# Systematic Testing of Autonomous Driving Systems Using Map Topology-Based Scenario Classification

Yun Tang†, Yuan Zhou†*, Tianwei Zhang†, Fenghua Wu†, Yang Liu†, Gang Wang‡
†Nanyang Technological University, Singapore 639798
Email: yun005@e.ntu.edu.sg, {y.zhou, tianwei.zhang, fenghua.wu, yangliu}@ntu.edu.sg
‡Alibaba DAMO Academy, Alibaba Group, China.
E-mail: wg134231@alibaba-inc.com
*Corresponding author

***Abstract*—Autonomous Driving Systems (ADSs), which replace humans to drive vehicles, are complex software systems deployed in autonomous vehicles (AVs). Since the execution of ADSs highly relies on maps, it is essential to perform global map-based testing for ADSs to guarantee their correctness and AVs' safety in different situations. Existing methods focus more on specific scenarios rather than global testing throughout the map. Testing on a global map is challenging since the complex lane connections in a map can generate enormous scenarios. In this work, we propose ATLAS, an approach to ADSs' collision avoidance testing using map topology-based scenario classification. The core insight of ATLAS is to generate diverse testing scenarios by classifying junction lanes according to their topology-based interaction patterns. First, ATLAS divides the junction lanes into different classes such that an ADS can execute similar collision avoidance maneuvers on the lanes in the same class. Second, for each class, ATLAS selects one junction lane to construct the testing scenario and generate test cases using a genetic algorithm. Finally, we implement and evaluate ATLAS on Baidu Apollo with the LGSVL simulator on the San Francisco map. Results show that ATLAS exposes *nine* types of real issues in Apollo 6.0 and reduces the number of junction lanes for testing by 98%.**

***Index Terms*—Autonomous Driving Systems, Collision Avoidance Testing, Scenario Classification**

## I. INTRODUCTION

Autonomous vehicles (AVs) will play an essential role in intelligent transportation systems to relieve traffic congestion and eliminate accidents, especially at junctions. Many companies have been devoting themselves to this domain, such as Google Waymo [1], Baidu Apollo [2], and Autoware [3]. In AVs, Autonomous Driving Systems (ADSs) are deployed to replace human drivers to provide high-level autonomy. To guarantee the safety and reliability of AVs, ADSs must be tested sufficiently to ensure their correctness.

Since on-road testing is risky, costly, and insufficient, developers carry out extensive testing on simulators. However, since the environments vary dramatically in different aspects, such as road networks, road participants, and traffic signs, the number of scenarios for ADS testing is infinite. Hence, researchers usually focus on the generation of critical scenarios, such as those causing collisions between the ego vehicle (i.e., the AV) and the NPC (non-player character) vehicles (i.e., other vehicles on the roads) [4]–[21]. These approaches usually consider different scenarios one by one, and only one specific kind of scenario is covered in each test. There are limited studies on systematic collision avoidance testing for ADSs concerning a map (e.g., the road map of a city) where the AVs operate [22], [23]. Road maps of modern cities are usually complex and consist of various intersections [24]. Different intersections manifest different scenarios and interactions between the ego and NPC vehicles, demanding different collision avoidance maneuvers by the ego vehicle. Intersections with different junction lanes also present a variety of scenarios due to different numbers of traffic participants. Hence, a map may contain various scenarios. Global testing on such a map with existing methods will be time-consuming.

In this paper, we propose an approach, ATLAS, to collision avoidance testing for ADSs using map topology-based scenario classification. Given a map, ATLAS first extracts the connection relations between one-way roads and junction lanes. Based on the connections, each junction lane is characterized by its road topology, i.e., the set of road pairs connected by its intersecting junction lanes. Such a road topology identifies the motion directions of NPC vehicles that may collide with the ego one. ATLAS then categorizes junction lanes with the same road topology into the same lane class. Thus, inspired by the traffic rules and common human-driving practices, the ego vehicle following the junction lanes in the same class is expected to perform similar collision avoidance maneuvers. Hence, by sampling only one lane from each class, ATLAS can create diverse scenarios at the minimum testing effort. There are different configurable parameters for each scenario, e.g., the relative speeds and distances between the ego and NPC vehicles. ATLAS applies the genetic algorithm (GA) to search for potential collision test cases.

We implement and evaluate ATLAS on Apollo running with the LGSVL simulator and San Francisco map. We use the ground truth perception data during the experiments to avoid perception uncertainties and focus on Apollo's motion part. ATLAS selects 13 out of 786 junction lanes for testing after map-topology-based classification, reducing the testing effort by 98%. ATLAS discovers nine types of issues in Apollo. The contributions of this paper are as follows:

- A junction lane classification approach is proposed, guaranteeing scenario diversity at the minimum testing effort.
- A GA-based black-box testing is proposed for collision avoidance testing in each junction lane class, which can generate collision test cases efficiently.

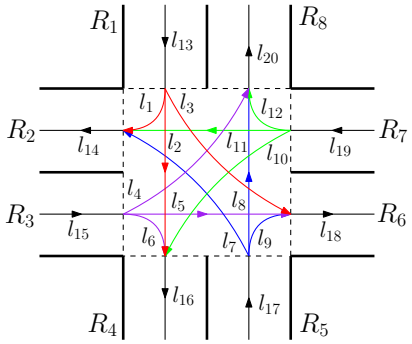


Fig. 1: Illustration of a junction $J_0$.

- Comprehensive evaluations on the Apollo ADS are conducted, and nine types of safety issues are discovered.

## II. Preliminaries and Problem Statement

Modern AVs operate with the guidance of High-Definition (HD) maps, which contain map geometry and semantic objects such as lane boundaries, intersections, crosswalks, parking spots, stop signs, and traffic lights [25]. A *lane* is the atomic geometry in a map. It is bounded by two boundaries and contains a reference path set as its centerline. Vehicles should move along the reference paths. We represent a lane by its reference path in the following discussion. A one-way *road* is a set of lanes, among which a vehicle can perform lane changing at any position. A *junction* contains a set of junction lanes that connect different roads. The ego vehicle cannot change lanes in a junction. Fig. 1 shows an example of a junction formed by eight roads. The bold black lines are the lane boundaries (also road boundaries in this case); the arrow lines are the lanes (the black ones are road lanes and the colored ones are junction lanes); $R_1$ - $R_8$ are eight roads connected by the junction, and each road contains one lane.

The problem we address can be described as follows: *Given an ADS and a map, our goal is to design an approach to evaluating the ADS' ability to avoid collisions on the map.*

We limit our discussion with the following assumptions. (1) An ADS can be roughly divided into the perception and the motion parts. This paper mainly focuses on the testing of the motion part. Testing of AI models in perception is orthogonal to our work. (2) The target ADS is a black box without disclosing the detailed designs, algorithms, or implementations. (3) Our discussion mainly focuses on systematic testing of collision avoidance with the NPC vehicles in junctions. However, ATLAS can extend to road testing on a map and collision avoidance with pedestrians.

## III. Methodology

### A. Junction Lane Classification

Given an HD map, let $l_i$ denote a lane in the map. A road $R_r$ containing $n_r$ lanes, say $l^r_1, \ldots, l^r_{n_r}$, is denoted as $R_r = l^r_1 \| \ldots \| l^r_{n_r}$, and the set of all roads is denoted as $R = \{R_1, R_2, \ldots, R_n\}$. A junction $J_j$ containing $n_j$ lanes is denoted as $J_j = \{l^j_1, \ldots, l^j_{n_j}\}$, and the set of all junctions and all junction lanes are denoted as $J = \{J_1, \ldots, J_m\}$ and $L^J = \bigcup_{j \in \mathbb{N}_m} J_j$, respectively, where $\mathbb{N}_m = \{1, 2, \ldots, m\}$. Given a junction lane $l^j$, $l^j = (l^{r_1}, l^{r_2})$ if $l^{r_1}$ and $l^{r_2}$ are connected directly by $l^j$, where $l^{r_1} \in R_{r_1}$ and $l^{r_2} \in R_{r_2}$; $R_{r_1}$ and $R_{r_2}$ are called the incoming and the outgoing roads of $l^j$, denoted as $I(l^j)$ and $O(l^j)$, respectively. The set of roads connected by the junction $J_j$ can be described as $\mathcal{R}^{\mathrm{I}}_j \cup \mathcal{R}^{\mathrm{O}}_j$, where $\mathcal{R}^{\mathrm{I}}_j = \bigcup_{l^j_i \in J_j} I(l^j_i)$ and $\mathcal{R}^{\mathrm{O}}_j = \bigcup_{l^j_i \in J_j} O(l^j_i)$ are called incoming and outgoing roads of $J_j$, respectively.

For an arbitrary junction lane $l \in J_j$, we introduce the $l$-index of the roads in $\mathcal{R}^{\mathrm{I}}_j \cup \mathcal{R}^{\mathrm{O}}_j$. First, all the roads are listed in a counter-clockwise order starting from $I(l)$: $C^j_l = (R_{r_1}, \ldots, R_{r_k})$, where $R_{r_1} = I(l)$ and $k = |\mathcal{R}^{\mathrm{I}}_j \cup \mathcal{R}^{\mathrm{O}}_j|$. Then, the $l$-index, denoted as $f^j_l$, can be determined as:

$$f^j_l(C^j_l(i)) = \begin{cases} i, & \text{if } C^j_l(i) = R_{r_i} \in R^{\mathrm{I}}_j. \\ -i, & \text{if } C^j_l(i) = R_{r_i} \in R^{\mathrm{O}}_j. \end{cases} \tag{1}$$

Indeed, the $l$-index describes the roads' relative orientations with respect to $I(l)$.

**Definition 1:** *Given a junction $J_j$ and two junction lanes $l^j_{i_1}, l^j_{i_2} \in J_j$, $l^j_{i_1}$ and $l^j_{i_2}$ are called in-junction intersecting if $l^j_{i_1}$ and $l^j_{i_2}$ are intersecting and with different start points. The set of intersecting junction lanes of $l^j_{i_1}$ is denoted as $IL(l^j_{i_1})$.*

**Definition 2:** *The road-topology characteristic of a junction lane $l$, denoted as $TC(l)$, is the set of incoming-outgoing road pairs of its intersecting lanes, i.e.,*

$$TC(l) = \bigcup\nolimits_{l^j_i \in IL(l)} \{[f^j_l(I(l^j_i)), f^j_l(O(l^j_i))]\}$$

For example, consider $l_2$ in Fig. 1. Since $I(l_2) = R_1$, we have $C^0_{l_2} = (R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8)$ and $f^0_{l_2} = (1, -2, 3, -4, 5, -6, 7, -8)$. The intersecting junction lanes of $l_2$ are $IL(l_2) = \{l_{11}, l_7, l_4, l_5, l_6, l_{10}\}$. Hence, $TC(l_2) = \{[7, -2], [5, -2], [3, -8], [3, -6], [3, -4], [7, -4]\}$.

The road-topology characteristic describes the relative directions from which the NPC vehicles may collide with the ego vehicle moving along the junction lane. It measures the complexity of scenarios for collision avoidance testing.

**Definition 3:** *Two junction lanes with the same road-topology characteristic are said to be topology equivalent.*

Based on this definition, we can divide all junction lanes on a map into a set of equivalence classes. In each equivalence class, the junction lanes have the same road-topology characteristic. It means if the ego vehicle drives along these lanes, it may collide with NPC vehicles that approach from the same relative directions. Thus, motivated by human driving behaviors, the ego vehicle is expected to perform similar maneuvers to avoid collisions. Hence, with a limited time budget, we can improve the diversity of the generated test cases by testing only one junction lane in each class.

### B. Test Case Generation via Genetic Algorithm

Each selected junction lane can construct a scenario from which we can generate test cases. Given a selected junction lane $l$, the scenario is constructed by parameterizing the initial position of the ego vehicle on the incoming road lane of

$l$ and the motion of the NPCs, each of which runs along an intersecting junction lane of $l$. Hence, test cases can be generated from the parameter space. Among many methods for test case generation, search-based methods are widely used [26], [27]. ATLAS uses GA for test case generation.

*Genetic Representation*. In our case, the solution domain is represented by an array of real numbers. Without loss of generality, we assume that each NPC vehicle moves with a constant speed since a variable motion can be segmented into a set of uniform motions for the testing purpose. Hence, the ego vehicle's safety is affected by the initial relative distances and speeds between the ego and the NPC vehicles. After fixing the start positions of NPC vehicles and the initial speed of the ego vehicle, each test can be parameterized by the initial position of the ego vehicle and the initial speeds of NPC vehicles. Hence, each individual in the solution domain can be described as a vector $(s, v_1, \ldots, v_{i_l})$, where $i_l$ is the number of junction lanes in $IL(l)$ based on Definition 1.

*Fitness Function*. In ATLAS, the fitness function for each test case $indv$ is defined as $f(indv) = 1/\min\{d_i : i = 1, \ldots, i_l\}$, where $d_i$ is the minimal distance between the ego and the $i$-th NPC vehicle during the entire motion process. If the ego vehicle collides with any NPC vehicle, the fitness value becomes infinity, and the GA stops.

*Selection, Crossover, and Mutation*. First, we use the 2-way tournament selection strategy [28] to select individuals from the population. Second, we adopt a fixed crossover rate (0.9 in our experiment) to select individual pairs for crossover. For each pair of crossover individuals, the two-point crossover is performed. Third, we implement a constant mutation rate (0.2 in our experiment) to select offspring for mutation, and due to its local convergence [29], the Gaussian mutation is applied to mutate each gene in an individual. Hence, for a selected individual $indv = (d, v_1, \ldots, v_{i_l})$, the mutation operator is:

$$g' = \min\{\max\{N(g, \delta), g_{\min}\}, g_{\max}\}, \forall g \in \{d, v_1, \ldots, v_{i_1}\},$$

where $g$ is the corresponding gene value in $indv$, $N(g, \delta)$ is a Gaussian distribution whose mean is $g$ and variance is $\delta$, $g_{\max}$ and $g_{\min}$ are the maximal and minimal values of the corresponding gene, and $\max$ and $\min$ functions are used to bound the mutated gene values.

## IV. EXPERIMENTS

To demonstrate the effectiveness and efficiency of ATLAS, we conduct evaluations on *Apollo*, one of the most popular ADSs in both the research community and industry. Two desktop computers (i7-6850K CPU with 64GB RAM, Xeon E5-2660 v4 CPU with 96GB RAM, and one GTX 1080 Ti GPU for either) are used to run the experiments in parallel.

We use the LGSVL simulator [30] to simulate real-world environment and vehicle dynamics, and Apollo 6.0 [31] as the system under testing. The global map is the San Francisco downtown area [32], which contains 84 signal-controlled junctions and 686 junction lanes with intersecting lanes. These lanes are divided into 34 classes. ATLAS selects 13 lane classes for testing, which subsume the remaining 21 classes. Since ATLAS is the first method for ADSs' systematic global map testing, we consider the random sampling approach the baseline for comparison with ATLAS. For GA, we set the population size as 20 and the maximal number of generations as 16. Thus, the maximal number of test cases per lane generated by either method is 320. Each experiment is repeated 16 times. The crossover rate and mutation rate are set heuristically to 0.9 and 0.2, respectively, based on preliminary experiments.

### A. Issues Discovered in Apollo

There are mainly three actions for collision avoidance: yielding (including stopping), following, and overtaking. We use ATLAS to successfully uncover nine types of issues in Apollo, which could cause collisions, traffic congestion, and deadlocks. These are never discussed in previous works.

1) *Soft braking in emergencies*: In some emergent situations, the ego vehicle brakes too gently and collides with NPC vehicles. They can be avoided with more braking power.
2) *Wrong overtaking*: Apollo may fail to overtake and thus collide with slow NPC vehicles. In such cases, it is safe for the ego vehicle to yield instead of trying to overtake.
3) *Wrong intersecting trajectory*: Sometimes, Apollo produces an incorrect trajectory that intersects with an NPC's driving path, resulting in an inevitable collision.
4) *Deadlock*: Apollo's aggressive path planning will block the motion of NPC vehicles; The stopped NPC vehicles, in turn, will block the motion of the ego vehicle. This leads to a deadlock.
5) *Wrong prediction*: Apollo may make incorrect motion predictions of the crossing NPC vehicles nearby, which significantly affects the generation of collision-free trajectory and eventually causes collisions.
6) *Planning delay*: Apollo sometimes suffers from planning delays, i.e., it cannot update real-time trajectory in time. Hence, the ego vehicle will follow the old trajectory, resulting in a collision.
7) *Unidentified traffic accidents*: When a traffic accident happens at the junction, Apollo cannot recognize it and will stop the ego vehicle inside the junction forever and block other vehicles' motion.
8) *Wrong switch from yielding to overtaking*: Sometimes Apollo is too aggressive in switching from yielding to overtaking before the crossing NPC vehicles pass through the collision regions.
9) *Prediction delay*: Apollo may experience a delay of obstacle predictions at runtime. In this case, it cannot make collision-free decisions and eventually cause the vehicle to collide with NPC vehicles.

We also discover one simulator-related issue; However, it is out of the scope of this paper and will not be discussed here. Detailed explanations and videos of all these issues can be found on https://av-testing.github.io/atlas/.

### B. Justification of Lane Classification

We analyze the rationality of lane classification based on the distribution of discovered issues in different junction lane

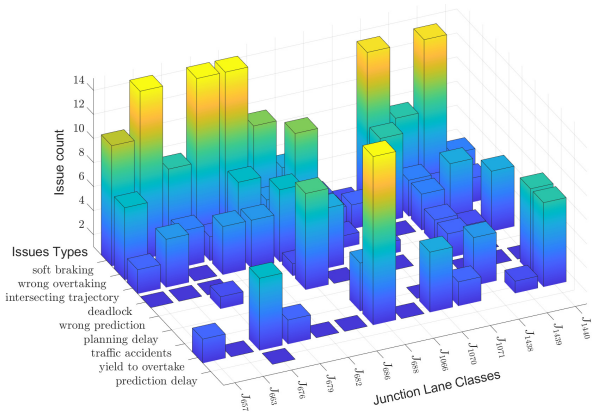


Fig. 2: Issues discovered in selected junction lane classes.

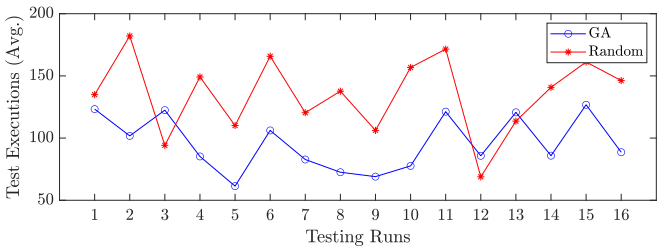


Fig. 3: Average number of test cases to trigger an issue.

classes, as shown in Fig. 2. We find that even though some issues (e.g., *soft braking* and *wrong overtaking*) are common in various classes, each junction lane class shows its unique distribution of discovered issues. For example, the issue of *wrong yielding to overtaking* is common in $\mathcal{J}_{688}$ and $\mathcal{J}_{1440}$, while the issue of *wrong prediction* occurs frequently in $\mathcal{J}_{686}$. Moreover, although we distinguish the situations that NPC vehicles come from the opposite relative directions (e.g., $\mathcal{J}_{657}$ and $\mathcal{J}_{663}$) due to different rights of way, Apollo does not identify such different situations during its collision avoidance. From Fig. 2, we observe that the distributions of discovered issues in $\mathcal{J}_{657}$ and $\mathcal{J}_{663}$ are similar.

In conclusion, ATLAS can distinguish the most representative intersection patterns, discover diverse types of issues, and assist in debugging and improving function.

### C. *Efficiency of* ATLAS

This section demonstrates the efficiency of ATLAS from different perspectives.

First, we evaluate the efficiency of the GA method. We record the average number of generated test cases to trigger an issue in each run during our experiments. The results are shown in Fig. 3. For all the 16 runs, the average number and standard deviation of the test cases executed by the GA method are 95.7 and 21.82, respectively. In contrast, the random sampling approach requires an average of 134.96 test cases with a standard deviation of 30.76. To further demonstrate the significance, we perform Levene test for equal variances and two-sample t-test for equal means [33], as shown in Table I. We observe a significant difference between the means of GA and random sampling. Hence, we conclude that the GA

TABLE I: Significance Testing between the two approaches

| | mean | std | Levene Test for Equal Variances | Two-Sample t-Test for Equal Means |
|---|---|---|---|---|
| GA | 95.70 | 21.82 | p-value: 0.274 | p-value: 2.429e-04 |
| Random | 134.96 | 30.76 | | |

method can reduce the number of test cases by 29.1% to discover an issue on average.

Next, we consider the time cost to perform testing on a global map. Based on our experiment, one test case takes about one minute. As Table I shows, the average number of executed tests with GA is 95.7. So the average testing time per junction lane is about 95.70 minutes, and the total testing time for all 686 valid junction lanes on the San Francisco map is about $95.7 * 686/60/24 \approx 45.59$ days. If we repeat the experiment 16 times, the total time is almost 730 days, which is computationally infeasible. On the other hand, by performing lane classification, ATLAS samples 13 lanes per run, reducing the testing effort by 98%. Thus, ATLAS offers a huge efficiency boost with lane classification.

## V. DISCUSSION AND CONCLUSION

**Discussion.** ADSs are complex and domain-specific software systems. On the one hand, regarding an ADS as a general software system, we can apply conventional software testing technologies (e.g., unit testing, fuzzing testing, and regression testing) and coverage criteria (function coverage, statement coverage, branch coverage, condition coverage, and line coverage). On the other hand, as a domain-specific software system, an ADS shows its own characteristics. For example, an ADS is implemented with various motion planning algorithms, so testing these algorithms needs domain-related methods to design diverse scenarios. Thus, in ADS testing, scenarios play an essential role in validating the correctness of ADSs. Combining scenario-related coverage and code-related coverage is a promising direction for ADS testing. This work makes a first attempt for ADS testing from scenario-related coverage.

**Conclusion.** In this paper, we propose ATLAS, a novel method for efficient ADS testing on a global map. It first generates a set of map topology-based scenarios for testing via junction lane classification and selection. For each selected lane, ATLAS parameterizes the scenario with the initial and target positions of the ego vehicle on the lane and the initial speeds of the NPCs moving along the lane's intersecting junction lanes. Finally, a genetic algorithm is adopted to generate test cases. The effectiveness and efficiency of ATLAS are validated via global testing of Apollo on a map.


## ACKNOWLEDGMENTS

This work was supported by Singapore MOE Academic Research Fund Tier 2 grant (MOE-T2EP20120-0004), Singapore Ministry of Education (MOE) AcRF Tier 1 RG108/19 (S), Singapore National Research Foundation (NRF) under its National Cybersecurity R&D Program (NRF2018NCR-NCR005-0001 and NRF2018NCR-NSOE003-0001), NRF Investigatorship (NRF-NRFI06-2020-0001), and Alibaba Group through Alibaba Innovative Research Program and Alibaba-NTU Joint Research Institute.

## REFERENCES


[1] Waymo, "Waymo," https://waymo.com/, 2016, online; accessed April 2020.

[2] Baidu, "Apollo open platform," http://apollo.auto/.

[3] Autoware.AI, "Autoware.AI," www.autoware.ai/.

[4] S. K. Bashetty, H. B. Amor, and G. Fainekos, "Deepcrashtest: Turning dashcam videos into virtual crash tests for automated driving systems," in *2020 IEEE International Conference on Robotics and Automation, ICRA*, Paris, France, 2020, pp. 11 353–11 360.

[5] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 257–267.

[6] W. G. Najm, S. Toma, J. Brewer *et al.*, "Depiction of priority light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications," National Highway Traffic Safety Administration, U.S. Department of Transportation, Washington, DC, Tech. Rep. DOT HS 811 732, Apr. 2013.

[7] P. Nitsche, P. Thomas, R. Stuetz, and R. Welsh, "Pre-crash scenarios at road junctions: A clustering method for car crash data," *Accident Analysis & Prevention*, vol. 107, pp. 137–151, 2017.

[8] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner, "Did we test all scenarios for automated and autonomous driving systems?" in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2950–2955.

[9] W. Ding, M. Xu, and D. Zhao, "Cmts: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4314–4321.

[10] C. Roesener, F. Fahrenkrog, A. Uhlig, and L. Eckstein, "A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour," in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, 2016, pp. 1360–1365.

[11] J.-P. Paardekooper, S. Montfort, J. Manders, J. Goos, E. d. Gelder, O. Camp, O. Bracquemond, and G. Thiolon, "Automatic identification of critical scenarios in a public dataset of 6000 km of public-road driving," in *26th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, 2019.

[12] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, 2017.

[13] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1326–1333.

[14] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2352–2358.

[15] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[16] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018.

[17] G. E. Mullins, A. G. Dress, P. G. Stankiewicz, J. D. Appler, and S. K. Gupta, "Accelerated testing and evaluation of autonomous vehicles via imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–7.

[18] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence testing for autonomous vehicles: A new approach," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, 2016.

[19] S. Masuda, H. Nakamura, and K. Kajitani, "Rule-based searching for collision test cases of autonomous vehicles simulation," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1088–1095, 2018.

[20] A. Calò, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Generating avoidable collision scenarios for testing autonomous driving systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 375–386.

[21] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part I: Methodology," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[22] Y. Tang, Y. Zhou, F. Wu, Y. Liu, J. Sun, W. Huang, and G. Wang, "Route coverage testing for autonomous vehicles via map modeling," in *IEEE Int. Conf. Robot.d Autom.*, 2021.

[23] Y. Tang, Y. Zhou, Y. Liu, J. Sun, and G. Wang, "Collision avoidance testing for autonomous driving systems on complete maps," in *2021 IEEE Intelligent Vehicles Symposium (IV21)*, 2021.

[24] K. Czarnecki, "Operational world model ontology for automated driving systems–part 1: Road structure," *Waterloo Intelligent Systems Engineering Lab (WISE) Report, University of Waterloo*, 2018.

[25] K. Chellapilla, "Rethinking maps for self-driving," https://medium.com/lyftlevel5/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6, Oct. 2018.

[26] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 143–154.

[27] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[28] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection." in *Proceedings of the 6th International Conference on Genetic Algorithms*, vol. 95. Citeseer, 1995, pp. 9–15.

[29] K.-T. Lan and C.-H. Lan, "Notes on the distinction of gaussian and cauchy mutations," in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, vol. 1, 2008, pp. 272–277.

[30] LG Silicon Valley Lab, "Lgsvl simulator," https://github.com/lgsvl/simulator/releases/tag/2020.06, 2019, online; accessed Oct 2020.

[31] Baidu, "Apollo 6.0," https://github.com/ApolloAuto/apollo/releases/tag/v6.0.0, 2019, online; accessed Oct 2020.

[32] LG Electronics, "Map for San Francisco," https://content.lgsvlsimulator.com/maps/sanfrancisco/, 2020.

[33] N. A. Heckert and J. J. Filliben, "Nist/sematech e-handbook of statistical methods; chapter 1: Exploratory data analysis," 2003.