SensoDat: Simulation-based Sensor Dataset of Self-driving Cars

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

Cyrill Rohrbach
cyrill.rohrbach@students.unibe.ch
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

Developing tools in the context of autonomous systems [22, 24], such as self-driving cars (SDCs), is time-consuming and costly since researchers and practitioners rely on expensive computing hardware and simulation software. We propose SensoDat, a dataset of 32,580 executed simulation-based SDC test cases generated with state-of-the-art test generators for SDCs. The dataset consists of trajectory logs and a variety of sensor data from the SDCs (e.g., rpm, wheel speed, brake thermals, transmission, etc.) represented as a time series. In total, SensoDat provides data from 81 different simulated sensors. Future research in the domain of SDCs does not necessarily depend on executing expensive test cases when using SensoDat. Furthermore, with the high amount and variety of sensor data, we think SensoDat can contribute to research, particularly for AI development, regression testing techniques for simulation-based SDC testing, flakiness in simulation, etc.
Link to the dataset: https://doi.org/10.5281/zenodo.10307479

ACM Reference Format:
Christian Birchler, Cyrill Rohrbach, Timo Kehrer, and Sebastiano Panichella. 2024. SensoDat: Simulation-based Sensor Dataset of Self-driving Cars. In *Proceedings of 21st International Conference on Mining Software Repositories (MSR 2024).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

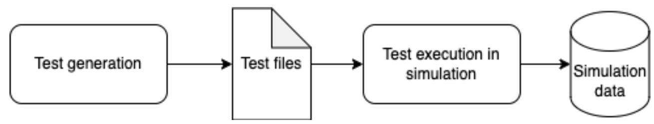

Figure 1: High-level dataset generation process

1 INTRODUCTION

Testing self-driving cars (SDCs) is crucial for maintaining high security levels and minimizing potential threats to humans. While infield SDC testing is costly, simulation technologies offer a safer alternative. However, conducting simulation-based tests demands increased computational resources, particularly GPUs for accelerated computation of physical dynamics [1, 27].

To address the challenge of minimizing the costs of simulation-based testing, recent research focused on various regression testing techniques for simulation-based tests. Those techniques aim to test SDCs cost-effectively while maintaining the system's safety. For instance, by test prioritization [2, 8], in which the tests of the test suite are prioritized, i.e., sorted in a way so that the testing phase reveals faults of the system earlier.

However, to conduct research on optimizing simulation-based testing, researchers rely on expensive test executions in simulation environments [1, 27]. These additional computational costs are mainly due to the expensive computation simulating the physics of the environment, which is not the case when testing traditional software systems. Thus, those tests are expensive and often not affordable when a large amount of data is required, e.g., for ML and DNNs.

To overcome the issue of running simulations to obtain the execution data of simulations, researchers can use existing datasets of executed simulation-based tests. There are only a few datasets available consisting of simulation data for SDCs [13, 26]. Despite the existence of those datasets, in most cases, the simulators used are not maintained anymore. Popular simulators like Udacity [21], Apollo [3], SVL [25], and DeepDrive [15] were used in the past for research purposes, but unfortunately, the active development of these simulators has been stopped by the maintainers or have long release cycles.

We propose a dataset consisting of simulation data of executed SDC test cases in the BeamNG.tech simulation environment. The BeamNG.tech simulator is known in academia [5, 7, 10, 16, 28, 29] and is based on the popular BeamNG.drive game, which is actively developed and maintained by BeamNG GmbH. The availability of simulation data of SDCs enhances the research on SDC testing in simulation. Researchers and practitioners do not rely on executing expensive test cases to develop and evaluate regression testing techniques. Having the dataset publicly available eases the research for the domain of SDC software, especially for researchers and practitioners who can not afford expensive computing hardware. Furthermore, the availability of an open dataset improves the reproducibility and comparability of research results in various areas.

MSR 2024, April 2024, Lisbon, Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 METHODOLOGY

As illustrated in Figure 1, to start a data collection process of sensor data of simulation-based SDC test cases, we have to generate test cases with different test generators (Section 2.1). After the generation of test cases, we execute those test cases in a simulation environment (Section 2.2). During the test execution, we collect the data from various simulated sensors of the SDC.

2.1 Test generation

We generate test cases that are based on three different test generators for the BeamNG.tech simulator. All of these test generators were developed in the context of tool competitions at the SBST [16, 28] and SBFT [5] workshop.

```
{
  "_id": {...},
  "OpenDRIVE": {
    "header": {
      "@name": "mt_set5/frenetic/test-440",
      "sdc_test_info": {
        "@test_id": "mt_set5/frenetic/test-440",
        "@test_outcome": "PASS",
        "@predicted_test_outcome": "null",
        "@test_duration": "11.067427158355713",
        "@is_valid": "True"
      },
      "run_config": {
        "@rf": "1.5",
        "@oob": "0.5",
        "@max_speed": "120",
        "@obstacles": "False",
        "@bump_dist": "20",
        "@delineator_dist": "None",
        "@tree_dist": "None",
        "@field_of_view": "120"
      }
    },
    "road": {...}
  },
  "execution_data": {...}
}
```

Listing 1: Metadata of a test execution consisting of simulation configuration and test outcome

2.1.1 *Frenetic.*

The Frenetic tool [11] uses a genetic algorithm to minimize the distance between the SDC and the edge of the road. For its computation, it leverages the concepts of frenet frames and curvature-based road representations. Thus, it converts the solutions back to the Cartesian space, which is required for the simulator.

2.1.2 *FreneticV.*

The FreneticV tool [12] is an extension of Frenetic, which is able to reduce the amount of invalid roads. For example, an invalid road has overly sharp turns, or if it intersects with itself. This definition of road validity is given by the SBST tool competition [16], for which the tool was developed for.

2.1.3 *AmbieGen.*

The AmbieGen [17] tool is a test generator that uses a multi-objective approach using NSGA-II [14]. Using the diversity preservation and the fault revealing power as the objectives, AmbieGen generates test cases that are more likely to reveal faults based on the OOB metric.

2.2 Simulation platform

We used the BeamNG.tech [4] simulator with SDC-Scissor [6] to execute the generated test cases. BeamNG.tech is a widely used simulator for researching testing of SDCs in academia [5, 16, 28]. It is a high-fidelity soft-body physics simulator that accurately simulates the car's dynamics, such as the pressure and deformation on the chassis and engine, clutch, and its related components. We conjecture obtaining as realistic data as possible using the BeamNG.tech simulator.

The vendor of BeamNG.tech provides researchers with a free academic license, which makes it highly accessible for researchers. Furthermore, the simulator provides a Python API to control the simulation process, including defining the environment with the road shape and gathering sensor data.

Table 1: Database storage size

Collection	# Documents	Storage size
campaign_2_ambiegen	973	108.36 MB
campaign_2_frenetic	928	109.46 MB
campaign_2_frenetic_v	944	41.93 MB
campaign_3_ambiegen	964	109.85 MB
campaign_3_frenetic	954	112.73 MB
campaign_4_ambiegen	965	111.98 MB
campaign_4_frenetic	964	113.87 MB
campaign_4_frenetic_v	525	63.43 MB
campaign_5_ambiegen	958	109.59 MB
campaign_5_frenetic	945	112.29 MB
campaign_5_frenetic_v	940	112.81 MB
campaign_6_ambiegen	959	111.76 MB
campaign_6_frenetic	944	111.11 MB
campaign_6_frenetic_v	764	91.14 MB
campaign_7_ambiegen	963	110.00 MB
campaign_7_frenetic	967	114.09 MB
campaign_7_frenetic_v	47	5.67 MB
campaign_8_ambiegen	952	110.76 MB
campaign_8_frenetic	952	112.25 MB
campaign_9_ambiegen	953	109.20 MB
campaign_9_frenetic	964	113.57 MB
campaign_10_ambiegen	971	63.95 MB
campaign_11_ambiegen	973	72.79 MB
campaign_11_frenetic	866	66.44 MB
campaign_11_frenetic_v	953	73.52 MB
campaign_12_frenetic	956	110.11 MB
campaign_12_freneticV	942	114.00 MB
campaign_13_ambiegen	954	68.83 MB
campaign_13_frenetic	959	72.52 MB
campaign_13_frenetic_v	951	71.48 MB
campaign_14_ambiegen	959	70.14 MB
campaign_14_frenetic	866	64.05 MB
campaign_14_frenetic_v	934	70.16 MB
campaign_15_ambiegen	952	110.50 MB
campaign_15_frenetic	870	102.61 MB
campaign_15_freneticV	949	114.67 MB
Total	**32,580**	**3.34 GB**

2.3 Data generation

2.3.1 *Process.*

We generated data by conducting 14 simulation campaigns. Every campaign follows the process as illustrated in Figure 1. We generate test cases with test generators (Section 2.1). After the generation, we execute the test cases in the simulation environment and collect the sensor data at runtime. Furthermore, we collect the information if the SDC violated the OOB safety metric and, therefore, passes or fails the test case.

2.3.2 *Infrastructure.*

The simulations were executed on a machine with the following hardware specifications: *AMD Ryzen 7 3800X* 8 core 16 threads, 64 GB DDR4 RAM, *NVIDIA GeForce GTX 1080* 8GB, *Windows 11*.

3 DATA STORAGE

We use MongoDB as database technology to efficiently query and perform analyses on the data. Since the raw data obtained from the simulations is mainly stored as JSON files, the mapping of JSON files

Table 2: Overview of available types of sensor data

fuel	steering_input	oil	exhaust_flow	4x brakeCoreTemperature	left_signal	airspeed	brake_input	signal_l
low fuel	rpm spin	lowhighbeam	fog_lights	4x brakeThermalEfficiency	signal_r	abs	tcs	parking
gear	airflow speed	lowbeam	fuel_volume	4x brakeSurfaceTemperature	right_signal	steering	ignition	hazard
odometer	lights	high beam	fuel_capacity	engineRunning	tcs_active	isYCBrakeActive	gearboxMode	clutch_input
brake	horn	brakelight_signal_R	gear_a	running	water_temperature	isTCBrakeActive	lightbar	abs_active
throttle	hasABS	brakelight_signal_L	gear_index	low-pressure	brake_lights	driveshaft	headlights	engine_throttle
parking brake	altitude	lowhighbeam_signal_R	gear_m	rpm	check_engine	wheelspeed	oil_temperature	esc_active
throttle_input	dseColor	lowhighbeam_signal_L	hazard_signal	clutch	clutch_ratio	esc	radiator_fan_spin	avg_wheel_av
reverse	virtualAirspeed	turn signal	is_shifting	parkingbrake_input	engine_load	smoothShiftLogicAV	rpm_tacho	freezeState

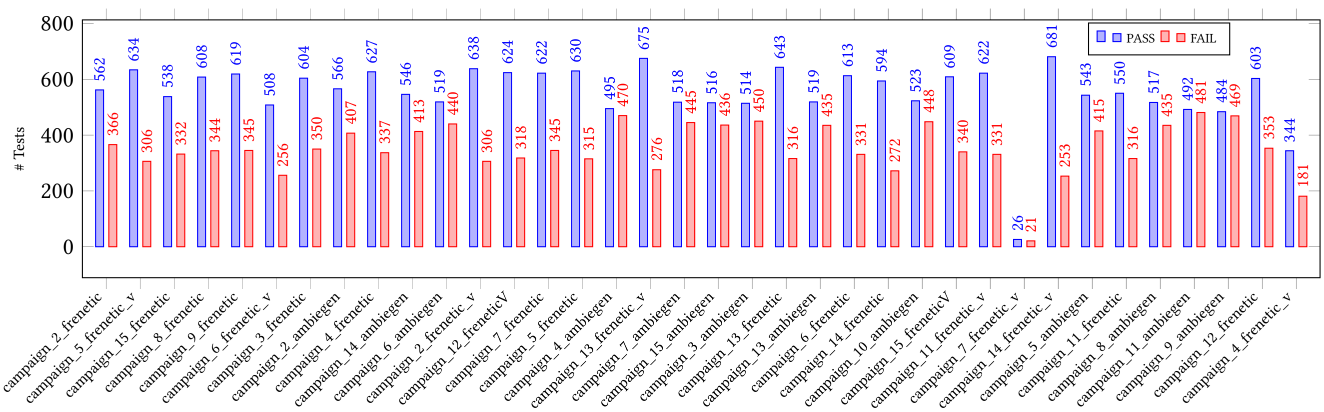

Figure 2: Test outcome distribution among MongoDB collections

to documents in MongoDB is straightforward For each simulation campaign, a dedicated collection is created in the database.

3.1 Database setup

First, to set up the database, download the raw data and code from the public repository as described in Section 7. Furthermore, it is required to have Docker [18] installed on the machine.

```
docker-compose -f ./environment/docker-compose.yml up -d --build
docker ps # Verify if container is up and running
docker cp ./data uploader:/app/data
docker exec uploader unzip ./data/data.zip -d ./data
docker cp ./code uploader:/app/code
docker exec -it uploader python ./code/fill_mongodb.py
```

Listing 2: Database setup with Docker

A locally deployed MongoDB instance will provide all simulation data by following the aforementioned instructions. The data is accessible by using a compatible MongoDB client (e.g., MongoDB Compass [19]) and the default credentials (*username*: `msr`, *password*: `fooBar`).

3.2 Database query

Given the database is up and running, querying the data is simple. The queries must conform to the MongoDB language specification as illustrated in a sample query in Listing 3. In this example, we use pymongo [20] as a client library to connect to MongoDB for the Python programming language. The query operates on the `campaign_2_frenetic` database collection and counts the number of passing test executions. For this specific query, we make use of the document structure as illustrated in Listing 1.

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

uri = "mongodb://msr:fooBar@localhost:27017/?authMechanism=DEFAULT"
client = MongoClient(uri, server_api=ServerApi('1'))
db = client.get_database('sdc_sim_data')

query_passing = \
    {"OpenDRIVE.header.sdc_test_info.@test_outcome": "PASS"}
nr_passing_tests = db['campaign_2_frenetic'] \
    .count_documents(filter=query_passing)
print(nr_passing_tests)
```

Listing 3: Sample query with pymongo

Adjusting the query string, a user can also query for specific sensor values. For example, the user can query data for a time range before a fault has occurred in the simulation. For this, the user needs to identify the location of the desired field in the database document and write the appropriate query string.

4 DATA CHARACTARISTICS

Table 1 overviews the total amount of collections and their corresponding required storage sizes. From the simulator in which we conducted 14 test campaigns, we obtained total data from 32,580 test executions. Figure 2 depicts the distribution of failing and passing test executions among 14 test campaigns. In total, we have 19,926 passing and 12,654 failing test executions. The distribution depends on the out-of-bound (OOB) metric, which acts as the oracle. For this dataset, we set $OOB = 0.5$, in order to have a fairly balanced dataset of passing and failing tests.

Next to the metadata (see Listing 1), we obtain data from the simulation as time series. For each data field, we obtain its state annotated with the timestamp. The timestamp reflects the delta between the start of the simulation and the time when the data was retrieved.

We capture two types of data as time series: (i) sensor-based data, and (ii) trajectory data. For the first type, we collect each sensor in Table 2 its state at every timestamp. Various sensors provide different data types, such as binary and continuous. For instance, on the one hand, `hasABS`, `parking`, or `signal_*` are binary and show if certain actions or features were active at the specific timestamp. On the other hand, `rpm`, `wheelspeed`, and `steering` are examples of data fields on the continuous range. The second type of data refers to the monitored trajectory of the SDC. We capture the current position of the SDC at each timestamp. The trajectories in the dataset are a time series of logged x,y, and z coordinates on the Cartesian coordinate system of the simulator. In conjunction with the first type of data, the whole state of the SDC is preserved for analysis purposes. Such analysis can be made for various areas to enhance SDC software research as discussed in Section 5.

5 USAGE & IMPLICATIONS

The availability of sensor and trajectory data enables the evaluation of testing methodologies without running expensive simulations. Furthermore, it lowers the barrier for researchers to conduct research since no complex infrastructure setup is required for simulating SDCs. In the following, we present various research directions for which our dataset can contribute.

5.1 Driver AI development

The field of SDCs is rapidly advancing, but the availability of AI models specifically designed for SDCs remains limited. Our goal is to contribute to the progress of open-source driving AI development. Regrettably, numerous researchers and practitioners face challenges due to the scarcity of computing resources required for generating training data for these models. By providing access to our comprehensive dataset, we aim to empower the research community to overcome these barriers and facilitate the creation of prototypes for driving AIs. We believe that this collaborative effort will accelerate advancements in self-driving technology and foster innovation within the field.

5.2 Regression testing in simulation

Various state-of-the-art regression testing techniques for simulation-based tests use different features, such as *road features* [7, 8]. Using these features enables a cost-effective testing of SDCs. However, using our dataset, we can identify and develop new features for new regression testing approaches.

Additionally, empirical evaluation of regression testing techniques for simulation-based tests is costly since tests must be executed to obtain the test outcome. The dataset mitigates the issue of running simulations for this kind of evaluation purposes of regression testing techniques.

5.3 CAN bus protocol support

Most modern vehicles have different components that communicate with each other over a shared bus system called *CAN bus*. Using the dataset consisting of sensor data, testing relevant CAN devices for the lane-keeping system is feasible with realistic CAN messages as test inputs. Recent studies [7, 10] already confirmed that the same type of data to develop testing approaches for CAN devices based on sensor data retrieved from simulation environments.

5.4 Flakiness in simulation

We encourage researchers to investigate with our dataset the flakiness of simulation-based tests. In other domains, such as for unmanned aerial vehicles (UAV), we observe different behaviors between the real world and the simulated test cases and within simulations [23].

These non-deterministic, i.e., flaky, behaviors of simulators bring up new challenges, especially in the context of *DevOps* pipelines. Having flaky tests in a continuous integration and delivery (CI/CD) pipeline reduces the time to deployment since these tests lead to build failures. The presence of flaky tests leads to failing tests that eventually fail the build process. Furthermore, flaky tests do not necessarily reveal bugs in the system since only the test code might cause the flaky behavior. In summary, flaky tests are a challenging problem for simulation-based tests as they are for traditional software systems. With the dataset, we aim to provide initial execution data for comparison with the replicated dataset of other researchers.

6 REMARKS

The aim of the dataset is to provide a variety of different sensor and trajectory data from 32,580 SDC simulations for research purposes. Researchers can profit from the dataset because they do not necessarily need to execute expensive simulation-based tests to obtain the required data.

We used the BeamNG.tech simulator since several well-known SDC simulators are outdated and not maintained anymore, which is crucial for developing new SDC technologies. BeamNG.tech is currently actively maintained by BeamNG GmbH and is also widely used in academia.

For future datasets, we suggest increasing the amount of data since training data for several AI techniques is crucial. Furthermore, we encourage researchers to replicate the dataset for analyses on research topics mentioned in Section 5.

7 DATA AVAILABILITY

All the data and code to set up the MongoDB and compute the statistics is made available on Zenodo [9].

CREDIT AUTHOR STATEMENT

Christian Birchler: Conceptualization, Resources, Data Curation, Writing - Original Draft, Visualization. **Cyrill Rohrbach**: Conceptualization, Methodology, Investigation, Data Curation. **Timo Kehrer**: Writing - Review & Editing, Supervision. **Sebastiano Panichella**: Conceptualization, Writing - Review & Editing, Supervision, Funding acquisition.

ACKNOWLEDGMENTS

We thank the Horizon 2020 (EU Commission) support for the project COSMOS, Project No. 957254-COSMOS.

REFERENCES

[1] Aitor Arrieta, Shuai Wang, Ainhoa Arruabarrena, Urtzi Markiegi, Goiuria Sagardui, and Leire Etxeberria. 2018. Multi-objective black-box test case selection for cost-effectively testing simulation models. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, Hernán E. Aguirre and Keiki Takadama (Eds.). ACM, 1411–1418. https://doi.org/10.1145/3205455.3205490

[2] Aitor Arrieta, Shuai Wang, Goiuria Sagardui, and Leire Etxeberria. 2019. Search-Based test case prioritization for simulation-Based testing of cyber-Physical system product lines. *J. Syst. Softw.* 149 (2019), 1–34. https://doi.org/10.1016/J.JSS.2018.09.055

[3] Apollo Auto. 2023. *apollo.* Retrieved December 8, 2023 from https://github.com/ApolloAuto/apollo

[4] BeamNG GmbH. [n. d.]. *BeamNG.tech.* https://www.beamng.tech/

[5] Matteo Biagiola, Stefan Klikovits, Jarkko Peltomäki, and Vincenzo Riccio. 2023. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2023, Melbourne, Australia, May 14, 2023.* IEEE, 45–48. https://doi.org/10.1109/SBFT59156.2023.00010

[6] 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 *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022.* IEEE, 164–168. https://doi.org/10.1109/SANER53432.2022.00030

[7] 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

[8] 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 (2023), 28:1–28:30. https://doi.org/10.1145/3533818

[9] Christian Birchler, Cyrill Rohrbach, Timo Kehrer, and Sebastiano Panichella. 2023. *SensoDat: Simulation-based Sensor Dataset of Self- driving Cars.* https://doi.org/10.5281/zenodo.10307479

[10] Christian Birchler, Cyrill Rohrbach, Hyeongkyun Kim, Alessio Gambi, Tianhai Liu, Jens Horneber, Timo Kehrer, and Sebastiano Panichella. 2023. TEASER: Simulation-Based CAN Bus Regression Testing for Self-Driving Cars Software. In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023.* IEEE, 2058–2061. https://doi.org/10.1109/ASE56229.2023.00154

[11] Ezequiel Castellano, Ahmet Cetinkaya, Cédric Ho Thanh, Stefan Klikovits, Xiaoyi Zhang, and Paolo Arcaini. 2021. Frenetic at the SBST 2021 Tool Competition. In *14th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2021, Madrid, Spain, May 31, 2021.* IEEE, 36–37. https://doi.org/10.1109/SBST52555.2021.00016

[12] Ezequiel Castellano, Stefan Klikovits, Ahmet Cetinkaya, and Paolo Arcaini. 2022. FreneticV at the SBST 2022 Tool Competition. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022.* IEEE, 47–48. https://doi.org/10.1145/3526072.3527532

[13] Birchler Christian, Khatiri Sajad, Derakhshanfar Pouria, Panichella Sebastiano, and Panichella Annibale. 2021. *Dataset Package for the paper "Single and Multi-objective Test Cases Prioritization for Self- driving Cars in Virtual Environments".* https://doi.org/10.5281/zenodo.5771017

[14] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2000. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II. In *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1917)*, Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julián Merelo Guervós, and Hans-Paul Schwefel (Eds.). Springer, 849–858. https://doi.org/10.1007/3-540-45356-3_83

[15] Deepdrive. 2023. *Deepdrive.* Retrieved December 8, 2023 from https://github.com/deepdrive/deepdrive

[16] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022.* IEEE, 25–32. https://doi.org/10.1145/3526072.3527538

[17] Dmytro Humeniuk, Giuliano Antoniol, and Foutse Khomh. 2022. AmbieGen tool at the SBST 2022 Tool Competition. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022.* IEEE, 43–46. https://doi.org/10.1145/3526072.3527531

[18] Docker Inc. 2023. *Get Started with Docker.* Retrieved December 4, 2023 from https://www.docker.com/get-started/

[19] MongoDB Inc. 2023. *Compass. The GUI for MongoDB.* Retrieved December 4, 2023 from https://www.mongodb.com/products/tools/compass

[20] MongoDB Inc. 2023. *PyMongo 4.6.1 Documentation.* Retrieved December 4, 2023 from https://pymongo.readthedocs.io/en/4.6.1/

[21] Udacity Inc. 2017. *A self-driving car simulator built with Unity.* Retrieved December 10 2023 from https://github.com/udacity/self-driving-car-sim

[22] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *2023 16th IEEE International Conference on Software Testing, Verification and Validation (ICST).*

[23] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based Test Case Generation for Unmanned Aerial Vehicles in the Neighborhood of Real Flights. In *IEEE Conference on Software Testing, Verification and Validation, ICST 2023, Dublin, Ireland, April 16-20, 2023.* IEEE, 281–292. https://doi.org/10.1109/ICST57152.2023.00034

[24] 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 *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[25] LGSVL. 2023. *SVL Simulator: An Autonomous Vehicle Simulator.* Retrieved December 8, 2023 from https://github.com/lgsvl/simulator

[26] Chengjie Lu, Tao Yue, and Shaukat Ali. 2023. DeepScenario: An Open Driving Scenario Dataset for Autonomous Driving System Testing. In *20th IEEE/ACM International Conference on Mining Software Repositories, MSR 2023, Melbourne, Australia, May 15-16, 2023.* IEEE, 52–56. https://doi.org/10.1109/MSR59073.2023.00020

[27] Toni Mancini, Igor Melatti, and Enrico Tronci. 2023. Optimizing Highly-Parallel Simulation-Based Verification of Cyber-Physical Systems. *IEEE Trans. Software Eng.* 49, 9 (2023), 4443–4455. https://doi.org/10.1109/TSE.2023.3298432

[28] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *14th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2021, Madrid, Spain, May 31, 2021.* IEEE, 20–27. https://doi.org/10.1109/SBST52555.2021.00011

[29] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2023. Efficient and Effective Feature Space Exploration for Testing Deep Learning Systems. *ACM Trans. Softw. Eng. Methodol.* 32, 2, Article 49 (mar 2023), 38 pages. https://doi.org/10.1145/3544792