

SimFusion: Multi-Fidelity Simulation-based Testing of Automated Driving Systems

Lev Sorokin¹, Berfin Dinler¹, Alessio Gambi^{2,3}, Andrea Stocco^{1,4}

¹*Technical University of Munich, Munich, Germany, {lev.sorokin, berfin.dinler, andrea.stocco}@tum.de*

²*AIT, Vienna, Austria, alessio.gambi@ait.ac.at*

³*Italian Institute of Artificial Intelligence (IAI), Torino, Italy*

⁴*fortiss GmbH, Munich, Germany, stocco@fortiss.org*

Abstract—Simulation is the primary means of validating autonomous driving systems (ADS). Existing testing approaches typically rely on a single simulator, forcing a trade-off between the execution speed of low-fidelity (LoFi) simulation and the realism of computationally intensive high-fidelity (HiFi) simulation. To balance this trade-off, we present SimFusion, a multi-fidelity search-based testing framework that dynamically orchestrates LoFi and HiFi simulators to optimize execution cost and simulation realism. SimFusion learns cross-simulator agreement to determine when inexpensive LoFi simulation is sufficient and when HiFi evaluation is necessary, enabling fast exploration while preserving reliable failure detection. We evaluate SimFusion on two ADS stacks, three driving scenarios, and four simulators. Compared with single-fidelity and surrogate-assisted baselines, SimFusion discovers up to 40% more failures, increases failure diversity, and reduces the number of expensive HiFi simulations. Our results show that learning cross-simulator agreement enables effective multi-fidelity testing, making simulation fidelity a first-class optimization dimension in ADS testing.

I. INTRODUCTION

Automated Driving Systems (ADS) are primarily validated through simulation before real-world testing [1], [2]. Simulators offer varying trade-offs between computational cost and realism [3]. High-fidelity (HiFi) simulators, such as CARLA [4] and BeamNG.tech [5], accurately model vehicle dynamics and perception, but their execution is time-consuming, requiring minutes to hours per scenario. Conversely, low-fidelity (LoFi) simulators, such as CommonRoad [6] and MetaDrive [7], abstract these components to enable faster execution at the cost of reduced realism [8], [9].

Despite the availability of multiple simulators, simulation fidelity remains a fixed design choice in current testing pipelines. LoFi simulators are typically used for rapid prototyping, whereas HiFi simulators are reserved for system-level validation [10]. Likewise, existing search-based testing executes all candidate scenarios in a single simulator, forcing developers to trade exploration throughput for realism [11].

Some approaches attempt to reduce HiFi executions using surrogate models that predict failures or fitness values [12]–[15]. However, these approaches require learning a surrogate of the entire ADS, i.e., a mapping from high-dimensional driving scenarios to complex closed-loop system behavior. Such behavior emerges from the interaction between perception, planning, control, and the environment, making accurate

approximation challenging. Other approaches reduce execution cost by adjusting simulator parameters, such as the frame rate, trading simulation accuracy for efficiency [3].

In this paper, we present SimFusion, a multi-fidelity search-based testing framework that dynamically selects the simulation fidelity at which each candidate scenario is evaluated. Rather than learning failures or ADS behavior, SimFusion learns a model of *cross-simulator agreement* between LoFi and HiFi simulators to predict whether both simulators are likely to produce the same test outcome for a given scenario [16]. Since search-based testing spends most of its computational budget evaluating non-critical scenarios, identifying when these scenarios can be evaluated reliably in the faster LoFi simulator offers the greatest opportunity to reduce testing cost. SimFusion therefore uses the agreement model to guide simulator selection, reserving expensive HiFi execution for scenarios where HiFi simulation is predicted to reach a different safety verdict than LoFi. To the best of our knowledge, no prior work dynamically orchestrates multiple simulators during search by learning when high-fidelity execution is required.

We evaluate SimFusion across two ADS stacks, three driving scenarios, and four simulators: CARLA [4] and BeamNG.tech [5] as HiFi simulators, and CommonRoad [6] together with a ROS-based kinematic simulator as LoFi simulators. Our results show that SimFusion consistently outperforms both single-fidelity and surrogate-assisted baselines, discovering more failures, achieving higher failure diversity, and substantially reducing the number of HiFi simulations.

Beyond these empirical improvements, our study provides two key insights. First, deciding when HiFi simulation is worthwhile is substantially easier than approximating fitness functions or entire ADS behaviors, making simulator orchestration more effective than surrogate-assisted testing. Second, treating simulator fidelity as a runtime decision yields better failure discovery than committing to a fixed simulator throughout the search, as it enables the search process to explore the scenario space more effectively.

The main contributions of this paper are:

SimFusion. A multi-fidelity search-based testing framework that dynamically orchestrates low- and high-fidelity simulators by learning cross-simulator agreement [17].

Empirical evaluation. A comprehensive evaluation across two ADS stacks, three driving scenarios, four simulators,



Fig. 1: Comparison of HiFi and LoFi simulations for an automated emergency braking scenario.

that comprises more than 11,000 simulation executions.

Findings. Empirical evidence that using cross-simulation agreement to guide simulator orchestration leads to the discovery of substantially more failures than both fixed-fidelity and surrogate-assisted approaches while improving testing efficiency.

II. BACKGROUND

A. Simulation Fidelity in ADS Testing

In ADS testing, *simulation fidelity* denotes how accurately a simulator reproduces sensing, vehicle dynamics, and interactions with the environment [18]. Simulators execute *driving scenarios*, which specify the initial state and behavior of the ego vehicle (i.e., the ADS under test), the surrounding traffic participants, and the environment.

High-fidelity (HiFi) simulators, such as CARLA [4] and BeamNG.tech [5], explicitly model sensing, vehicle dynamics, and environmental interactions using detailed physics and photorealistic rendering. These simulators generate realistic sensor inputs (e.g., camera images or LiDAR point clouds) that are processed by the ADS perception stack, yielding realistic yet computationally expensive simulations.

Low-fidelity (LoFi) simulators simplify sensing, dynamics, or both to reduce execution time. They typically replace sensor simulation with ground-truth state information and approximate vehicle motion using kinematic models. Examples include CommonRoad [6], MetaDrive [7], and esmini [19]. These abstractions substantially accelerate simulation but may miss failures that depend on realistic perception or physics.

Figure 1 compares the HiFi and LoFi implementations of an automated emergency braking (AEB) scenario used throughout this paper. HiFi relies on LiDAR-based perception and physics-based vehicle dynamics, whereas LoFi uses ground-truth perception and a kinematic bicycle model. In our setting, these abstractions reduce the average execution time from 175s to 48s per scenario (a $3.6\times$ reduction), illustrating the potential computational savings, at the cost of potentially changing whether a scenario is classified as safe or unsafe.

B. Simulator Orchestration Problem

Previous work has shown the benefits of orchestrating multiple HiFi simulators to reduce simulator-dependent failure

Algorithm 1: SimFusion

Input : (ADS, D, F, O) : SBST problem instance
 S_{lofi} : LoFi simulator
 S_{hifi} : HiFi simulator
 C : agreement classifier
 th : classifier confidence threshold
Output: *Failures*: set of failure-inducing test inputs

```

1 INIT( $T$ )
2  $A \leftarrow \emptyset$  //  $T$ : archive of evaluated scenarios
3  $P_0 \leftarrow \text{SAMPLING}(ADS, D)$ 
4  $P_0 \leftarrow \text{RUNORCHESTRATOR}(P_0, S_{lofi}, S_{hifi}, C, th, F, ADS)$ 
5 while budgetAvailable do
6    $Par \leftarrow \text{TOURNAMENTSELECTION}(P_i)$ 
7    $P_i \leftarrow \text{CROSSOVERMUTATE}(Par)$ 
8    $P_i \leftarrow \text{RUNORCHESTRATOR}(P_i, S_{lofi}, S_{hifi}, C, th, F, ADS)$ 
9    $T \leftarrow T \cup P_i$ 
10   $P_i \leftarrow \text{SURVIVAL}(P_i, O)$ 
11   $P_i \leftarrow \text{REPOPULATE}(P_i, D, T)$ 
12  $Failures \leftarrow \{x \in T \mid O(x) = 1\}$ 
13 return Failures

```

discovery [16], [20]. However, executing multiple HiFi simulators substantially increases computational cost of testing.

In contrast, we investigate orchestration across simulation fidelities, leveraging inexpensive LoFi simulation while selectively invoking HiFi execution only when necessary. However, orchestrating multiple simulators introduces a fundamental challenge: the same scenario may produce different outcomes depending on the simulator used [21]. These discrepancies complicate the use of LoFi simulation to guide the search process. Scenarios deemed safe in LoFi may reveal failures in HiFi, and vice versa, causing the search to discard promising scenarios while exploring uninformative ones.

Addressing this challenge requires determining when two simulators are likely to produce equivalent outcomes and when additional simulation fidelity is necessary.

III. SEARCH-BASED SCENARIO GENERATION DRIVEN BY SIMULATION ORCHESTRATION

SimFusion is a multi-fidelity search-based testing framework that dynamically orchestrates low- and high-fidelity simulators during scenario generation. Rather than executing every candidate scenario in a fixed simulator (e.g., HiFi), SimFusion trains a classifier to predict whether executing a scenario in HiFi would produce the same outcome as executing the same scenario in LoFi (see Section III-B). During generation, it uses the classifier to determine whether a newly generated scenario requires HiFi evaluation or can be safely executed in the faster LoFi simulator, thereby enabling efficient exploration of the scenario space while preserving reliable failure detection.

A. Multi-Fidelity Search-based Testing

SimFusion integrates simulator orchestration, i.e., the decision of which simulation fidelity to use for executing a scenario, into a search-based testing loop that iteratively generates, evaluates, and evolves driving scenarios. SimFusion takes as input the ego-vehicle ADS , i.e., system under test, the search domain D , which defines the validity range of the parameters defining the scenario, a fitness function F , which

evaluates the “goodness” of the scenario, and an oracle O , which distinguishes passing and failing scenarios. SimFusion leverages a pair of LoFi and HiFi simulators that can execute the same scenarios against the ego-vehicle and evaluate them using the same fitness function and oracle, although at different fidelity levels.

Algorithm 1 outlines the overall workflow of the scenario generation process. It starts by creating an initial population sampling search domain using Latin Hypercube Sampling [22] and evaluating the individual through the simulator orchestrator (Lines 2–3). Then, the search loop repeatedly generates new candidate scenarios using evolutionary operators, evaluates them through the orchestrator, updates the archive of executed tests, and applies survival and repopulation operators (Lines 5–10). Once the testing budget is exhausted, all scenarios that satisfy the oracle are returned as witnesses to failures (Lines 12–13).

B. Simulation Agreement and Orchestration

A scenario s executed in a simulator $S \in \{\text{LoFi}, \text{HiFi}\}$ produces an oracle outcome $o \in \{\text{failure}, \text{non-failure}\}$ indicating whether the ADS violates a system requirement, e.g., a safety property. In this paper, we focus on safety violations such as collisions, violations of minimum safety distance, and critical time-to-collision values [23]. Consequently, a scenario is classified as a failure whenever the ego vehicle violates one of these safety criteria.

Let $o_{\text{LoFi}}(s)$ and $o_{\text{HiFi}}(s)$ denote the oracle outcomes obtained by executing scenario s in the LoFi and HiFi simulators, respectively. The two simulators are said to *agree* if $o_{\text{LoFi}}(s) = o_{\text{HiFi}}(s)$, and *disagree* otherwise. Two types of disagreement may occur:

Case 1 LoFi reports a failure whereas HiFi does not.

Case 2 HiFi reports a failure whereas LoFi does not.

The goal of SimFusion is to predict whether a scenario belongs to an agreement or disagreement region and exploit this information to determine when HiFi simulation is necessary.

The simulator orchestrator determines which simulator evaluates each candidate scenario. It bases this decision on a binary classifier trained to predict whether the two simulators will agree or disagree on the candidate scenario outcome. Unlike surrogate-assisted approaches, the agreement classifier is not required to approximate the behavior of the ADS or predict continuous fitness values. Instead, it learns whether increased simulation fidelity is likely to change the final oracle verdict. This is a substantially simpler learning problem, as it focuses only on identifying regions where simulator outcomes diverge, while still providing the information required for simulator orchestration. Indeed, from the perspective of search-based testing, the two disagreement cases (Section III-B) have different consequences. Case 1 may waste search effort by following spurious LoFi failures, whereas Case 2 is more critical because it causes genuine HiFi failures to be missed. Consequently, SimFusion adopts a conservative orchestration strategy that favors HiFi whenever disagreement is predicted.

Algorithm 2: Simulators Orchestration

Input : $P = \{\text{test}_i\}_{i=1}^N$: population of candidate scenarios
 S_{hiFi} : high-fidelity simulator
 S_{loFi} : low-fidelity simulator
 C : agreement classifier
 th : confidence threshold
 F : fitness function
 ADS : ADS

Output: P : population with assigned fitness values

```

1 for  $i \leftarrow 1$  to  $N$  do
2   if  $P(\text{test}_i) > th$  then
3      $S \leftarrow S_{\text{loFi}}$ 
4   else
5      $S \leftarrow S_{\text{hiFi}}$ 
6    $result \leftarrow S(\text{test}_i, ADS)$ 
7    $P[i].fitness \leftarrow F(result)$ 
8 return  $P$ 

```

The binary classifier estimates the probability that the LoFi and HiFi simulators will produce the same criticality judgment for a scenario. If the predicted agreement exceeds the confidence threshold th , the faster LoFi simulator is selected; otherwise, the scenario is conservatively evaluated in HiFi.

Algorithm 2 describes the orchestrator procedure. The selected simulator executes the scenario and produces a simulation trace. The trace is used both to compute the fitness value that guides the search and to evaluate the safety oracle that determines whether the scenario exposes a failure.

C. Scenario Generation

New candidate scenarios are generated using evolutionary operators, namely selection, crossover, and mutation [24].

Selection. At each iteration, a subset of scenarios is selected via tournament selection for follow-up processing (i.e., evolution). This selection mechanism favors higher-fitness scenarios while maintaining population diversity.

Crossover. Given two parent scenarios $s^{(1)} = (p_1^{(1)}, \dots, p_d^{(1)})$ and $s^{(2)} = (p_1^{(2)}, \dots, p_d^{(2)})$, where each parameter corresponds to a scenario variable (e.g., ego speed, pedestrian distance), an offspring scenario is generated through Simulated Binary Crossover [25]. For each parameter index j , the offspring inherits p_j from parent $s^{(1)}$ with probability μ_{cx} and from $s^{(2)}$ otherwise. For example, recombining the scenarios $s^{(1)} = (\text{EgoSpeed} = 3.0, \text{PedDist} = 10, \text{PedSpeed} = 1.5)$ and $s^{(2)} = (\text{EgoSpeed} = 4.5, \text{PedDist} = 2, \text{PedSpeed} = 1.0)$ may produce $s' = (\text{EgoSpeed} = 3.0, \text{PedDist} = 2, \text{PedSpeed} = 1.0)$ by exchanging the pedestrian speed.

Mutation. Mutation perturbs scenario parameters of single scenarios to explore nearby regions of the search space. SimFusion applies Polynomial Mutation [25] with probability μ_{mut} to each parameter p_j in scenario $s = (p_1, \dots, p_d)$.

Duplicate Handling. To avoid redundant simulations, scenarios whose parameter vectors match those of previously generated tests are discarded, and new scenarios are generated.

D. Fitness Function

The search is guided by a user-defined fitness function, which quantifies how close a scenario is to violating a target

system requirement. Unlike the oracle, which provides a binary pass/fail verdict, the fitness function provides a continuous optimization objective that directs the evolutionary search toward increasingly critical scenarios.

E. Failure Identification

The search process terminates when the predefined testing budget is exhausted. All executed scenarios are evaluated using the oracle O , which determines whether the system behavior violates safety requirements. The algorithm returns the set of scenarios that cause the system under test to fail.

IV. EVALUATION

A. Research Questions

RQ₁ (outcome consistency). *How consistently do LoFi and HiFi simulations agree on the outcome of a scenario?*

The aim is to investigate to what extent low-fidelity and high-fidelity simulators agree on the outcome of the same driving scenarios. This question serves as a prerequisite for our approach, which relies on predicting when a scenario can be safely evaluated in the low-fidelity simulator. It assesses whether simulator agreement is sufficiently systematic to enable reliable simulator orchestration.

RQ₂ (effectiveness). *How effective is SimFusion at detecting failures compared to single-simulation and surrogate-based approaches?* This research question evaluates whether dynamic orchestration of LoFi and HiFi simulations improves failure detection compared to executing scenarios in a single simulator or relying on surrogate models.

RQ₃ (efficiency). *How efficient is SimFusion at detecting failures compared to single-simulation and surrogate-based approaches?*

This research question evaluates whether SimFusion reduces the computational cost of testing while maintaining effective failure detection through selective HiFi execution.

RQ₄ (failure diversity). *How diverse are the failures identified by SimFusion compared to failures obtained through single-fidelity simulation approaches?*

This research question evaluates the diversity of the failure-inducing scenarios identified by SimFusion and compares their distribution across the scenario space with those obtained using single-simulation and surrogate-based approaches.

B. Objects of Studies

To evaluate the generality of our approach, we consider two realistic ADS stacks and three driving scenarios. Both ADS stacks implement contemporary architectural paradigms and perception/planning pipelines, are developed in different simulation ecosystems, and deployed on real vehicles.

1) *Autoware Mini*: Autoware Mini [26] is a Python port of the autonomous driving stack developed by the Autoware Foundation [27]. The system includes obstacle detection, global and local planning, control, and a camera-based traffic-light detector. Autoware Mini is deployed on an experimental autonomous vehicle (a Lexus RX450h) at the University of Tartu [28]. As HiFi simulation, we use CARLA [4]. As LoFi

TABLE I: FrenetiX scenario variables and ranges.

Parameter	Range	Description
goal_center_x	[150, 160]	Goal longitudinal position
goal_center_y	[0, 0.5]	Goal lateral position
goal_velocity_min	[10, 15]	Minimum goal velocity m/s
goal_velocity_max	[20, 25]	Maximum goal velocity m/s
num_npc	[2, 4]	Number of NPC vehicles
lane_count	[2, 5]	Number of lanes
npc_x0	[9, 12]	Initial NPC position (x)
npc_v_max	[15, 20]	Maximum NPC velocity m/s
npc_seed	[0, 99]	NPC behavior seed
cutin_tx	[0.5, 1.0]	Scalar cut-in duration
cutin_delay	[0, 0.5]	Scalar cut-in activation delay

simulator we use the ROS message exchange bundled with Autoware Mini. Compared to CARLA, which uses LiDAR perception and rigid-body physics, the LoFi simulator provides ground-truth data on obstacles and traffic lights. Furthermore, it uses the bicycle model to simulate vehicle dynamics. To ensure that outcome differences are attributable to simulator fidelity rather than mismatched scenario executions, we executed a pilot study to assess trace alignment across simulator executions using distance metrics (i.e., Euclidean, DTW [29], and Hausdorff [30]) as detailed in our replication package [17].

We evaluate Autoware Mini on two abstract scenarios, widely used in prior work [31]: (a) an automated emergency-braking (AEB) scenario in which the ego vehicle drives on a straight road segment while a pedestrian crosses the road in front of it; (b) a cut-in scenario (CUT-IN) in which a slower vehicle traveling in front of the ego-vehicle in the left lane performs a cut-in maneuver into the ego vehicle's lane. Scenarios take place on the Town1 and Town4 maps, respectively. The AEB scenario is parameterized by the ego vehicle's target velocity *EgoSpeed*, the pedestrian trigger distance *PedDist*, and the pedestrian velocity *PedSpeed*. Parameter ranges are derived from prior studies and preliminary experiments [32]. The CUT-IN scenario is parametrized by the ego vehicle's target velocity *EgoSpeed*, the adversary vehicle triggering distance *DistStartCutin*, the cut-in duration *LengthCutin*, and the speed difference between the vehicles *SpeedDiff*.

2) *FrenetiX*: FrenetiX [33] is an open-source motion planning framework implementing an efficient sampling-based trajectory planning in Python and C++. FrenetiX is deployed in the EDGAR research vehicle (a Volkswagen T7 Multivan) at the Technical University of Munich [34]. As HiFi simulator, we use BeamNG.tech [5], which implements photo-realistic environments and soft-body physics, whereas as LoFi simulator, we use CommonRoad [6], which implements a lightweight 2D simulation. We use the default simulator configuration and a highway template from CommonRoad [6] and ensure consistent scenario execution using MultiDrive [9].

We evaluate FrenetiX in a highway scenario involving a lane-change maneuver on a straight, multi-lane road with surrounding traffic [9]. The scenario is parameterized by 11 variables that control the target position and velocity of the ego vehicle, the number of lanes and vehicles, and the cut-in maneuvers of surrounding traffic (Table I).

TABLE II: Algorithm configuration for the approaches under test LoFi, HiFi, and SimFusion, summarized by case study.

Parameter	Autoware (AEB/CUT-IN)	FrenetiX
Simulators	Carla [4], SimpleSim [26]	BeamNG [5], CommonRoad [6]
Test Generator	NSGA-II	NSGA-II
Population size	20/15	10
Time budget	3/2.5 hours	6 hours
Mutation rate	0.33	0.09
Crossover rate	0.6	0.6
Fitness functions	2	3
Search variables	3/4	10

C. SimFusion Configurations

To generate critical scenarios, i.e., scenarios in which the ego-vehicle collides at high speed, we configure SimFusion based on preliminary runs and best practices (see Table II). The search loop runs for 3.0 hours and evolves a population of 20 individuals in the AEB scenario, while it runs for 2.5 hours and evolves a population of 15 individuals in the CUT-IN scenario. Mutations and crossover trigger with probabilities 0.2 and 0.8 in both scenarios.

Autoware Mini. In the AEB scenario, the test oracle flags a failure if the ego-vehicle is too close to the pedestrian crossing in front of it ($F_{dist} < 1$) while the ego-vehicle is moving ($F_v > 0$). Similarly, in the CUT-IN scenario, it flags a failure when the TTC falls below 0.5 s [35].

FrenetiX. We configure SimFusion to minimize the distance between the ego-vehicle and the other vehicles (F_{dist}) while maximizing the distance of the ego-vehicle from the goal area it must reach (F_{goal}). To account for longer simulations in BeamNG.tech, SimFusion evolves a population of 10 individuals for 6 hours. Mutation and crossover trigger with the same probabilities used for Autoware Mini. The test oracle flags a failure if $F_{dist} < 0.5m$ or $F_{goal} > 2m$, indicating unsafe proximity or failure to reach the goal.

D. Baselines

We compare SimFusion against two baselines: single-fidelity testing and surrogate-assisted testing. We instantiate single-fidelity in two versions: single-fidelity testing using only HiFi and single-fidelity testing using only LoFi.

The surrogate-assisted baseline follows the approach proposed by Jodat et al. [15]. It leverages a regression model to predict the fitness values of newly generated scenarios before executing them. The surrogate model is trained on labeled execution data from HiFi simulations. Specifically, the surrogate-assisted baseline uses the prediction uncertainty to determine whether the prediction interval, estimated from the validation-set mean absolute error, lies entirely within either the pass or fail region. If the interval lies entirely within a single region, the prediction is considered reliable, and the surrogate result is used without executing the HiFi simulator; otherwise, the scenario is evaluated with HiFi.

To ensure a fair comparison, all approaches use the same search-based testing framework and optimization as SimFusion, differing only in how candidate scenarios are evaluated.

TABLE III: Features used for diversity analysis.

Feature	FrenetiX	Autoware AEB	Autoware CUT-IN
min_TTC	✓	✓	✓
min_dist	✓	✓	✓
top2_min_dist	✓	-	-
num_neighbors_within_5m	✓	-	-
time_with_neighbors_5m	✓	-	✓
min_lateral_gap	✓	-	✓
min_longit_gap	✓	-	✓
min_lateral_signed	-	✓	-
min_longit_signed	-	✓	-
avg_speed_ego	✓	-	✓
speed_at_min_dist	-	✓	✓
max_speed_ego	✓	✓	✓
max_deceleration_ego	✓	✓	✓

E. Procedure and Metrics

RQ₁. We generated scenarios using both random sampling and NSGA-II search to obtain a balanced dataset of passing and failing scenarios for analyzing outcome consistency. Next, we executed all scenarios in both LoFi and HiFi and labeled them with the corresponding LoFi and HiFi simulation outcomes; in total, we collected 592 scenarios across all case studies, resulting in 1,084 simulator executions. Finally, we quantify outcome consistency between the simulators by comparing their labels scenario-wise, as follows: (i) Agreement on Pass (AP), when both simulators report a non-failure outcome; (ii) Agreement on Fail (AF), when both simulators report a failure; (iii) Disagreement Pass (DP), when the HiFi simulator reports a pass whereas the LoFi simulator reports a failure; and (iv) Disagreement Fail (DF), otherwise. AP and AF represent *agreement* cases, while DP and DF correspond to *disagreements* between simulators. We quantify consistency among simulators as the percentage of scenarios that fall into AF and AP within the evaluated set of scenarios.

RQ₂/RQ₃. We used the dataset collected in RQ₁ to train binary classifiers to predict simulator agreement in the various scenarios. In particular, we used all scenario input parameters for training, i.e., we used 3 features (Pedestrian Speed, Ego Speed, Pedestrian Distance) for Autoware AEB, 4 features for Autoware CUT-ING, and 10 features for FrenetiX. To identify an appropriate classifier, we trained several models, including k-Nearest Neighbors (kNN), Random Forests (RF), Gradient Boosting (GBoost), and Support Vector Machines (SVM) [15]. For each model, we optimized hyperparameters via grid search and evaluated performance using 5-fold cross-validation (see Table IV). For each use case, we set the threshold th to 0.8 based on the Youden index [36], which maximizes the sum of sensitivity and specificity. This value was selected based on preliminary experiments; results for other thresholds are available in our replication package [17].

For RQ₂, we measure the number of failures detected. We validate reported failures in HiFi simulation. For RQ₃, we measure the rate at which failures are discovered during the search process. Efficiency is the cumulative number of failures discovered over wall-clock time.

To compare SimFusion with the surrogate-assisted approach

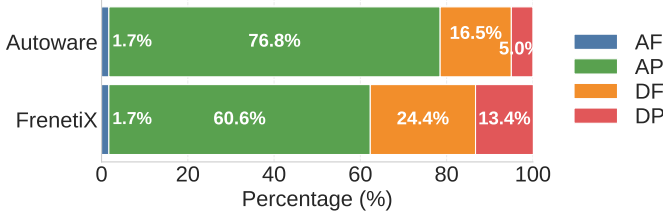


Fig. 2: RQ₁: Distribution of simulator outcome agreement. AP/AF denote agreement on pass/fail outcomes, whereas DF/DP denote disagreement where only HiFi or LoFi, respectively, detects a failure.

proposed by Jodat et al. [15], we trained a regression model to predict the fitness value for each generated scenario, using the same dataset collected for RQ₁. Specifically, we used only data from the HiFi runs, which serve as our ground truth, and split the data using the standard 80/20 train-validation split for validation, as proposed in the original work. We evaluated different dataset configurations, including balanced and imbalanced sets, and applied SMOTE as suggested in prior work [15]. We evaluated several regression models, including Gaussian Processes (GP), Random Forest (RF), and a Multi-Layer Perceptron (MLP). For each scenario, we used all input parameters as input features for the regression models and all fitness values as the models’ target variables (Section IV-B).

RQ₄. We use for the scenario encoding the scenario-specific feature sets reported in Table III and measured the diversity of failures using clustering [37]. First, we encode failing scenarios from RQ₂ across all testing approaches, as numerical feature vectors that capture key properties of the execution. Next, we cluster the feature vectors using HDBSCAN [38] with the Euclidean distance metric. Finally, using the identified clusters, we compute the cluster coverage and the failure entropy, which quantifies the uniformity of failure distribution [37], [39]. Based on preliminary experiments, we set the minimum cluster size to 3 for Autoware and 6 for FrenetiX.

Summary. To address nondeterminism associated with search-based testing and improve the robustness of our conclusions, we repeated the experiments for RQ₂ and RQ₃ $n = 6$ times. Following established guidelines [40], we assessed statistical significance using the Wilcoxon rank-sum test ($\alpha = 0.05$) and effect sizes with Vargha–Delaney $\hat{A}_{12}$ [41]. In total, our experiments involved more than 11,000 simulations, which correspond to approximately 480 hours of computation. All experiments were conducted on a dedicated machine equipped with an NVIDIA RTX 4090 GPU (24 GB VRAM), 32 GB RAM, and an Intel i7 16-core CPU.

F. Results

1) *Outcome consistency (RQ₁):* Figure 2 summarizes the distribution of agreement and disagreement across all generated scenarios. Overall, the simulators agree on 78.5% of scenarios for Autoware and 62.3% for FrenetiX.

TABLE IV: RQ₁: Agreement/disagreement classification performance used to instantiate SimFusion. The best-performing model for each case study is highlighted in bold.

Case Study	Model	Recall	Precision	F ₁
Autoware AEB	kNN	0.97	0.79	0.87
	RF	0.92	0.78	0.84
	GBoost	0.78	0.79	0.79
	SVM	0.03	0.72	0.06
Autoware CUT-IN	kNN	0.81	0.83	0.81
	RF	0.91	0.81	0.86
	GBoost	0.93	0.80	0.86
	SVM	0.95	0.77	0.85
FrenetiX	kNN	0.67	0.73	0.70
	RF	0.76	0.74	0.75
	GBoost	0.69	0.74	0.72
	SVM	0.54	0.82	0.65

Most agreement corresponds to AP, whereas AF is comparatively rare, accounting for only 1.7% of all scenarios. Among disagreement cases, DF is substantially more frequent than DP, indicating that LoFi more often misses failures detected by HiFi than the converse. These results indicate that simulator agreement is concentrated on non-failing scenarios, whereas disagreement primarily occurs in safety-critical situations.¹ This behavior is consistent with the fidelity gap between the simulators and motivates our orchestration strategy, which reserves computationally intensive HiFi execution for scenarios where increased simulation fidelity influences the outcome.

Table IV reports the classification performance. The best-performing models were kNN ($k = 7$) for Autoware AEB, Gradient Boosting for Autoware CUT-IN, and Random Forest for FrenetiX, achieving F₁ scores of 0.87, 0.86, and 0.75, respectively. We use these classifiers to instantiate SimFusion in the remaining research questions, demonstrating that simulator agreement can be predicted with sufficient accuracy to support dynamic simulator orchestration.

These findings indicate that disagreement occupies a relatively small but safety-critical portion of the scenario space. Consequently, simulator orchestration can substantially reduce HiFi executions while preserving access to the scenarios where increased fidelity is most valuable.

RQ₁ (outcome consistency). *LoFi and HiFi agree on 78.5% of scenarios for Autoware and 62.3% for FrenetiX. Agreement is concentrated on non-failing scenarios and can be accurately predicted from scenario parameters ($F_1 > 0.75$), providing the foundation for simulator orchestration.*

¹Considering failure as the positive class, failure precision and recall are computed as Precision = $AF/(AF + DP)$ and Recall = $AF/(AF + DF)$. For Autoware, precision/recall are 6.5%/11.3%; for FrenetiX, they are 9.3%/25.4%. Considering pass as the positive class, precision and recall are Precision = $AP/(AP + DF)$ and Recall = $AP/(AP + DP)$, yielding 81.9%/71.3% for Autoware and 93.9%/82.3% for FrenetiX.

TABLE V: RQ_2 : Effectiveness results for three case studies. Failure count and ratio are computed for validated failures (in HiFi). Runtime/test denotes the average runtime per executed test case (s). Best results across methods are highlighted in bold.

	Run	LoFi				HiFi				Surrogate				SimFusion			
		Tests $\uparrow$	Fail. $\uparrow$	Ratio $\uparrow$	Time/Test $\downarrow$	Tests	Fail.	Ratio	Time/Test	Tests	Fail.	Ratio	Time/Test	Tests	Fail.	Ratio	Time/Test
Autoware AEB	Run 1	660	10	0.01	16.36	340	69	0.20	31.76	280	181	0.65	38.57	378	289	0.76	28.57
	Run 2	680	5	0.00	15.88	320	32	0.10	33.75	300	185	0.62	36.00	440	257	0.58	24.55
	Run 3	620	35	0.05	17.42	380	150	0.39	28.42	400	124	0.31	27.00	460	254	0.55	23.48
	Run 4	680	7	0.01	15.88	380	74	0.19	28.42	320	146	0.45	33.75	420	166	0.39	25.71
	Run 5	680	15	0.02	15.88	360	85	0.23	30.00	280	198	0.71	38.57	380	234	0.61	28.42
	Run 6	660	28	0.04	16.36	380	47	0.12	28.42	280	150	0.53	38.57	380	281	0.73	28.42
	Avg.	663.33	16.66	0.02	16.30	360.00	76.16	0.20	30.13	310.00	164.00	0.55	34.84	409.66	246.83*	0.60	26.53
Autoware CUT-IN	Run 1	615	0	0	14.6	330	135	0.41	27.27	930	123	0.13	9.68	315	197	0.62	28.57
	Run 2	630	0	0	14.3	375	120	0.32	24.00	705	30	0.04	12.77	315	148	0.47	28.57
	Run 3	630	0	0	14.3	345	182	0.53	26.09	300	15	0.05	30.00	285	181	0.64	31.57
	Run 4	600	0	0	15	300	165	0.55	30.00	570	125	0.21	15.79	315	215	0.68	28.57
	Run 5	600	0	0	15	270	59	0.22	33.33	1290	117	0.09	6.98	270	181	0.67	33.33
	Run 6	600	0	0	15	300	111	0.39	30.00	1155	116	0.10	7.79	330	142	0.43	27.29
	Avg.	612.5	0	0	14.7	320.00	128.67	0.40	28.45	825.00	87.67	0.10	13.83	305	177.33*	0.58	29.65
FrenetiX	Run 1	258	4	0.01	83.72	90	26	0.28	240.00	140	16	0.11	154.29	110	37	0.34	196.36
	Run 2	260	6	0.02	83.08	90	20	0.22	240.00	370	22	0.06	58.38	110	47	0.43	196.36
	Run 3	258	4	0.01	83.72	60	10	0.16	360.00	100	28	0.25	216.00	109	45	0.41	198.17
	Run 4	260	2	0.00	83.08	60	9	0.15	360.00	350	47	0.13	61.71	120	39	0.32	180.00
	Run 5	290	0	0.00	74.48	60	17	0.28	360.00	290	50	0.17	74.48	130	48	0.37	166.15
	Run 6	310	21	0.06	69.67	60	15	0.25	360.00	350	56	0.16	61.71	130	47	0.36	166.15
	Avg.	272.66	6.16	0.02	79.63	70.00	16.16	0.22	320.00	266.67	36.50	0.15	81.00	118.17	43.83*	0.37	183.86
Avg.		516.16	7.61	0.01	36.88	250.00	73.66	0.27	126.19	467.22	96.06	0.27	43.22	277.61	155.99*	0.52	80.01

* Statistically significant improvement over the best baseline (Wilcoxon rank-sum test, $\alpha = 0.05$) with large effect size (Vargha-Delaney $\hat{A}_{12}$).

2) *Effectiveness (RQ_2)*: Table V reports for each of the three scenarios, six runs, and four approaches (the two single-fidelity baselines, the surrogate-assisted baseline, and SimFusion), the number of generated (col. *Tests*) and failed (col. *Fail.*) tests, the pass/fail ratio (col. *Ratio*), and the average test execution time (col. *Time/Test*). The table also reports results aggregated across the runs (rows *Avg*) and highlights statistically significant improvements over the best baseline.

We observe that LoFi executes the largest number of tests across all configurations but identifies the fewest failures. In contrast, HiFi executes substantially fewer tests because of its higher computational cost, yet detects more failures than LoFi. These findings are consistent with the expected trade-off between the two simulation fidelities: LoFi enables efficient exploration of the scenario space but has limited failure detection capability due to its lower fidelity, whereas HiFi provides more effective failure identification at the expense of reduced exploration. The surrogate-assisted baseline generates, on average, a similar number of tests as LoFi; however, it identifies up to $12\times$ more failures than LoFi. Compared to HiFi, it identifies a higher number of failures in two out of three case studies, i.e., FrenetiX and Autoware AEB.

These results highlight the benefits of using a surrogate model to balance execution cost and failure identification. However, for each case study, our approach SimFusion achieves the highest number of failures: for the Autoware AEB case study, SimFusion identifies up to 50% more failures than surrogate-based search, while for Autoware CUT-IN it finds up to 37% more failures than HiFi. For FrenetiX, it achieves up to 20% more failures than the baselines. On average, SimFusion identifies up to $2.1\times$ more failures than single simulator-based

testing in HiFi. All differences between SimFusion and other approaches are statistically significant ($p < 0.05$) with large effect sizes. The complete table of statistical test results is provided in the replication package [17].

Ablation Study. To isolate the contribution of the orchestration mechanism, we replaced the classifier-guided simulator selection with random LoFi/HiFi selection while keeping all remaining components of SimFusion unchanged. This ablation evaluates whether the observed improvements stem from informed simulator selection rather than simply combining two simulators. Across six independent runs, SimFusion improved failure discovery by 17% for Autoware and 22% for FrenetiX compared to random orchestration. These differences were statistically significant, indicating that the observed improvements stem from informed simulator orchestration rather than simply alternating between LoFi and HiFi simulations. These results demonstrate that the performance gains originate from learning where HiFi execution is beneficial, rather than from the availability of multiple simulators alone.

RQ_2 (effectiveness). Compared to single-fidelity and surrogate assisted testing, SimFusion substantially improves failure detection by finding, on average, 40% more failures than HiFi for Autoware and 20% more failures for FrenetiX.

3) *Efficiency (RQ_3)*: Figure 3 reports the average time required to detect a failure. Across all case studies, SimFusion consistently achieves the shortest detection time. The best result is obtained for the Autoware CUT-IN scenario (0.86 min), compared to 1.3 min for HiFi. FrenetiX is the most challenging case study for all approaches; nevertheless, SimFusion

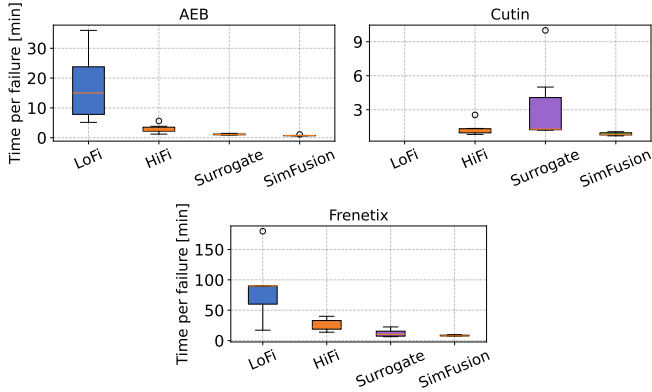


Fig. 3: Efficiency Results: Average time per failure.

detects failures in 8.30 min on average, 30% faster than the surrogate-based approach. In addition, SimFusion consistently exhibits the lowest runtime variability across all scenarios. On average across all scenarios, SimFusion identifies up to $15.85 \times$ faster failures than LoFi (52.32 minutes). Compared to HiFi, SimFusion is up to $3 \times$ faster than HiFi (9.93 minutes), and $1.7 \times$ faster than the surrogate-based approach (5.54 minutes). All differences between SimFusion and other approaches are statistically significant with large effect sizes [17].

RQ₃ (efficiency). *SimFusion discovers failures faster than baseline approaches. SimFusion finds on average up to $3 \times$ faster failures than HiFi and up to $1.7 \times$ faster than surrogate-assisted testing.*

4) *Diversity (RQ₄):* The entropy and coverage results for the evaluation of the failure diversity are provided in Table VI.

We can observe that SimFusion achieves for both Autoware case studies the highest coverage of 94% and 100%, substantially outperforming HiFi with 60% and 83%, LoFi with 23% and 0%, and the surrogate baseline with 11% and 17%. For Autoware CUT-IN SimFusion achieves the highest entropy, while for the Autoware AEB scenario the surrogate-based approach achieves a higher value.

For FrenetiX, both SimFusion and the surrogate baseline achieve higher failure coverage and entropy than the single-fidelity baselines. SimFusion achieves the highest coverage overall (0.88) and the highest entropy (0.58), indicating that it explores a broader set of failure behaviors.

These results indicate that SimFusion explores previously unseen regions of the scenario space rather than repeatedly identifying similar failure instances.

RQ₄ (diversity). *SimFusion achieves the highest failure diversity overall, with an average failure coverage of 94% and an average entropy of 0.45 across all scenarios. In the majority of cases, SimFusion outperforms all the baseline approaches regarding failure diversity.*

TABLE VI: RQ₄: Failure coverage and entropy (in %).

Approach	Autoware AEB		Autoware CUT-IN		FrenetiX	
	Coverage $\uparrow$	Entropy $\uparrow$	Coverage	Entropy	Coverage	Entropy
LoFi	23	64	0	0	22	31
HiFi	60	80	83	32	77	48
Surrogate	11	91	17	7	55	54
SimFusion	94	43	100	34	88	58

V. THREATS TO VALIDITY

1) *Internal Validity:* Differences in simulator implementations may affect scenario alignment and execution outcomes. For Autoware, we verified the alignment of pedestrian trajectories between the LoFi and HiFi simulators and selected the CARLA step size based on preliminary experiments (0.15 steps/s), while otherwise using default configurations. For FrenetiX, we rely on the MultiDrive framework [9], which ensures consistent scenario execution across the LoFi and HiFi simulators. Since LoFi outcomes may be unreliable (as shown in RQ₁), all failures by all approaches in LoFi were validated in HiFi. Implementation errors could also affect the results. To mitigate this threat, we built SimFusion on top of the OpenSBT framework [42], which provides a modular and well-tested infrastructure for search-based testing of ADS.

2) *Construct Validity:* Simulator flakiness could influence the observed failures. To mitigate this threat, we re-executed scenarios in the high-fidelity simulators whenever inconsistent outcomes occurred. Preliminary experiments did not indicate substantial flakiness. Another threat concerns whether our metrics accurately capture the effectiveness and diversity of test generation. To address this, we rely on widely used metrics for failure discovery and diversity analysis, and we repeated each experiment multiple times using randomized search-based algorithms and assessed statistical significance.

3) *External Validity:* Our evaluation considers one LoFi/HiFi pair per ADS stack. Although the orchestration strategy is independent of the underlying simulators, additional studies involving different simulator pairs and fidelity gaps would further assess the generality of the proposed approach. Our results may depend on the specific ADS stacks, simulators, and scenarios used in our evaluation. To mitigate this threat, we evaluated SimFusion on two distinct ADS systems (Autoware and FrenetiX) across four simulators with varying fidelity levels and simulation paradigms, as well as three abstract scenarios.

4) *Reproducibility:* We release SimFusion implementation, experimental scripts, case studies, and additional details to foster open science and public scrutiny [17].

VI. QUALITATIVE EVALUATION

We manually inspected for all three case studies, a subset of scenarios in which the simulators agree and disagree. Figure 4 shows a selection of test cases for agreements and disagreements. For instance, for Autoware we can observe agreements on scenarios where the ego vehicle brakes because of the approaching pedestrian but is not able to stop in time

because of its relatively high velocity and collides in HiFi with the pedestrian or violates the safety distance in LoFi (Figure 4.a). Disagreements happen mostly when the ego vehicle in HiFi does not break and collides to a running pedestrian on the left tail of the vehicles, while in LoFi a breaking makes the vehicle stop before colliding (Figure 4.b).

For FrenetiX, we observe agreements of failures, especially of different types: while in HiFi collisions happen because the ego breaks and steers slightly left, ending up in a collision with a vehicle coming from the left, while in LoFi the ego overtakes before the goal a blocked vehicle, making him violate the goal distance (Figure 4.c). In another case, in HiFi ego collides with a forwarding vehicle, while in LoFi ego breaks but collides with another vehicle from the back. We observe disagreements as illustrated in Figure 4d), when, e.g., in HiFi the ego does not let a cut-in vehicle pass and fails because of a minimal distance violation, while on LoFi the ego brakes to let the NPC merge into the lane.

VII. DISCUSSION

Applicability. The effectiveness of SimFusion depends primarily on the relationship between the two simulation fidelities rather than on the specific simulators employed. In particular, SimFusion is expected to provide the greatest benefit when LoFi and HiFi agree on the majority of scenarios, while disagreements remain concentrated in a relatively small subset of safety-critical situations. Under these conditions, inexpensive LoFi executions can safely filter most candidate scenarios, allowing computationally intensive HiFi simulation to be reserved for cases where increased realism is likely to influence the testing outcome.

Applying SimFusion to additional simulator pairs also requires that equivalent scenarios can be executed consistently across both environments. As simulators inevitably differ in map representations, vehicle dynamics, and timing models, validating scenario alignment is essential to ensure that observed disagreements arise from differences in simulation fidelity rather than inconsistencies in scenario execution. In our case studies, we verified alignment using trajectory similarity metrics, although alternative validation strategies may be appropriate depending on the simulation platforms.

Predicting Simulator Agreement vs. Approximating Simulation. Recent surrogate-assisted testing approaches reduce simulation cost by learning mappings from scenario parameters to fitness values or directly to test outcomes [15], [43]. Such approaches implicitly attempt to approximate the behavior of the complete autonomous driving system, whose outputs emerge from complex interactions among perception, planning, vehicle dynamics, control, and the environment.

SimFusion adopts a different perspective. Rather than approximating simulator outputs, it predicts whether two simulators are likely to agree on a scenario’s outcome. Our results suggest that learning simulator agreement is a substantially simpler classification problem than approximating ADS behavior itself, while still providing sufficient information to guide simulator orchestration. We believe this represents

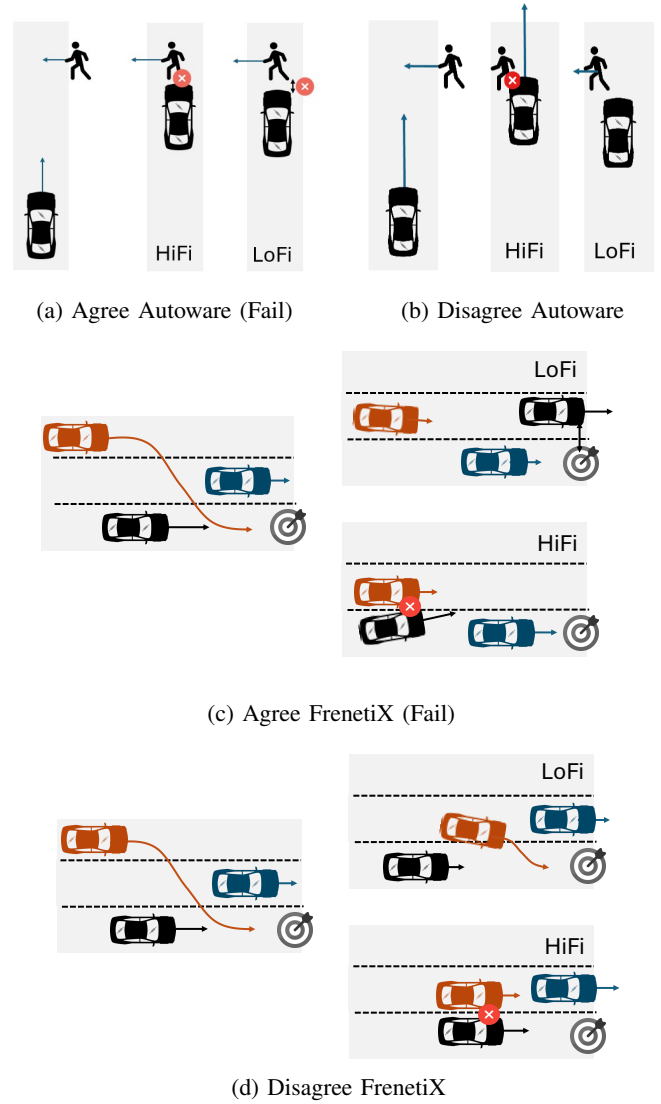


Fig. 4: Examples of simulator failure agreements and disagreements for Autoware in Fig. a), b) and FrenetiX in Fig. c). Left most figure depicts the scenario defined, while remaining figures depict simulator specific executions.

a complementary direction for reducing simulation cost in search-based testing of ADS.

Understanding Simulators Disagreement. Our empirical analysis shows that disagreements between LoFi and HiFi are not uniformly distributed across the scenario space. Instead, they are concentrated in safety-critical situations, whereas the vast majority of nominal scenarios are classified consistently by both simulators. This behavior is consistent with the abstractions introduced by low-fidelity simulation, which simplify sensing, perception, and vehicle dynamics while largely preserving nominal driving behavior.

This asymmetry explains why multi-fidelity orchestration is effective. Executing every scenario in HiFi provides little additional information for the large fraction of scenarios where both simulators agree. Conversely, selectively invoking HiFi

in regions where disagreement is likely enables computational resources to be focused where increased simulation fidelity has the greatest influence on testing outcomes.

Scenario Complexity. The benefits of SimFusion become more pronounced as the complexity of the interaction space increases. While the Autoware case studies involve relatively constrained interactions, the FrenetX scenario contains multiple vehicles, lane changes, and richer traffic dynamics, resulting in a substantially larger space of possible behaviors. In such settings, rapid exploration through LoFi simulation allows considerably more candidate scenarios to be evaluated within a fixed testing budget, while selective HiFi validation preserves the ability to identify fidelity-dependent failures.

Another perspective is that each simulator induces a different search landscape over the same scenario space. Search guided exclusively by a single simulator explores failures exposed under that simulator’s abstractions [16]. In contrast, SimFusion exploits both landscapes by concentrating computational effort on regions where simulator disagreement is most likely to occur. Similar ideas have recently been explored by combining multiple high-fidelity simulators to expose simulator-dependent failures [16], [20]. Our work shows that analogous benefits can also be achieved by orchestrating simulators across different fidelity levels.

VIII. RELATED WORK

We review existing scenario-based testing approaches that either (i) use a single simulator, (ii) use multiple simulators, or (iii) use surrogate models.

Single-simulator testing approaches generate scenarios automatically via search-based software testing or fuzzing [44]. Most current test generation methods use search-based approaches to automatically create test scenarios for DNN-based ADS [31], [45]–[52]. In these approaches, test generators vary scenario attributes, including road topologies [46], [53], traffic configurations [54], road users [31], [47], [48], and environmental elements [55]. Relevant work in fuzzing includes DriveFuzz [56], which leverages the vehicle’s physical state and oracles grounded in real-world traffic rules to guide the fuzzer in uncovering potential misbehaviors; AutoFuzz [57], which focuses on fuzzing the test scenario specifications and employs a binary classifier to identify highly critical scenario seeds; and AV-Fuzzer [54], which extends a genetic algorithm with information about the positioning of globally monitored non-player characters in each driving scenario. More recently, Li et al. [3] proposed to adaptively reconfigure MetaDrive to achieve different fidelity levels during the test generation process. Cheng et al. [58] introduce BehaviorMiner, an unsupervised model that extracts temporal features from predefined scenarios and employs clustering-based abstraction to group behaviors with similar features into abstract states. Because these approaches rely on a single simulator, their cost-effectiveness depends strongly on the simulator’s fidelity, necessitating a trade-off between computational time (for higher-fidelity simulations) and reliability of the results [59].

In contrast, our work leverages multiple simulators with different fidelity levels, enabling more efficient exploration while preserving accurate failure assessment.

Multi-simulator testing approaches use multiple simulators to improve test reliability. Borg et al. [21] and Amini et al. [60] showed that relying on a single simulator yields unreliable conclusions about the quality of the system under test because some failures stem from simulator-specific characteristics rather than issues in the SUT. To address this issue, Biagiola et al. [16] and Sorokin et al. [20] used ensembles of simulators to identify and resolve discrepancies in scenario executions across simulators. However, using multiple simulators reduces testing efficiency because test cases must be executed once per simulator. More recently, Kaufeld et al. [9] developed a co-simulation framework that integrates a low-fidelity and a high-fidelity simulator to facilitate scenario transfer and analysis across simulators of different fidelity levels. MFBO-Drive [3] applies multi-fidelity Bayesian optimization by adaptively selecting both candidate scenarios and the fidelity level at which they are evaluated, reducing computational cost while maintaining test quality. In contrast, our approach orchestrates simulators based on predicted cross-fidelity agreement, rather than optimizing fidelity selection through Bayesian optimization. Compared to these approaches, our work improves testing efficiency by combining low- and high-fidelity simulators by limiting the execution of expensive simulations while preserving ADS testing reliability.

Surrogate-based scenario generation improves testing efficiency by guiding scenario generation using computationally efficient approximations based on machine learning or probabilistic models. Mullins et al. [61] employed Gaussian Processes to steer the search toward unexplored regions of the input space, thereby increasing coverage. Biagiola et al. [37] trained a classifier on past execution data to guide failure discovery in Deep Reinforcement Learning systems. Ul Haq et al. [13] extended many-objective search to utilize surrogate models that mimic simulators’ output. Jodat et al. [15] proposed learning interpretable failure models that approximate test outcomes using multiple surrogate predictors. These approaches reduce testing costs by approximating simulator executions through predictive models. In contrast, SimFusion leverages existing simulators with different fidelity levels and dynamically orchestrates their use during testing.

IX. CONCLUSIONS

We presented SimFusion, a multi-fidelity search-based testing framework that improves the effectiveness and efficiency of search-based scenario testing by orchestrating simulators across multiple fidelity levels. Using a classifier that predicts cross-simulator agreement, SimFusion selectively executes scenarios in LoFi or HiFi simulation, combining efficient exploration with reliable failure detection. Our evaluation across two case studies, four driving simulators, and three abstract scenarios shows that using simulation agreement to guide fidelity selection identifies more diverse failures than single-fidelity and surrogate-assisted approaches.

As future work, we plan to investigate how SimFusion can be extended to leverage multiple parallel LoFi simulators combined with multiple parallel HiFi simulators. Such configurations could further accelerate exploration of the scenario space while maintaining reliable failure validation. More broadly, we aim to study how adaptive multi-fidelity simulation pipelines can support continuous testing and faster development cycles for safety-critical autonomous systems.

REFERENCES

- [1] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y. Li, L. Ma, Y. Xue, and Y. Liu, "A survey on automated driving system testing: Landscapes and trends," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, pp. 124:1–124:62, Jul. 2023. [Online]. Available: <https://doi.org/10.1145/3579642>
- [2] P. Kaur, S. Taghavi, Z. Tian, and W. Shi, "A survey on simulators for testing self-driving cars," in *2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD)*, 2021, pp. 62–70.
- [3] O. Osikowicz, P. Mcminn, W. Xing, and D. Shin, "Multi-fidelity bayesian optimization for simulation-based autonomous driving systems testing," in *IEEE Intelligent Vehicles Symposium, IV 2026*. IEEE, 2026.
- [4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [5] BeamNG.research, "BeamNG GmbH," <https://www.beamng.gmbh/research>, 2022, online; accessed 18/08/2019.
- [6] M. Althoff, M. Koschi, and S. Manzing, "Commonroad: Composible benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.
- [7] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3461–3475, 2023.
- [8] S. Sagmeister, P. Kounatidis, S. Goblirsch, and M. Lienkamp, "Analyzing the impact of simulation fidelity on the evaluation of autonomous driving motion control," in *IEEE Intelligent Vehicles Symposium, IV 2024, Jeju Island, Republic of Korea, June 2–5, 2024*. IEEE, 2024, pp. 230–237. [Online]. Available: <https://doi.org/10.1109/IV55156.2024.10588858>
- [9] M. Kaufeld, K. Moller, A. Gambi, P. Arcaini, and J. Betz, "Multidrive: A co-simulation framework bridging 2d and 3d driving simulation for av software validation," in *IEEE 28th International Conference on Intelligent Transportation Systems (ITSC 2025)*, Nov. 2025.
- [10] P. Koopman, "UL 4600: Standard for safety for the evaluation of autonomous products," 2022.
- [11] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, pp. 15–24, 04 2016.
- [12] C. Birchler, S. Khatiri, B. Bosshard, A. Gambi, and S. Panichella, "Machine learning-based test selection for simulation-based testing of self-driving cars software," *Empir. Softw. Eng.*, vol. 28, no. 3, p. 71, 2023. [Online]. Available: <https://doi.org/10.1007/s10664-023-10286-y>
- [13] F. U. Haq, D. Shin, and L. C. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, USA*. ACM, 2022, pp. 811–822.
- [14] F. U. Haq, D. Shin, and L. C. Briand, "Many-objective reinforcement learning for online testing of dnn-enabled systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023.
- [15] B. A. Jodat, A. Chandar, S. Nejati, and M. Sabetzadeh, "Test generation strategies for building failure models and explaining spurious failures," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 4, apr 2024. [Online]. Available: <https://doi.org/10.1145/3638246>
- [16] M. Biagiola, A. Stocco, V. Riccio, and P. Tonella, "Two is better than one: digital siblings to improve autonomous driving testing," *Empirical Softw. Engg.*, vol. 29, no. 4, May 2024. [Online]. Available: <https://doi.org/10.1007/s10664-024-10458-4>
- [17] "Replication package," 2026. [Online]. Available: <https://anonymous.4open.science/r/simfusion-C957/README.md>
- [18] A. Stocco, B. Pulfer, and P. Tonella, "Mind the Gap! A Study on the Transferability of Virtual Versus Physical-World Testing of Autonomous Driving Systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 04, pp. 1928–1940, apr 2023.
- [19] esmini Project, "esmini: A lightweight openscenario player for traffic simulation," <https://github.com/esmini/esmini>, 2020, accessed: 2026.
- [20] L. Sorokin, M. Biagiola, and A. Stocco, "Simulator ensembles for trustworthy autonomous driving systems testing," *Empirical Softw. Engg.*, vol. 31, no. 4, Feb. 2026. [Online]. Available: <https://doi.org/10.1007/s10664-026-10821-7>
- [21] M. Borg, R. B. Abdesslem, S. Nejati, F. Jegeden, and D. Shin, "Digital twins are not monozygotic - cross-replicating ADAS testing in two industry-grade automotive simulators," *CoRR*, vol. abs/2012.06822, 2020. [Online]. Available: <https://arxiv.org/abs/2012.06822>
- [22] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979. [Online]. Available: <http://www.jstor.org/stable/1268522>
- [23] S. Liu, X. Wang, O. Hassanin, X. Xu, M. Yang, D. Hurwitz, and X. Wu, "Calibration and evaluation of responsibility-sensitive safety (rss) in automated vehicle performance during cut-in scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 103037, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0968090X21000681>
- [24] L. Sorokin, D. Safin, and S. Nejati, "Can search-based testing with pareto optimization effectively cover failure-revealing test inputs?" *Empirical Software Engineering*, vol. 30, no. 1, p. 26, November 2024. [Online]. Available: <https://doi.org/10.1007/s10664-024-10564-3>
- [25] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*. New York, NY, USA: ACM, 2007, pp. 1187–1194. [Online]. Available: <http://doi.acm.org/10.1145/1276958.1277190>
- [26] UT-ADL, "Autoware Mini: a minimalistic python-based autonomy software," https://github.com/UT-ADL/autoware_mini, UT Autonomous Driving Lab, 2023, accessed: 2026-02-24.
- [27] A. Foundation, "Autoware core/universe," <https://github.com/autowarefoundation/autoware>, 2022–2025.
- [28] Autonomous Driving Lab, University of Tartu, "Vehicle," <https://adl.cs.ut.ee/lab/vehicle>, 2023, accessed: 2026-03-06.
- [29] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.
- [30] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.
- [31] R. B. Abdesslem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, M. Chaudron, I. Crnkovic, M. Chechik, and M. Harman, Eds. ACM, May 2018, pp. 1016–1026. [Online]. Available: <https://doi.org/10.1145/3180155.3180160>
- [32] S. Nejati, L. Sorokin, D. Safin, F. Formica, M. M. Mahboob, and C. Menghi, "Reflections on surrogate-assisted search-based testing: A taxonomy and two replication studies based on industrial adas and simulink models," *Information and Software Technology*, vol. 163, p. 107286, 2023.
- [33] R. Trauth, K. Moller, G. Würsching, and J. Betz, "Frenetix: A high-performance and modular motion planning framework for autonomous driving," *IEEE Access*, vol. 12, p. 127426–127439, 2024. [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2024.3436835>
- [34] P. Karle, T. Betz, M. Bosk, F. Fent, N. Gehrke, M. Geisslinger, L. Gressenbuch, P. Hafemann, S. Huber, M. Hübner, S. Huch, G. Kaljavesi, T. Kerbl, D. Kulmer, T. Mascetta, S. Maierhofer, F. Pfab, F. Rezabek, E. Rivera, S. Sagmeister, L. Seidlitz, F. Sauerbeck, I. Tahiraj, R. Trauth, N. Uhlemann, G. Würsching, B. Zarrouki, M. Althoff, J. Betz, K. Bengler, G. Carle, F. Diemeyer, J. Ott, and M. Lienkamp, "Edgar: An autonomous driving research platform – from feature development to real-world application," 2024. [Online]. Available: <https://arxiv.org/abs/2309.15492>
- [35] L. Quante, M. Zhang, K. Preuk *et al.*, "Human performance in critical scenarios as a benchmark for highly automated vehicles," *Automotive*

- Innovation*, vol. 4, no. 3, pp. 274–283, 2021. [Online]. Available: <https://doi.org/10.1007/s42154-021-00152-2>
- [36] W. J. Youden, “Index for rating diagnostic tests,” *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.
- [37] M. Biagiola and P. Tonella, “Testing of deep reinforcement learning agents with surrogate models,” *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 3, Mar. 2024. [Online]. Available: <https://doi.org/10.1145/3631970>
- [38] L. McInnes, J. Healy, and S. Astels, “hdbscan: Hierarchical density based clustering,” *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. [Online]. Available: <https://doi.org/10.21105/joss.00205>
- [39] L. Sorokin, I. Vasilev, K. E. Friedl, and A. Stocco, “STELLAR: A search-based testing framework for large language model applications,” in *Proceedings of the 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 2026.
- [40] A. Arcuri and L. Briand, “A hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering,” *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.
- [41] A. Vargha and H. D. Delaney, “A critique and improvement of the cl common language effect size statistics of mcgraw and wong,” *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.
- [42] L. Sorokin, T. Munaro, D. Safin, B. H.-C. Liao, and A. Molin, “OpenSBT: A modular framework for search-based testing of automated driving systems,” New York, NY, USA, p. 94–98, 2023. [Online]. Available: <https://github.com/opensbt/opensbt-core>
- [43] C. Birchler, N. Ganz, S. Khatiri, A. Gambi, and S. Panichella, “Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor,” in *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022*. IEEE, 2022, pp. 164–168. [Online]. Available: <https://doi.org/10.1109/SANER53432.2022.00030>
- [44] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, and Y. Liu, “A survey on automated driving system testing: Landscapes and trends,” *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, Jul. 2023. [Online]. Available: <https://doi.org/10.1145/3579642>
- [45] M. H. Moghadam, M. Borg, M. Saadatmand, S. J. Mousavirad, M. Bohlin, and B. Lisper, “Machine learning testing in an adas case study using simulation-integrated bio-inspired sbt,” *J. Softw.: Evol. Process*, p. e2591.
- [46] V. Riccio and P. Tonella, “Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing,” in *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE ’20, 2020.
- [47] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, “Testing autonomous cars for feature interaction failures using many-objective search,” in *Proceedings of ASE ’18*, ser. ASE 2018. ACM, 2018, pp. 143–154. [Online]. Available: <http://doi.acm.org/10.1145/3238147.3238192>
- [48] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, “Testing advanced driver assistance systems using multi-objective search and neural networks,” in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2016, pp. 63–74.
- [49] K. Pei, Y. Cao, J. Yang, and S. Jana, “Deepxplore: Automated whitebox testing of deep learning systems,” in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP ’17. New York, NY, USA: ACM, 2017, pp. 1–18. [Online]. Available: <http://doi.acm.org/10.1145/3132747.3132785>
- [50] Y. Tian, K. Pei, S. Jana, and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of ICSE ’18*, ser. ICSE ’18. New York, NY, USA: ACM, 2018, pp. 303–314. [Online]. Available: <http://doi.acm.org/10.1145/3180155.3180220>
- [51] 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 ASE ’18*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 132–142. [Online]. Available: <http://doi.acm.org/10.1145/3238147.3238187>
- [52] J. Kim, R. Feldt, and S. Yoo, “Guiding deep learning system testing using surprise adequacy,” in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE ’19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 1039–1049. [Online]. Available: <https://doi.org/10.1109/ICSE.2019.00108>
- [53] A. Gambi, M. Mueller, and G. Fraser, “Automatically testing self-driving cars with search-based procedural content generation,” in *Proceedings of ISSTA ’19*, 2019.
- [54] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, “Av-fuzzer: Finding safety violations in autonomous driving systems,” in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 25–36.
- [55] Y. Sun, H. Wang, C. M. Poskitt, and J. Sun, “Natural adversaries: Fuzzing autonomous vehicles with realistic roadside object placements,” in *Proc. IEEE International Conference on Software Testing, Verification and Validation (ICST’26)*. IEEE, 2026, to appear.
- [56] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, “Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1753–1767. [Online]. Available: <https://doi.org/10.1145/3548606.3560558>
- [57] Z. Zhong, G. Kaiser, and B. Ray, “Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles,” *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, p. 1860–1875, Apr. 2023. [Online]. Available: <https://doi.org/10.1109/TSE.2022.3195640>
- [58] M. Cheng, Y. Zhou, and X. Xie, “Behavexplor: Behavior diversity guided testing for autonomous driving systems,” in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 488–500. [Online]. Available: <https://doi.org/10.1145/3597926.3598072>
- [59] S. Wagner, K. Groh, T. Kühbeck, and A. Knoll, “Towards cross-verification and use of simulation in the assessment of automated driving,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1589–1596.
- [60] M. H. Amini, S. Naseri, and S. Nejati, “Evaluating the impact of flaky simulators on testing autonomous driving systems,” *Empirical Softw. Engg.*, vol. 29, no. 2, feb 2024. [Online]. Available: <https://doi.org/10.1007/s10664-023-10433-5>
- [61] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, “Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles,” *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121217302546>