Metamorphic Testing of Advanced Driver-Assistance Systems: Implementing Euro NCAP Standards on OpenStreetMap

Muhammad Iqbal *
School of Computing and IT
University of Wollongong
Wollongong, NSW 2522, Australia
{mi759@uowmail.edu.au}

2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET) | 979-8-3503-0176-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/MET59151.2023.00008

***Abstract*—Simulation testing is considered the supplement to ensure the safety of Autonomous driving (AD) and advanced driving assistance (ADAS's) systems in terms of time and costs. However, it is very difficult and challenging when the simulation results are unexpected. This work presents a simulation-based metamorphic testing (MT) approach to test the ADAS system, implementing the European new car assessment program (Euro NCAP) standards on OpenStreetMap (OSM). We first defined input patterns and relations related to autonomous driving, following the principles of MT. To assess the approach, we executed three tests in two steps at both the design and system levels. Our results show that none of the three (source) tests detected any collisions. However, for follow-up test cases, the ego vehicle failed to apply brakes to avoid a collision when the speed changed. A real-life issue in the system was immediately revealed and confirmed by the development team. We then designed a mechanism and continued the test to check whether the recorded collisions were avoidable. Our results (rate of 5.8%) indicate the fault detection effectiveness when testing the ADAS system. Although we applied the approach to testing the ADAS driving performance, it can be applied to other AD systems. This research, therefore, provides a systematic way to design and test autonomous driving technologies and integrate testing standards with metamorphic testing.**

***Index Terms*—Simulation testing, Advanced driver assistance systems (ADAS), Metamorphic testing, Metamorphic relation pattern, Testing standards, OpenStreetMap**

I. Introduction

Simulation testing is considered the cornerstone of testing Advanced driver-assistance systems (ADASs) and autonomous driving (AD) features [19], [32]. It can quickly produce high-quality results at a relatively low cost among the various available testing approaches [14], [42].

Testing the Advanced driver-assistance systems (ADASs) involves exposing a vehicle to situations that trigger the system to intervene, then examining the outcome to assess the system's performance [1]. However, dealing with such oracles is difficult when the simulation result is unexpected. For example, given a certain driving scenario for the system, the performance should be stable and safe for all expected situations. The inability to determine whether the results are correct is known as the oracle problem [6].

Current studies show that researchers from both industry and academia try to improve the performance of both ADAS's and autonomous driving (AD) systems, using different tools and approaches [22], [23], [38]. Due to the lack of uniqueness in the testing approaches, it is challenging to address system performance, often involving simulations.

Also, to minimize the problem effectively, different task assessment methods are used by combining them with varying standards of testing like the European new car assessment program (Euro NCAP) [19], National highway traffic safety assessment (NHTSA) [7], ISO/PASS:214448 safety of intended functionality (SOTIF) [1], ISO:26262 [20].

Metamorphic testing (MT) [11], [12], [33] is one of the most popular approaches, alleviating the oracle problem and has been successfully adopted in different domains [45], [8], [13], [5], [17], [19], [35], [40], [41], [43]. Instead of focusing on the correctness of individual outputs, MT examines the relations called *metamorphic relations* (MRs), among the *multiple* executions of the SUT. An MR violation typically indicates the existence of a bug. One of the main challenges with metamorphic testing (MT) is the identification and design of metamorphic relations (MRs), which may require testers' domain knowledge [12].

To test the performance of ADAS and AD systems and to design metamorphic relations, research has also been conducted by integrating MT with different techniques. The combination of MT and fuzzing has been used to test the performance of the real-life self-driving system [17], [43]. They detected the previously unknown fatal bugs in the LiDAR obstacle perception and distinguished between genuine failures and false alarms.

OpenStreetMap[1] is one of the simplest and most user-friendly web-based HD map databases and is used by researchers to evaluate different testing approaches to testing autonomous driving systems [3], [4], [9], [28].

This paper reports on an experiment to evaluate the approach, executing three tests based on collision. Our experiment involves implementing the European New Car Assess-

[1]https://www.openstreetmap.org.

DOI 10.1109/MET59151.2023.00008

ment Programme (Euro NCAP) and MT, utilizing the OSM on MATLAB[2] and Simulink [3] platform. Although this paper focuses on ADASs based testing, our approach is equally applicable to more advanced AD systems.

The key contributions of this paper are:

- In this research, we successfully designed and presented a simulation-based MT approach to test the ADAS system behaviors by implementing the Euro NCAP standards on OpenStreetMap.
- We have presented a promising research direction, presenting a useful framework for testing valid scenarios on real roads. This work laid the foundation to combine MT with testing protocols to tackle an important problem of testing ADAS by evaluating the empirical study.
- We applied MT to test ADAS driving behavior at both the design and system levels. Our results indicate the approach's effectiveness in detecting faults.
- A systematic method is presented to demonstrate the recorded collisions as avoidable.

The rest of the paper is organized as follows: Section II introduces the concept of testing protocols, the OSM, and related features. Section III proposes our method to conduct the empirical study. Section IV presents the tests executions steps. Section V presents the experimental results and brief discussions of our findings. Section VI reviews some related work, and Section VII concludes the paper.

II. Preliminaries

This section introduces the testing standards, and a brief overview of the OpenStreetMap used to implement the approach.

A. Test protocols:

Testing protocols and standards are the keys to designing the driving scenario and establishing the environment to test the AD and ADAS systems in real-world and simulation environments.

The European new car assessment program (Euro NCAP) [2], is a voluntary car safety performance assessment program. The standards provide a primary foundation, design rules, and a driving scenario to test autonomous systems. A prebuilt scenario catalogues are available on different platforms [26] and tools[4]. Scenarios can be downloaded, categorized, and parametrized to establish the testing environment and evaluate the testing approaches [19].

For example, when testing the AEB function, two scenarios can be considered based on the Euro NCAP protocols: car-to-car (C2C) and Vulnerable Road users (VRU). For each scenario type, a number of critical driving situations can be considered for testing, such as a vehicle insertion in front of the side, the sudden appearance of conflicting objects (pedestrians, bicyclists), etc. To evaluate the testing framework, we considered one of the VRU driving scenarios in this work.

[2]urlhttps://au.mathworks.com/products/matlab.html
[3]https://au.mathworks.com/products/simulink.html
[4]https://simone.51aes.com/casemanage.

Figure 1 shows the test overview. The protocols assume that vehicles travel on the right side of the road; therefore, the right side is the side closest to the ego vehicle. At collision time, the pedestrian is 50% of the way across the width of the ego vehicle [26].

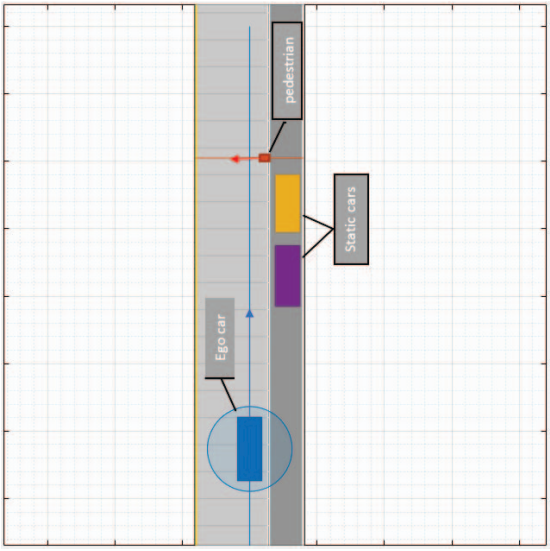


Fig. 1: **An overview of a prebuilt Euro NCAP test.**

The scenario has an ego vehicle (blue car) and three actors. One of the three actors is a pedestrian, the moving target with which the ego vehicle collides. The protocols also assume that collision occurs only once during the design phase, which must be avoided when executing the scenario with the AEB system. The ego vehicle and the non-ego actors must travel at constant speeds. Hence, the ego vehicle and the target actor must each have a scalar speed value.

B. An OpenStreetMap (OSM):

OpenStreetMap (OSM) is a free, open geographic database and is freely licensed under the Open Databases license (ODbl), hosted by OpenStreetMap Foundation[5] in England. It uses a topological data structure with four elements (also known as data primitives).

- Nodes are points with a geographic position, stored as coordinates, representing map features such as specific locations, mountain peaks, etc.
- Ways are ordered lists of nodes and can be polylines or polygons (closed ways), representing streets, highways, and parks, among others.
- Relations represent the relationship of existing nodes and ways—for example, turn restrictions on roads, routes that span several existing ways, etc.
- Tags are used as key or value pairs to store map objects' properties. They provide information about the elements of the map, especially about points of interest (POI) such as hotels, lakes, historical places, etc. Tags also provide

[5]https://en.wikipedia.org/wiki/OpenStreetMap.

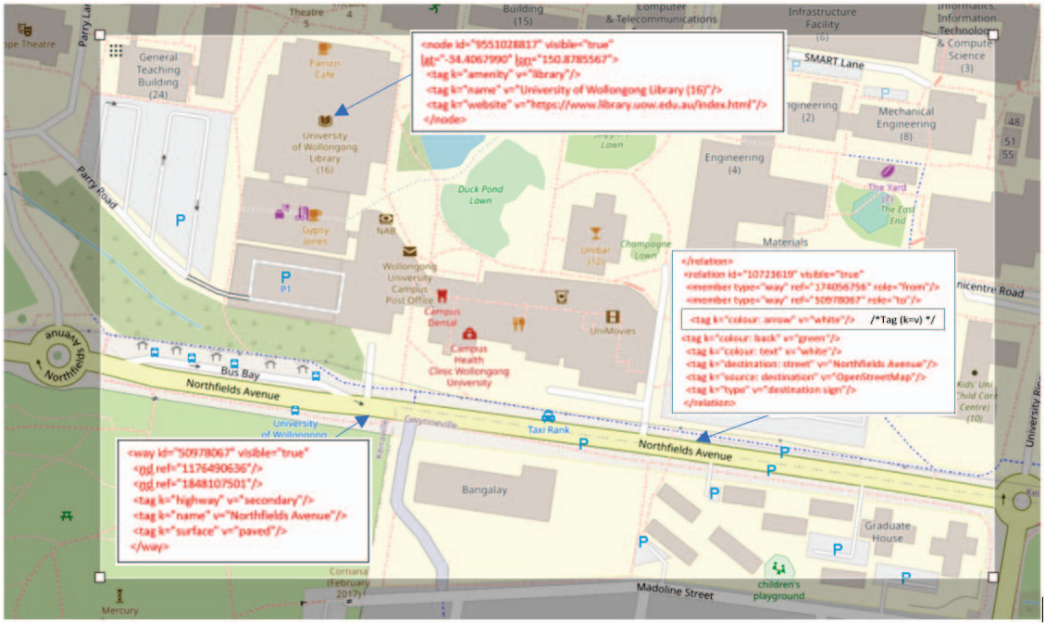


Fig. 2: **An example of OpenStreetMap.**

helpful information about streets and highways, which is central in OSM to describe routes from one point to another.

Figure 2 shows one example of an OSM from the surrounding of Wollongong City, NSW, Australia.

The nodes are described (as in the triangle on the top) by the node label, including the node identifier and coordinate as attributes, together with sub-label tag (key and value) pairs to the node. For example, the key is 'amenity', and the value is 'library.' The ways represent the geometries (as in the triangle in the left corner). The references to the nodes of the geometry are stated in addition to the tags. Additionally, relations are used to represent the sets of nodes/ways to group them (as in the right rectangle). Relations describe more complex structures like roads, bus stops, buildings on the same estate, etc.

The OSM provides an interactive web-based [6] interface editor with an online map and geo data search engine. Users can freely sign into the editor to access its features. Figure 3 shows the editor GUI of OpenStreetMap, which can be used to design a scenario in an OSM format interactively.

Using the editor feature of the tools in the right bar of Figure 3, the user can set the map by accessing the background setting, map data, and preferences. Once these instances have been selected, further refinements can be made by choosing the desired road or highway from the center and configuring (e.g., speed limits, lanes, surfaces, etc.) through the left-hand pane.

OSM uses its topology to store geographical features, which can then be exported onto other platforms to design driving behaviors and analysis from different perspectives.

[6]https://www.openstreetmap.org/edit#map=18/-34.40812/150.87892.

This feature helps us to import the real road geometry to the driving scenario designer (DSD) to establish the testing environment implementing the Euro NCAP standards to test ADAS behaviors.

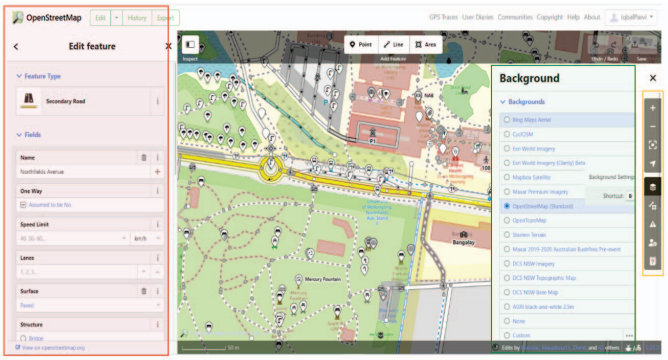


Fig. 3: **An interective OSM editor.**

III. Our Approach

Quality of service metrics is the key factor in both AD and ADASs to evaluate the driving experience [21], [37] and [5]. We considered the idea and selected the following metric to design our MR.

Time to Collision point (TCP):

The time required for a vehicle to reach the collision point (the position at which the ego vehicle and target actor collide) is termed the time to the collision point (TCP). We assume that the ego vehicle and the target actor always collide with a point on the front edge of the ego vehicle, keeping all other conditions the same.

We configured three tests on different road geometry (Euro NCAP and OSM). In each test the actor dimensions, positions,

speed values, and trajectories are set as per the Euro NCAP test protocol requirements. We considered the speed of the ego vehicle as an input pattern to establish the relation and execute the test systematically. A metamorphic relation input pattern (MRIP) describes input relations between the source and follow-up test cases. We define the following MRIP:

MRIP$_1$:Increasing or decreasing the speed $(\pm V)$**:**

This pattern represents the relation where the follow-up test case is constructed by increasing or decreasing the speed of the ego car. When this happens the time to the collision point should be similar to the source test case, to avoid a collision.

MR_1:

"For a given driving scenario S, in which the system detects the target obstacle 'O' (which arrives at the point "P" in time "t"), and applies a brake to avoid a collision. The system should perform the same when the speed of the ego car is changed to produce a follow-up scenario S' ".

One may argue that this situation (increased speed) makes the scenario aggressive. It should be noted, however, that to examine the system's behavior in high-speed situations is exactly one of our testing purposes. In any case, the ADAS system needs to react and drive safely.

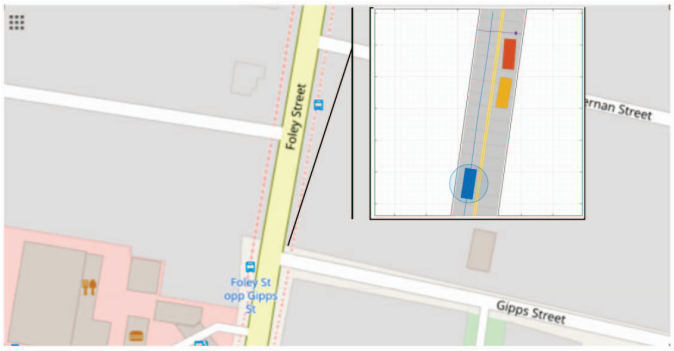
IV. Test Executions

The scenario design and test execution are separate steps. Hence, we considered both MATLAB API and the AEB Simulink system to execute the tests. To evaluate the approach, we performed three tests. Each test was executed in two steps. All the source and follow-up test cases are executed in both steps.

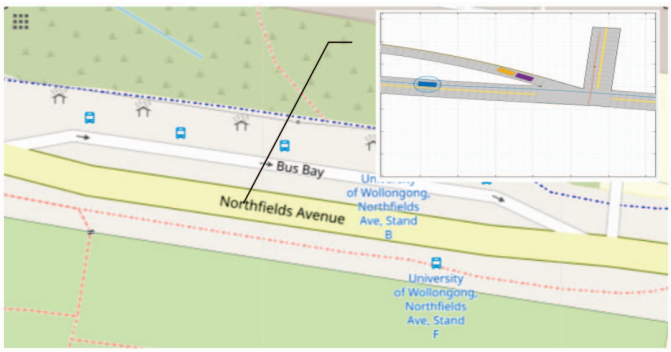
Stage-I

For *Test_01*, we have executed one of the Euro NCAP prebuilt driving scenario, i.e., $AEB\_PedestrianChild\_Nearside\_50width$. The scenario is compatible with both MATLAB API & and with the Simulink AEB systems. In the first stage, the source test cases for each test were executed, then a follow-up test was generated by increasing the speed of the ego car by 1 km/h. When executing the follow-up test case by increasing the speed, two types of collisions (front and side) between the ego car and the pedestrian was recorded. A scenario where the collision occurs on the right (missing TCP) of the ego car instead of front-angled is critical and is considered for further evaluation.

Test_02 and *Test_03* were executed by importing real-world roads (maps) from OpenStreetMap (OSM) to the Driving scenario designer. Both tests are performed by following the collision strategy of the Euro NCAP protocols. The actor dimensions, positions, speed values, and trajectories are set accordingly. The two real roads (Foley Street & Northfield Avenue) from the surrounding of the University of Wollongong, Australia, are considered to design the tests. The same collision was recorded with varying speeds. Figure 4 shows the implementation of testing standards on OSM.



(a)



(b)

Fig. 4: **Implementation of Euro NCAP pedestrian collision test on OSM: (a) Foley Street and (b) Northfield Avenue.**

Table I summarized parameters involved in the implementation of the test. For all tests, the initial speed of the ego car was 20 km/h, while the speed of the target pedestrian was 5 km/h.

TABLE I: **Summary of test parameters**

Test. No	Road type (straight)	Map adoption	Actors initial speed (km/h)	
			ego car	pedestrian
Test_01	highway	Euro_NCAP	20	5
Test_02	residential	OpenStreetMap		
Test_03	commercial			

Stage-II:

All the source and follow-up test cases were re-executed in this stage to see whether the recorded collision at stage-I was avoidable. We designed and executed a script on the MATLAB API (e.g., ABC.m) using some prebuilt helper functions and evaluated the approach systematically. To meet the requirement of MR_1, our scripts run the same source scenario and execute the test automatically by refreshing the whole scenario. In each execution, for all follow-up test cases (where speed increases or decreases), the mechanisms evaluate

the actor's behaviors to keep the arrival time to the collision point the same.

According to the MT methodology, three (euro NCAP and OSM-based) source test cases were executed to produce three source outputs. In each round of the test, the speed of the ego car changed randomly to produce follow-up test cases. A total of 85 rounds of tests were conducted (15 to 100 km/h), giving (85*3)=255 follow-up test cases and hence producing 255 execution outputs (called follow-up outputs).

The pairs of source and follow-up outputs were automatically checked against MR_1. Violations of metamorphic relations were recorded as a system failure.

V. Experimental Results and Discusions

When executing the three tests, no source test case reveals any crash or collision between the actors. This is summarized in the first two columns of Table II. In the follow-up executions, however, a collision was detected at different speeds when the speed of the ego car was changed by 1 km/h. The last two columns of Table II summarized the follow-up execution results.

TABLE II: **Results of experiment**

Test_Name	Source execution behavior	No. of scenario with collision		Minimum speed range that violated MR1	
		(front)	(side)	km/h	rate
Euro_NCAP _ped_collision	successfully avoided a collision	4	2	40	5.80%
Foley_Street _Ped_Collision		7	9	60	
NorthField_Avenue _Ped_Collision		2	4	52	

Scenarios can be designed/transformed programmatically using MATLAB API or the Driving Scenario Designer app. Figure 5 shows an example of MR_1 violation. The collision occurs on the side (missing TCP) of the ego car on MATLAB API after changing the ego car speed to 40, 60, and 52 km/h, respectively of the source scenarios.

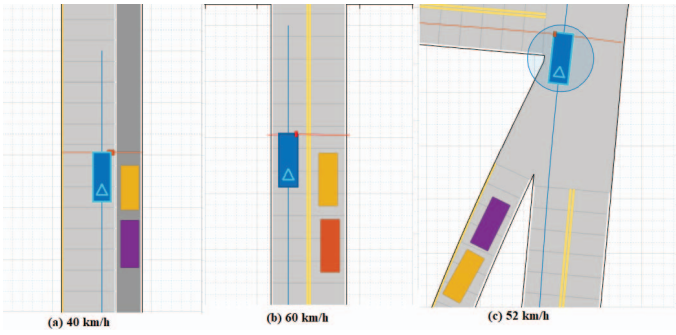


Fig. 5: **Voilations of MR at DSD.**

When the tests were executed with the Simulink AEB systems, the controller behavior was as expected for all source scenarios. However, the same collision (side of ego car) was recorded for follow-up executions. Figure 6 shows examples of MR_1 violation behaviors of AEB systems.

Hence, a relatively minor change can lead to the system's incorrect behavior: For example, in *Test_01* the ego speed of 39 km/h passed, but the follow-up, with the increment of 1 km/h speed that is 40 km/h, failed.

In the second stage, our automated script loads and executes the seed scenario systematically. The function executes the source scenario to store the actor's IDs. Once the IDs of the ego car and target actor are specified, the new speeds have been set to generate the follow-up scenarios. Then compared and utilized different prebuilt functions to control the time to collision point (TCP). The steps involved in the demonstration are detailed in Algorithm 1.

Algorithm 1: Steps to demonstrate stage-II

1 Driving scenarios (S, S');
Input : Speed V and V'
Output: $TCP(Drive(S)) = TCP(Drive(S'))$
2 Load$--$>seed scenario
3 Specify actor IDs to compute the (TCP)
$EgoID : 1; TargetID : 2, ...$
4 Specify the new speeds
$Condition1 - - > V >= V'$;
$Condition2 - - > V <= V'$
5 Compare speeds & define methods to control the actors
6 **if** $V >= V'$ **then**
7 | execute $(Method - A)$;
8 **else**
9 | execute $(Method - B)$;
10 **end**
11 Repeat step 4 to generate a new scenario
12 Plot and inspect the behaviors
13 Finish

Simulink used graphical block diagramming tools and a customizable set of block libraries (e.g. Scenario Reader block) to read the driving scenario [27].

We have considered the prebuilt ADAS module to evaluate the approach and did not know the internal mechanisms of the model. So, all the possible transformations were only done during the design phase and then pass to advanced driver assistance (ADAS) or automated driving systems (resulting in an equivalent effect).

Figure 7 shows the execution of *Test_01* on the second stage. The results also indicate that the collisions obtained in Figure 5 & Figure 6 are avoidable. Due to space limitations, we could not include more graphical results for other tests.

Out of 255 driving scenarios, 28 collisions are detected, each of which is checked against the MR_1, indicating 15 avoidable collisions. For all violations, the pedestrian collided with the right side of the ego car, resulting in missing the time to the collision point. The results (5.80%) indicate the effectiveness and efficiency of our approach to detecting faults.

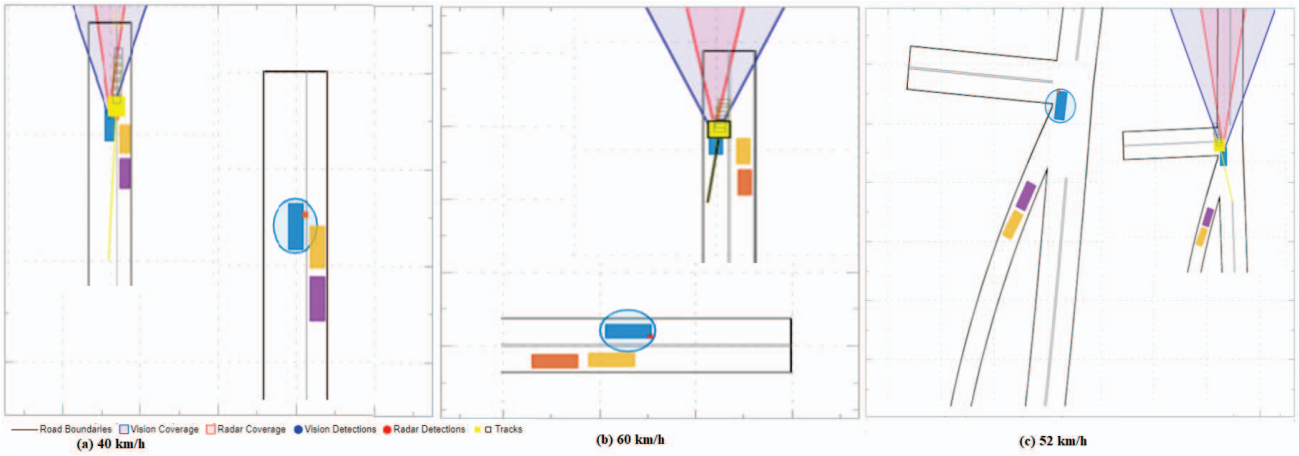


Fig. 6: **Voilations of MR at Simulink: Figures (a) *Test_01*, (b) *Test_02* and (c) *Test_03*.**

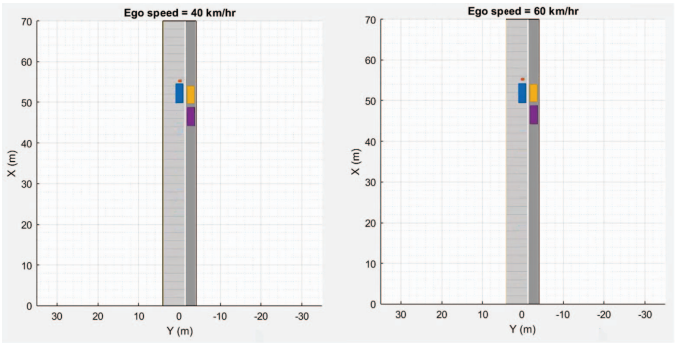


Fig. 7: **Execution of *Test_01* at Stage-II: The pedestrian always collides in front of the ego car, keeping the TCP constant.**

We have carefully reviewed and discussed all 15 collisions with the development team and found a common pattern, revealing the failure root cause. When the speed of the ego car changes, there are two possibilities for missing the time to the collision point: The ego car will arrive at the collision point before the actor and pass away, or the target actor will arrive at the collision point before the ego car and pass away. The system cannot apply brakes to avoid a collision in both cases.

We found that inside the *Simulink Decision logic block*, the value of road width is fixed (i.e., 3.6 m). The ego car is assumed to be in the middle of its lane. This causes the *call-back function* to mismatch the input values. For example, the road width is 10 m in the prebuilt scenario. However, inside the system, it was fixed at 3.6 m. To perform the simulation accordingly, the user needs to change the value manually. We understand that typically the length of a highway is 3.6 m. However, it is also possible that some roads in the real environment have different widths where the ADAS or AD system should behave accurately.

To verify the argument, we measured and randomly collected real road data from our surroundings and found that different widths exist and verified with the developer, who investigated and confirmed the shortcoming in the SUT by commenting: *"The mentioned parameterization of lane width aspect is currently not taken into consideration in the current AEB system, and the model pre-assumes the lane width as 3.6 m. However, we believe that more analysis needs to be done to handle the scenarios with varying lane widths that, includes complex scenarios like junctions where we don't have lane markings. We will consider updating the AEB model by handling this as an enhancement in the future."*

We are not sure if this limitation relates to the AEB systems or the platform itself. However, to increase the reliability of the AEB system, the development team should initiate a method or modifies the function to control the behavior of the actors at varying speeds.

One may argue that to avoid the collision, either increase the speed or change the start point of the pedestrian actor in the scenario. It should be noted. However, a change in speed is possible in the real environment when the situations change suddenly, and the ADAS systems behave incorrectly [7].

In conventional testing, it is difficult for the tester to judge whether the collision is avoidable due to the lack of a test oracle. However, utilizing MT to distinguish between avoidable and non-avoidable collisions [17], in this work, we can configure the collisions as avoidable and identify a failure caused by a system's limitations.

VI. Related Work

Modeling and simulations are among the growing number of software testing mechanisms where both verification and validation of the system can be done to overcome the oracle problem [18], [29]–[31] in different domains. However, it is very challenging to know that the simulation model was properly developed and to decide whether the test results are reliable and trustworthy. It is also becoming more popular

[7]https://www.youtube.com/watch?v=uEjwLKE0cCQ

for testing ADAS and AD systems. Therefore, researchers worldwide focus on finding bugs and their nature to expose harmful driving scenarios [17], [41], [43].

Similar to our approach, Klück et al. investigated a testing approach that could identify critical scenarios that may lead to unsafe SUT behavior [24]. Using genetic algorithms and a search-based approach, they identified specific configurations that could cause the AEB system to fail. However, only rely on limited factors, which gave us to carry out the system as a case with more factors following the principles of MT.

By combining metamorphic testing with another method, i.e fuzzing (scene switching), research was also conducted to test the performance of autonomous systems to distinguish critical and noncritical situations [17], [43]. The idea was inspirational and was adopted to demonstrate the approach in this empirical study.

Toohey et al. [36], focused on the transformations of images to train the neurons to investigate the impact of transformed inputs and to predict the steering angle (SA). This study was limited to the deep neural network (DNN) model with the two selected metrics for evaluation; however, the use of the transformation technique with MT was inspiring to indicate the system's weaknesses.

Zhou et al., applied geometric transformations to test software in completely different domains and noticed the great research serendipity in there studies [44].They further investigate and propose a more general concept of a *symmetry* metamorphic relation pattern to guide MR identification. The symmetry pattern refers to the existence of different viewpoints from which the system appears the same.

Tian et al. [35]; and Zhang et al. [41] generated MT-based synthetic driving scenes to test autonomous systems. However, their work is limited to the verification of ADAS or autonomous systems or subsystems. This helps us enhance the MT approach's effectiveness by integrating the safety standards to classify the identified scenarios to verify and validate the system performance.

The driving attributes can be extracted from high-definition (HD) maps data available on high digital map databases (e.g., Google Map, HERE Map, OpenStreetMap, etc.) [16]. These maps provide rich environmental information, such as static roads, buildings, and traffic infrastructures. The environmental information is useful for machine learning approaches [15], and for the vehicle to understand the driving functionalities, therefore contributing to accessing the automated vehicle behaviors.

Recent studies show that researchers are also trying to evaluate the safety of ADAS and AD systems using formal verification methods, which require a lot of effort in terms of time and cost [10], [25], [34], [39] but have a lack of empirical evaluations.

Our method in this paper shows the effectiveness of MT implementing Euro NCAP standards utilizing OpenStreetMap to test driving behaviors. We assessed the approach, executing the tests on both design and system levels and providing a way to replicate the real environment tests.

VII. Conclusion and future work

In this paper, we have presented a simulation-based MT testing framework to observe driving behaviors. The approach is evaluated at both system and design levels. We executed three tests, implementing the Euro NCAP standards to design the scenaros. In each test, an *"avoidable collision"* was recorded.

A mechanism was designed at the design phase and satisfied the metamorphic relation. However, MR satisfaction at the system level is expected in future releases.

We used the real road from OSM to design the test, presenting an effective and novel approach to replicating the real road driving test. We also discuss the property of the failures, which may or may not be a real-life issue for the system, but it helps us to understand the system more in-depth, referring to the concept of metamorphic exploration (ME) [44].

A threat to the validity of our conclusion is that we have evaluated the approach with the closed simulation platform, However, our results indicate the simplicity and effectiveness of the approach and are platform-independent; hence, it can be applied to verify and validate other ADAS and AD systems.

To increase the proposed approach's effectiveness, we plan to design more MRs and implement the approach with open-source driving platforms (e,g. Carla, AirSim, Baidu Apollo, esmini, etc.).

In the future, we plan to implement the approach in a real testing environment, collaborating with industries (like Euro NCAP, ANCAP[8], tfNSW[9], etc.). This research, therefore, also provides a cost-efficient and beneficial gateway for industries to integrate and evaluate MT principles with the safety standards and protocols for V&V of ADAS and AD systems.

Acknowledgments

This work was supported in part by the Higher Education Commission Pakistan through the University of Wollongong, Australia, under project HRD/UESTPs/UETs/Batch-VI-2018, under award number 4995, a linkage grant of the Australian Research Council (project ID: LP160101691), an Australian Government Research Training Program Scholarship, and a Western River entrepreneurship grant. We wish to thank Morphick Solutions Pty Ltd, Australia, and the MATLAB team for their support.

References

[1] ISO/PASS:21448 Road vehicles - Safety of intended functionality (SOTIF). Reference Number: ISO/PASS 21448:2019 (E), Accessed: January, 2019. [Online]. Available: https://www.iso.org/standard/70939.html.

[2] The European New Assessment Programme (Euro NCAP). Accessed: 2023. [Online]. Available: https://www.euroncap.com/en.

[8] urlhttps://www.ancap.com.au/

[9] https://www.transport.nsw.gov.au/

[3] J. M. Almendros-Jimenez, A. Becerra-Terón, M. Merayo, and M. Núnez, "Using metamorphic testing to improve the quality of tags in OpenStreetMap," *IEEE Transactions on Software Engineering*, 2022.

[4] J. M. Almendros-Jiménez, A. Becerra-Terón, M. G. Merayo, and M. Núñez, "Metamorphic testing of OpenStreetMap," *Information and Software Technology*, vol. 138, p. 106631, 2021.

[5] J. Ayerdi, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "QoS-aware metamorphic testing: An elevation case study," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 104–114.

[6] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[7] C. Becker, J. C. Brewer, L. Yount *et al.*, "Safety of the intended functionality of lane-centering and lane-changing maneuvers of a generic level 3 highway chauffeur system," United States. National Highway Traffic Safety Administration. Electronic, Tech. Rep., 2020.

[8] J. Brown, Z. Q. Zhou, and Y.-W. Chow, "Metamorphic testing of navigation software: A pilot study with Google Maps," in *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS-51)*, 2018, pp. 5687–5696, available: http://hdl.handle.net/10125/50602.

[9] G. Cao, F. Damerow, B. Flade, M. Helmling, and J. Eggert, "Camera to map alignment for accurate low-cost lane-level scene interpretation," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 498–504.

[10] L. Capito and K. A. Redmill, "Methodology for hazard identification and mitigation strategies applied to an overtaking assistant ADAS," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3972–3977.

[11] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing without the need of oracles," *Information and Software Technology*, vol. 45, no. 1, pp. 1–9, 2003.

[12] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[13] T. Y. Chen, F.-C. Kuo, W. Ma, W. Susilo, D. Towey, J. Voas, and Z. Q. Zhou, "Metamorphic testing for cybersecurity," *Computer*, vol. 49, no. 6, pp. 48–55, 2016.

[14] D. L. Fisher, J. K. Caird, M. Rizzo, and J. D. Lee, "Handbook of driving simulation for engineering, medicine and psychology: an overview," *Handbook of driving simulation for engineering, medicine, and psychology*, 2011.

[15] M. Hacar, "Analyzing the behaviors of OpenStreetMap volunteers in mapping building polygons using a machine learning approach," *ISPRS International Journal of Geo-Information*, vol. 11, no. 1, p. 70, 2022.

[16] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[17] J. C. Han and Z. Q. Zhou, "Metamorphic fuzz testing of Autonomous Vehicles," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW '20)*. ACM, 2020.

[18] M. Iqbal, J. C. Han, Z. Q. Zhou, and D. Towey, "Enhancing Euro NCAP Standards with Metamorphic Testing for Verification of Advanced Driver-Assistance Systems," in *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. IEEE, 2021, pp. 37–41.

[19] M. Iqbal, J. C. Han, Z. Q. Zhou, D. Towey, and T. Y. Chen, "Metamorphic testing of Advanced Driver-Assistance System (ADAS) simulation platforms: Lane keeping Assist System (LKAS) case studies," *Information and Software Technology*, vol. 155, p. 107104, 2023.

[20] ISO. ISO:26262 Road vehicles - functional safety. Accessed: December, 2018. [Online]. Available: https://www.iso.org/standard/68383.html.

[21] G. Jahangirova, A. Stocco, and P. Tonella, "Quality metrics and oracles for autonomous vehicles testing," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 194–204.

[22] S. Khastgir, S. Brewerton, J. Thomas, and P. Jennings, "Systems Approach to Creating Test Scenarios for Automated Driving Systems," *Reliability Engineering & System Safety*, vol. 215, p. 107610, 2021.

[23] D. Kibalama, P. Tulpule, and B.-S. Chen, "AV/ADAS Safety Critical Testing Scenario Generation from Vehicle Crash Data," SAE Technical Paper, Tech. Rep., 2022.

[24] F. Klück, M. Zimmermann, F. Wotawa, and M. Nica, "Genetic algorithm-based test parameter optimization for ADAS system testing," in *IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, 2019, pp. 418–425.

[25] A. Lyamani, T. Hajji, I. Elhassani, and T. Masrour, "Scenarios for ADAS Testing: Modeling and Design," in *International Conference on Digital Technologies and Applications*. Springer, 2022, pp. 753–762.

[26] MathWork. Euro NCAP Driving Scenarios in Driving Scenario Designer. Accessed: 1994-2022. [Online]. Available: https://au.mathworks.com/help/driving/ug/euro-ncap-driving-scenarios-in-driving-scenario-designer.html.

[27] Mathworks and Simulink. Math.Graphic.Programming. Accessed: 1994-2022. [Online]. Available: https://au.mathworks.com/products/matlab.html.

[28] P. Mooney, M. Minghini *et al.*, "A review of OpenStreetMap data," 2017.

[29] C. Murphy, M. S. Raunak, A. King, S. Chen, C. Imbriano, G. Kaiser, I. Lee, O. Sokolsky, L. Clarke, and L. Osterweil, "On effective testing of health care simulation software," in *Proceedings of the 3rd workshop on software engineering in health care*, 2011, pp. 40–47.

[30] M. Olsen and M. Raunak, "Increasing validity of simulation models through metamorphic testing," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 91–108, 2018.

[31] M. S. Raunak and M. M. Olsen, "Metamorphic Testing on the Continuum of Verification and Validation of Simulation Models," in *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. IEEE, 2021, pp. 47–52.

[32] F. A. Schiegg, J. Krost, S. Jesenski, and J. Frye, "A novel simulation framework for the design and testing of advanced driver assistance systems," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[33] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[34] J. Sini, M. Violante, V. Dodde, R. Gnaniah, and L. Pecorella, "A novel simulation-based approach for ISO 26262 hazard analysis and risk assessment," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 253–254.

[35] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the IEEE/ACM 40th International Conference on Software Engineering (ICSE '18)*. ACM, 2018, pp. 303–314.

[36] J. R. Toohey, M. S. Raunak, and D. Binkley, "From neuron coverage to steering angle: Testing autonomous vehicles effectively," *Computer*, vol. 54, no. 8, pp. 77–85, 2021.

[37] P. Valle, "Metamorphic testing of autonomous vehicles: a case study on Simulink," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2021, pp. 105–107.

[38] M. Waltz and O. Okhrin, "Two-Sample Testing in Reinforcement Learning," *arXiv preprint arXiv:2201.08078*, 2022.

[39] X. Xing, T. Zhou, J. Chen, L. Xiong, and Z. Yu, "A Hazard Analysis Approach based on STPA and Finite State Machine for Autonomous Vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 150–156.

[40] Y. Xu, Z. Q. Zhou, X. Zhang, J. Wang, and M. Jiang, "Metamorphic testing of named entity recognition systems: A case study," *IET Software*, pp. 386–404, 2022.

[41] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. 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 '18)*. ACM, 2018, pp. 132–142.

[42] J. Zhou, R. Schmied, A. Sandalek, H. Kokal, and L. del Re, "A framework for virtual testing of ADAS," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 9, no. 1, pp. 66–74, 2016.

[43] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Communications of the ACM*, vol. 62, no. 3, pp. 61–67, March 2019.

[44] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2018.

[45] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2016.