Diversity-guided Search Exploration for Self-driving Cars Test Generation through Frenet Space Encoding

Timo Blattner
timo.blattner@students.unibe.ch
University of Bern
Switzerland

Christian Birchler
christian.birchler@{zhaw,unibe}.ch
Zurich University of Applied Sciences
University of Bern
Switzerland

Timo Kehrer
timo.kehrer@unibe.ch
University of Bern
Switzerland

Sebastiano Panichella
sebastiano.panichella@zhaw.ch
Zurich University of Applied Sciences
Switzerland

ABSTRACT

The rise of self-driving cars (SDCs) presents important safety challenges to address in dynamic environments. While field testing is essential, current methods lack diversity in assessing critical SDC scenarios. Prior research introduced simulation-based testing for SDCs, with Frenetic, a test generation approach based on Frenet space encoding, achieving a relatively high percentage of valid tests (approximately 50%) characterized by naturally smooth curves. The "minimal out-of-bound distance" is often taken as a fitness function, which we argue to be a sub-optimal metric. Instead, we show that the likelihood of leading to an out-of-bound condition can be learned by the deep-learning vanilla transformer model. We combine this "inherently learned metric" with a genetic algorithm, which has been shown to produce a high diversity of tests. To validate our approach, we conducted a large-scale empirical evaluation on a dataset comprising over 1,174 simulated test cases created to challenge the SDCs behavior. Our investigation revealed that our approach demonstrates a substantial reduction in generating non-valid test cases, increased diversity, and high accuracy in identifying safety violations during SDC test execution.

1 INTRODUCTION

The increasing autonomy and widespread adoption of autonomous systems [11, 13] such as self-driving cars (SDCs) pose significant challenges regarding their operational safety in dynamic environments [2, 3, 10, 15]. While field testing is crucial to assess the behavior of autonomous systems in different conditions, there is still a big gap in testing SDC features in diverse and critical scenarios that faithfully replicate the complexity of real-world dynamics [6, 12].

Previous work proposed simulation-based testing environments in the context of SDCs to assess the lane-keeping ability of the vehicle, with important results in efficiently identifying test cases that effectively reveal faults of SDCs in different environmental conditions [3–5, 16]. Notably, the Frenetic test generation approach [14, 18], based on Frenet space encoding, emerged as one of the most successful approaches for generating a high percentage of valid test cases (approximately 50%) with naturally smooth curves. However, Frenetic still has two important limitations: (i) it generates a high percentage of invalid test cases (e.g., excessive curvature) [14, 18]; and (ii) it relies on the "out of bound distance", the distance between the center of the center line of the road, which we argue to be a sub-optimal fitness function: good drivers will always push to the outside first before turning in to reduce the centripetal acceleration and extend the corner. Consequently, the metric might show high fitness for a test where the driver actually drives correctly.

To address this gap, we propose an approach named FRENILLA, which employs a transformer model that has been trained to predict the likelihood of an out-of-bound condition at each point on the road. We combine this discriminator with a genetic algorithm to guide the search exploration toward more diverse and valid test cases while keeping a competitive fault detection ability. To validate our approach, we conducted a large-scale empirical study on a dataset comprising over 1,174 simulated test cases created to challenge the SDC's behavior, which allowed us to explore the relation between road shape compositions and SDC safety violations.

2 BACKGROUND AND RELATED WORK

The FRENILLA approach works based on the concept of Frenet frames. Hence, we overview that model along with how FRENETIC was using it in previous work.

The road shape of an SDC test case can be modeled as a series of curves $s(t)$. Furthermore, certain mathematical properties can be expressed on curves. For instance, the *Frenet-Serret formulas* describe the dynamics of a moving object among a curve. Specifically, (T, N, B) is a *frenet frame*, where T is the tangent on the curve
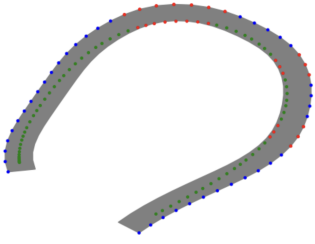
Figure 1: The road in grey, with the simulation trace in green showing the out-of-bounds conditions in red

heading the object's movement, N is the normal unit vector of T, and B is a unit vector by the cross product of T and N.

Moreover, $\frac{d}{ds}$ is the derivative with respect to arclength, κ is the curvature, and τ is the torsion of the curve. The two scalars κ and τ effectively define the curvature and torsion of a space curve. The associated collection, T, N, B, κ, and τ, is called the Frenet–Serret apparatus. Intuitively, curvature measures the failure of a curve to be a straight line, while torsion measures the failure of a curve to be planar.

Castellano et al. [9] make use of the Frenet representation in their genetic algorithm, which was shown to be one of the most successful approaches at the SBFT 2021 competition. Peltomäki et al. [17] also use the Frenet representation with a Wasserstein Generative Adversarial Network to estimate the fitness of tests before having to execute them on the simulator. Intuitively, they generate tests with higher fitnesses and the discriminator gets better at identifying tests with lower fitnesses. However, they limit themselves to segments with 6 points and struggle with generating valid tests.

3 THE FRENILLA APPROACH

While we utilize the dynamic encoding technique of Frenetic, we tackle the abovementioned limitations of Frenetic in this section. Specifically, in terms of novelty, our approach integrates a transformer model that implicitly learns the out-of-bound condition and expected outcome, supposed to improve the overall robustness and effectiveness of state-of-the-art approaches.

3.1 Seed Data

Given several failing simulation tests, we know where the car went out of bounds (Figure 1). We match those points to the closest curvature point in our road segment, which we label as out-of-bound. So, we have a pairing of curvature points, and if they lead to an out-of-bound condition, we can use them for training.

We re-used previously in simulation-executed SDC test cases consisting of information regarding the SDC's trajectory. In total, FRENILLA is trained on 12,561 road segments having a length of 50 meters. The seed data is also available in the replication package of [5].

3.2 Discriminator

In our framework, each road is represented by a series of vectors (c_i, s_i), where c_i indicates the curvature and s_i the cumulative step size. Each vector is encoded into a 128-dimensional vector, which is fed to a vanilla transformer with 6 heads and 6 layers, with a linear
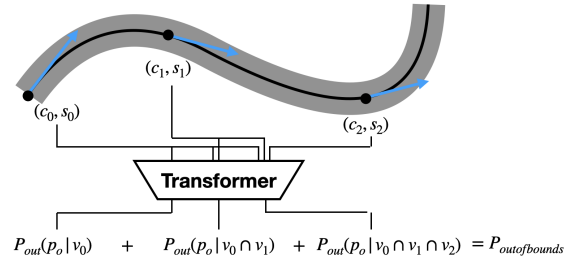

Figure 2: The discriminator model takes the previous road points (curvature, step size) as input to predict for each point the likelihood of an OOB condition

layer at the end predicting the in and out of bounds. The model is trained with a block size of 50 (representing the length of a road), batch size of 1024, and Adam optimizer with a learning rate of 3e-4 with a dropout of 0.2 in the linear layers.

We use the summed probability output of being out of bound as a fitness function F_1 for each road (Figure 2), which we use in our genetic algorithm. As a second fitness metric F_2, we use the median Euclidian distance to every other road segment in the pool, which represents our diversity metric. We trained the discriminator for 500 epochs until we reached our peak sensitivity of 0.52 and specificity of 0.96 in our validation set.

3.3 Genetic Algorithm

For our genetic algorithm, we start with AmbieGen as a template but adapt it to our problem. We encode our road as a series of 50 curvature points c_i with length cumulative length s_i. Therefore, we use the following operators:

(1) Crossover operator: Given two roads, a curvature index is randomly chosen, splitting the roads in half with an exchange of segments;
(2) K-Crossover operator: It operates as the crossover operator but with two splitting points;
(3) Swap operator: We take two random segments of the road between lengths 5 and 15 and interchange them;
(4) Mutation operator: We randomly chose an index, sample a random curvature value between $[-0.7, 0.7]$, and set the index $+-3$ to the curvature value. This has the effect of randomly introducing a curve to the road.

We initialize a population of random roads where, at each iteration, there is a probability of 0.8 for the crossover operator, 0.4 for 2-crossover, 0.4 for swap, and 0.2 for mutation. We then select the top 3,000 roads based on our F_1 fitness score and further select 2,000 of our population based on our diversity metric F_2. This has the effect of prioritizing fitness but maintaining diversity.

We observed that roads with high fitness were characterized by non-smooth segments. Consequently, we added a univariate cubic spline interpolation (smoothing factor: 0.01) for the newly generated road segments to produce naturally smooth road segments. To avoid duplicates, we also check for matching road segments that have an Euclidean distance of less than 0.2, which we consequently exclude from reproduction. A secondary validity test checks for curvature constraints $[-0.1; 0.1]$, boundary constraints

(inside of the 200x200 limit), and not self-intersecting, which are consequently removed from the pool. We run the algorithm for 50 epochs, reaching a mean probability of being out of bounds of 33% and a median distance of 0.4 to every other road segment.

4 EMPIRICAL EVALUATION

To empirically evaluate FRENILLA, we apply the same experimental setting of FRENETIC at the SBST tool competition [16] to compare the results.

4.1 Research question

In the first part of the evaluation of FRENILLA, we want to know to what extent the approach generates feasible test cases. The BEAMNG.TECH simulator and the test runner have certain restrictions, e.g., the test case must not contain roads that intersect with themselves or that comprise curves being too sharp. If the test does not fulfill the abovementioned requirements, then it is considered an *invalid test case*. Thus, our first research question is:

> RQ_1: *Does diversity-guided search exploration in Frenet space with* FRENILLA *lead to more valid test cases compared to Frenetic?*

We investigate to what extent our approach produces valid test cases compared to Frenetic. An approach that produces many invalid test cases is not optimal, especially considering that the generation process requires a significant amount of time/testing budget. Assuming the approach produces only a few invalid test cases, we can focus on effective testing, i.e., test cases that will eventually fail. Hence, our second research question is as follows:

> RQ_2: *Does diversity-guided search exploration in Frenet space with* FRENILLA *lead to higher fault detection compared to Frenetic?*

We execute 1,174 test cases generated by FRENILLA. By measuring the execution time and logging the test outcomes, we compute the number of test failures, i.e., the approach's effectiveness.

4.2 Test subject

We use the BEAMNG.TECH simulation environment with its built-in self-driving agent to evaluate FRENILLA. Concretely, we use the BEAMNG.TECH *v0.24.0.2*, which is a more recent version than the version in the context of SBST'21. The driving agent steers the car with the goal of keeping the SDC within the lane. The SDC must stay within the lane by at least 85% of its area; otherwise, the test will fail.

4.3 Procedure

To address RQ_1 and RQ_2, we compare the results of FRENETIC with FRENILLA, and apply the same procedure as in SBST'21 [1]. We provide FRENILLA to generate and execute test cases with a time budget of two hours. The driving agent is configured to have a maximum speed of 70km/h, and the oracle was set to an out-of-bound (OOB) [5, 6] of 30%.

Test generation. We generated 1,174 test cases with our approach FRENILLA. The tests are stored as JSON files compatible with the SDC-SCISSOR tool to execute the tests in BEAMNG.TECH.

Test execution. We execute all 1,174 generated test cases in BEAMNG.TECH. For configuring the runtime parameters, such as the OOB tolerance and test runner, we use SDC-SCISSOR [4], which is a reimplementation of the test runner provided by SBST'21 with more up-to-date dependencies.

Infrastructure. We used a Windows 10 machine with an Intel(R) Core(TM) i9-10900KF CPU at 3.70GHz, 32 GB RAM, and a 10GB NVIDIA GeForce RTX 3080 graphics card.

4.4 Data collection

For each test execution, we log the test outcome and simulation time in JSON files. Concretely, we have in the JSON file the following fields: (i) `test_outcome`, which can be either "PASS" or "FAIL", and (ii) `test_duration`, for the simulation time measured in seconds.

4.5 Data analysis

We count the number of faults detected during the test execution as well the number of invalid test cases produced by FRENILLA within the given time budget. Since we ran 1,174 test cases, which require more execution time than the time budget limit, we randomly selected the test results with the logged simulation time until we reached the time budget of two hours. We performed the random selection with replacement 100 times and computed statistics. Concretely, we compare the number of invalid test cases (RQ_1) and fault detection rate (RQ_2) with the results of FRENETIC from [16].

All data and code are available in a replication package (Section 7). For the analysis, we set up a MONGODB with the simulations data to effectively query the data with our analysis scripts.

5 RESULTS & DISCUSSION

With FRENILLA, we ran a total of 1,174 test cases, which is also our population of test cases from which we randomly sample sets of test cases, where each set consists of test cases with a total simulation time within the time budget, i.e., we construct a sample set by taking tests from the population until we reach the time budget. We sampled 100 times with replacement so that we can compute the statistics, such as the minimum, maximum, and mean of invalid and failing test cases. As indicated in Table 1, from all 100 samples, we did not identify any invalid road, i.e., a road that intersects with itself. Considering large testing campaigns, the FRENILLA is able to generate valid test cases effectively and does not introduce overhead in the generation process by generating invalid test cases. FRENILLA produces failing tests in most cases; on average,

Table 1: Descriptive statistics of 100 samples with a simulation time budget of two hours.

Statistic	# Executed	# Invalid	# Faults
Min.	778	0	482
Avg.	798.47	0	508.58
Max.	815	0	536

Frenilla generates 508.58 test cases leading to an OOB violation within a time budget of two hours. When taking into consideration that, within the budget, 796.47 tests are executed on average, we have 64% failing test cases. In the tool competition [16], Frenetic generated on average 20-25 (boxplot in [16]) failing test cases. With Frenilla, we have an approach that effectively generates test cases, which helps to reduce the testing costs since we spend less time on executing passing ("uninformative") test cases.

5.1 Threats to validity

Due to the large number of training examples in comparison to the number of tests we generate, we find a number of generated tests that appear similar to the ones in our training set. We derived the mean minimal distance for every test to every training example, which we find to be an average of 0.15, which is below our duplication criterium of 0.2. However, the average median distance (our diversity measure) is higher, i.e., at 0.6, showing a relatively high diversity. It is important to note that there is no explicit conditioning of our discriminator model on the generation of tests, but that through the fitness metric, the model might favor examples that appear similar to what it has seen in its training. Nevertheless, we also find 300 examples with a minimal distance above 0.2, proving that we are able to generate completely novel tests.

Regarding RQ$_1$, an invalid road is defined as a road that intersects with itself or has overly sharp turns. As far as we know, no well-defined definition of an overly sharp turn in the literature is available. Hence, we can not exclude that Frenilla would still generate invalid roads. Furthermore, with respect to RQ$_2$, Frenilla would benefit from an evaluation with statistical tests to show its performance against the benchmark with more confidence. We did not perform experiments with Frenetic but only used the reported results from [16]. Thus, the comparison of Frenilla with Frenetic would benefit from a very identical experimental setup.

6 CONCLUSION

In summary, we have shown that the OOB metric can be implicitly learned by a model without explicitly formulating the problem. In combination with a genetic algorithm, we have shown that a diverse population of tests can be generated, showing high validity and relatively high fault-revealing power.

Future works should focus on leveraging the ability of transformer models to work with variable road lengths, possibly increasing the diversity of tests that can be generated. We could improve on diversity by extending the diversity measure to the training set and generated test, thus favoring more diversity and human perception [7] overall.

Currently, we also rely on a large number of failing tests, which are needed beforehand. A reinforcement model that continuously improves its ability to predict the likelihood of failure could work as a discriminator for a genetic algorithm, identifying the tests that might produce failures before having to execute them.

7 DATA AVAILABILITY

The code and dataset are publicly available on Zenodo [8].

CREDIT AUTHOR STATEMENT

Timo Blattner: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft. **Christian Birchler**: Investigation, Resources, Data Curation, Writing - Original Draft. **Timo Kehrer**: Writing - Review & Editing, Funding acquisition. **Sebastiano Panichella**: Writing Original Draft, Supervision, Funding acquisition.

ACKNOWLEDGMENTS

REFERENCES

[1] 2021. *Intl. Workshop on Search-Based Software Testing*. IEEE. https://doi.org/10.1109/SBST52555.2021

[2] National Highway Traffic Safety Administration. 2022. *Summary Report: Standing General Order on Crash Reporting for Level 2 Advanced Driver Assistance Systems*. Technical Report DOT HS 813 325. 1200 New Jersey Avenue, SE Washington, D.C. 20590.

[3] Matteo Biagiola, Stefan Klikovits, Jarkko Peltomäki, and Vincenzo Riccio. 2023. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. In *Intl. Workshop on Search-Based and Fuzz Testing*. https://doi.org/10.1109/SBFT59156.2023.00010

[4] Christian Birchler, Nicolas Ganz, Sajad Khatiri, Alessio Gambi, and Sebastiano Panichella. 2022. Cost-effective Simulation-based Test Selection in Self-driving Cars Software with SDC-Scissor. In *Intl. Conference on Software Analysis, Evolution and Reengineering*. 164–168. https://doi.org/10.1109/SANER53432.2022.00030

[5] Christian Birchler, Sajad Khatiri, Bill Bosshard, Alessio Gambi, and Sebastiano Panichella. 2023. Machine learning-based test selection for simulation-based testing of self-driving cars software. *Empir. Softw. Eng.* 28, 3 (2023), 71. https://doi.org/10.1007/S10664-023-10286-Y

[6] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. 2023. Single and Multi-Objective Test Cases Prioritization for Self-Driving Cars in Virtual Environments. *ACM Trans. Softw. Eng. Methodol.* 32, 2 (apr 2023). https://doi.org/10.1145/3533818

[7] Christian Birchler, Tanzil Kombarabettu Mohammed, Pooja Rani, Teodora Nechita, Timo Kehrer, and Sebastiano Panichella. 2024. How does Simulation-based Testing for Self-driving Cars match Human Perception?. In *ACM International Conference on the Foundations of Software Engineering*.

[8] Timo Blattner, Christian Birchler, Timo Kehrer, and Sebastiano Panichella. 2024. Diversity-guided Search Exploration for Self- driving Cars Test Generation through Frenet Space Encoding [REPLICATION PACKAGE]. https://doi.org/10.5281/zenodo.10527295

[9] Ezequiel Castellano, Ahmet Cetinkaya, Cédric Ho Thanh, Stefan Klikovits, Xiaoyi Zhang, and Paolo Arcaini. 2021. Frenetic at the SBST 2021 Tool Competition. In *Intl. Workshop on Search-Based Software Testing*. IEEE, 36–37. https://doi.org/10.1109/SBST52555.2021.00016

[10] Andrea Di Sorbo, Fiorella Zampetti, Aaron Visaggio, Massimiliano Di Penta, and Sebastiano Panichella. 2023. Automated Identification and Qualitative Characterization of Safety Concerns Reported in UAV Software Platforms. *ACM Trans. Softw. Eng. Methodol.* 32, 3 (2023). https://doi.org/10.1145/3564821

[11] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *Intl. Conference on Software Testing, Verification and Validation (ICST)*.

[12] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based Test Case Generation for Unmanned Aerial Vehicles in the Neighborhood of Real Flights. In *Conference on Software Testing, Verification and Validation*. IEEE, 281–292. https://doi.org/10.1109/ICST57152.2023.00034

[13] Sajad Khatiri, Prasun Saurabh, Timothy Zimmermann, Charith Munasinghe, Christian Birchler, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024 - CPS-UAV Test Case Generation Track. In *Intl. Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.

[14] Stefan Klikovits, Ezequiel Castellano, Ahmet Cetinkaya, and Paolo Arcaini. 2023. Frenetic-lib: An extensible framework for search-based generation of road structures for ADS testing. *Sci. Comput. Program.* 230 (2023), 102996. https://doi.org/10.1016/J.SCICO.2023.102996

[15] NPR. [n. d.]. Nearly 400 car crashes in 11 months involved automated tech, companies tell regulators. npr.org. www.npr.org/2022/06/15/1105252793/nearly-400-car-crashes-in-11-months-involved-automated-tech-companies-tell-regul

[16] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *Intl. Workshop on Search-Based Software Testing*. IEEE, 20–27. https://doi.org/10.1109/SBST52555.2021.00011

[17] Jarkko Peltomäki, Frankie Spencer, and Ivan Porres. 2022. Wasserstein generative adversarial networks for online test generation for cyber physical systems. In *Workshop on Search-Based Software Testing*. 1–5.

[18] Jesper Winsten and Ivan Porres. 2023. WOGAN at the SBFT 2023 Tool Competition - Cyber-Physical Systems Track. In *Intl. Workshop on Search-Based and Fuzz Testing*. IEEE, 43–44. https://doi.org/10.1109/SBFT59156.2023.00009