Scenario-Based Testing of Automated Driving Functions Using Sampling and Iterative Optimization in a Modular Framework

Martin Kirchengast*, Gerhard Benedikt Weiß*, Bernhard Hillbrand*, Selim Solmaz*
*Virtual Vehicle Research GmbH
Graz, AUSTRIA
Email: martin.kirchengast@v2c2.at

***Abstract*—In scenario-based testing of automated driving functions (ADF) the instantiation of concrete scenarios and test cases from logical scenario descriptions is an essential task. This work compares an open-loop method, where the scenario samples are defined prior to simulation execution, with a closed-loop procedure. The latter one specifically searches for scenario parameter combinations for ADF falsification and exploration of critical parameter space regions. The open-loop approach is based on statistical sampling and pseudo-random number generation for space filling. In contrast, the closed-loop approach involves iterative scenario simulation and evaluation of the ADF behavior by a black box optimizer. This approach comprises a modular framework for virtual testing of autonomous driving, structured around a base layer. A fundamental aspect of the framework's architecture is the harmonization of subsystem interfaces within the base layer, enabling a truly modular design. To ensure seamless integration, adopting an open standard for defining subsystem interfaces is advantageous. In this context, the ASAM Open Simulation Interface (OSI) standard is particularly suitable, as it facilitates efficient and straightforward compatibility between the various subsystems. A co-simulation platform manages the execution of subsystems, while the proposed method selects scenario parameters to identify critical concrete scenarios efficiently, avoiding exhaustive exploration of the entire parameter space.**

***Index Terms*—Parameter sampling, optimization, harmonization, simulation framework**

I. Introduction

Automated driving has been a topic of intensive research for years. At the same time, the need for virtual testing of the developed driving functions has also increased. Scenario-based testing is becoming increasingly important in the development, verification and validation of automated driving systems (ADS). In order to ensure the correct functionality of the ADS, various scenarios must be tested to verify the right behavior in situations within the Operational Design Domain (ODD). But even within a scenario, the various parameter ranges must be covered. This results in a huge number of test cases and the question of how to find critical test cases that could lead to potential safety risks.

There are already several methods to identify critical scenarios that can be divided into different categories. In [1], such a categorization was made. Some methods explore logical scenarios without parameter trajectories. These methods are divided into naive search (sampling and combinatorial testing) and guided search (optimization and learning-based testing). Sampling methods include near-random sampling, such as Latin Hypercube sampling [2] and exhaustive search [3], [4], while combinatorial testing generates a minimum set of test cases to satisfy N-wise coverage [5]. Optimization methods use objective functions representing criticality measures and employ various heuristic techniques [6]. Learning-based testing combines model checking with model inference algorithms [7]. Another category is the exploration methods for logical scenarios that include parameter trajectories. These methods can be categorized into optimization, path planning, and reinforcement learning. Optimization assumes that the values at different time steps of a trajectory are independent and includes a cost function designed to guide the search for a trajectory that forces the system of interest to fail (e.g., Bayesian optimization) [8]. Path planning includes algorithms like Rapidly Exploring Random Trees (RRT) [9] and optimal control [10]. Reinforcement learning programs agents by giving rewards based on their actions without specifying how the task should be accomplished. It includes algorithms like Monte Carlo tree search and deep reinforcement learning. These methods model the problem as a Markov decision process (MDP) and use techniques like deep Q-learning to learn the behavior of adversarial agents [11].

This paper formulates the test case generation as an optimization problem and therefore facilitates guided search of critical regions in the parameter space II-C. To execute the tests a harmonized simulation framework (introduced in [12]) is built, which ensures modularity, interoperability, and scalability and is described in section II-D.

II. Methodology

All ADFs below SAE-Level 5 are restricted to a designated ODD. The ODD defines the set of conditions, situations, and circumstances under which an ADF has been designed to function and operate safely (e.g., environmental or geographic restrictions, the presence or absence of certain traffic or road geometry elements, etc.) [13]. A common approach for testing and verification of ADFs relies on the definition of a set of scenarios containing all safety-relevant situations which can occur within (or at the border of) the given ODD. These so-called logical scenarios, which include among others road geometry, static objects, and dynamic object paths / trajectories, still comprise adjustable parameters and their constraints. As soon as a single parameter is real-valued, one logical scenario has infinitely many concrete realizations [14]. Selecting feasible values for all parameters yields a concrete scenario and together with some assessment criteria one obtains a test case. The evaluation of the ADF behavior in those test cases builds the foundation for safety argumentation. Assuming the selected logical scenarios to sufficiently cover the ODD, there remains the question of how to choose individual test cases from a potentially infinite number. This process is denoted as *concretization* in the rest of the document.

Due to the typically high number of test cases, it is common practice to execute a major amount in simulation environments to reduce time and effort [14]. Thus, the concretization approach is combined with a modular simulation framework (Section II-D) to demonstrate its applicability for seeking the safety limits of an ADF.

A. Scenario Parameters

To derive concrete scenarios from logical scenarios feasible values must be picked for all parameters. This work focuses on parameters such as initial conditions of dynamic systems or sensor parameters, which remain constant during simulation, contrary to parameter trajectories [1]. Assume a logical scenario comprises parameters from $\mathbb{R}$, $\mathbb{Z}$, and some finite sets $\mathbb{F}_1$, $\mathbb{F}_2$, ... Then the parameter space is

$$\mathcal{P} = \mathbb{R}^{N_r} \times \mathbb{Z}^{N_z} \times \mathbb{F}_1^{N_{f_1}} \times \mathbb{F}_2^{N_{f_2}} \times \dots \tag{1}$$

with total parameter number $N = N_r + N_z + N_{f_1} + N_{f_2} + \dots$ and $N_r, N_z, N_{f_1}, N_{f_2}, \dots \in \mathbb{N}$. A concrete scenario reads as

$$\mathbf{s} \in \mathcal{S} \quad \text{with} \quad \mathcal{S} \subseteq \mathcal{P} \tag{2}$$

where $\mathcal{S}$ is the space of feasible concrete scenarios due to potential parameter constraints.

B. Scenario evaluation

Let $N_o \in \mathbb{N}$ be the number of other traffic participants in the scenario and $T_{sim} > 0$ the simulation time. The test cases are assessed as *passed* or *failed* depending on predefined thresholds for the following metrics:

1) Minimum Time-to-collision: The time-to-collision (TTC) between two entities is defined as time until a collision would occur if they continued on their current path and speed [15]. The minimum TTC for concrete scenario $\mathbf{s}$ is defined as

$$\underline{T}_{tc}(\mathbf{s}) = \min_{\substack{i \in \{1 \dots N_o\} \\ 0 \le t \le T_{sim}}} T_{tc}(\mathbf{s}, i, t) \tag{3}$$

with $0 \le T_{tc}(\mathbf{s}, i, t) \le \infty$ yielding the TTC between ego and i-th entity at time instant t.

2) Minimum distance: Let $\mathbf{p}_{ego}, \mathbf{p}_{o,i} : \mathcal{S} \times \mathbb{R} \to \mathbb{R}^2$ be the time-varying positions of ego and i-th other entity in the inertial reference frame, respectively. The minimum distance to others the ego reaches in scenario $\mathbf{s}$ reads as

$$\underline{d}(\mathbf{s}) = \min_{\substack{i \in \{1 \dots N_o\} \\ 0 \le t \le T_{sim}}} ||\mathbf{p}_{ego}(\mathbf{s}, t) - \mathbf{p}_{o,i}(\mathbf{s}, t)||_2. \tag{4}$$

Even if $\underline{T}_{tc}(\mathbf{s}) = \infty$, which means no collision has occurred, the distance between ego and others could be extremely small. By demanding a lower limit for (4) this safety issue is addressed. Furthermore, it proved beneficial to include (4) in optimization (Section II-C2).

A test case scores *failed* if $\underline{T}_{tc}(\mathbf{s}) < T_{tc,thr}$ or $\underline{d}(\mathbf{s}) < d_{thr}$ with $T_{tc,thr} \ge 0$ and $d_{thr} > 0$ being TTC and distance thresholds.

C. Scenario Concretization

Concretization selects scenarios from (2) based on one of the following strategies:

- Open-loop: The parameter combinations are determined from (2) prior to test case execution and therefore unaffected by the performance of the ADF under test. These approaches are often denoted as sampling methods and their results as samples [1].
- Closed-loop: Initially a certain number of test cases is generated and executed. The subsequent points in $\mathcal{S}$ are selected based on simulation outcomes.

1) Open-loop: For most practically relevant scenarios, exhaustive search in $\mathcal{S}$ is computationally intractable, especially when real-valued parameters are involved. This approach solves this issue via statistical sampling and quasi-random sequences. The modular implementation supports any generator function whose outputs $\mathbf{c} = [c_1 \dots c_N]^T$ lie within the unit hypercube, i.e., $\forall i : c_i \in \mathbb{R} \big| 0 \le c_i \le 1$. Afterwards those samples are transformed into admissible parameter space $\mathcal{S}_{OL}$, which is constructed from individual parameter bounds

$$\mathcal{S}_{OL} = \mathcal{S}_1 \times \dots \times \mathcal{S}_{N_r+N_z}. \tag{5}$$

For every numeric parameter one can specify $\mathcal{P}_i$ separately, e.g., for real parameters

$$\mathcal{S}_i = \left\{ r \in \mathbb{R} \middle| (l_1 \leq r \leq u_1) \vee (l_2 \leq r \leq u_2) \ldots \right\}. \quad (6)$$

Constraints involving several parameters (e.g., $0 \leq r_1 \leq r_2$) are currently not available in the open-loop approach.

One supported generator function is Latin Hypercube Sampling (LHS) which draws near-random samples from multidimensional distributions. Its advantage over pure random sampling is that $\mathcal{S}$ is evenly explored, requiring fewer samples to get a good representation [16]. Alternatively to LHS, samples can be created from Sobol pseudo-random sequence, which exhibits good space filling properties in high-dimensional spaces. The corresponding implementations from [17] are utilized to generate $\mathbf{c}$.

2) Closed-loop: The open-loop approach potentially needs a large sample number to reach unsafe test cases, especially in high-dimensional $\mathcal{S}$. Each sample requires a possibly time-consuming simulation, which leads to high effort in total. To address this issue the paper proposes to specifically search for critical parameter combinations via optimization. Starting with an initial guess (consists of one or several distinct samples) the simulation results are feed into an objective function, which is a numerical representation of the safety measure. Gradients or Hessians are not available as the objective evaluation involves execution of the modular simulation framework with exchangeable components (ADF, vehicle dynamics, etc.) described in Section II-D. Therefore, derivative-free solution methods are applied. Assuming that $\mathbf{s} \in \mathbb{R}^N$ the optimization problem's generic form is

$$\min_{\mathbf{s} \in \mathbb{R}^N} J(\mathbf{s}) \qquad \text{subject to} \quad (7a)$$

$$\mathbf{l}_b \leq \mathbf{s} \leq \mathbf{u}_b \quad (7b)$$

$$\mathbf{h}(\mathbf{s}) \leq \mathbf{0} \quad (7c)$$

with lower and upper parameter bounds $\mathbf{l}_b, \mathbf{u}_b \in \mathbb{R}^N$, zero vector $\mathbf{0} \in \mathbb{R}^N$, and $N_c \geq 0$ nonlinear constraints $\mathbf{h} : \mathbb{R}^N \to \mathbb{R}^{N_c}$. To promote unsafe test cases the objective function in (7a) is chosen as combination of the evaluation metrics (3) and (4). Since $\underline{T}_{tc}(\mathbf{s}) = \infty$ if there is no collision an auxiliary function

$$\underline{T}^*_{tc}(\mathbf{s}) = \begin{cases} \underline{T}_{tc}(\mathbf{s}) & \text{if } \underline{T}_{tc}(\mathbf{s}) < \infty \\ T_{tc,max} & \text{else} \end{cases} \quad (8)$$

is introduced, that limits TTC to a finite $T_{tc,max} > 0$. Using (8) the objective function is stated as

$$J(\mathbf{s}) = w_1 \left| \underline{d}(\mathbf{s}) - d_{des} \right| + w_2 \left| \underline{T}^*_{tc}(\mathbf{s}) - T_{tc,des} \right| \quad (9)$$

where $d_{des} \geq 0$, $T_{tc,des} \geq 0$, $w_1 > 0$ and $w_2 > 0$ are desired distance and TTC, and the corresponding weighting factors, respectively. Including $\underline{d}(\mathbf{s})$ in (9) significantly improves convergence, since whenever there is no collision $\underline{T}^*_{tc}(\mathbf{s}) = T_{tc,max}$, which does not provide any direction information to the solver. With non-zero d_{des} and $T_{tc,des}$ one can guide the search rather into the boundary region between passed and failed than to the worst result. To solve (7) the following algorithms from the *Global Optimization Toolbox* of [17] are applied:

a) Pattern Search: (PS) starts at a random $\mathbf{s} \in \mathcal{S}$. During polling phase it evaluates $J(\mathbf{s})$ at a set of points (the mesh points) around the current point. If a mesh point with lower objective function value is found, it becomes the new current point. The mesh size is adjusted based on the success of the poll. If a better point is found, the mesh size is expanded, otherwise, it is contracted. Now polling starts again until a stopping criterion is met.

b) Genetic Algorithm: (GA) begins with an initial population of candidate samples. In selection phase the individuals with the best $J(\mathbf{s})$ are selected for generating the next population. Crossover combines parameters from selected samples to create new offspring. Mutation introduces random changes to generate new samples. The next generation (best, crossover, and mutation samples) is expected to contain better solutions than the current one and start again with selection.

c) Particle Swarm: (PSW) starts with an initial particle population, each with random velocity. During evaluation $J(\mathbf{s})$ is obtained for each particle, identifying the lowest function value and the corresponding location. Each particle's velocity is updated based on its current velocity, its encountered best position, and neighboring best positions. The new particle positions are the old ones plus the velocities, adjusted to satisfy (7b) and evaluation phase is repeated. PSW does not support (7c).

d) Surrogate Optimization: (SO) determines $J(\mathbf{s})$ at a set of initial points and builds an internal surrogate model by fitting a radial basis function. For several thousand random samples fulfilling the constraints a merit function, considering both the surrogate value and the distances from points where the expensive simulation has been previously executed, is evaluated. The best sample w.r.t. the merit function is selected as adaptive point where $J(\mathbf{s})$ is evaluated and the surrogate model adapted accordingly. The search for the best merit function value is repeated until stopping criteria are met.

D. Simulation Framework

An adequate simulation framework that incorporates all relevant systems is crucial for virtual testing of ADS as described in [18]. In the EU Project SUNRISE, a harmonized simulation framework is introduced as part of the broader Safety Assurance Framework [12]. The objective is not to prescribe specific tools, but rather to develop a simulation architecture that facilitates interoperability. By employing standardized interfaces and common data formats, the framework enables users

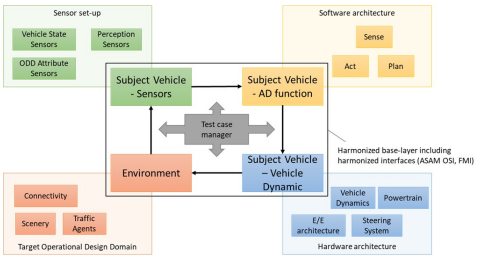


Fig. 1: Overview of the harmonized Simulation Framework proposed by [12].

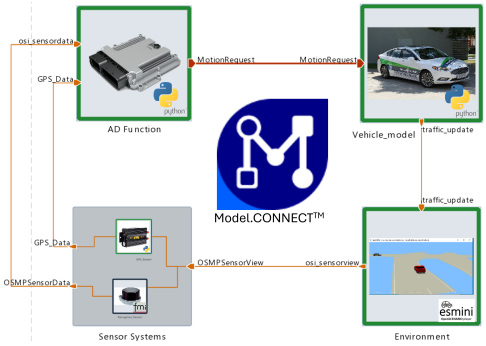


Fig. 2: Simulation topology of the framework implemented in Model.CONNECT

to switch between tools seamlessly without needing to redesign the overall system. This approach supports the creation of a modular, flexible, and scalable simulation framework that can adapt to varying project requirements and technological developments.

Fig. 1 shows the final version of the harmonized simulation framework. At its foundation is a base layer consisting of four interconnected subsystems: the Subject Vehicle – Vehicle Dynamics, the Subject Vehicle – Sensors, the Subject Vehicle – AD Function, and the Environment in which the vehicle operates. This base layer represents the core structure that can be harmonized, as these four subsystems are essential to all simulation configurations. Consequently, standardized interfaces, such as ASAM OSI, can be utilized to guarantee modularity and interoperability among these components. For simplicity, the Environment block also incorporates traffic simulation and scenario engine, even though standard interface definitions for these elements are currently lacking. Beyond the functional interconnections, a Test Case Manager is included to coordinate the execution of test cases and underlying scenarios. It also facilitates the calculation of key performance indicators (KPIs). The base layer is closely aligned with the ASAM OSI standard [19]. Moreover, ASAM OpenSCENARIO [20] and OpenDRIVE [21] are proposed as common formats for scenario and road network descriptions, promoting consistency and interoperability within the framework.

Based on the harmonized simulation framework, Fig. 2 shows the simulation topology used in this paper:

1) Test Case Manager: The simulation platform AVL Model.CONNECT [22], jointly developed by ViF and AVL, is used as the co-simulation master. It provides the capability to integrate models based on standardized interfaces (Functional Mockup Interface, FMI) as well as based on specific interfaces to a wide range of well-known simulation tools. The optimization process described in Section II-C is implemented in Matlab and acts as test case manager according to Fig.1.

2) Subject Vehicle - AD Function and Vehicle Dynamics: As outlined in [18] and [23] a python module named RoMPaC (Robust Motion Planning and Control) has been developed by ViF to support rapid prototyping of ADFs, with a particular focus on path planning and control tasks. This framework primarily consists of dedicated components for ADF and vehicle model. The ADF follows a block-based structure, encapsulating the critical sub-tasks such as perception, state estimation, decision making, trajectory planning, and trajectory tracking. The vehicle model is implemented as a low-fidelity, offering different variants such as single-track and double-track vehicle models with adjustable levels of detail. This setup allows the establishment of a closed-loop architecture that supports fast and efficient algorithm testing.

The vehicle subsystem receives actuation commands (gas pedal, brake pedal, and steering angle) from the ADF. These commands correspond to the OSI *MotionRequest* message format. Conversely, the ADF subsystem receives two inputs: object list from the perception sensor (*OSI SensorData*) and ego vehicle pose.

3) Subject Vehicle - Sensor: The perception sensor model is based on the approach described in [24]. It processes the ground truth object list and outputs detected objects after a two-stage filtering process: field of view (FOV) filter, occlusion filter. The FOV filter can be configured by adjusting parameters such as the detection range and opening angle, while for the occlusion filter, a visibility threshold is configured to determine whether an object's bounding box is considered visible. The perception model is implemented in C++ and utilizes the OSMP (OSI Sensor Model Packaging) framework [25], which defines standardized methods for packaging sensor models based on the FMI 2.0 standard.

4) Environment: As photorealistic rendering is not needed, esmini is used as simple environment simulator. It supports ASAM OpenSCENARIO for dynamic content simulation, ensuring compatibility with standardized scenario descriptions. The ego position is embedded into the OSI *Traffic Update* signal, which is subsequently received by esmini. As output, esmini provides the ideal object list from the environment's OSI *SensorView*, serving as basis for further sensor model processing.

5) Co-Simulation Settings: Another important aspect of the framework is the co-simulation configuration,

which reads as:

- All subsystems use coupling step size $0.02s$
- Sequential scheduling is employed to reduce coupling errors and maintain higher accuracy
- The execution sequence begins with the vehicle dynamics subsystem, as extrapolating object-based signals (such as OSI *SenorView*) is more complex than extrapolating scalar signals (such as gas pedal or brake pedal positions).

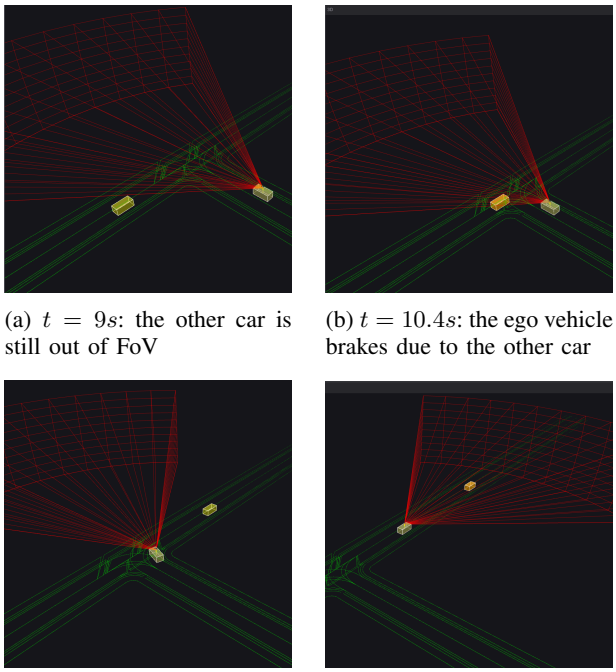
Additionally, Zero-Order Hold is selected as the extrapolation technique,specifically from the ADF to the vehicle dynamics subsystem, ensuring a simple and stable coupling between components.

III. Results

A right-turn maneuver at a T-junction is selected as the use case for this study. The ego vehicle, which is controlled by the ADF, arrives at the junction and needs to give right of way. Another vehicle drives across the junction with constant velocity, uninfluenced by the ego vehicle's behavior (see fig. 3). To identify potentially critical scenarios the following parameters are varied: FoV (ϕ_h), range (r_{sensor}), other vehicle's starting position (s_{other}), and velocity (v_{other}). The ego vehicle's target velocity is defined by the road speed limit (17.88 m/s). Fig. 3 presents an example of a near-critical scenario. Fig. 4 shows the most relevant time-series data for the scenario. In the upper and middle subplots, it is evident that the opponent vehicle is detected between 10s and 12s, prompting the ego vehicle to decelerate. Once the turn is completed, the ego vehicle accelerates back to the speed limit using ACC. Because the opponent vehicle travels at a lower speed, the ego vehicle eventually slows down again to maintain a safe following distance. The lower subplot of Fig. 4 shows the distance between the two vehicles over time, including a critical threshold of 4 meters. At 11 seconds, the actual distance approaches but does not fall below this threshold. Consequently, the scenario is classified as non-critical.

A. Open-loop

Open-loop sampling is utilized to demonstrate that simulation framework and scenario evaluation generate meaningful results. Furthermore, it facilitates the visual inspection of the evaluation for variations of up to three parameters. After specifying number of cases and sampling method the test cases are independent and can be executed in parallel. Fig. 5 depicts the results for varying the two sensor parameters while the others remain constant. In that setting ϕ_h seems to be the major influencing factor regarding evaluation. In Fig. 6 one recognizes that the other vehicle must start within a small region to induce interaction with the ego. Fig. 7 shows 300 samples varying v_{other}, s_{other}, and r_{sensor}



(a) $t = 9s$: the other car is still out of FoV

(b) $t = 10.4s$: the ego vehicle brakes due to the other car

(c) $t = 14.4s$: the ego vehicle proceeds with the right turn while the other car is again outside its FoV

(d) $t = 18s$: the other car appears in the FoV, and Adaptive Cruise Control (ACC) is active

Fig. 3: Scenario visualized with Lichtblick

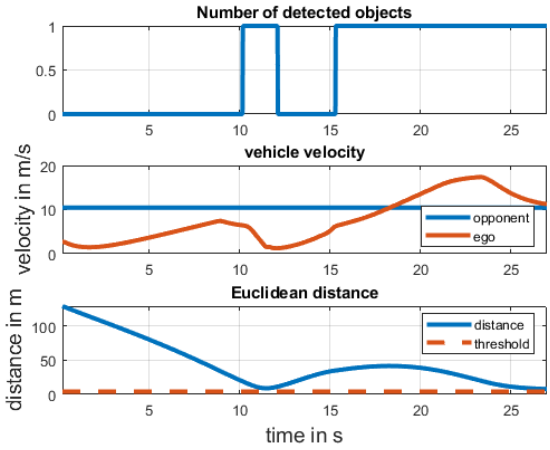


Fig. 4: most relevant time-series data of scenario

with constant ϕ_h. It illustrates that v_{other} and s_{other} must match to reach the junction at a time such that the ego must react. Looking at regions with only passed evaluation results, it is tempting to conclude that they do not contain safety critical cases. However, the isolated failed sample in Fig. 7 proves this misleading even for simple scenarios.

B. Closed-loop

The optimization-based approach searches for critical cases by minimizing (9). The results in this section

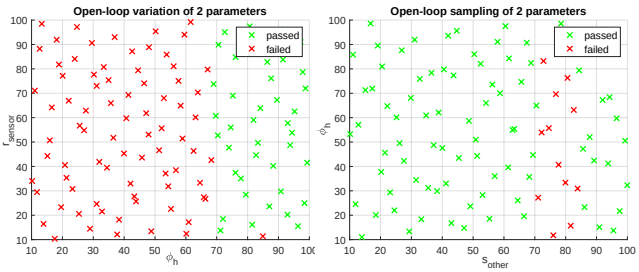


Fig. 5: Evaluation results for open-loop variation of ϕ_h and r_{sensor}.

Fig. 6: Open-loop variation shows critical cases for $\phi_h < 85$ for certain s_{other}.

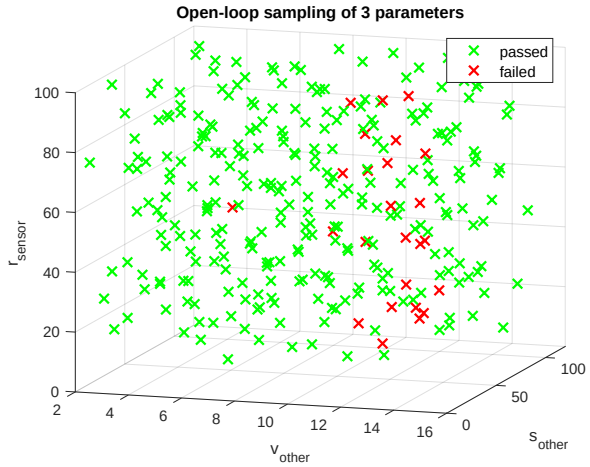


Fig. 7: Critical cases occur when s_{other} and v_{other} cause the other vehicle to stay outside the ego's detection area, while reaching the junction just after the ego.

are obtained with the following objective function parameters: $w_1 = w_2 = 1$, $d_{des} = T_{tc,des} = 0$, and $T_{tc,max} = 15$. Four solvers are compared for their applicability to the problem and their parameters can be found in Table I. The optimization and open-loop sampling results for all two-, three-, and four-dimensional parameter combinations are shown in Fig. 8, Fig. 9, and Fig. 10, respectively. Each bar color represents one distinct combination of varied and constant parameters (the legends list the varied parameters). The solver's maximum number of function evaluations has been set to the number of open-loop samples. Although the surrogate optimization implementation in [17] has no objective tolerance parameter, this feature was realized via a custom implementation. The bottom bar charts in Fig. 8 and Fig. 9 indicate the objective function value (the smaller the better) and whether at least one failed test case w.r.t. (3) and (4). is found. Minimum TTC and distance together with their corresponding limits for scoring *failed* are depicted. SO and PSW find the most critical cases for all parameter combinations with a modest number of function evaluations, which involve the costly simulation executions. The open-loop samples

TABLE I: Solver parameters.

Parameter	Value	Applied to solver
Objective tolerance	0.05	PS, GA, PSW, SO
Step tolerance	0.1	PS
Max. stall iterations	4	GA, PSW
Swarm / pop. size	10	GA, PSW
Min. sample dist.	0.1	SO
Max. fun eval. 2D	120	PS, GA, PSW, SO
Max. fun eval. 3D	300	PS, GA, PSW, SO
Max. fun eval. 4D	400	PS, GA, PSW, SO

also contain critical cases for all variations, but with higher numbers of function evaluations. PS and GA sometimes struggle to find failed test cases.

IV. Conclusion

This paper proposes a closed-loop scenario concretization approach and compares different algorithms for solving the related optimization problems. Its applicability is demonstrated using a modular simulation framework for executing right-turn test cases with two-, three-, and four-dimensional parameter combinations. SO found the most critical cases in all tests while requiring only a low number of function evaluations. PSW achieved similar results especially for three-, and four-dimensional problems. Although open-loop sampling also found critical cases for all combinations it requires a higher amount of costly function evaluations compared to closed-loop approach. Future work will incorporate the application of the approach for other scenarios including multiple traffic participants and impact studies of choosing different objective functions.

Acknowledgment

The publication was written at Virtual Vehicle Research GmbH in Graz, Austria and partially funded by the research project "SUNRISE", which has received funding from the European Union's Horizon 2020 Research & Innovation Actions under grant agreement No. 101069573. The authors would like to acknowledge the financial support within the COMET K2 Competence Centers for Excellent Technologies by the Austrian Federal Ministry for Innovation, Mobility and Infrastructure (BMIMI), Austrian Federal Ministry for Economy, Energy and Tourism (BMWET), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

References

[1] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica, and H. Felbinger, "Finding critical scenarios for automated driving systems: A systematic literature review," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2110.08664

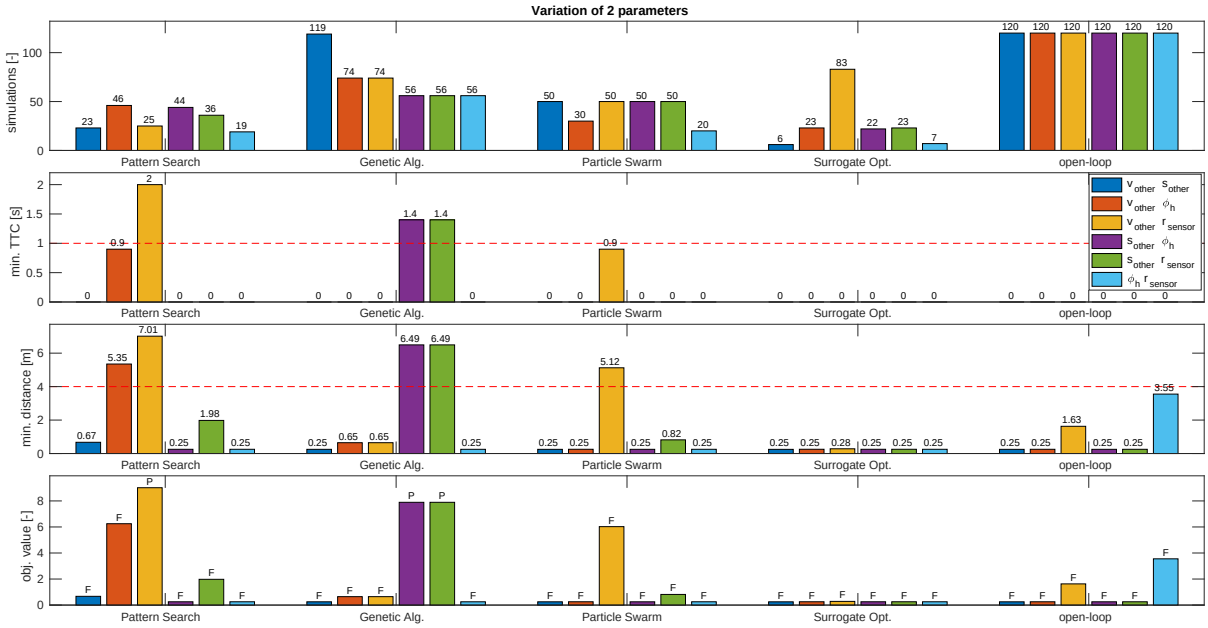


Fig. 8: Comparison of closed-loop with open-loop sampling for all combinations of two out of four parameters. The top chart shows the number of simulations and the two middle charts the evaluation criteria (3) and (4). At the bottom the objective function value is visualized together with the test case evaluation result (passed or failed).

[2] F. Batsch, A. Daneshkhah, M. Cheah, S. Kanarachos, and A. Baxendale, "Performance boundary identification for the evaluation of automated vehicles using gaussian process classification," *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 419–424, 2019.

[3] X.Wang, H. Peng, and D. Zhao, "Combining reachability analysis and importance sampling for accelerated evaluation of highway automated vehicles at pedestrian crossing," *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 1, 2011.

[4] J. Zhou and L. del Re, "Safety verification of adas by collisionfree boundary searching of a parameterized catalog," *Annual American Control Conference (ACC)*, pp. 4790–4795, 2018.

[5] D. R. Kuhn, R. Bryce, F. Duan, L. S. Ghandehari, Y. Lei, and R. N. Kacker, "Combinatorial testing: Theory and practice," *Advances in Computers*, vol. 99, pp. 1–66, 2015.

[6] M. J. Kochenderfer and T. A. Wheeler, "Algorithms for optimization," *Mit Press*, 2019.

[7] K. Meinke, F. Niu, and M. Sindhu, "Learning-based software testing: a tutorial," *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, p. 200–219, 2011.

[8] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," *IEEE International Conference on Robotics and Automation*, vol. 2019-May, p. 8271–8277, 2019.

[9] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," 1998. [Online]. Available: https://ci.nii.ac.jp/naid/10014962955

[10] R.Sargent, "Optimal control," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 361–371, 2000.

[11] L. P. Kaelbling, M. L. Littman, , and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 237–285, 1996.

[12] B. Hillbrand, "SUNRISE Deliverable D4.4.: Report on the Harmonised V & V simulation framework," European Commission, Horizon Europe Project SUNRISE, Brussels, Belgium, EU Deliverable D4.4, September 2024, public Deliverable. [Online]. Available: https://ccam-sunrise-project.eu/deliverable/d4-4-report-on-the-harmonised-vv-simulation-framework/

[13] ISO Central Secretary, "Road vehicles — test scenarios for automated driving systems — specification for operational design domain," International Organization for Standardization, Geneva, CH, Standard ISO 34503:2023(E), 2023. [Online]. Available: https://www.iso.org/standard/62711.html

[14] J. Allen, W. Koo, D. Murugesan, and C. Zagorski, "Testing methods and recommended validation strategies for active safety to optimize time and cost efficiency," in *SAE Technical Papers*, 04 2020.

[15] S. S. Mahmud, L. Ferreira, M. S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," *IATSS Research*, vol. 41, no. 4, pp. 153–163, 2017.

[16] 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.

[17] MathWorks. MATLAB. Accessed on 02 May 2025. [Online]. Available: https://www.mathworks.com/help/matlab/index.html

[18] P. Weissensteiner, G. Stettinger, J. Rumetshofer, and D. Watzenig, "Virtual validation of an automated lane-keeping system with an extended operational design domain," *Electronics*, vol. 11, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/1/72

[19] ASAM. Open simulation interface. Accessed on 28 April 2025. [Online]. Available: https://github.com/OpenSimulationInterface

[20] ——. Openscenario. Accessed on 28 April 2025. [Online]. Available: https://www.asam.net/standards/detail/openscenario

[21] ——. Opendrive. Accessed on 28 April 2025. [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[22] AVL. Model.connect. Accessed on 29 April 2025. [Online]. Available: https://www.avl.com/en/simulation-solutions/software-offering/simulation-tools-a-z/modelconnect

[23] K. Festl, J. Rumetshofer, M. Stolz, and D. Watzenig, "A best practice for the lean development of automated driving function concepts to reduce integration risks," 09 2020, pp. 1–6.

[24] S. Genser, S. Muckenhuber, C. Gaisberger, S. Haas, and T. Haid, "Occlusion model—a geometric sensor modeling approach for

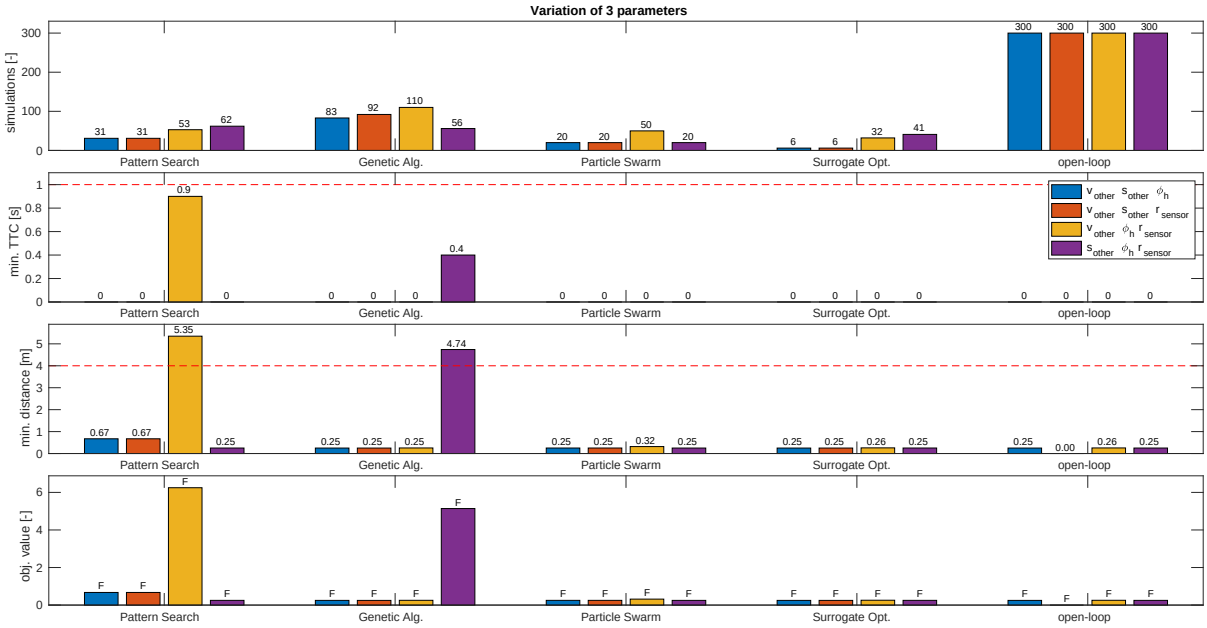


Fig. 9: Closed-loop and open-loop methods for all combinations of three out of four parameters.

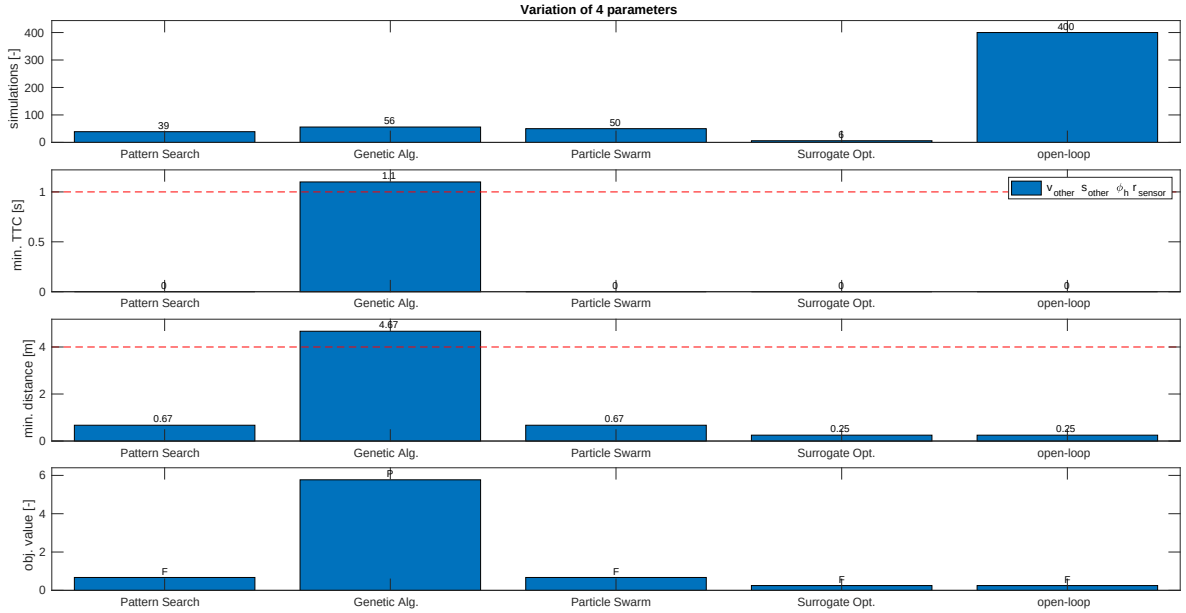


Fig. 10: Closed-loop and open-loop results with all parameters being variable.

virtual testing of adas/ad functions," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 439–455, 2023.

[25] Institute of Automotive Engineering at TU Darmstadt. Modular OSI Sensor Model Packaging FMU Framework with Example Strategies. Accessed on 02 May 2025. [Online]. Available: https://github.com/TUDa-FZD/Modular-OSMP-Framework