Preliminary Evaluation of Path-aware Crossover Operators for Search-Based Test Data Generation for Autonomous Driving

Seunghee Han*
KAIST
Daejeon, Republic of Korea
shhan1755@kaist.ac.kr

Jaeuk Kim*
KAIST
Daejeon, Republic of Korea
kju5789@kaist.ac.kr

Geon Kim*
KAIST
Daejeon, Republic of Korea
skygun88@kaist.ac.kr

Jaemin Cho*
KAIST
Daejeon, Republic of Korea
jjm7307@kaist.ac.kr

Jiin Kim*
KAIST
Daejeon, Republic of Korea
kji1362@kaist.ac.kr

Shin Yoo
KAIST
Daejeon, Republic of Korea
shin.yoo@kaist.ac.kr

***Abstract*—As autonomous driving gains attraction, testing of autonomous vehicles has become an important issue. However, testing in the real world is not only dangerous but also expensive. Consequently, a virtual test method has emerged as an alternative. Recently, a novel testing technique based on Procedural Content Generation (PCG) and Genetic Algorithm (GA), AsFault, has been proposed to test the lane-keeping functionality of autonomous vehicles. This paper proposes new crossover operators for AsFault that can better preserve the coupling between genotype (representations of road segments) and phenotype (occurrences of interesting self-driving behaviour). We explain our design intentions and present a preliminary evaluation of the proposed operators using the Simulink autonomous driving simulator. We report promising early results: the proposed operators can lead not only to Out of Bound Episodes (OBEs) but also causes more vision errors in the simulation when compared to the original.**

***Index Terms*—Autonomous Driving, Test Data Generation, Search Based Software Engineering**

I. Introduction

With advances in AI techniques, the autonomous driving industry has become one of the most valuable industries [8]. Consequently, the safety assurance of autonomous vehicles is receiving much attention. To improve the safety of autonomous vehicles, it is crucial to test and detect problematic behaviour of self-driving cars as early as possible. The most realistic way of testing autonomous vehicles would be in the real traffic. However, it would be impractical for a number of reasons [3]: the testing itself may pose safety risks to other cars, not to mention the cost and time required for testing. Testing within computer simulation is a widely accepted alternative [9].

AsFault [2] is a search-based test data generation technique that can evolve road networks to test the lane-keeping functionality of autonomous vehicles on. AsFault combines Procedural Content Generation (PCG) and Genetic Algorithm (GA) to automatically evolve random road maps that can be used to expose problems in self-driving software.

*Equal contribution

While AsFault has been shown to be capable of producing realistic and effective road layouts, we believe its genetic operators can be improved. AsFault uses two types of crossover methods to generate new road maps by combining different segments of each parent road map. Here, the road segments on the map are genotype, while the self-driving behaviour on the roads is phenotype. In principle, if we select a candidate solution based on the fitness of its phenotype, we would want to pass on the corresponding genotype: if a road map reveals particularly erratic self-driving behaviour, we would like to keep the road segment that affected the driving behaviour negatively. However, the crossover operator used by AsFault does not consider the specific road segment that was traversed by the simulation, and chooses a random segment for the exchange. This leads to the failure to preserve the dominant trait of the parent road network that resulted in the high fitness.

This paper proposes new crossover operators that can better preserve the dominant trait of a given parent road network in AsFault: these operators are aware of the actual path last traversed by the simulation to yield the fitness value, and always operate on a road in the traversed path. We also introduce a new fitness function that captures more details of OBEs. For a preliminary evaluation of these components, we present a framework called ATeSS (Automatic Test Data Generator System for Self-Driving Car). Our preliminary comparison of ATeSS and AsFault are potentially promising and show that the new operators, guided by the new fitness function, can potentially generate road networks that induce more OBEs than AsFault.

The remainder of this paper is organised as follows. Section II describes some background knowledge about this study including conventional AsFault. Section III describes ATeSS, with an emphasis on the differences in crossover operators.

Section IV presents related work, and Section V concludes.

II. AsFault

AsFault [2] combines PCG and GA to generate road networks that can expose the problems in the lane-keeping functionality of autonomous vehicles. PCG refers to the algorithmic generation of game content with limited or no human input. AsFault uses PCG to initialise the road network population. The road network generation is an incremental process that builds road networks from road segments using PCG: it is depicted in Fig 1. First, it constructs a short straight road segment, either horizontally or vertically: this segment becomes the starting point of the road (Fig 1 (a)). Subsequently, PCG grows the road by randomly creating additional new road segments, which are concatenated to form a single road (Fig 1 (b)). The road generation is completed when the road reaches the map boundary (Fig 1 (c)). Multiple roads are placed on the same map to create a road network (Fig 1 (d)). During the process, PCG also applies some validation checks. For example, a road cannot intersect itself. When an invalid road or road network is detected, the generation process fails and PCG creates a new road or road network.

Each time the self-driving car software under test deviates from the road counts as an Out of Boundary Episode (OBE). AsFault uses GA to evolve road networks that produce more OBEs. After an initial population is generated, AsFault selects road networks with more OBEs using tournament selection. Next, selected networks produce offsprings using crossover and mutation. AsFault can successfully generate road networks that produce OBEs through the evolutionary search.

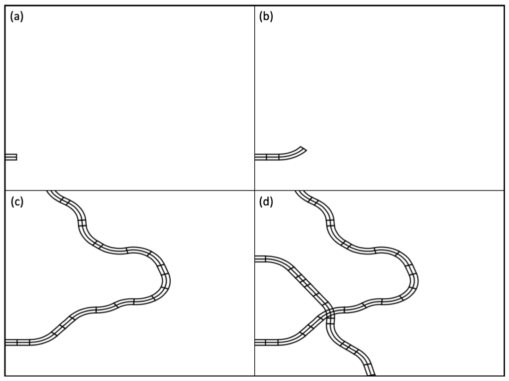

Fig. 1. An example of road network generation process.

III. ATeSS

ATeSS (Automatic Test Data Generator System for Self-Driving Car) is essentially AsFault with improved genetic operators and fitness function. It inherits PCG and GA for AsFault to evolve road networks. ATeSS uses tournament selection with $k = 5$, and an elitism mechanism that preserves the top 10% of parent generation according to their fitness. We describe the fitness function, as well as the crossover and the mutation operator, of ATeSS in the remainder of this section.

A. *Fitness Calculation*

We argue that the number of OBEs may not be the ideal fitness function to generate test data (i.e., road networks) for lane-keeping functionality. Essentially being a count-based metric, OBEs can be prone to ties. The number of OBEs as fitness function also fails to capture the severity of each OBE, as both minor and major deviations will count as one. To address these issues, we propose Bounded Lateral Deviation (BLD) as the fitness value: it represents the maximum distance between the centre of the road and the car during self-driving. Figure 2 shows when OBE occurs, and how the fitness value is measured.

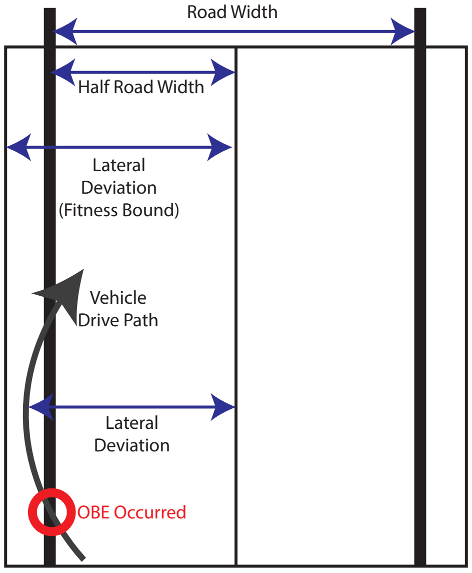

Fig. 2. Fitness value calculation scenario. The lateral deviation is used to calculate the fitness value, and OBE occurs when the lateral deviation becomes larger than the half road width.

When the lateral deviation is greater than half the width of the road, an OBE occurs. However, during our evaluation, we found that exceptionally large lateral deviations can cause the simulator to malfunction, hindering the search process. Consequently, we bound the lateral deviation by a value larger than half of the road width.

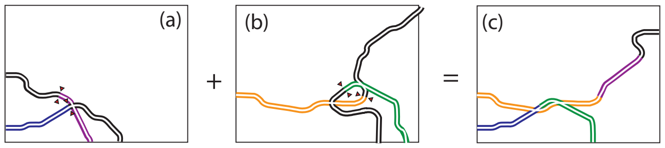

Fig. 3. Example of the original join crossover: in (c), the blue segment of (a) and the green segment of (b) have been joined, while the purple segment of (a) and the yellow segment of (b) have been joined.

B. *Crossover Operators*

1) AsFault: AsFault uses join and merge crossover operators. The join crossover splits each road into two roads at a randomly assigned segment and cross-connects the divided parts to create one new road. Subsequently, it creates random road segments, so that the road can be reached to the map boundary. Figure 3 shows an example of the join crossover operator.

Merge crossover, on the other hand, randomly chooses two or three roads from two road networks without any variation on the road itself in the road networks. Figure 4 shows an example of the merge crossover operator application.

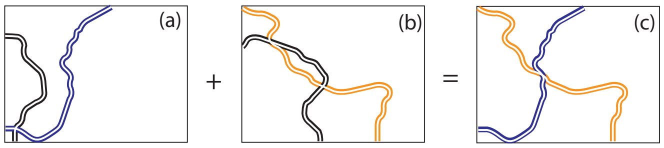

Fig. 4. Example of the original merge crossover: in (c), the blue road of (a) and the yellow road of (b) have been merged.

The original crossover operators proposed by AsFault operate at the map level. However, the fitness of a map is actually only based on a single road that the vehicle traverse during the fitness evaluation. This, combined with the random sampling of the cut or merge point, means that segments of road that have little to do with the map fitness can be chosen for the crossover, based on the map level fitness.

2) ATeSS: ATeSS join crossover randomly chooses a join point from the path traversed in the latest fitness evaluation of the road network. Figure 5 shows an example of an improved join crossover operator application. The red lines at the centre of the roads in Figure 5 (a) and Figure 5 (b), respectively, denote the paths traversed in the fitness evaluation. Note that a traversed path exists across multiple roads. ATeSS first randomly chooses a road used by the traversed path, then subsequently chooses a cut point on the chosen road. Consider Figure 5 (a) for an example: the traversed path involves two roads. ATeSS first chooses the left one that connects the top and the bottom of the map tile, then chooses a random cut point to retrieve the segment to perform crossover with (road segment with the blue lanes). Similarly, the traversed path in Figure 5 (b) involved both roads: ATeSS chooses the curved road at the bottom, and subsequently the cut point, and the road segment with the green lanes. These two chosen segments are combined into the child road in Figure 5 (c), which is added to the child road network map tile. First, one of the chosen segments is placed randomly at the edge of the child map tile; the next one is joined at the end, and the road is randomly extended using PCG, until it crosses the map edge.

ATeSS performs the same validity check as used by AsFault. If a child road is not valid, ATeSS will make up to ten attempts with the same pair of chosen roads (each time with different cut points), after which it chooses another pair of roads from the parent. It will repeat the join crossover until the number of roads in the child map tile is equal to that of the parent map tile with fewer roads.

Similarly to its join crossover, the merge crossover of ATeSS operates on the path that was traversed in the fitness evaluation.

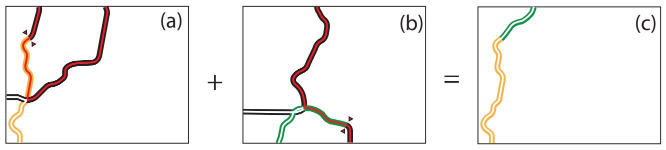

Fig. 5. Example of the improved join crossover operator: the traversed path is shown in red. The yellow road from (a) and the green road from (b) have been chosen from the traversed paths. After cutting them at a random join points marked with triangles, the yellow and green segments have been joined in (c).

Figure 6 shows an example of the improved merge crossover operator. The red line in Figure 6 (a) and Figure 6 (b) shows the traversed path. ATeSS randomly chooses one or two roads that are traversed from one parent (roads with blue lanes in Figure 6 (a)), and merges them with one or two traversed roads from the other parent (the road with green lanes in Figure 6 (b)) , while keeping the maximum number of roads to three. The chosen roads are merged into the child road network in Figure 6 (c) consists of only blue and green lines. Merge crossover operator also follows the same validation and iteration policy of the join crossover.

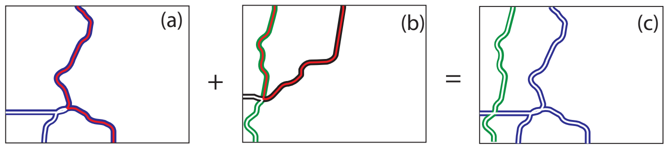

Fig. 6. Example of the improved merge crossover: the traversed path is shown in red. Both blue roads of (a) have been chosen from the traversed path, while only the green road of (b) has been chosen. These are merged in (c).

With the bias towards the traversed path, we hope to preserve the dominant trait of the road that led to the fitness of the parent road network map.

C. Preliminary Evaluation

In our preliminary evaluation of ATeSS using Simulink autonomous driving simulator, the path-aware crossover operators, combined with the new fitness function, resulted in 14–29% more OBEs than AsFault, depending on the size of the map, using a population size of 25 and 40 generations of search budget. ATeSS can also consistently outperform random search. These early findings are potentially promising and in line with our expectation that path-aware crossovers may improve the effectiveness of AsFault. We leave a more rigorous comparison with statistical analysis as future work.

IV. Related Work

Testing of AI-based systems have received much attention recently, as machine learning models using Deep Neural Networks (DNNs) are rapidly incorporated into safety-critical system such as autonomous driving vehicles [5], [7], [10], [13], [15]. Many existing work focus on test adequacy criteria, i.e., the problem of finding a criterion that can be used to choose a test input that is likely to reveal unexpected behaviour in the

DNN model under test. However, existing work tends to study a component of an autonomous driving vehicle, such as steering control module [13] or object segmentation module [6] in separation. System level testing of autonomous driving vehicle is relatively under-explored, although uncertainty-based approaches are starting to be investigated [12].

Unlike the study of test adequacy, actual test data generation for DNN models remain as a challenge due to the high dimensionality of the input space [1], [14]. Riccio and Tonella proposed a model-based approach, where a search-based approach is applied to parameterised models of test inputs to DNN image classifiers [11]. Kang et al. introduced a VAE based model, where a search is performed in the latent embedding space of the input domain derived by Variational Autoencoder [4]. AsFault can be considered as a model-based approach, but it uses discrete ingredient genes, rather than parameterised models [2]. Our work focuses on the GA-based search part of AsFault, and proposes alternative crossover operators that are more customised for the input domain (i.e., road maps).

V. Conclusion

We proposed a new and improved crossover operator for search-based test data generation for the testing of lane-keeping functionalities of autonomous vehicles. The crossover operators are improved versions of those used by AsFault, an existing test data generation framework. Unlike existing operators, our new search operators are aware of the path traversed by the vehicle in the latest fitness evaluation, and focuses on the traversed road segments when performing crossover. Consequently, we expect the new crossover operators to better preserve the parts of the road network that resulted in its fitness value, providing better coupling between genotype and phenotype. We also present a more detailed fitness function to capture Out-of-Bound Episodes. A preliminary comparison of ATeSS, our extension of AsFault with new operators and fitness function, to AsFault, shows potentially promising performance. We leave more rigorous experimentation and comparison as future work.

Acknowledgement

This work was supported by the Korean MSIT (Ministry of Science and ICT), under the National Program for Excellence in SW (2016-0-00018), supervised by the IITP (Institute of Information & communications Technology Planning &Evaluation).

References

[1] Taejoon Byun and Sanjai Rayadurgam. Manifold for machine learning assurance. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, Jun 2020.

[2] Alessio Gambi, Marc Müller, and Gordon Fraser. Asfault: Testing self-driving car software using search-based procedural content generation. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 27–30. IEEE, 2019.

[3] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[4] Sungmin Kang, Robert Feldt, and Shin Yoo. Sinvad: Search-based image space navigation for dnn image classifier test input generation. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, SBST 2020, pages 521–528, 2020.

[5] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the 41th International Conference on Software Engineering*, ICSE 2019, pages 1039–1049. IEEE Press, 2019.

[6] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE Industry Track)*, ESEC/FSE 2020, pages 1466–1476, 2020.

[7] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems. *CoRR*, abs/1803.07519, 2018.

[8] Market Data Forecast Inc. Self-driving cars market worth usd 220.44 billion by 2025: Industry report 2020-2025 https://www.marketdataforecast.com/market-reports/self-driving-cars-market.

[9] Satoshi Masuda. Software testing design techniques used in automated vehicle simulations. In *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 300–303. IEEE, 2017.

[10] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 1–18, New York, NY, USA, 2017. ACM.

[11] Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 876–888, 2020.

[12] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 359–371, 2020.

[13] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 303–314, New York, NY, USA, 2018. ACM.

[14] Shin Yoo. Sbst in the age of machine learning systems-challenges ahead. In *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*, pages 2–2. IEEE, 2019.

[15] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 132–142, New York, NY, USA, 2018. ACM.