– DRiVERSITY–
Synthetic Torture Testing to Find Limits of Autonomous Driving Algorithms*

Extended Abstract

Daniel Frassinelli
CISPA Helmholtz-Zentrum i.G. GmbH
Saarbrücken, Germany
daniel.frassinelli@cispa.saarland

Alessio Gambi
Passau University
Passau, Germany
alessio.gambi@uni-passau.de

Stefan Nürnberger ✉
CISPA Helmholtz-Zentrum i.G. GmbH
Saarbrücken, Germany
nuernberger@cispa.saarland

Sohyeon Park
CISPA Helmholtz-Zentrum i.G. GmbH
Saarbrücken, Germany
sohyeon.park@cispa.saarland

ABSTRACT

Autonomous driving is expected to significantly improve road safety. Car makers are conducting extensive testing of their autonomous vehicles on proofing grounds and in virtual pre-defined scenarios. Because proofing grounds do not offer a deterministic test field and are time-consuming, virtual hardware- and software-in-the-loop testing is en vogue as it provides reproducibility. However, pre-defined tests (real or virtual) represent low coverage in comparison to all physically possible driving scenarios. Furthermore, they are unlikely to systematically discover corner cases that emerge due to software bugs or absurd but possible scenarios.

In this abstract, we introduce the DRiVERSITY framework for systematic testing of autonomous driving algorithms. Our DRiVERSITY framework builds scenes on the fly – adapting to how a car handles situations while driving. DRiVERSITY pronounces misbehaviour by tailoring new scenes based on monitored driving bahaviour during fuzzing stimuli. DRiVERSITY shall provide a standard testing framework to evaluate and compare driving algorithms in a reproducible and controllable way.

CCS CONCEPTS

• **Software and its engineering** → **Software safety**; • **Computer systems organization** → *Embedded software*;

1 INTRODUCTION

Autonomous driving is an industrial and academic effort that promises the elimination of 90% of all accidents [1]. Car makers are conducting intensive research in different fields, from computer vision to embedded computing, and are pushing hard to release fully autonomous cars in the near future. However, as of today, autonomous cars can by no means safely drive in all imaginable scenarios, may it be urban, snowbound or in bright sunlight.

Problem. To address this, it is vital to thoroughly test the decision-making algorithms as their ultimately makes the difference between life and death. Despite being tested, available autonomous vehicles caused accidents including fatalities [5, 7]; this calls for novel ways and systematic approaches to better test autonomous cars.

Solution. Systematic testing can point to weaknesses in driving implementations by automatically generating large sets of *diverse* tests which better cover the input space compared to the current practice of manually pre-selecting driving scenarios. This advantage is twofold:

(1) Systematic testing identifies universally *critical* driving situations that are hard or even impossible to handle for computers and humans alike.
(2) Testing specific implementations can reveal peculiarities, which are the result of bugs, over-approximations, or vague scene processing.

Our Approach. DRiVERSITY makes use of search-based procedural content generation to test autonomous cars by *automatically synthesising diverse, critical, but still realistic, virtual driving scenarios.* This project not only aims to discover functional problems and safety issues of autonomous cars, but it also tests non-functional properties, such as driving style, passenger comfort, and economic driving.

Contributions. The proposed research system goes well beyond the state-of-practice and state-of-art. The final aim is to develop a testing framework for systematically testing autonomous cars, which can be used by developers to

(a) achieve more in-depth testing of autonomous vehicles by generating and executing test cases orders of magnitudes faster than physical driving;
(b) find critical situations quickly;
(c) maximise coverage of driving scenarios;
(d) generate tests which expose functional and non-functional related issues; and,
(e) compare different autonomous cars.

2 DRIVERSITY - DRIVING DIVERSITY

It has been shown that digital environments can be used as a substitute for real-world driving [4]. We use synthesised, realistic scenery to automatically create test beds for already trained and programmed driving algorithms.

2.1 Idea

The core functionality of DRiVERSITY is an open-world driving environment in which scenes are created on-the-fly. This is enabled by leveraging the fact that sensors can only detect their environment in a perimeter of a few hundred meters. The scenery outside that perimeter is undefined and generated only when the car under test moves in a certain direction. Our approach combines powerful *evolutionary techniques* for test case generation [2] with *procedural content generation*, the core component of modern hyperrealistic computer games [8].

Our search-based framework evaluates the fitness of driving behaviour for a given scene and uses it as input to synthesize an infinitely large on-the-fly environment. We divide the infinite environment in areas of a few m^2 that we call a *scene*. Each scene is defined by parameters such as the 3D assets that appear in a scene, their physical attributes, their geographical layout and the weather conditions. A naive approach would result in a potentially infinite number of virtual scenes. This is why the main building block of DRiVERSITY is to discriminate *relevant* from *irrelevant* scenes. More specific, each newly synthesised scene should ideally fulfil the following criteria:

- **Diversity:** Test cases should be different from each other in terms of the car's behaviour that they expose;
- **Reproducibility:** Scenes must be completely deterministic such that test parameters are sufficient to completely recreate the same execution conditions every time;
- **Utility:** The more likely a crash or uncomfortable a passenger, the more useful;
- **Realism:** The visual, geometrical and physical quality of the generated scene must be realistic;
- **Corner Case Discovery:** The level of unexpectedness should be high (e.g. a kangaroo crossing a street [6]).

Our approach is organized as a continuous loop following the best practices for hardware-in-the-loop testing [3]. It is depicted in Figure 1.

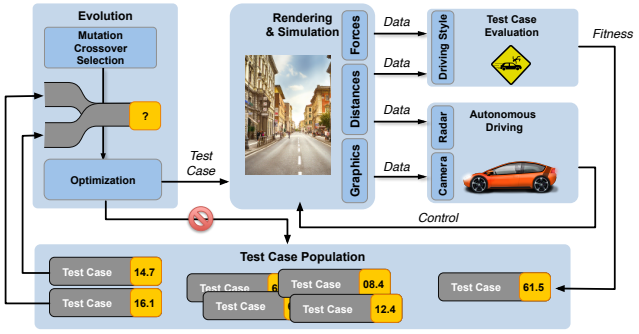

Figure 1: Approach Overview

The loop starts by generating a population of test cases (*Test Case Population*), which are then selected, combined, and mutated to create challenging driving scenarios (*Evolution*). Then, test cases are executed (*Rendering & Simulation*). Test case execution includes rendering the virtual environment and feeding synthetic data (camera, radar, lidar) that simulate the environment to the self-driving algorithm (*Autonomous Driving*). The self-driving algorithm reacts by providing control commands back to the simulation (steering, acceleration, braking), which influence the physics of the virtual car inside the DRiVERSITY simulation. This creates a tight feedback loop between the self-driving algorithm and the driving simulator. Simulation data are also used to objectively assess the quality of the test cases by calculating their fitness score (*Test Case Evaluation*). The fitness score is propagated back to the population of test cases and drives their evolution by favouring those test cases which exposed critical behaviours of the self-driving algorithm.

2.2 Realism and Search-Space Pruning

During the evolution, as a means to optimise the efficiency of the test generation, we discard those test cases which contribute little to the final goal of exposing self-driving algorithm critical behaviours. For example, we filter out test cases which are too similar to the already experienced ones.

The proposed testing framework includes constraints that can be used to eliminate unrealistic test cases. Those unrealistic cases are detected by means of domain knowledge, such as traffic regulations, along with geometrical, topological, and physical constraints. DRiVERSITY prevents the generation of test cases where cars fly or houses are built in the middle of a road. By incorporating those constraints into the search algorithms, we enable the search to create valid virtual worlds by design, thus saving substantial computing time during the generation.

2.3 Extensibility

We achieve extensibility at functional level by adopting a plug-in design: specific implementations of the primary functions will be wrapped inside components which can be easily plugged together to fit the diverse developers' needs. For example, search algorithms and fitness functions will be provided as Plug-Ins and integrated into the framework.

2.4 Seamless Transitions

Finally, creating test cases by instantiating and combining distinct geometrical elements, such as road segments, requires a seamless transition between them. For instance, the combination of possibly very different driving scenes, such as highways and cities, must be presented in a smooth fashion to the autonomous cars. To resolve these challenges, we use element stitching. We introduce intermediate geometrical elements that seamlessly combine otherwise incompatible geometries and scenes.

3 CONCLUSION

We presented a new framework for tailored testing of driving algorithms without hard-coded scenes. This allows the probabilistic discovery of bugs, faults and safety issues and enables an objective comparison of autonomous driving implementations.

REFERENCES

[1] Michele Bertoncello and Dominik Wee. 2015. Ten ways autonomous driving could redefine the automotive world. http://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world. *McKinsey & Company, Automotive & Assembly* (June 2015).

[2] Mark Harman. 2007. The Current State and Future of Search Based Software Engineering. In *2007 Future of Software Engineering (FOSE '07)*. IEEE Computer Society, 342–357.

[3] Taehun Hwang, Jihoon Roh, Kihong Park, Jeongho Hwang, Kyu Hoon Lee, Kangwon Lee, Soo-Jin Lee, and Young-Jun Kim. 2006. Development of HILS systems for active brake control systems. In *SICE-ICASE, 2006. International Joint Conference*. IEEE, 4404–4408.

[4] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for Data: Ground Truth from Computer Games. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*. Springer, Cham, 102–118.

[5] The Guardian. 2016. Google self-driving car caught on video colliding with bus. https://www.theguardian.com/technology/2016/mar/09/google-self-driving-car-crash-video-accident-bus. (March 2016).

[6] The Guardian. 2017. Volvo admits its self-driving cars are confused by kangaroos. https://www.theguardian.com/technology/2017/jul/01/volvo-admits-its-self-driving-cars-are-confused-by-kangaroos. (2017).

[7] The Guardian. 2018. Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian. https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe. (February 2018).

[8] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (Sept 2011), 172–186.