ego}}, v_{\text{ego}}, v_{\text{npc}})$ represents the **parameter vector** encoding the initial configuration of S, and $\mathbb{S} = (r_{\text{npc}}, \mathcal{A})$ represents the **sequence components** of S.

The dissimilarity between any two scenarios S_a and S_b is quantified by computing the distance between their respective parameters and sequences. The **parameter distance** is defined as the Euclidean norm of their parameter vectors:

$$\mathcal{D}_{\text{param}}(S_a, S_b) = |\mathbb{P}_a - \mathbb{P}_b|^2 \tag{1}$$

while the **sequence distance** is computed based on the topological structures of their NPC routes:

$$\mathcal{D}_{\text{seq}}(S_a, S_b) = \text{Diff}(\mathcal{T}(r_a), \mathcal{T}(r_b)) \tag{2}$$

where $\mathcal{T}$ represents the function to calculate the topological representations of NPC route and $\text{Diff}(\cdot)$ measures the structural dissimilarity between two topological representations. In this work we use symmetrized segment-path distance (SSPD) [2] as $\text{Diff}(\cdot)$. By operating on route topologies rather than raw trajectories, we significantly reduce computational complexity while preserving essential structural information.

Then we can calculate the dissimilarity between any two scenario:

$$\mathcal{D}(S_a, S_b) = \mathcal{D}_{\text{param}}(S_a, S_b) + \mathcal{D}_{\text{seq}}(S_a, S_b) \tag{3}$$

and the diversity score of any scenario S_i can be defined as the dissimilarity between this scenario S_i and the initial scenario S_0:

$$\mathcal{D}_{S_i} = \mathcal{D}(S_i, S_0) \tag{4}$$

2.3 Diversity-guided genetic algorithm

We employ a diversity-guided genetic algorithm (GA) to generate a diverse set of scenarios. The algorithm is initialized with initial scenario S_0, which constitutes the initial population $\mathcal{P}^0 = \{S_0\}$. For each generation g, the current population $\mathcal{P}^g$ serves as the parent population. We can calculate the diversity scores of all scenarios in the g-th population:

$$\mathcal{D}_{\mathcal{P}^g} = \{\mathcal{D}_{S_0^g}, \mathcal{D}_{S_1^g}, \dots, \mathcal{D}_{S_n^g}\} \tag{5}$$

where $S_0^g, S_1^g, \dots, S_n^g \in \mathcal{P}^g$ and n is the length of g-th population. For scenario $S_i \in \mathcal{P}^g$, we calculate its diversity rate:

$$\mathcal{R}_{S_i} = |\mathcal{D}_{S_i} - avg(\mathcal{D}_{\mathcal{P}^{(g-1)}}, \mathcal{D}_{\mathcal{P}^{(g-2)}}, \dots, \mathcal{D}_{\mathcal{P}^{(g-w)}})| \tag{6}$$

where w represents the length of the recent-population window. The diversity rate for a current scenario is defined by the deviation of its diversity score from the average diversity across the last w populations. Specifically, a larger diversity rate indicates greater difference relative to recent history. In this work, the recent-population window length is set to 2. Scenarios can then be ranked by their diversity rate to select the most diverse candidates for propagation to the next population. Consequently, scenarios with higher diversity rates have a higher probability of being chosen to generate the next population.

To enhance scenario diversity in the evolutionary process, each scenario selected from the parent population undergoes mutation, where one or more parameters are randomly perturbed to produce the next generation.

2.4 Fuzzing for critical scenario searching

Each scenario generated by the genetic algorithm is executed in the simulator to calculate its temporal loss. In this work, we use time-to-collision (TTC) as the metric for computing temporal loss. Based on the calculated temporal loss, the fuzzing module perturbs the scenario using three methods: ❶ **Change Position** modifies the initial position of the ego vehicle or NPC vehicles, ❷ **Change Velocity** adjusts the initial velocity of the ego vehicle or NPC vehicles, and ❸ **Change Action** alters the action sequence of NPC vehicles.

In the fuzzing process, **Change Position** and **Change Velocity** introduce random perturbations to the initial positions and velocities of both ego and NPC vehicles:

$$\begin{aligned} s_{\text{ego}}^{i+1} &= s_{\text{ego}}^{i} + \Delta s_{\text{ego}} \\ s_{\text{npc}}^{i+1} &= s_{\text{npc}}^{i} + \Delta s_{\text{npc}} \\ v_{\text{ego}}^{i+1} &= v_{\text{ego}}^{i} + \Delta v_{\text{ego}} \\ v_{\text{npc}}^{i+1} &= v_{\text{npc}}^{i} + \Delta v_{\text{npc}} \end{aligned} \tag{7}$$

where Δs_{ego} and Δs_{npc} denote the random perturbations to the initial positions of the ego and NPC vehicles, respectively, and Δv_{ego} and Δv_{npc} denote the perturbations to their initial velocities. **Change Action** modifies the action sequence of the scenario by either appending an additional action to the sequence or altering the last action.

In the i-th iteration, the fuzzing module applies all three methods to scenario S_i to generate a new scenario S_{i+1}, which is then executed in the simulator. After obtaining the simulation results and calculating the temporal loss, the fuzzing module evaluates the outcome: if the temporal loss of S_{i+1} decreases compared to S_i, indicating a more critical scenario, S_{i+1} is retained and becomes the basis for further fuzzing in the next iteration; conversely, if the temporal loss increases, S_{i+1} is discarded and the fuzzing process continues from S_i.

3 Results

3.1 Settings

3.1.1 Environment

All experiments were conducted on Ubuntu 22.04, equipped with an Intel i9-14900 CPU, an Nvidia RTX 4090 GPU, and 32 GB of memory. We employed CARLA [3] version 0.9.15 as the simulation platform, utilizing its built-in Autopilot as the ADS under test.

3.1.2 Scenario

To comprehensively evaluate our method, we selected three different scenarios from official maps provided by CARLA [3]: ❶ **Scenario (a)** is a straight road near spawn point No.54 in the "Town01" map; ❷ **Scenario (b)** is a T-junction near spawn point No.94 in the "Town01" map; and ❸ **Scenario (c)** is a cross junction near spawn point No.204 in the "Town05" map.

For other scenario parameters, each simulation tick in the simulator is set to 0.05 seconds. The speed limit is set to 60 km/h, and the duration of the actions `ACC` and `DEC` is set to 10 simulation ticks (0.5 seconds).

3.2 Experiment Design

To comprehensively evaluate the performance of our method, we compare it with two baseline methods: random sampling and conflict-based generation [15].

We execute our method and two baseline methods across three scenarios, each for 10,000 iterations. The initial parameters (e.g., positions of the ego vehicle and NPC vehicle) remain the same when testing within the same scenario. To ensure adequate reaction time, the ego vehicle is positioned at least 10 meters away from the NPC vehicle at initialization. To enhance realism, the NPC vehicle is controlled by CARLA's Autopilot to simulate human driving behavior, with manual overrides applied when specific actions are required. When counting collisions, we only attribute collisions to the ego vehicle by analyzing the accelerations of both vehicles at the moment of impact: if the ego vehicle's acceleration exceeds that of the NPC vehicle, the collision is considered to be caused by the ego vehicle. To reduce the impact of randomness, each experiment is repeated three times and the results are averaged.

3.3 RQ1 (Effectiveness): Does our method generate more safety-critical scenarios than existing methods?

To evaluate the effectiveness of our method, we compare its performance with two baseline methods (*i.e.* random sampling and conflict-based generation) across three scenarios. The results are presented in Table 2:

Table 2: Collision Counts for Three Methods Across Three Scenarios

Method	**#Collision**		
	Scenario (a)	Scenario (b)	Scenario (c)
Random	3	37	45
Conflict-based [15]	15	109	204
Our method	**17**	**155**	**306**

The results demonstrate that our method discovered the highest number of collision scenarios across all three test cases. In **Scenario (a)**, our method generated 17 collisions, substantially outperforming Random Sampling. In **Scenarios (b) and (c)**, our method identified 155 and 306 collision scenarios, respectively, significantly exceeding the collision counts of both baseline methods. In contrast, Random Sampling achieved the lowest collision counts across all three scenarios. These findings indicate that our method is more effective at generating safety-critical scenarios than the two baseline approaches.

3.4 RQ2 (Diversity): Does our method generate more diverse safety-critical scenarios than existing methods?

RQ1 demonstrates that our method generates more critical scenarios than the two baseline methods, while the diversity of these scenarios remains to be evaluated. Following the approach in MOSAT [17], we measure diversity using Euclidean distance between critical scenarios. Specifically, we calculate the Euclidean distance between the sequences of NPC vehicle states, including position and controller states (*i.e.*, throttle, brake, and steering), to quantify the dissimilarity between two scenarios. The average Euclidean distances are presented in Table 3:

Table 3: Average Euclidean distance of generated safety-critical scenarios

Method	**Avg. Euclidean Distance**		
	Scenario (a)	Scenario (b)	Scenario (c)
Random	**37.685**	71.368	94.077
Conflict-based [15]	16.724	56.422	86.016
Our method	24.377	**78.205**	**122.497**

The results show that Random Sampling achieved the highest average Euclidean distance in **Scenario (a)**, while our method achieved the highest average Euclidean distance in **Scenarios (b) and (c)**. This indicates that our method generates the most diverse critical scenarios in **Scenarios (b) and (c)**. Although Random Sampling produced more diverse scenarios in **Scenario (a)**, it only generated 3 critical scenarios total—substantially fewer than the other methods—limiting the significance of this result.

3.5 RQ3 (Efficiency): Does our method generate safety-critical scenarios more efficiently than existing methods?

RQ1 demonstrates that our method generates more safety-critical scenarios than existing methods. However, does this improvement come at the cost of increased computational time? To assess the efficiency of our method, we calculated the average execution time per collision scenario. The results are presented in Table 4:

Table 4: Average Execution Time (hours per collision)

Method	**Avg. Execution time (hours/collision)**		
	Scenario (a)	Scenario (b)	Scenario (c)
Random	2.845	0.234	0.193
Conflict-based [15]	0.504	0.071	0.036
Our method	**0.438**	**0.049**	**0.025**

The results indicate that our method achieves lower average execution time per collision scenario than the two baseline methods across all three scenarios. This demonstrates that our method is more efficient than both Random Sampling and the conflict-based approach in generating safety-critical scenarios.

4 Discussion

4.1 Threats to Validity

In Section 2.2, we use SSPD to calculate the difference between NPC routes as it is suitable for most situations. However, our method is not restricted to SSPD; other distance metrics can also be used, such as Hausdorff distance and Fréchet distance, which may perform better in certain cases.

In Section 2.2, our method uses topological structures of NPC routes to calculate dissimilarity. This reduces computational cost but introduces some distortion in representing NPC routes, which may affect accuracy. While we consider this trade-off acceptable for our purposes, applications requiring higher precision may benefit from using raw NPC routes instead of their topological structures.

In this work, while we implement our method on the CARLA [3] simulator using its built-in Autopilot, our approach is platform-independent and can be readily applied to other simulators and ADSs. We plan to implement our method on other simulators and evaluate additional ADSs.

5 Related Work

5.1 Scenario-based testing

Scenario-based testing is widely employed for ADS testing and evaluation. This methodology treats the ADS as a black box and conducts system-level evaluation. Early research leveraged formal methods [4, 6, 16, 19] to discover critical scenarios and detect system faults. For instance, Fremont *et al.* [6] applied formal methods to model system behavior and specify scenarios for system analysis. Other work has utilized metamorphic testing approaches. Han *et al.* [7] employed metamorphic fuzz testing to generate unrealistic scenarios for robustness evaluation and failure identification. While these methods demonstrate effectiveness, they incur high computational costs for sampling and optimization and often generate redundant scenarios.

Unlike these early approaches, search-based scenario testing methods [8, 10, 11, 12, 17, 20] have demonstrated both effectiveness and efficiency in critical scenario generation. AV-Fuzzer [11], which employed genetic algorithms to identify critical scenarios and fuzzing to detect ego-vehicle-caused collisions; DriveFuzz [10], which introduced traffic-rule-based test oracles to identify ADS misbehavior; and MOSAT [17], which applied multi-objective genetic algorithms to generate scenarios exposing ADS safety violations.

In search-based methods, the search strategy is critical for efficient and effective discovery of critical scenarios. However, existing methods focus primarily on efficiency and effectiveness while neglecting scenario diversity. In this work, we demonstrate that incorporating diversity metrics into genetic algorithms significantly improves the diversity of discovered critical scenarios.

6 Conclusion

In this paper, we propose a diversity-guided genetic algorithm for safety-critical scenario generation. Our method introduces a diversity metric to quantify scenario dissimilarity and leverages a diversity rate—computed between current scenarios and recent populations—to guide the genetic algorithm's search process. This improvement ensures the generation of both critical and diverse test scenarios. We evaluate our method on the CARLA simulator, using the built-in Autopilot system as the ADS under test. Experiments across three distinct driving scenarios compare our approach against two baseline methods. Results demonstrate that our diversity-guided approach effectively and efficiently generates diverse safety-critical scenarios, significantly improving test coverage and exposing various ADS collision vulnerabilities.

References

[1] Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Goiuria Sagardui, and Leire Etxeberria. Search-based test case generation for cyber-physical systems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 688–697, 2017.

[2] Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, 2016.

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

[4] Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. Verifai: A toolkit for the formal design and

analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*, pages 432–442. Springer, 2019.

[5] Shuo Feng, Yiheng Feng, Haowei Sun, Yi Zhang, and Henry X. Liu. Testing scenario library generation for connected and automated vehicles: An adaptive framework. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1213–1222, 2022.

[6] Daniel J Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[7] Jia Cheng Han and Zhi Quan Zhou. Metamorphic fuzz testing of autonomous vehicles. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 380–385, 2020.

[8] Yuqi Huai, Yuntianyi Chen, Sumaya Almanee, Tuan Ngo, Xiang Liao, Ziwen Wan, Qi Alfred Chen, and Joshua Garcia. Doppelgänger test generation for revealing bugs in autonomous driving software. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2591–2603. IEEE, 2023.

[9] Baekgyu Kim, Takato Masuda, and Shinichi Shiraishi. Test specification and generation for connected and autonomous vehicle in virtual environments. *ACM Trans. Cyber-Phys. Syst.*, 4(1), November 2019.

[10] Seulbae Kim, Major Liu, Junghwan" John" Rhee, Yuseok Jeon, Yonghwi Kwon, and Chung Hwan Kim. Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1753–1767, 2022.

[11] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st international symposium on software reliability engineering (ISSRE)*, pages 25–36. IEEE, 2020.

[12] Chengjie Lu, Huihui Zhang, Tao Yue, and Shaukat Ali. Search-based selection and prioritization of test scenarios for autonomous driving systems. In *International Symposium on Search Based Software Engineering*, pages 41–55. Springer, 2021.

[13] Wassim G. Najm, Raja Ranganathan, Gowrishankar Srinivasan, John D Smith, Samuel Toma, Elizabeth Swanson, and August Burgett. Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. 2013.

[14] Wassim G. Najm, Samuel Toma, and John N. Brewer. Depiction of priority light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. 2013.

[15] Hua Qi, Siyuan Chen, Fuyuan Zhang, Tomoyuki Tsuchiyak, Michio Hayashi, Manabu Okada, Lei Ma, and Jianjun Zhao. Conflict-based Scenario Generation for Autonomous Driving System Validation . In *2025 IEEE/ACM 1st International Workshop on Software Engineering for Autonomous Driving Systems (SE4ADS)*, pages 5–11, Los Alamitos, CA, USA, April 2025. IEEE Computer Society.

[16] Yang Sun, Christopher M Poskitt, Jun Sun, Yuqi Chen, and Zijiang Yang. Lawbreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[17] Haoxiang Tian, Yan Jiang, Guoquan Wu, Jiren Yan, Jun Wei, Wei Chen, Shuo Li, and Dan Ye. Mosat: finding safety violations of autonomous driving systems using multi-objective genetic algorithm. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 94–106, 2022.

[18] Xintao Yan, Shuo Feng, Haowei Sun, and Henry X. Liu. Distributionally consistent simulation of naturalistic driving environment for autonomous vehicle testing, 2022.

[19] Xiaodong Zhang, Wei Zhao, Yang Sun, Jun Sun, Yulong Shen, Xuewen Dong, and Zijiang Yang. Testing automated driving systems by breaking many laws efficiently. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 942–953, 2023.

[20] Ziyuan Zhong, Gail Kaiser, and Baishakhi Ray. Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles. *IEEE Transactions on Software Engineering*, 49(4):1860–1875, 2022.