This is a repository copy of *Digital Twins Are Not Monozygotic – cross-replicating ADAS testing in two industry-grade automotive simulators*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/206013/

Version: Accepted Version

Proceedings Paper:
Borg, M., Abdessalem, R.B., Nejati, S. et al. (2 more authors) (2021) Digital Twins Are Not Monozygotic – cross-replicating ADAS testing in two industry-grade automotive simulators. In: 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST). 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST), 12-16 Apr 2021, Porto de Galinhas, Brazil. Institute of Electrical and Electronics Engineers (IEEE) , pp. 383-393. ISBN 9781728168371

https://doi.org/10.1109/icst49551.2021.00050

© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works. Reproduced in accordance with the publisher's self-archiving policy.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose
university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Digital Twins Are Not Monozygotic – Cross-Replicating ADAS Testing in Two Industry-Grade Automotive Simulators

Markus Borg*, Raja Ben Abdessalem†, Shiva Nejati‡†, François-Xavier Jegeden§ and Donghwan Shin†
*RISE Research Institutes of Sweden and Lund University, Lund, Sweden, markus.borg@ri.se
†University of Luxembourg, Luxembourg, Luxembourg. raja.benabdessalem@uni.lu, donghwan.shin@uni.lu
‡University of Ottawa, Ottawa, Canada, snejati@uottawa.ca
§ESI Group, Nantes, France, Francois-Xavier.Jegaden@esi-group.com

***Abstract*—The increasing levels of software- and data-intensive driving automation call for an evolution of automotive software testing. As a recommended practice of the Verification and Validation (V&V) process of ISO/PAS 21448, a candidate standard for safety of the intended functionality for road vehicles, simulation-based testing has the potential to reduce both risks and costs. There is a growing body of research on devising test automation techniques using simulators for Advanced Driver-Assistance Systems (ADAS). However, how similar are the results if the same test scenarios are executed in different simulators? We conduct a replication study of applying a Search-Based Software Testing (SBST) solution to a real-world ADAS (PeVi, a pedestrian vision detection system) using two different commercial simulators, namely, TASS/Siemens PreScan and ESI Pro-SiVIC. Based on a minimalistic scene, we compare critical test scenarios generated using our SBST solution in these two simulators. We show that SBST can be used to effectively generate critical test scenarios in both simulators, and the test results obtained from the two simulators can reveal several weaknesses of the ADAS under test. However, executing the same test scenarios in the two simulators leads to notable differences in the details of the test outputs, in particular, related to (1) safety violations revealed by tests, and (2) dynamics of cars and pedestrians. Based on our findings, we recommend future V&V plans to include multiple simulators to support robust simulation-based testing and to base test objectives on measures that are less dependant on the internals of the simulators.**

***Index Terms*—search-based software testing, advanced driver-assistance systems, automotive simulators, replication**

I. Introduction

There is a growing trend to increase the level of vehicle automation driven by the recent advances in technologies such as, among others, Machine Learning (ML) and Deep Neural Networks (DNN), computer vision, and sensor fusion. However, in parallel with this technological growth, there is an increase in the number of accidents and crashes that involve self-driving cars and pedestrians [1]. Many of these accidents are due to an interplay between software, often containing complex ML-based components, and advanced electronics, e.g., cameras and LiDAR technologies, that are used in today's modern vehicles. To prevent such accidents and crashes, there is a need to perform Verification and Validation (V&V) techniques for self-driving vehicles at *system-level* to ensure that they are safe and reliable before letting them drive on public roads [2].

Currently, in industry, the major bulk of system-level testing of self-driving vehicles is carried out through on-road testing or using naturalistic field operational tests. These activities, however, are expensive, dangerous, and ineffective [3]. A feasible and efficient complementary approach is to conduct system-level testing through computer simulations that can capture the entire self-driving vehicles and their operational environment using effective and high-fidelity physics-based simulators. There is a growing number of commercial and public-domain simulators that have been developed over the past few years to support realistic simulation of self-driving systems [4], [5], [6]. In the ISO/PAS 21448 Safety of the Intended Function (SOTIF) candidate standard [7], an ongoing standardization initiative covering automotive ML, simulation is recognized as one of the main V&V means for self-driving cars. This has led to the development of a large number of system-level testing approaches in the literature that rely on such simulators.

Existing testing techniques are often focused on devising algorithms and techniques to generate test cases [8], [9], [10] or to generate test oracles [11]. There is, however, little research on studying the role of simulators when testing is based on a simulation environment. Recently, Sotiropoulos *et al.* [12] provided an empirical study comparing testing results of robot function models obtained based on a simulator with those obtained from their physical field testing. Ul Haq *et al.* [13] and Codevilla *et al.* [14] compare testing of DNN-based automated driving systems based on real-world and simulator-generated images and videos. We pose a cornerstone question that has not been previously studied in the simulation-based testing literature: Can we obtain similar or consistent test results from different simulators? Answering this question requires replicating testing techniques in different simulators and studying the results. We refer to such studies as *cross-simulator (X-sim)* replications.

Ben Abdessalem *et al.* have conducted several studies on ADAS testing using the simulator TASS/Siemens PreScan [10], [15], [9]. These papers show how Search-Based Software Testing (SBST) [16] can be used to effectively find

input values to generate test scenarios that stress individual ADAS components [10]. The stress test scenarios, which are also referred to as critical test scenarios, are obtained such that they break or are close to breaking safety requirements of the ADAS under test, and hence, result in a *safety violation*.

In this paper, we investigate if the results obtained from ADAS testing are consistent across different simulators. To this end, we present a X-sim replication study in which we ported the solution by Ben Abdessalem *et al.* [10] to the ESI Pro-SiVIC simulator [17] which is an alternative commercial automotive simulator. The *original study* applies SBST to an ADAS example, i.e., the Pedestrian Detection Vision based system (PeVi). Specifically, the original study is focused on testing PeVi using simulations capturing the ego car (i.e., the car augmented with ADAS) driving on a straight urban street and a pedestrian crossing the street from the right. We adhere to the definitions of *scene* and *scenario* proposed by Ulrich *et al.* [18], which are also used in SOTIF. A scene is "a snapshot of the environment including the scenery, dynamic elements, and all actor and observer self-representations, and the relationships between those entities". A scenario describes "the temporal development between several scenes in a sequence of scenes". In collaboration with the original authors, we simplified the PreScan scene that was used for test generation to support porting to Pro-SiVIC with minimal differences. By controlling as many variables as possible related to the (initial) scene, we focus this study to compare the scenarios generated based on the initial scene. In addition, we ported PeVi to Pro-SiVIC so that the replication and the original study use the same ADAS under test.

In line with the terminology used by Cartwright [19] and Gomez *et al.* [20], we refer to our work as a series of *reproductions*. Three research questions guide our study:

RQ1 Is SBST an effective approach to ADAS testing if we replace PreScan with Pro-SiVIC?

RQ2 Is the diagnostic information obtained by applying SBST using PreScan reproducible if we use Pro-SiVIC?

RQ3 Given a minimalistic scene, to what extent can critical test scenarios identified in PreScan be reproduced in Pro-SiVIC, and vice versa?

Our results show that SBST can be used to effectively generate critical test scenarios in both simulators, and the test results obtained from the two simulators can reveal several weaknesses of PeVi (the ADAS under test). However, the test scenarios obtained by PreScan and Pro-SiVIC do not lead to consistent and conclusive characterizations of safety violations for PeVi. In particular, the only consistent diagnostic information that we identify in our study is that, in both PreScan and Pro-SiVIC, PeVi likely violates its safety requirement when the car moves fast (more than 72 km/h). Finally, reproducing critical scenarios between PreScan and Pro-SiVIC can result in discrepancies that might not only be due to the implementation of PeVi, but can originate in differences in the dynamic models of the simulators or the off-the-shelf sensors available in the simulators' libraries. This research concludes by two lessons-learned and recommendations that have the potential to influence future simulation-based testing of ADAS.

Paper organization. Section II presents related work on ADAS testing and introduces the original study. The process of porting scenes to Pro-SiVIC is described in Section III. Section IV explains the research method and Section V presents the results. Finally, Section VI discusses the main threats to validity, Section VII provides lessons learned, and Section VIII concludes the paper.

II. Background and Related Work

This section presents a brief overview of related work and details about the original study reproduced in this paper.

A. Simulation-based CPS and ADAS Testing

Digital twins [21] are defined as digital and virtual representations of physical assets enabled through data and simulators for monitoring, controlling, optimization and verification purposes. There is an increasing demand for fast, agile and high fidelity digital twins in the domain of cyber physical systems (CPS). For ADAS and self-driving systems, there is even a higher demand for digital twins and simulators since real-world testing and verification of such systems is expensive, dangerous, and ineffective. Various simulators such as those relying on physics-based modeling (e.g., Pro-SiVIC and PreScan) or those that rely on game engines (e.g., [8], [22]) have been used for testing of self-driving systems and ADAS. Due to the large search space for ADAS and self-driving systems, achieving any form of coverage over the space of all possible simulation scenarios is rather infeasible. Hence, search-based software testing has been advocated as an effective and efficient strategy to generate test scenarios for such systems when they are tested within a simulation environment [8], [22], [10], [23]. While the focus of the current research is on devising testing techniques, in this paper, we evaluate the impact of simulators on the test results through a replication study performed using two physics-based ADAS simulators: PreScan and Pro-SiVIC.

B. Description and Definitions of the Original Study

The original study used SBST and PreScan to test PeVi as part of an industrial ADAS case study [10]. In this section, we provide details of the original work.

Study subject. Briefly, PeVi's function is to determine whether there is any pedestrian in a rectangular *Acute Warning Area* (AWA) in front of the car, and if so, it shows a warning message to the driver. The size of the AWA depends on the speed of the car and the shape of the road. Figure 1 shows the AWA for a car driving on a straight road. PeVi uses data received from a sensor component to identify the position and the speed of the objects in or near the AWA. It also receives the Time To Collision (TTC) as computed by the sensor component. TTC measures the time until impact between the ego car and an object if both continue with the same velocities [24]. When an object is detected in or near AWA ($\leq 0.2m$ from the boundaries), and when the TTC is below a defined threshold, the object position is sent to the camera to

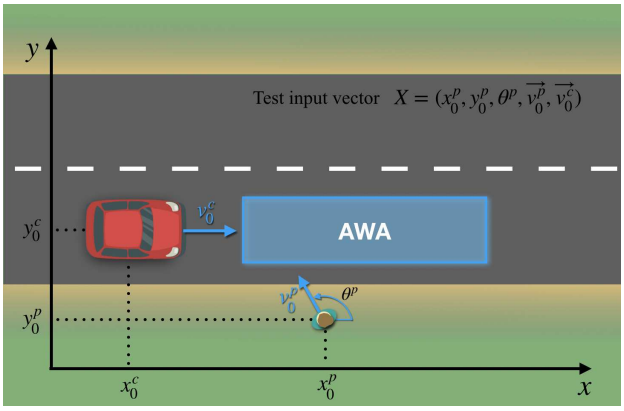


Fig. 1. Input variables in the initial scene.

detect object types and shapes after receiving their positions from the sensor component. Specifically, the vision component determines whether the object is a pedestrian. Then, PeVi will show a warning message to the driver indicating that the car may risk a collision with a pedestrian.

Safety Requirement. The test generation aims to verify the following functional safety requirement of PeVi: *"PeVi shall detect pedestrians in or near AWA ($\leq$ 0.2m from the boundaries) when there is a risk of collision with the pedestrians and when the pedestrians are close to the car"*.

This requirement originated with customers (car manufacturers) where the statement "there is a risk of collision with the pedestrians and when the pedestrians are close to the car" was not detailed. As we describe later, the above requirement is detailed and formalized using quantifiable fitness functions through interactions with the engineers who developed PeVi.

Scope of Testing. A number of simulation-based testing studies [9], [15], vary both static elements (e.g., different weather conditions, different road shapes, and different background scenes) and dynamic elements of simulators (e.g., the speed, the position, and the trajectory of cars and pedestrians). The original study, however, fixed the initial scene to include the ego car driving on a straight urban street and a pedestrian crossing the street from the right. The test generation then focuses on varying the dynamics, namely, the speed of the car, and the speed, position and orientation of the pedestrian. Replicating this study (using an even further simplified scene) allows us to focus on comparing the generated scenarios within a plain and simple scene.

Note that we have to include PeVi in a simulation environment and perform system-level testing to verify PeVi against its safety requirement. However, the faults identified using system-level testing may not necessarily be due to faults or errors in PeVi's implementation and may be due to errors in the simulators or in third party models of hardware components (e.g., sensors and cameras), or due to the real world and physical constraints [15].

Input Representation. According to the original study, the test input space of PeVi, which is also depicted in Figure 1, consists of vectors $(v_0^c, x_0^p, y_0^p, \theta^p, v_0^p)$ where v_0^c is the car speed, x_0^p and y_0^p specify the (initial) position of the pedestrian, θ^p is the orientation of the pedestrian, and v_0^p is the pedestrian speed. Note that the initial car position is fixed at (x_0^c, y_0^c). The variables in the search space are further constrained as follows: $1 \leq v_0^c\ (m/s) \leq 25$; $x_0^c + 20 \leq x_0^p\ (m) \leq x_0^c + 85$; $y_0^c - 15 \geq y_0^p\ (m) \geq y_0^c - 2$; $40 \leq \theta^p(^\circ) \leq 160$ and $1 \leq v_0^p\ (m/s) \leq 5$. Each value assignment to the vector $(v_c, x_0, y_0, \theta, v_p)$ represents a test input for PeVi.

Fitness Functions. SBST exercises PeVi with respect to its requirement guided by minimizing three fitness functions:

FF1 The minimum distance between the ego car and the pedestrian over the test scenario.

FF2 The minimum distance between the AWA and the pedestrian over the test scenario.

FF3 The minimum TTC between the ego car and the pedestrian over the test scenario.

The outputs of each test scenario (simulation) include a vector of distances at each simulation time step between: the ego car and the pedestrian and the AWA and the pedestrian, as well as a vector of TTCs at each simulation time step between the ego car and the pedestrian. We select the minimum value of these vectors to compute the fitness functions. Note that test scenarios do not stop upon detection by PeVi. We run each test scenario for a time duration and stop them when any of these conditions holds: (1) the car has driven 100 m (i.e., the length of the road segment under analysis), or (2) the pedestrian has crossed the road, or (3) the car has passed the pedestrian (either there was a collision or the pedestrian did not yet reach the road). The original study discusses how *minimizing* the above three fitness functions pushes PeVi into breaking its requirement.

The Computational Search Algorithm. The search algorithm used to test PeVi is the Non-dominated Sorting Genetic Algorithm version 2 (NSGA-II) [25], a well-known multi-objective search algorithm that has been used in many different domains. Note that we need to use a multi-objective search algorithm to test PeVi, since breaking the safety requirement of PeVi requires us to minimize the three fitness functions defined above. The following summarizes the choice of operations and parameters of NSGA-II used in the original study [10]:

- Selection. We use a binary tournament selection with replacement that has been used in the original implementation of the NSGA-II algorithm.
- Crossover. We use the Simulated Binary Crossover operator (SBX). SBX creates two offsprings from two selected parent individuals. The difference between offsprings and parents is controlled by a distribution index (η): When η is large, the offsprings are closer to the parents, while a small η increases the difference. Analagous to the original study, we chose a high value for η (i.e., $\eta = 20$) based on the guidelines by Deb and Agrawal [25].
- Mutation. Mutation is applied after crossover to the genes of the children chromosomes with a certain probability (mutation rate). Given a gene x (i.e., any of the variables v_0^c, x_0^p, y_0^p, θ^p, v_0^p), our mutation operator shifts x by

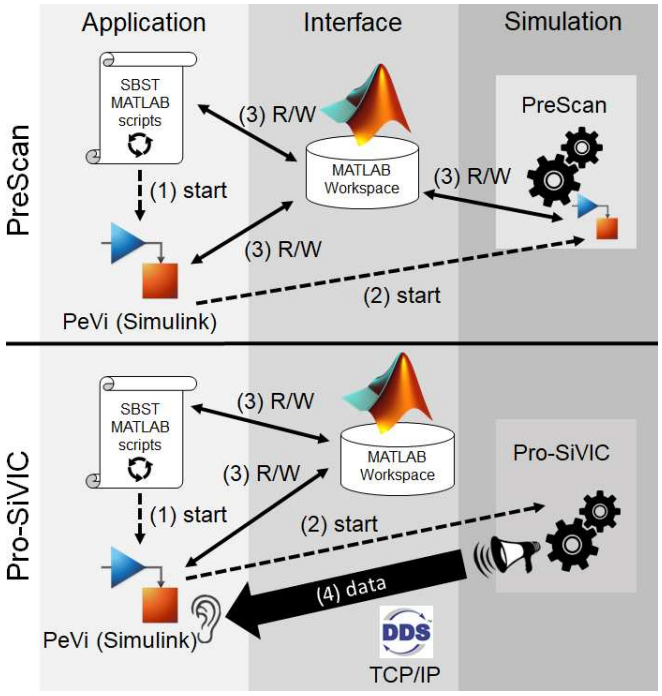


Fig. 2. Simulation setups used for PreScan and Pro-SiVIC. Dashed arrows depict function calls that happen once per scenario, solid arrow show read/write operations to shared variables, and the big arrow represents DDS communication.

a value selected from a normal distribution with mean $\mu = 0$ and variance σ^2. To avoid invalid offsprings from crossover or mutation, we use cutoffs corresponding to the end points of the ranges.

The NSGA-II search parameters were selected as follows: the crossover rate was set to 0.9, the mutation rate to 0.5, and the population size to 10. In this replication study, we reuse the same search parameters as in the original study.

Testing Time Budget. In the original study, NSGA-II ran within a restricted execution time budget of 150 min. The time budget was selected in consultation with the supplier of PeVi. The experiments reported in the original study show that the time budget was sufficient to find failure revealing test scenarios and also to demonstrate that NSGA-II outperforms random search testing (the sanity check experiment in SBST [26]). We use the same time budget for the replication study.

Critical and Safety Violation Scenarios. In this replication study, we discuss critical scenarios and safety violations using the following definitions. A *critical* scenario either results in a collision between the ego car and the pedestrian or a near miss. A near miss is defined as a scenario with FF1 $\leq$ 1 m or FF3 $\leq$ 0.5 s. In addition, we define a *safety violation* as a *critical* scenario where PeVi has failed to detect the pedestrian. Note that in critical scenarios, PeVi may or may not have detected the pedestrian. We are interested in generating both critical and safety violation scenarios. While safety violation scenarios indicate clear violations of the PeVi requirement, critical scenarios represent situations where the car and pedestrian may have a collision or a near miss. For a critical scenario, even if there is a detection, it is important to know the time gap between the detection and collision (or near miss) to determine if the gap is sufficiently large so that the driver can react and avoid the collision.

III. Porting from PreScan to Pro-SiVIC

Together with the lead developer of the original study [10], we ported the SBST algorithm as well as the PeVi component from PreScan to Pro-SiVIC. The process required considerable engineering effort, made possible through physical co-location during a two month research visit. This section describes similarities and differences between the two simulation setups.

Figure 2 depicts the simulation setups used for ADAS testing with PreScan and Pro-SiVIC, respectively. The figure is organized into three layers:

- **Application** – MATLAB/Simulink implementations of the SBST algorithm and PeVi.
- **Interface** – The interface between the simulator and both the SBST algorithm and PeVi.
- **Simulation** – The simulator tool which includes mechanisms to construct initial scenes as well as physics-based and mathematical models that simulate sensors and dynamic objects such as cars and pedestrians.

For the *Application* layer, porting the original implementation of the SBST algorithm was straightforward. For both PreScan and Pro-SiVIC, MATLAB scripts implement the NSGA-II algorithm and call the Simulink model of PeVi to initialize it with specific test inputs once per test scenario. The PeVi model, in turn, calls the simulator to start generating the output corresponding to the given test input (see links labelled (1) and (2) in Figure 2). In the Application layer, there is a one-to-one mapping between the elements used in the PreScan setting and those used in the Pro-SiVIC setting. PeVi from the original study was reused without modifications in the replication study. Still some engineering work was needed, primarily in relation to configuring the sensor model of Pro-SiVIC and some data type conversion to ensure that Pro-SiVIC could generate the input formats required by PeVi.

The main differences between the PreScan and Pro-SiVIC setups are related to the *Interface* layer. PreScan uses Simulink internally for both modeling the physics and the motion behavior of vehicles and pedestrians as well as sensor modeling – a local Simulink installation is even a prerequisite to run PreScan. As a result, external Simulink models (such as the model of PeVi) can easily be integrated with PreScan since they can read and write to shared variables in the same MATLAB workspace (see the links labelled (3) in Figure 2).

In contrast, Pro-SiVIC does not depend on Simulink for internal modeling. In Pro-SiVIC, elements communicate through the Data Distribution Service (DDS) [27], a message-based middleware protocol implementing a publish-subscribe pattern (see the link labelled (4) in Figure 2). Hence, PeVi communicates with the internal models of PreScan synchronously, while the DDS-based communication between PeVi and Pro-SiVIC is asynchronous. To initiate the communication, the external Simulink model (PeVi) starts the Pro-SiVIC scenario,

Fig. 3. Crossing pedestrians in PreScan and Pro-SiVIC. Note that we later disabled shadows in PreScan.

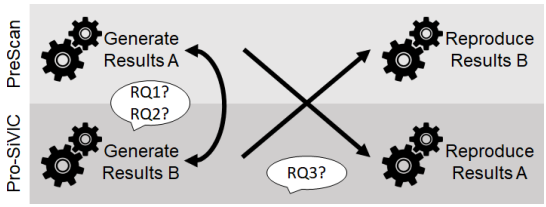


Fig. 4. Comparisons relevant to the RQs.

and then, each DDS enabled element in Pro-SiVIC (e.g., the sensors and the car) begins broadcasting DDS messages to the subscribing Simulink blocks of PeVi each 40 ms (25 Hz).

The Simulink simulation was not fast enough to receive DDS messages at this frequency. We measured a uniform packet loss of 20%, i.e., roughly every fifth DDS message was not received by PeVi. This means that 20% of the DDS messages corresponded to an 80 ms measurement interval instead of 40 ms. To mitigate the risk of losing DDS messages containing data with a minimum FF, each Pro-SiVIC scenario was repeated 20 times and the mode of the FF measurements (i.e., the most frequently generated outputs) were used for the subsequent analysis. We found that this mitigation strategy generated simulation results that were not impacted by the packet loss. Indeed, in these scenarios, due to the high frequency of the messages sent from Pro-SiVIC, the content of the lost messages were redundant or were very similar to the messages coming immediately before or after them and processed by the Simulink model.

For the *Simulation* layer, there are inevitable differences between the initial scenes in PreScan and Pro-SiVIC. As we discussed in Section II-B, one of the reasons that we choose the study of Ben Abdessalem [10] for replication is because this study is mainly focused on varying simulation dynamics (i.e., the position and speed of objects) and the background scene is unchanged over different test scenarios. To mitigate the potential threats to internal validity and to make sure that we compare the dynamic behavior of simulators rather than their motifs and initial scene construction abilities, we reduced the complexity of the scene of PreScan used in the original study [10], i.e., we created a novel *minimalistic* PreScan scene with removed buildings along the road and no shadows from the pedestrian. In PreScan, we built the minimalistic scene from scratch. On the other hand, Pro-SiVIC scenes are typically built using existing road snippets or adapted from pre-made standard scenes. Thus, we implemented the minimalistic scene using a straight road segment from the standard scene "horsering-ground" with a similar skydome and illumination settings as in the PreScan scene.

While we attempted to create equivalent initial scenes, some differences are obvious, including the visual appearance of approaching pedestrians as shown in Figure 3. We report three major differences:

1) The default pedestrian in PreScan is male, whereas the Pro-SiVIC pedestrian is female. The man runs with a swinging arm movement while the woman pumps the arms like a sprinter. Furthermore, the pedestrians wear different clothes.
2) The horizon is visible in the PreScan scene, while in Pro-SiVIC, it is occluded by mountains in the distance.
3) The road in the Pro-SiVIC scene has a narrow dirt shoulder, but the PreScan scene has no shoulder at all.

Finally, we used the same test input characterization with the same constraints and ranges when generating test scenarios for both PreScan and Pro-SiVIC. However, as the coordinates for the initial position of the car differ between the two scenes in PreScan and Pro-SiVIC, we implemented a translation function. Furthermore, similar translations were needed for the orientation of the pedestrian and conversions between m/s and km/h. All source code for the replication study is available on GitHub under a BSD 2-Clause [28]. The source code and data related to the original study is available on BitBucket [29].

IV. Research Method

This section describes the design of the empirical study.

A. Experimental Design

Figure 4 shows an overview of our experimental process. While we describe our empirical work as sequential steps, most experiments with PreScan and Pro-SiVIC were conducted in parallel – typically running overnight due to long execution times.

1) RQ1 – X-sim Reproduction of Principal Findings: RQ1 concerns the high-level replicability of the original study [10]. Can we show that SBST enabled by NSGA-II is an effective approach to ADAS testing even if we replace PreScan with another simulator?

To answer this question, we executed both the PreScan and the Pro-SiVIC setups for 40 times to account for the randomness in the NSGA-II algorithm. For each run of each setup, we used the testing time budget of 150 min from the original study. Note that each run of NSGA-II, being a multi-objective algorithm, generates 10 solutions (i.e., equivalent to the population size for NSGA-II provided in Section II-B). Thus, we obtained 400 scenarios in total.

To answer RQ1, we analyze the outputs from the PreScan and Pro-SiVIC setups to compare the quality of the generated test cases. In particular, we want to determine if SBST can generate fault revealing and critical test cases in both setups. We use two types of metrics for this purpose: (1) The number of test scenarios representing a critical or safety violation situation (see Section II-B for the definitions of critical and safety violations). (2) An assessment of the NSGA-II outputs using the hypervolume (HV) indicator [30]. The HV indicator has been commonly used in the literature (including the original study [10]) to evaluate multi-objective search algorithms since their outputs create a Pareto front [25]. Briefly, HV represents the size of the space covered by members of a Pareto front generated by a search algorithm [30]. The higher the HV values, the better the Pareto front outputs are. To compare the statistical differences in HV values generated by PreScan and Pro-SiVIC, we use the Mann–Whitney U test at $\alpha = 0.05$.

2) RQ2 – X-sim Reproduction of Diagnosis Information: While RQ1 is focused on the reproduction of test outputs in the two simulators, RQ2 investigates the consistency of the diagnostic information that can be derived from the test inputs generated by the application of SBST in the two simulators. In general, there is little research on producing diagnosis or debugging support for self-driving systems and ADAS. One proposed approach is to apply classification decision trees to identify conditions on test inputs that best explain and characterize failures [15]. Decision tree learning is a supervised learning classification technique [31]. To answer RQ2, we use the same results generated by the experiment we performed for RQ1. But this time, we study the distributions of the test inputs, and further, we use a decision tree classifier to infer conditions on the test inputs that can best characterize safety violations in the two simulators.

3) RQ3 – X-Sim Reproduction of Critical Test Scenarios: RQ3 addresses the reproduction of test scenarios in another simulator. If a scenario is found to be critical in PreScan, will the same scenario also be critical in Pro-SiVIC and vice versa? Recall that Section II-B provides the definitions for critical scenarios and safety violations. Our goal is to understand to what extent test inputs leading to critical scenarios or safety violations in one simulator remain critical or yield safety violations when executed in another simulator.

For this question, we converted the test inputs corresponding to the 400 test scenarios generated in PreScan for RQ1 to their Pro-SiVIC counterparts (as described under the Application layer in Section III). The converted test scenarios were then executed in Pro-SiVIC. Then, we repeated the analogous procedure to re-execute the scenarios generated by Pro-SiVIC in PreScan.

To answer RQ3, we analyze the outputs from PreScan and Pro-SiVIC from two perspectives: (1) the fraction of safety violations that remain after X-sim reproduction and if any new appear, and (2) the absolute differences of the results from the three fitness functions (FF1, FF2, and FF3) when reproducing scenarios across simulators.

B. Hardware and Software Setups

The PreScan and Pro-SiVIC setups used standalone licenses linked to the physical MAC addresses of specific devices. As the simulator vendors granted licenses to different organizations, we were not able to install a license server accessible over an internal network to execute PreScan and Pro-SiVIC on the same device. As a result, we conducted the simulations on separate computers. While this might introduce confounding factors, we believe this does not have an impact on our conclusions since our analysis is not focused on computational performance. In particular, in our analysis, we do not compare the time performance of the two simulator setups. The setup used to run the PreScan experiments was a MacBook Pro with a 2.5 GHz CPU and 16 GB RAM with PreScan version 2019.1 and MS Windows 10. We conducted the Pro-SiVIC experiments on a desktop PC running MS Windows 10 equipped with an Intel Core i7-3770 CPU @ 3.40 GHz, 32 GB RAM, and an Nvidia 1080Ti graphics card. The software version used was ESI Pro-SiVIC 2018.0.

V. Results and Discussion

This section presents results from the X-sim reproductions and discuss their practical implications.

A. RQ1: X-sim Reproduction of Principal Findings

Figure 5 (the left part) presents the number of critical scenarios and safety violations generated in PreScan and Pro-SiVIC (see Section II-B for the definitions of safety violation and critical scenarios). All 800 scenarios generated by SBST are critical, i.e., FF1 $\leq$ 1 m or FF3 $\leq$ 0.5 s in all scenarios.

Among the 400 scenarios generated by PreScan and Pro-SiVIC, 229 (57.3%) and 236 (59.0%) scenarios led to safety violations, respectively. For the remaining 171 (42.8%) scenarios of PreScan and the remaining 164 (41.0%) scenarios of Pro-SiVIC, the pedestrian was detected by PeVi. However, in all those non-safety violation scenarios, the simulators still recorded collisions between the car and the pedestrian. Note that PeVi only provides a warning and does not apply any braking. More precisely, we found that 396 out of 400 (99.0%) of the PreScan scenarios resulted in collisions between the car and the pedestrian. For Pro-SiVIC, the corresponding figure was 345 (86.3%).

The right part of Figure 5 shows the distributions of the HV indicators computed based on the Pareto front outputs obtained from different runs of NSGA-II in PreScan and Pro-SiVIC, respectively. There is no statistically significant difference between the two HV distributions, indicating that the quality of the Pareto front outputs obtained from PreScan and Pro-SiVIC are comparable.

> The principal findings from the original PreScan study can be reproduced using Pro-SiVIC. SBST is an effective approach to ADAS testing and the quality of the generated scenarios is comparable across simulators.

	PreScan	Pro-SiVIC
#Critical	400	400
#Safety Violations	229	236
#Detections	171	164
#Collisions	396	345

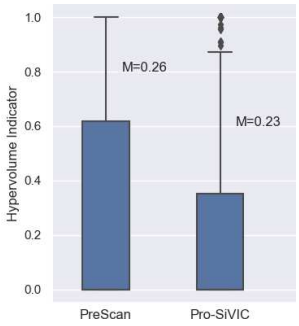


Fig. 5. Overview of the 800 generated test scenarios by PreScan and Pro-SiVIC. The boxplots show the hypervolume indicators, reflecting the quality of the Pareto fronts.

B. RQ2: X-sim Reproduction of Diagnosis Information

Figure 6 depicts swarm plots for the input parameters of the 800 test scenarios generated using SBST in PreScan and Pro-SiVIC, respectively. The red points represent safety violations (229 for PreScan and 236 for Pro-SiVIC). Note that box plots are not an appropriate visualization format, as some distributions are not only skewed but there are also considerable gaps in the data resulting in multimodal distributions. As Figure 6 shows, SBST found effective test inputs (i.e., test inputs revealing critical behaviors of PeVi) in most areas of the input space, but there is a notable exception. Using PreScan, no effective test scenarios involved the car driving slower than 10 m/s (v_0^c), i.e., 36 km/h. Moreover, for both simulators, there are certain parameter ranges that are considerably sparser compared to the rest the input space, e.g., $55 \leq x_0(\text{m}) \leq 68$ in Pro-SiVIC and $y_0 \leq 38(\text{m})$ in PreScan. Those ranges are not consistent between PreScan and Pro-SiVIC, illustrating internal variations when generating test scenarios using different simulators.

Further analysis of Figure 6 shows that diagnostic information obtained by PeVi testing are different in the two simulators. Specifically, in that figure, red points show test inputs resulting in safety violations. The distribution of red points show how the PeVi safety violations cluster for some parameter ranges – but again, the results are not consistent between the two simulators. For example, Pro-SiVIC testing with $15 \leq v_c\ (m/s) \leq 17$ result in many safety violations, for which the PreScan counterpart paints a different picture. Another divergent example is visible at $v_p \approx 4.4$ m/s, where Pro-SiVIC identifies safety violations but PreScan does not.

Figure 7 displays two decision trees that we have built based on the 400 test inputs generated by PreScan (a) and Pro-SiVIC (b). The test inputs for both simulators are labelled by True (when they lead to a safety violation) and by False, otherwise. For example, Figures 7(a) shows that in 181 safety violation scenarios generated by PreScan, the speed of the car was more than 18.92 m/s (68 km/h), and in 45 other safety violation scenarios, the speed of the car was less than 18.92 m/s (68 km/h) but the orientation of the pedestrian was more than 120.24°. The latter conditions, however, characterize a leaf of the tree with a mix of True and False-labelled output, and hence, cannot be taken as a characterization of failures. Similarly, Figure 7(b) shows that for Pro-SiVIC the majority of safety violations are characterized by the conjunction of two conditions: θ ($\geq 100.52°$) and $v^p \geq 1.17m/s$.

Overall, based on the test results in Figure 6 and the decision trees in Figures 7(a) and (b), we can identify only one condition that can consistently characterize safety violations identified by both Pro-SiVIC and PreScan. In particular, in both PreScan and Pro-SiVIC, PeVi performs worse when the car moves fast. More specifically, when $v_c \geq 20$ m/s (72 km/h), all test scenarios generated by PreScan and Pro-SiVIC lead to safety violations. However, apart from this condition, there are few patterns that can characterize safety violations for PeVi.

> The results obtained by PreScan and Pro-SiVIC do not generally lead to consistent and conclusive characterizations of safety violations for PeVi. The only consistent conclusion is that PeVi likely violates its safety safety requirement when the car moves fast ($\geq$ 72 km/h).

C. RQ3: X-Sim Reproduction of Critical Test Scenarios

Recall from Section V-A that all the 800 scenarios generated by PreScan and Pro-SiVIC were critical but some led to safety violations and some did not (see the definitions of critical and safety violations in Section II-B). To simplify the discussion, we refer to scenarios as either *unsafe* (when they lead to a safety violation) or *safe* otherwise. Figure 8 displays the results from X-sim reproduction of critical scenarios between PreScan and Pro-SiVIC. Before discussing the figure, we present the six possibilities that can happen when executing a critical scenario generated by one simulator (SimA) in another simulator (SimB). The references in parentheses below refer to rows in Figure 8.

- An unsafe scenario in SimA can: (1a) also be unsafe in SimB (the detection failed in both SimA and SimB); (1b) be critical but become safe in SimB (the detection failed only in SimA); and (1c) be non-critical in SimB (in SimB, neither FF1 nor FF3 is small enough to warrant the scenario as critical).
- A safe scenario in SimA can: (2a) be unsafe in SimB (the detection failed only in SimB); (2b) be both critical and safe in SimB (the detection works in both SimA and SimB); and (2c) be non-critical in SimB (same reason as in 1c).

As shown in Figure 8, after reproducing the scenarios generated by PreScan in Pro-SiVIC, we obtain the following results: Out of the 229 unsafe scenarios generated by PreScan, 78 (34.1%) are unsafe (1a) and 151 are safe (65.9%) (1b+1c) in Pro-SiVIC, but 45 of these 151 scenarios are still critical (1b) in Pro-SiVIC and the remaining 106 scenarios turn out to be non-critical and safe (1c) in Pro-SiVIC. Specifically, in the 45 scenarios (1b), the PeVi detection fails in PreScan but works in Pro-SiVIC, and in the 106 scenarios (1c), the distances between the ego car and the pedestrian in time and space in Pro-SiVIC are large enough to no longer constitute a critical scenario.

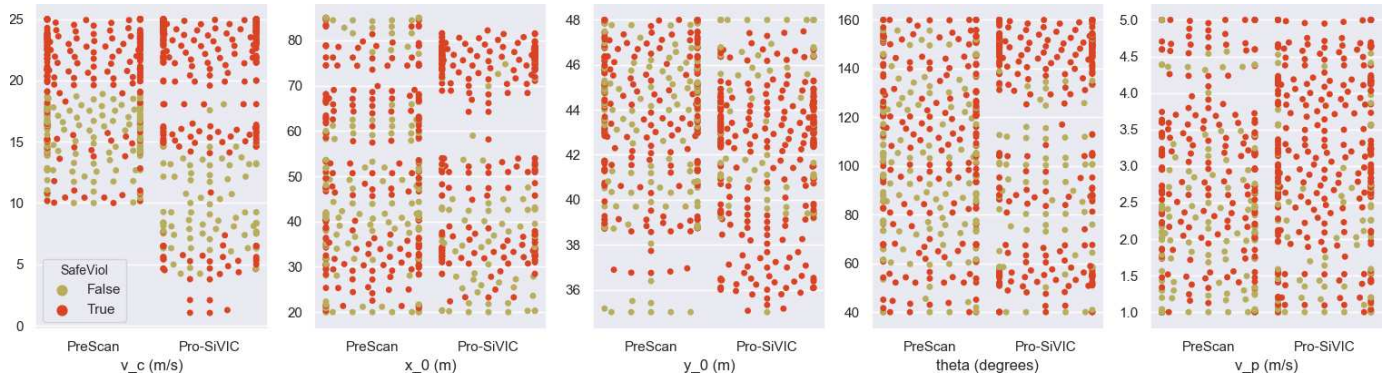


Fig. 6. Distribution of input parameters ($v_0^c, x_0^p, y_0^p, \theta^p$, and v_0^p) for the 800 test scenarios generated by applying SBST in PreScan and Pro-SiVIC. Red points denote safety violations (229 in PreScan and 236 in Pro-SiVIC).

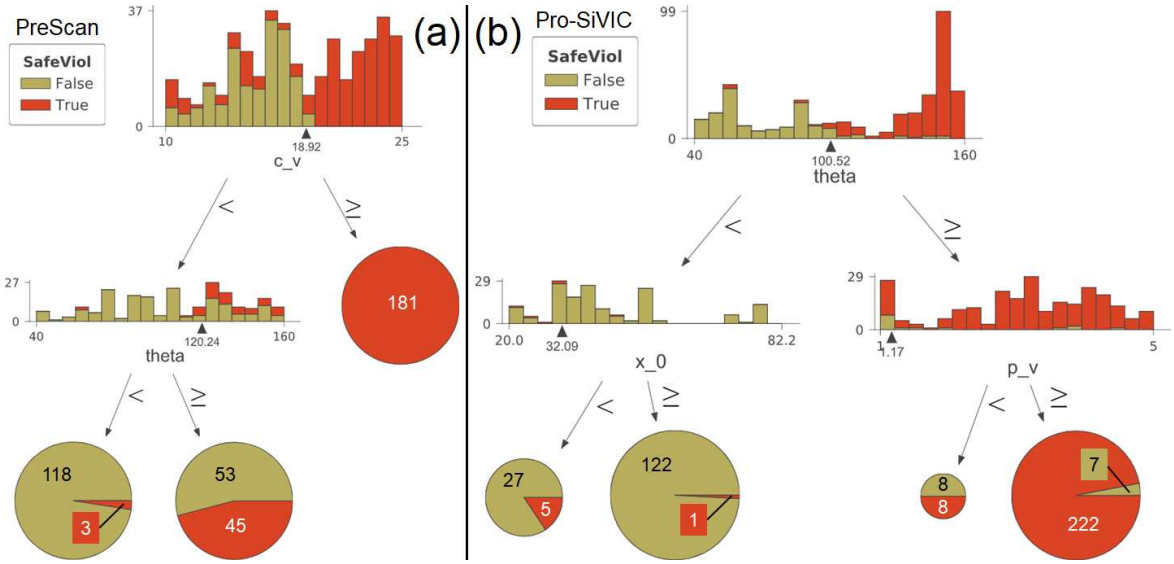
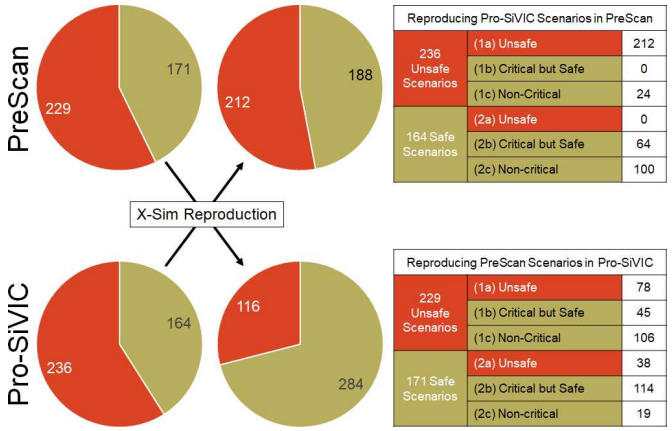


Fig. 7. Decision trees explaining when safety violations occur in PreScan (a) and Pro-SiVIC (b), respectively.



Reproducing Pro-SiVIC Scenarios in PreScan		
236 Unsafe Scenarios	(1a) Unsafe	212
	(1b) Critical but Safe	0
	(1c) Non-Critical	24
164 Safe Scenarios	(2a) Unsafe	0
	(2b) Critical but Safe	64
	(2c) Non-critical	100

Reproducing PreScan Scenarios in Pro-SiVIC		
229 Unsafe Scenarios	(1a) Unsafe	78
	(1b) Critical but Safe	45
	(1c) Non-Critical	106
171 Safe Scenarios	(2a) Unsafe	38
	(2b) Critical but Safe	114
	(2c) Non-critical	19

Fig. 8. Results from X-Sim reproduction of critical scenarios between PreScan and Pro-SiVIC.

Among the 171 safe but critical scenarios generated by PreScan, 133 (78.8%) are safe (2b+2c), out of which 114 (2b) are still critical while 19 (2c) are no longer critical. The remaining 38 scenarios (2a) change from being safe (but critical) in PreScan to unsafe in Pro-SiVIC, indicating that PeVi detected the pedestrian in PreScan but failed in Pro-SiVIC. In short, the discrepancies in X-sim reproduction of critical scenarios from PreScan and Pro-SiVIC were due the following factors: (1) Inconsistencies in detecting the pedestrian (for 45 scenarios, the PeVi detection worked in Pro-SiVIC but failed in PreScan; and in 38 scenarios, the detection worked in PreScan but not in Pro-SiVIC); (2) Changes in the distances between the ego car and the pedestrian in time and space. Specifically, 125 scenarios (1c+2c) that were critical in PreScan turned out to be non-critical in Pro-SiVIC.

When reproducing the 400 Pro-SiVIC scenarios in PreScan, among the 236 unsafe scenarios in Pro-SiVIC, 212 (89.8%) are unsafe in PreScan (1a), and all the 164 safe scenarios in Pro-SiVIC are safe in PreScan as well (2b+2c). Among the 236 unsafe scenarios in Pro-SiVIC, 24 are safe (10.2%) because the scenarios are no longer critical (1c). We observed no discrepancies in PeVi's pedestrian detection after X-sim

reproduction from Pro-SiVIC to PreScan (1b+2a). However, we found that 100 of the 164 (61.0%) safe scenarios became non-critical when reproducing in Pro-SiVIC (2c), indicating changes in the distances between the ego car and the pedestrian in time and space.

The X-sim reproductions show that the dynamic simulator models can substantially influence the test results. First, critical scenarios often turned non-critical as distances changed (1c+2c). Second, the PeVi detections are not necessarily consistent between the simulators, i.e., the overall test verdicts related to safety violations frequently differ when reproducing scenarios in Pro-SiVIC (1b+2a). This might also be due to the dynamic modeling of the pedestrian, but it can also be explained by the implementation of PeVi or differences in the off-the-shelf sensors available in the simulators' libraries.

To know which fitness function (among FF1, FF2, and FF3) contributes most to the discrepancies in the X-sim reproduction results of critical scenarios between PreScan and Pro-SiVIC, we measure how big the FF differences are after the X-sim reproductions. Figure 9 shows distributions of absolute differences for the three FFs. The top row shows the results from reproducing the 400 critical Pro-SiVIC scenarios in PreScan. For most scenarios, the difference in FF1 is less than 4 m, i.e., $FF1 \leq 4$ m. However, for 59 scenarios (14.8%), the absolute difference in FF1 is $\geq$ 5 m. Furthermore, the distribution of differences for FF2 resembles FF1 with an absolute difference of $\leq 1m$ in 348 scenarios (87.0%).

The absolute differences in FF3 display a bimodal distribution. Due to a PeVi implementation choice, an FF3 value of 4 s either means that the sensor did not detect the pedestrian or the pedestrian remained far away during the entire scenario. Consequently, an absolute difference close to 4 s for equivalent test scenarios in PreScan and Pro-SiVIC is the result of either (1) large variations in how close the pedestrian gets to the car (as shown for the absolute difference in FF1) or (2) conflicting results of the sensors in PreScan and Pro-SiVIC. Reproduction of 400 critical Pro-SiVIC scenarios in PreScan resulted in 156 scenarios with an absolute difference for FF3 of $\leq 0.5s$ (41.3%) and 195 scenarios of $\geq 3.5s$ (48.8%).

The bottom row in Figure 9 depicts the results from reproducing the 400 critical PreScan scenarios in Pro-SiVIC. The results largely resembles the Pro-SiVIC to PreScan reproduction. For FF1, the absolute difference is $\geq 4m$ in 62 scenarios (15.5%). For FF2, the absolute difference is $\geq 1m$ in 48 scenarios (12.0%). Finally, a bimodal distribution is again the result for FF3, the absolute difference is $\leq 0.5s$ in 214 scenarios (53.5%) and $\geq 3.5s$ in 118 scenarios (29.5%).

In short, FF2 values were the most consistent in the two simulators; FF1 values were largely consistent with a few outlier scenarios for which FF1 differences were large between PreScan and Pro-SiVIC; but FF3 values were the most inconsistent between the two simulators where we observed large FF3 differences in the results from PreScan and Pro-SiVIC for several scenarios.

> Reproducing critical scenarios between PreScan and Pro-SiVIC frequently results in discrepancies regarding distances and PeVi detections. Among the three fitness functions used for scenario generation, FF2 values were the most consistent after X-sim reproductions whereas FF3 values differed substantially.

VI. Threats to Validity

In this section, we discuss the most important threats to internal and external validity [32].

Internal validity concerns inferences regarding casual relationships. We designed our experiment setup in a way to mitigate internal threats. For the X-sim reproductions, we focused on controlling as many variables as possible, both in the initial minimalistic scene and in the SBST setup. We used the same NSGA-II parameters X-sim and carefully created highly similar minimalistic scenes in PreScan and Pro-SiVIC. As described in Section III, there are some minor visual differences related to the initial scene. Furthermore, there are several other variables embedded in the simulators whose effects we study rather than control, e.g., the modeling of the optics in the cameras, sensor resolution and the radar cross-section of pedestrians. To mitigate this threat, we tried to ensure that sensors used in PreScan and Pro-SiVIC were configured the same to the extent possible. Finally, the packet loss measured in the Pro-SiVIC setup might have influenced the results, but as reported in Section IV-A3, we mitigated this by repeating the Pro-SiVIC experiments.

External validity reflects generalizability and often contradicts the internal validity of a study. As we carefully controlled variables through a minimalistic scene, we have no evidence regarding simulation of more complex environments or traffic scenarios. Future work should explore X-sim reproduction for more complex scenes, including variations in elevation and curvature of roads as well as scenery and traffic density. Another variable that deserves future study is how different weather simulations influence the sensor models. However, since we contribute evidence of prevalent X-sim discrepancies for a minimalistic scene, we have no reason to believe they would disappear for more complex scenes. Finally, as we limited the study to X-sim reproduction between PreScan and Pro-SiVIC, we cannot claim that the magnitude of differences would be the same for other simulators such as CARLA [5].

VII. Lessons Learned

Conducting X-sim reproductions between two commercial simulators provided insights that go beyond the RQs. In this section, we provide two important lessons learned.

Lesson 1. Validating ADAS testing results in multiple simulators is beneficial. Automotive simulators are complex software tools. Each simulator depends on the priorities, background and expertise of its vendor and is focused on certain aspects of ADAS and self-driving systems. For example, while PreScan is mostly focused on the fidelity of physics-based and mechanical models of self-driving systems, Pro-SiVIC is specialized in developing accurate sensor models. Typically, one simulator alone may not be able to perfectly capture

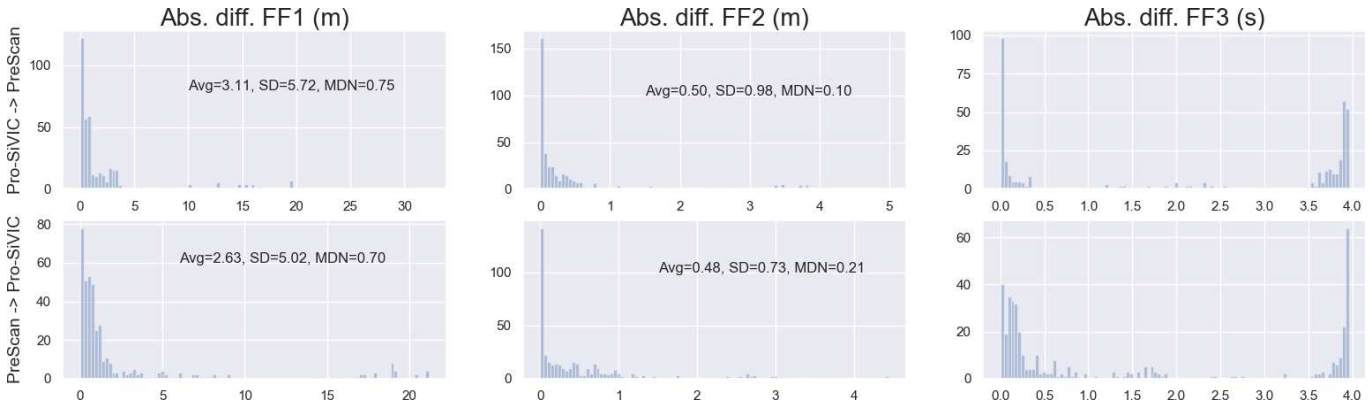


Fig. 9. Absolute differences of the three FFs when reproducing 400 critical Pro-SiVIC scenarios in PreScan (top row) and 400 critical PreScan scenarios in Pro-SiVIC (bottom row).

all the subtleties and complexities of ADAS and self-driving systems. As a result, replicating simulations in multiple tools with complementary strengths, identifying simulations that are consistent and robust across tools, and using those simulations for failure analysis and fault localization can help improve the accuracy of ADAS V&V activities. If the ADAS performance cannot generalize to another simulator, it would be overly optimistic to expect generalization to the real world.

Lesson 2. Fitness functions (test objectives) should be defined in such a way that they are minimally impacted by variations, weaknesses or potential faults in the internals of simulators. In our replication study, we observed different results for the three fitness functions that were proposed in the original study. Specifically, FF2 values were the most consistent in the two simulators and FF1 values were largely consistent with a few outlier scenarios; but FF3 values were the most inconsistent between the two simulators. While FF1 and FF2 largely depend on the physics-based models of simulators, FF3 depends on their sensor models and could, at least in part, explain the measured differences. Our observation is that testing results might be more consistent across different simulators if fitness functions do not depend on specific and non-standard components (e.g., sensors) that likely vary across different simulators. For example, we conjecture that if we repeat our study where we use the same SBST method with FF1 and FF2 only and remove FF3, we likely obtain more consistent test outputs between PreScan and Pro-SiVIC.

VIII. Conclusion and Future Work

We presented a replication study of applying a search-based software testing (SBST) solution to an Advanced Driver-Assistance System (ADAS) case study using two different commercial simulators, namely, TASS/Siemens PreScan and ESI Pro-SiVIC. Our results suggest that while SBST is effective in finding failure revealing test scenarios using both simulators, there are considerable differences in the specific details of the scenarios generated using the two simulators.

We present two recommendations for research and practice. First, simulation-based ADAS testing should not rely on a single simulator. Ideally, the test result analysis should primarily be based on the ADAS testing results that generalize to multiple simulators. Second, SBST for ADAS testing should be based on fitness functions (test objectives) that are minimally impacted by the internals of simulators and in particular by third party models of hardware components (e.g., sensors and cameras) included in the simulators.

For future, we will elaborate on how our findings relate to the evaluation of the residual risk as mandated in ISO/PAS 21448 Safety of the Intended Function (SOTIF). Specifically, we will provide actionable recommendations for the standardization efforts related to SOTIF Part 11.2 – Method K "Simulation of selected scenarios". As our primary interests relate to testing of SOTIF compliant perception systems that use deep neural networks, our next step will be development of testing recommendations tailored for machine learning.

Acknowledgment

This work was carried out within the SMILE II and SMILE III projects financed by Vinnova, FFI, Fordonsstrategisk forskning och innovation under the grant numbers: 2017-03066 and 2019-05871. Further, the work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852 (VALU3S), the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694277), and the NSERC of Canada under the Discovery program. Donghwan Shin was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2019R1A6A3A03033444).

References

[1] "List of self-driving car fatalities," https://www.wikiwand.com/en/List_of_self-driving_car_fatalities, accessed: 2020-10-18.

[2] M. Borg, C. Englund, K. Wnuk, B. Durann, C. Lewandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, and J. Törnqvist, "Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry," *Journal of Automotive Software Engineering*, vol. 1, no. 1, pp. 1–19, 2019.

[3] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[4] R. Math, A. Mahr, M. M. Moniri, and C. Müller, "Opends: A new open-source driving simulator for research," *GMM-Fachbericht-AmE 2013*, vol. 2, 2013.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[6] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Springer, 2018, pp. 621–635.

[7] "Road Vehicles - Safety of the Intended Functionality," International Organization for Standardization, Tech. Rep. ISO/PAS 21448:2019, 2019.

[8] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proc. of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 257–267.

[9] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing Autonomous Cars for Feature Interaction Failures Using Many-objective Search," in *Proc. of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 143–154.

[10] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 63–74.

[11] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proc. of the 42nd International Conference on Software Engineering*, 2020, pp. 359–371.

[12] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? An exploratory study," in *Proc. of the 2017 IEEE International Conference on Software Quality, Reliability and Security*, 2017, pp. 150–159.

[13] F. Ul Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *Proc. of the 13th IEEE International Conference on Software Testing, Validation and Verification*, 2020, pp. 85–95.

[14] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[15] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. of the 40th International Conference on Software Engineering*, 2018, pp. 1016–1026.

[16] P. McMinn, "Search-based software testing: Past, present and future," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011, pp. 153–163.

[17] A. Belbachir, J.-C. Smal, J.-M. Blosseville, and D. Gruyer, "Simulation-Driven Validation of Advanced Driving-Assistance Systems," *Procedia - Social and Behavioral Sciences*, vol. 48, pp. 1205–1214, 2012.

[18] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *Proc. of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 982–988.

[19] N. Cartwright, "Replicability, Reproducibility, and Robustness: Comments on Harry Collins," *History of Political Economy*, vol. 23, no. 1, pp. 143–155, 1991.

[20] O. Gómez, N. Juristo, and S. Vegas, "Replication, Reproduction and Re-analysis: Three Ways for Verifying Experimental Findings," in *Proc. of the 1st International Workshop on Replication in Empirical Software Engineering Research*, 2010.

[21] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[22] A. Gambi, M. Müller, and G. Fraser, "Automatically Testing Self-driving Cars with Search-based Procedural Content Generation," in *Proc. of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[23] O. Bühler and J. Wegener, "Automatic testing of an autonomous parking system using evolutionary computation," SAE Technical Paper, Tech. Rep., 2004.

[24] R. van der Horst and J. Hogema, "Time-to-collision and collision avoidance systems," in *Proc. of the Workshop of the International Cooperation on Theories and Concepts in Traffic Safety*, 1993, pp. 109–121.

[25] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[26] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–61, 2012.

[27] "Data Distribution Service (DDS) version 1.4," https://www.omg.org/spec/DDS/, Object Management Group, Tech. Rep., 2015.

[28] "Search-based Software Testing for ADAS in ESI Pro-SiVIC," https://github.com/mrksbrg/adas-pro-sivic.

[29] "Testing Advanced Driver Assistance Systems (TestingADAS)," https://bitbucket.org/rbenabdessalem/testingadas/src/master/.

[30] D. Brockhoff, T. Friedrich, and F. Neumann, "Analyzing hypervolume indicator based algorithms," in *Proc. of the 10th International Conference on Parallel Problem Solving from Nature*, 2008, pp. 651–660.

[31] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann Publishers Inc., 2011.

[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.