F. Beringhoff, J. Greenyer, C. Roesener and M. Tichy, "Realizing Scenario-Based Verification Tests of Automated Vehicles With an AI-Controlled Surrounding Vehicle in a Practice-Relevant Context," 2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC), Edmonton, AB, Canada, 2024, pp. 4196-4203, doi: 10.1109/ITSC58415.2024.10920179. https://ieeexplore.ieee.org/abstract/document/10920179

Realizing Scenario-Based Verification Tests of Automated Vehicles with an AI-Controlled Surrounding Vehicle in a Practice-Relevant Context*

Felix Beringhoff[1], Joel Greenyer[2], Christian Roesener[1] and Matthias Tichy[3]

***Abstract*— Scenario-based testing is seen as a key for the verification and validation of automated vehicles (AV). In a test scenario, the AV is tested under pre-defined traffic conditions. However, realizing those traffic conditions is challenging due to the autonomously behaving AV. The AV decides on its own, whether it gets into the test scenario conditions at all, e.g., by deciding on which lane it wants to drive or at which velocity it is driving. In order to influence the AV's decision making in a required way to realize a test scenario condition, we implement a novel AI method for controlling a surrounding vehicle (SV) of the AV. The AI-controlled SV (AISV) consists of a reinforcement learning (RL) agent which is trained to, e.g., nudge the AV into changing lanes. In contrast to current common practice, this approach does not require the manual tailoring of triggers and actions for controlling the SV. In this paper, we report on a working design of the RL framework and experiments regarding three different training scopes for the RL agent. We distinguish specialized agents which are trained to reach a single scenario condition and two sorts of generalized agents which shall be capable of reaching a set of scenario conditions. The results show that specialized agents perform the best with a success rate of up to 100 %. However, the generalized agents perform better in realizing scenario conditions which are not known to the agent from training. We also report on an implementation of the approach in a hardware-in-the-loop-simulation (HIL) test bench used in industrial practice and discuss a first try-out.**

I. INTRODUCTION

In the past, much research has been conducted on verification and validation methods for automated vehicles (AV) [3]–[5]. Among that, a multi-pillar approach emerges to be a commonly agreed path to verify the functionality and safety of an AV [6]. Here, the scenario-based approach plays an important role. In a scenario, the AV is exposed to relevant traffic scenarios for testing. Surveys like [7], [8] show the extent of research which has already been done and make the scenario-based approach state of research in testing AV.

Now, it is the responsibility of the test engineers to enable the test execution of test scenarios with complex, industrial grade Automated Driving System (ADS) on practice relevant test benches. Since the AV behaves autonomously and is not remotely controllable during test execution, only the environment of the AV contains the controllable variables in the test setup. The scenario realization challenge can be regarded as a scheduling, control or planning problem for the vehicle guidance of the surrounding vehicles (SV). The specific nature of the problem comes from the hard to predict AV behavior. To complicate matters further, the AV behavior certainly changes between evolving software versions. It is therefore required to find a robust control method for the SV to enable regression tests. This is a new challenge in comparison to tests of simpler driver assistance features where a human test driver or test automation tools can control the ego-vehicle according to the test script [1].

Our vision of a solution to the scenario realization challenge consists of an automatic orchestration of the surrounding traffic just by providing a scenario description. Given that, we investigate whether and how this goal can be achieved by a reinforcement learning (RL) [12] approach to AI-controlled SV (AISV). Arguments in favor of this method lie in the automatic learning of the interaction with the black-box AV, scalability potentials (number of AISVs), and easier to achieve real-time capabilities during deployment than, e.g., with model-predictive-controllers. Our intention is to proof the concept in a relatively simple traffic scenario context with one AISV, but with the use of practice-relevant software tools, i.e. MATLAB/Simulink. This shall allow an easy transfer of a prototype to a practice relevant test bench for AV.

Typically, many tests at OEM-level are conducted on system or full-vehicle level on hardware-in-the-loop-simulation (HIL) test benches [9]. On an end-to-end (from sensors to actuators) HIL test bench as sketched on the right side in Fig. 2, the real Electrical/Electronic (E/E) compound of the vehicle is connected to a simulator. Although HIL tests are conducted in a virtual environment, the presence of real electronic control units (ECU) in the simulation loop requires a realistic treatment of the ego-vehicle and its surroundings. For example, like in the real world, a test drive starts at standstill, the vehicle must be made ready to drive, the Automated Driving System (ADS) needs to locate itself in the environment, and the AD function must be engaged. Unlike non-HIL simulations, for tests with the end-to-end HIL setting, it is not possible to just spawn the ego-vehicle in the initial scene of a test scenario. However, the latter is common practice for model- or software-in-the-loop simulations [10], [11] and motivates the research for new solutions to the scenario realization problem further.

In this paper, we report on a working design and conducted experiments of a concept for the automatic realization of test conditions with an RL-based AI-controlled SV. The purpose

*The results, opinions and conclusions expressed in this publication are not necessarily those of Volkswagen Aktiengesellschaft.

[1]Felix Beringhoff and Christian Roesener are with Volkswagen AG, Wolfsburg, Germany {felix.beringhoff, christian.roesener1}@volkswagen.de

[2]Joel Greenyer is with the FHDW Hannover, Hannover, 30173, Germany joel.greenyer@fhdw.de

[3]Matthias Tichy is with the Institute for Software Engineering and Programming Languages, Ulm University, Germany matthias.tichy@uni-ulm.de

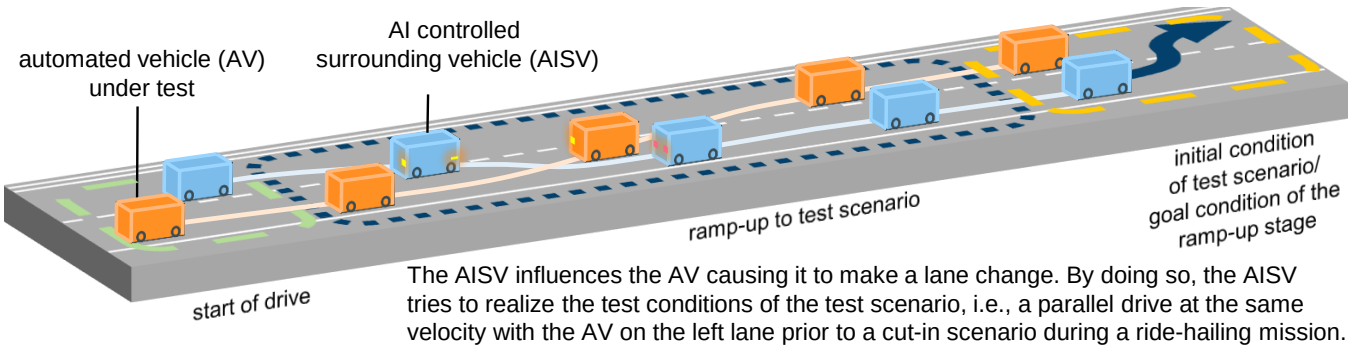

Fig. 1. Concept for the automatic realization of initial conditions of test scenarios through an AI controlled surrounding vehicle, concept based on [1], graphic based on [2].

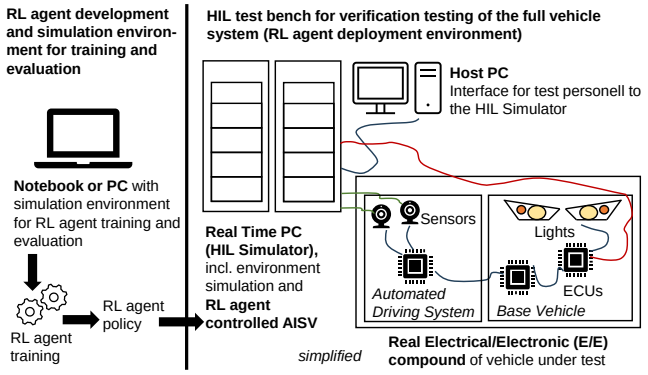

Fig. 2. RL agent development, training, evaluation environment (left) and hardware-in-the-loop-simulation test bench for verification testing (right).

of this study is to analyze whether it is worth to split the scenario realization challenge into several smaller challenges and train a specialized agent for each. Smaller tasks per agent could ease the learning problem and make finding a working RL setup easier, what anyway is a very laborious task. We compare the performance and training duration of:

- *specialized agents*, trained to reach one concrete traffic condition;
- *moderately-generalized agents*, trained to reach a logical set of concrete traffic conditions;
- *super-generalized agents*, which aim for reaching any traffic condition.

The results show, that the specialized agents perform the best in solving their designated learned tasks with success rates between 91 % and 100 %, while the moderately- and super-generalized agents are worse with 78 % to 97 % success rates. However, the generalized agents show indeed generalization capabilities with which the training duration can be reduced.

The contributions of this paper are:

- description of an RL configuration with which an agent can be trained for the scenario realization challenge;
- description and results of controlled experiments regarding the learning task and scope of a single RL agent in the scenario realization challenge.

II. RELATED WORK

The scenario realization problem is not per se new. Tests with advanced driver assistance systems (ADAS) equipped cars in the real world or user studies in driving simulators also bring up the challenge. The related work is split into the parts of the scenario realization problem and the RL concept.

The scenario realization approach commonly is to manually script the test procedures. Schöner et al. describe in [14] two ways of doing so. First, by prerecording humanly driven trajectories and, second, by using graphical tools to design and fine-tune maneuvers or trajectories of vehicles prior to test execution. In the KO-HAF project [15] a tool called test manager is developed for tests of ADAS-driven vehicles in the real world. Here, the human drivers of SV are given a graphical feedback whether they drive according to the test scenario and how they should adapt their driving.

For tests in a driving simulator, the scenario realization problem is long known as seen by the publication of [16] from 1995 where state machines are utilized to create reactive behavior of SV to realize desired situations. In driving simulators, the participants of a study usually should drive naturally and uninfluenced by an instructor. Similarly to the test of an AV, a direct control of the participant respectively the AV is not possible. A more recent approach for driving simulators by Xiong et al. use an automated action planning strategy to plan the behavior of the surrounding traffic [17].

For tests in simulations in general, a test scenario is usually scripted prior to the test execution. Weber et al. distinct in [18] two forms of scenario descriptions: declarative and imperative scenarios. Imperative scenario descriptions explicitly define the actions of the actors, e.g., by defining conditions for triggers and for the action execution. The ASAM OpenSCENARIO XML [19] format mostly complies to this definition. Declarative scenario descriptions formulate conditions of scenario states but do not describe how or with which action they are realized. The ASAM OpenSCENARIO DSL [20] and GeoScenario [21] represent this type of scenario descriptions. Declarative scenario descriptions require intelligent interpreters and algorithms to realize the conditions described in the scenario and transform the formulated conditions from the scenario to movement of the actors in a simulation. Those declarative scenario descriptions therefore

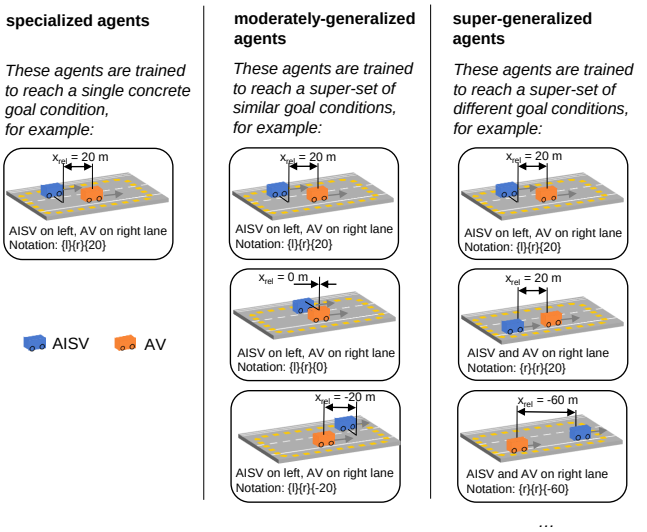

Fig. 3. The specialized, moderately-generalized, and super-generalized agents differ in the composition of conditions they are trained for to reach.

are currently supported by only very few simulation tools with unknown performance to the public.

The second part of related work focuses on RL applications in driving control relevant to our concept. RL can be used as an algorithm for the behavior decision of AV, e.g., for lane change decisions as shown in [22], [23]. More general, an RL design and training framework for behavior decision tasks for automated driving is presented in [24]. Here, high-level actions, e.g., lane change decisions, in combination with low-level controllers for the steering are discussed. In [25], the authors show how RL agents can be used to control vehicles in lane free traffic. The RL agent could learn how it needs to position itself on a road in order to let another vehicle past. Further publications focus on directly finding defects in the AV by changing the environment with the goal of provoking crashes [26]–[28]. In [26], [27] the drivable path of the AV is obstructed by controllable objects.

In comparison to our work, RL approaches from literature focus on the development of driving algorithms or on the identification of relevant test cases, but do not tackle the problem of realizing concrete conditions from test scenarios.

III. CONCEPT

Our concept is to train a surrounding vehicle (SV) of an AV to realize scenario conditions by goal-oriented influencing of the AV. The SV is controlled by an artificial intelligence (AI), trained with RL. The AI-controlled SV (AISV) learns how to influence the AV in order to trigger actions of the AV. If a scenario condition describes that the AV should drive on the left of two lanes, the AISV will learn how to nudge the AV into driving on the desired lane. Fig. 1 shows how the AISV is used to achieve such expemplary test conditions. The intention is to train the agent in a simple simulation environment and then deploy the trained agent on a HIL simulation test-bench of an AV, as shown in Fig. 2.

TABLE I
ACTIONS OF THE RL AGENT

number	longitudinal	lateral
1	keep velocity	keep lane
2	accelerate (1 m/s^2)	keep lane
3	accelerate (4 m/s^2)	keep lane
4	decelerate (-1 m/s^2)	keep lane
5	decelerate (-4 m/s^2)	keep lane
6	keep velocity	lane change left
7	keep velocity	lane change right

In order to abstract the scenario realization problem, we define three subtasks. They should address the constraints from the hard-to-predict and changing AV behavior and the need to realize a variety of conditions in a structured manner:

- reach a goal from a set of different start states;
- reach a goal for a set of different AV behaviors;
- reach a set of different goals.

Next, the modeled action space, observation space, reward function, and environment of the RL agent are described, based on prior systematic design variations.

A. Action space

Table I shows the seven actions the agent can choose from. When performing a longitudinal action, a steering controller keeps the vehicle in the center of the current lane. When changing lanes, the current longitudinal velocity is kept. A lane change can only be performed when there is a lane in the target direction, the velocity is greater than 1 m/s and the vehicle does not already perform a lane change. A pure pursuit controller [29] is used as low-level steering controller in the training environment. A lane change is completed when the vehicle is within 0.25 m of the target lane center.

B. Observation space

The modeled observation space (see Table II) includes continuous signals describing the longitudinal velocity vx and current lane affiliation of AISV $lane_{AISV}$ and AV $lane_{AV}$ (0: right lane, 1: left lane) as well as their relative position x_{rel}, y_{rel}, relative longitudinal velocity vx_{rel} and acceleration ax_{rel}. Furthermore, the observation space contains Boolean-like signals about the current state of a relational grid describing the relative positioning of the AISV and the AV, see Fig. 4. The relational grid distinguishes fourteen states, with only one being true at a time.

Additionally, the agent receives information about the deviation from goal conditions via the observation channels 23 to 32. In the following described experiments, we train the agent to reach goals defined by the lane affiliation of AISV $lane_{AISV,goal}$ and AV $lane_{AV,goal}$, their relative longitudinal positioning $x_{rel,goal}$, and relative longitudinal velocity $vx_{rel,goal}$. The observation space and (later described) reward function have placeholders for more goal parameters.

TABLE II

OBSERVATIONS OF THE RL AGENT

channel	symbol [unit]	range [min, max]
	kinematic states	
1	$vx_{AISV}[m/s]$	[0, 30]
2	$lane_{AISV}[1]$	[0,1]
3	$x_{rel}[m]$	[-150, 150]
4	$vx_{rel}[m/s]$	[-30, 30]
5	$ax_{rel}[m/s^2]$	[-10, 10]
6	$y_{rel}[m]$	[-10, 10]
7	$vx_{AV}[m/s]$	[0, 30]
8	$lane_{AV}[1]$	[0,1]
	relational grid states	
9	state no. 0	[0,1]
10-21	...	[0,1]
22	state no. 13	[0,1]
	deviation from goal	
23	$x_{rel,goal} - x_{rel}[m]$	[-150, 150]
24	(n/a) $y_{rel,goal} - y_{rel}[m]$	[-10, 10]
25	$vx_{rel,goal} - vx_{rel}[m/s]$	[-50, 50]
26	(n/a) $lane_{rel,goal} - lane_{rel}[1]$	[-4, 4]
27	(n/a) $x_{AISV,goal} - x_{AISV}[m]$	[-1000, 1000]
28	$lane_{AISV,goal} - lane_{AISV}[1]$	[-2, 2]
29	(n/a) $vx_{AISV,goal} - vx_{AISV}[m/s]$	[-50, 50]
30	(n/a) $x_{AV,goal} - x_{AV}[m]$	[-1000, 1000]
31	$lane_{AV,goal} - lane_{AV}[1]$	[-2, 2]
32	(n/a) $vx_{AV,goal} - vx_{AV}[m/s]$	[-50, 50]

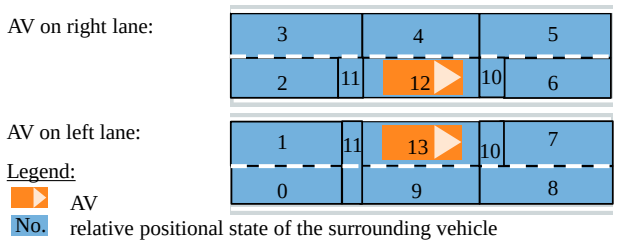

Fig. 4. Relational grid for the observation space of the RL agent, based on [22]. The value of placeholder channels (n/a) is constantly zero.

C. Reward function

The reward function is the result of many prior experiments. This sparse reward function is very simple and only rewards conditions that are easy to describe. This has the advantage that the agent is not guided much by the course of the reward function during training. Instead, the agent has to learn a behavior very freely by random action choices with which it arbitrarily has to reach goal conditions to be rewarded. Additionally, the function punishes a collision of the vehicles. The rewards are given only once when the goal conditions are true for the first time. The reward function $R(s)$ of state s is as follows:

$$R(s) = R_{rel.\ pos}(s) + R_{rel.\ pos\ and\ v}(s) + \\ + R_{rel.\ pos\ and\ abs.\ pos}(s) + R_{all\ goals}(s) + R_{collision}(s)$$

The term $R_{rel.\ pos}(s)$ returns 1/6 of the max. reward when the deviation of the relative positioning of the vehicles and the goal parameter values $x_{rel,goal} - x_{rel}$ and $y_{rel,goal} - y_{rel}$ are within the tolerances (i.e., +-4 m in our experiments) and the vehicles are on the goal lanes $lane_{AISV,goal}$ and $lane_{AV,goal}$.

The term $R_{rel.\ pos\ and\ v}(s)$ returns 1/6 of the max. reward when the relative positioning goals are reached (see $R_{rel.\ pos}(s)$) and the velocity related deviation signals $vx_{rel,goal} - vx_{rel}$ and $vx_{AISV,goal} - vx_{AISV}$ and $vx_{AV,goal} - vx_{AV}$ are within the targeted boundaries (i.e., +-1.1 m/s in our experiments).

Analogously, the term $R_{rel.\ pos\ and\ abs.\ pos}(s)$ returns 1/6 of max. reward when the conditions defined in $R_{rel.\ pos}(s)$ are true and the deviation from absolute position goals $x_{AISV,goal} - x_{AISV}$ and $x_{AV,goal} - x_{AV}$ are within a defined threshold. In the following experiments, the absolute position goals are not defined and therefore the signals are always within their thresholds and the reward is given when $R_{rel.\ pos}(s)$ also returns the reward.

Additionally, 1/2 of max. reward is returned when the conditions defined in $R_{rel.\ pos}(s)$, $R_{rel.\ pos\ and\ v}(s)$, and $R_{rel.\ pos\ and\ abs.\ pos}(s)$ are true at the same time. When a collision between both vehicles is detected, the term $R_{collision}(s)$ returns -10,000 points as punishment.

In total, the agent can get a max. reward of 200,000 points. A training episode is finished when all goal conditions are met, the vehicles collide, after a driven distance of 770 m or when reaching the max. number of defined simulation steps.

D. Training environment and AV model

The RL agent is trained in a time-discrete MATLAB/Simulink 2022a simulation environment with a sample time of 0.1 s. The simulated driving environment consists of a two lane, straight road. The vehicle dynamics are modeled by a kinematic single-track model. Each time step, the agent choses an action which is executed by lower-level driving controller of the AISV. Additionally, in each time step, the AV model reads the relative and absolute values of the distance, lane indices and velocity of AISV and AV, decides for a lateral behavior (stay on lane, change to the left or right) while adjusting its speed to a fixed target velocity.

From the perspective of the RL agent, the AV model is part of the environment. The AV model consists of a very simple rule-based lane change decision model, a pure pursuit steering controller and a simple velocity controller. The AV behavior can be varied by adjusting three AV behavior parameters: target velocity $vx_{AV,target}$, as well as the relative velocity threshold $vx_{rel,lanechange}$ and the relative longitudinal distance threshold $x_{rel,lanechange}$ to trigger a lane change. It should be noted that the AV does not have a following control of a slower moving vehicle in front. The AV switches from the right to the left lane, when a surrounding vehicle is within $x_{rel,lanechange}$ infront of the AV and the SV drives slower than $v_{AV} - v_{rel,lanechange}$. When the AV is on the left lane with no other vehicle in a safe distance on the right lane, the AV switches to the right.

IV. EXPERIMENTS

The experiments take place in the training environment of the RL agent. In the experiments, we train specialized, moderately-generalized and super-generalized agents. The agents differ in their deployment scope, as shown in Fig. 3.

Exp. 1 deals with *specialized agents*. A specialized agent has the task of reaching a single goal. Exp. 2 deals with *moderately-generalized agents* and one *super-generalized agent*. A moderately-generalized agent has the task of reaching a super set of goals which contains more than one goal from the same logical group of goals. In our definition, goals from the same logical group differ only in the values of one goal parameter, i.e., $x_{rel,goal}$. A super-generalized agent has the task to reach goals from different logical groups of goals.

For the evaluation of the agents generalization capabilities, the trained agents are tested in the same simulation environment but with previously unseen goal conditions.

Procedure of the experiments: During the experiments, the agents are trained and afterwards tested regarding their success in the trained tasks. The procedure of both experiments is as follows:

- train three agents for the same set of goals with different randomization seeds
- measure the amount of training episodes that is needed until a reward threshold is reached

→ average and aggregated training duration of the three agents is shown in Fig. 6

- deploy the agents with a deterministic policy on each concrete task from their training and measure how often an agent reaches the goal and calculate the $successrate = |reachedgoals|/|tasks|$

→ results of the best of three agents is shown in Fig. 6

A training session consists of several episodes, each including a drive from a start state at stand still until a stop is triggered. Between the end and start of a new episode, a reset function sets the vehicles back to a start state. Within the reset function, a randomization algorithm randomly combines new start state, AV behavior, and goal parameters from the given sets. Although the task seems more difficult, results from prior experiments show that varying start states enhances the speed of learning. Additionally, we vary the AV behavior during the training to make the agent robust against AV behavior changes. For exp. 1, the size of the set of goals per agent is one, while it is greater than one for exp. 2.

The sets of start states and AV behaviors is equal for exp. 1 and 2. The set of start states is (values refer to SI units):

$$
\begin{aligned}
startstates = \\
&\{(lane_{AISV,0}, lane_{AV,0}, x_{rel,0})| \\
&lane_{AISV,0} \in \{rightlane, leftlane\}, \\
&lane_{AV,0} \in \{rightlane, leftlane\}, \\
&x_{rel,0} \in \{-100, -50, -25, -10, 10, 25, 50, 100\}\}
\end{aligned}
$$

The set of AV behaviors is (values refer to SI units):

$$
\begin{aligned}
AVbehaviors = \\
&\{(vx_{AV,target}, x_{rel,lanechange}, vx_{rel,lanechange})| \\
&vx_{AV,target} \in \{4, 6, 8, 10\}, \\
&x_{rel,lanechange} \in \{-35, -45, -55\}, \\
&vx_{rel,lanechange} \in \{0, -2, -3.9\}\}
\end{aligned}
$$

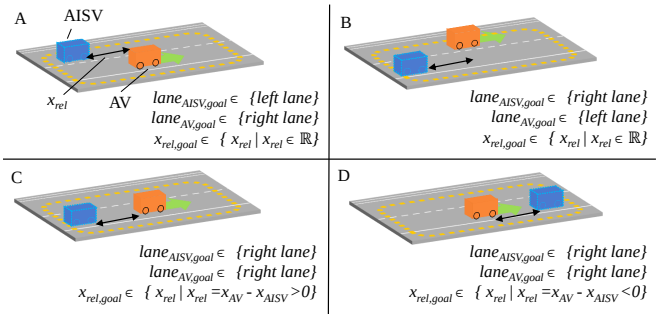

Fig. 5. logical groups of goals

A negative relative velocity means that the AISV travels faster than the AV and a negative relative distance means that the AISV is in front of the AV.

In analogy to the start states, the goal conditions describe the targeted relative longitudinal distance between both vehicles and their targeted lanes. The set of goals is:

$$goals = \{(lane_{AISV,goal}, lane_{AV,goal}, x_{rel,goal})\}$$

The combination of start state, AV behavior and goal define a concrete task for the training session. The super set $tasks$ is as follows:

$$tasks = startstates \times AVbehaviors \times goals$$

RL agent settings: For the design and training of the RL agent, we use the MATLAB/Simulink 2022a reinforcement learning toolbox. Initially, we experimented with PPO and DQN algorithms, but achieved the first successes with a double DQN algorithm and then further fine tuned its parameters. The agents critic consists of the default network from the toolbox with two hidden layers and 256 neurons. The structure of the neural network was not yet the main focus of the parameter variation. The other parameters were identified through systematic variations and showed a robust learning success for varying random number generator seeds. The experience buffer is set to 500000, the minibatch size is 32, the learning rate is 0.001. An epsilon of 1 with a decay of 0.00003 and a minimum of 0.01 is used along with an n-step lookahead of 16. In the experiments, we conduct three training sessions per agent with a different randomization seed for the RL framework. The trainings are conducted on a consumer notebook or PC and are set to run on the CPU. The training of 1000 episodes takes approx. 85 minutes.

In exp. 1, each agent is trained for a maximum of 10000 episodes with max. 700 time steps before aborting. In exp. 2, the max. allowed amount of time steps is 2000. A training is prematurely finished when the reward threshold of 195000 points in average over the last 200 episodes is reached. If the training does not finish within 10000 episodes, the agent at the time of the highest average reward is saved and used for further evaluation.

A. Experiment 1) training of specialized agents

In exp. 1, we train agents to reach a single concrete goal. We obtain three (three randomization seeds) times twelve

Training conditions		Training results		
logical goal group	**agent** ag(exp.).{$lane_{AISV,goal}$}{$lane_{AV,goal}$}{$x_{rel,goal}$}	**best success rate in trained tasks**	**average learning duration [episodes]**	**aggregated learning duration [episodes]**
specialized agents				
A	ag1.{l}{r}{-20}	98.44%	3226.67	9680
	ag1.{l}{r}{0}	92.97%	3405.33	10216
	ag1.{l}{r}{20}	94.62%	988.00	2964
	average of group	95.34%	2540.00	**22860**
B	ag1.{r}{l}{-20}	96.88%	769.00	2307
	ag1.{r}{l}{0}	100.00%	997.67	2993
	ag1.{r}{l}{20}	97.66%	1843.00	5529
	average of group	98.18%	1203.22	**10829**
C	ag1.{r}{r}{20}	99.74%	927.33	2782
	ag1.{r}{r}{40}	91.41%	1078.00	3234
	ag1.{r}{r}{60}	95.31%	2104.67	6314
	average of group	95.49%	1370.00	**12330**
D	ag1.{r}{r}{-100}	97.66%	703.33	2110
	ag1.{r}{r}{-60}	97.57%	3897.67	11693
	ag1.{r}{r}{-80}	97.66%	3859.67	11579
	average of group	97.63%	2820.22	**25382**
moderately-generalized agents				
A	ag2.{l}{r}{-20;0;20}	82.29%	4115.33	**12346**
B	ag2.{r}{l}{-20;-5;0;5;20}	97.03%	4934.00	**14802**
C	ag2.{r}{r}{20;40;60}	88.08%	1937.67	**5813**
D	ag2.{r}{r}{-60;-80;-100}	94.88%	1272.67	**3818**
super-generalized agent				
mixed	ag2.{l,r}{r}{-60;20;40}	78.04%	4746.67	**14240**

Fig. 6. Results of experiment 1 and 2. The agents naming notation include information about the goal conditions they are trained with.

(twelve goal conditions) agents, each specialized to realize one concrete goal. From those twelve concrete goals, three each belong to one logical group A, B, C, or D, see Fig. 5. With the varying 36 start positions and 32 AV behaviors each agent is trained in 1152 different concrete tasks.

Fig. 6 shows the performance of the best specialized agents per goal when deployed in their 1152 concrete tasks from training and the average and aggregated learning duration of the three agents (three randomization seeds). The average success rate over all specialized agents is 78 % and 96 % success rate in average for the best performing agents per task. Not every trained agent which reached the reward threshold in training performs well in deployment. However, by training three agents per task with different random generator seeds, an agent with a success rate higher than 91 % is obtained for every task. Agent ag1.{r}{l}{0} reaches the goal in all 1152 concrete tasks while ag1.{r}{r}{20} misses in three tasks.

The training duration varies strongly between agents and used randomization seed. The agent with highest success rate does not necessarily take longer in training than lower performing agents. Some agents reach the reward threshold in less than 1152 episodes which means that these agents could not see every concrete tasks during training.

B. Experiment 2) training of moderately- and super-generalized agents

In exp. 2, we train agents to reach a set of concrete goal conditions which has three to six elements. Agents ag2.{l}{r}{-20,0,20}, ag2.{r}{r}{20,40,60}, ag2.{r}{r}{-60,-80-100} are trained with three randomly varied goals, making it 3456 different concrete learning tasks. Agent ag2.{r}{l}{-20,-5,0,5,20} has five different goals and is trained in 5760 different concrete tasks while ag2.{l,r}{r}{-60,20,40} sees six different goals and 6912 different concrete learning tasks.

Fig. 6 shows the performance of the moderately- and super-generalized agents per goal-set. A comparison with the results from exp. 1 suggest that the training of specialized agents leads to a higher success rate in reaching the goals. However, in total the aggregated training duration of one moderately-generalized agent with three randomization seeds for logical groups A, C, and D is less than training three times three specialized agents. The performance of the super-generalized agent is on the lower end of the moderately-generalized agents while the average training duration is on par with the longest average duration of the moderately-generalized agents.

C. Evaluation

To evaluate the performance, the agents are tested in tasks they have not seen before during training. Similar to the training set, the test set consists of freely selected but different test conditions. The set of test tasks include two known start states (r,r,-25),(r,r,25) and one known AV behavior (8,-45,-2) together with 14 new goals per logical group A, B, C, D and 28 new goals for the mixed group. This results in 28, respectively 56 different test tasks for each agent. The tests are conducted with the best agents from the experiments.

Fig. 7 shows the goal parameter values of the tests together with the results. The general performance level is, with exception of ag2.{r}{l}{-20,-5,0,5,20}, below the success rate in the learned tasks. In Test A and B, the moderately-generalized agents reach a higher success rate than the specialized agents, which meets the expectations. Agent ag2.{r}{l}{-20,-5,0,5,20} even reaches 100 % success rate which is higher than in the learned task. This might be the result of the omission of several start states and AV behaviors. Surprisingly, two specialized agents outperform the moderately-generalized agents in Test C and D.

The performance of the super-generalized agent is compared to three specialized agents. Here, the super-generalized agent shows a success rate of 79 % and the specialized agents of less than 25 %. Keeping in mind that the Test E includes goals from different logical groups, the results shows the effect of learning with varying goals through the adaptability capability of the super-generalized agent in comparison to the failing specialized agents.

V. DISCUSSION

A. Discussion of the experiment results

The findings of this study show that the specialized agents perform with the highest success rate in their learned tasks while the moderately-generalized agents are capable of reaching sets of different goals with less aggregated learning effort than needed for the training of several specialized

Training conditions		Test conditions and success rates				
logical goal group	agent ag(exp.).{lane$_{AISV,goal}$}{lane$_{AV,goal}$}{x$_{rel,goal}$}	Test A	Test B	Test C	Test D	Test mixed
		{l}×{r}×{-26:4:26}	{r}×{l}×{-26:4:26}	{r}×{r}×{8:5:73}	{r}×{r}×{-113:5:-48}	{l,r}×{r}×{-82:5:-52, 18:5:48}
specialized agents						
A	ag1.{l}{r}{-20}	21.43%				
	ag1.{l}{r}{0}	35.71%				
	ag1.{l}{r}{20}	25.00%				
B	ag1.{r}{l}{-20}		28.57%			
	ag1.{r}{l}{0}		28.57%			
	ag1.{r}{l}{20}		28.57%			
C	ag1.{r}{r}{20}			67.86%		
	ag1.{r}{r}{40}			78.57%		
	ag1.{r}{r}{60}			92.86%		
D	ag1.{r}{r}{-60}				89.29%	
	ag1.{r}{r}{-80}				89.29%	
	ag1.{r}{r}{-100}				64.29%	
mixed	ag1.{l}{r}{0}					5.36%
	ag1.{r}{r}{60}					25.00%
	ag1.{r}{r}{-80}					25.00%
moderately-generalized agents						
A	ag2.{l}{r}{-20,0,20}	53.57%				
B	ag2.{r}{l}{-20,-5,0,5,20}		100%			
C	ag2.{r}{r}{20,40,60}			67.86%		
D	ag2.{r}{r}{-60,-80,-100}				78.57%	
super-generalized agent						
mixed	ag2.{l,r}{r}{-60,20,40}					78.57%

Fig. 7. Success rates of actually reaching the goals. The test conditions are not known to the agents from the training.

agents. However, a specialized agent can easily trained and added if new test conditions become relevant in practical use.

The super-generalized agent was trained with the highest variety in the set of goals and shows the worst success rate in its learned tasks. This leads to the early assumption that the scope of an agent shall be limited for higher performance. Possibly the generalized agents do not interpret the observed deviation-from-goal signals right. Here, further research is needed, e.g., regarding the critic network since the used default network could be the limiting factor and has not yet been touched. However, the results of the tests with a mixed test condition set gives confidence about the general idea of generalized agents.

The experiments show a high difference in performance of the agents trained with different randomization seeds. This underlines the influence of luck during training and tuning of the RL framework. Potentially good modeling concepts can be concealed by a bad randomization seed and sorted out too quickly during development. Further fine-tuning of the learning setting could increase the overall performance of the agents, however, this is a time consuming task as experienced during the realization of the concept.

Threats to validity: Threats to the internal validity of the experiments are caused by training only the super-generalized agent with the goals $\{l\} \times \{r\} \times \{-60, 40\}$. It cannot be ruled out that the lower performance of the agent is due to challenges with these concrete goals and is not caused by the larger scope of goals. This is due to finding a set of goals $\{lane_{AISV,goal}\} \times \{lane_{AV,goal}\} \times \{x_{rel,goal}\}$ which does not include invalid goals (e.g., (r,r,0) describes a crash, or (l,l,40) violates the obligation to drive on the right), but contain goals from different logical groups. Additionally, the moderately-generalized agent of goal group B contains two additional goals than there are specialized agents for group B. Both aspects are a weakness in the experiment design.

Furthermore, regarding the zero-shot performance of the agents in the test runs, the obtained results could be distorted by the simplicity of the traffic scenario. The specialized agents of group C and D perform well in the test of unknown tasks, because they found a general solution which arbitrarily is capable of reaching similar goals. For example, in Test D an agent drives 100 m in front of the AV and 1 m/s slower. The relative velocity is within the tolerances of the goal condition. Therefore, the AISV gets closer to the AV and reaches several goals of goal class D. The success of the specialized agents in the evaluation tasks is only possible due to the simplicity of the scenario. However, the average success rate of the specialized agents in the mixed test is clearly lower than the super-generalized agent, because the beforementioned strategy can not work for realizing goals from different logical groups.

Regarding the external validity and practice relevance of the experiments, the influence of the very simple AV model on the performance is unknown. The level of performance of the trained agent might change completely when a different AV model is used. This is an important aspect since the RL agent is trained in a simple simulation framework but should be deployed on an industrial test bench with a real ADS in the loop. Therefore, the validation of the used AV model for training should be future work.

B. Discussion of the deployment of the method in practice

The intended deployment environment of the RL agent is an end-to-end hardware-in-the-loop-simulation test bench with almost the full Electrical/Electronic (E/E) vehicle compound connected as real hardware to the simulator, as sketched in Fig. 2. The E/E compound of the vehicle-under-test consists of, e.g., ECUs, sensors with injection interfaces of the ADS, actors (e.g., steering rack, head lights), and HMI (e.g., steering wheel, pedals). The real time PC (HIL simulator) simulates the ego-vehicle dynamics and inertia sensor data (e.g., speed, acceleration, rotation), a virtual environment including road and traffic (this is where the AISV is located), ego-vehicle environment sensor data (e.g., camera video stream, object lists for radar), and restbus simulation of missing components (e.g., battery and motors).

For a first prototypical try-out, we transferred a trained AISV to a HIL-simulation test bench. We prototypical integrate the trained policy of an RL agent of logical goal group B in a dSPACE Automotive Simulation Models Traffic (ASM Traffic) simulation model. Adapter functions ensure that the AISV receives the same observation signals as in the training environment. Adapter functions for the action space include low-level controller which implement the high-level action choice of the agent into vehicle movement. Engineering of those adapter functions can widen the area of use of the agent. Here, for example, we use the ASM internal lane-following, lane-change and velocity controller

and hand-over the target velocity and desired lane index of the agent. This also allows the agent to function on curvy roads, although the agent was trained on a straight road only.

To get a first impression of the use case of the RL agent, we use the AISV in the surrounding of a human controlled vehicle (instead of an AV) on an endless straight road. The AISV showed interesting human-like behavior when trying to nudge the human driven vehicle into a lane change. That shows another potential use-case of the AISV in driver-in-the-loop simulators. Anyhow, limits of the AISV become clear, when the vehicle-under-test drives at much higher speeds than the AV model did during training. However, all in all the prototype leaves a positive subjective impression on the involved test engineers for the use of this method for selected test cases.

VI. CONCLUSIONS

Overall, this study is a starting point for further research on RL approaches to the scenario realization problem. In comparison to action-trigger-based approaches like those used in OpenSCENARIO XML implementations, the RL approach does not require the explicit definition of triggers and actions for the SV. Since proprietary scenario formats and OpenSCENARIO are widely used and sufficient for most use cases, the RL approach can act as an user-defined external controller used in symbiosis with the scenario scripts. The RL approach helps in acts of the scenario script, in which an interaction between an SV and the autonomously acting AV should lead to actions of the AV.

First inhouse tests of the deployment of a simple prototype to a HIL test bench showed promising results worth further research. With the current AISV model, it is recommended to use specialized-agents, as these show a higher success rate in realizing the desired scenario conditions. However, with the generalized-agents can a real benefit be derived from the AI approach, as a new agent does not have to be trained for each new scenario condition. Future research should focus on enhancing the generalized-agents performance, adapting the approach to more complex scenarios on realistic road networks, and on the AV model used during training.

REFERENCES

[1] F. Beringhoff, J. Greenyer, C. Roesener, and M. Tichy, "Thirty-one challenges in testing automated vehicles: Interviews with experts from industry and research," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 360–366.

[2] F. Beringhoff and C. Roesener, "A method for automatic test realization for automated vehicles," in *Commercial Vehicles 2023*, ser. VDI-Berichte, vol. 2417, 2023, pp. 179–190.

[3] M. Wood, *et al.*, "Safety first for automated driving," 2019, white paper of different companies.

[4] PEGASUS, "Schlussbericht für das gesamtprojekt," 2020. [Online]. Available: www.pegasusprojekt.de

[5] Y. Cao, *et al.*, "Code of practice for the development of automated driving functions," 2022. [Online]. Available: www.L3Pilot.eu

[6] UNECE, GRVA and IWG on VMAD, "New assessment/test method for automated driving," 2022, wP.29, 187th session.

[7] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[8] D. Nalic, T. Mihalj, M. Baeumler, M. Lehmann, A. Eichberger, and S. Bernsteiner, "Scenario based testing of automated driving systems: A literature survey," 2020.

[9] K. Juhnke, M. Tichy, and F. Houdek, "Challenges concerning test case specifications in automotive software testing: assessment of frequency and criticality," *Software Quality Journal*, vol. 29, pp. 39–100, 2021.

[10] A. Tenbrock, A. König, T. Keutgens, and H. Weber, "The conscend dataset: Concrete scenarios from the highd dataset according to alks regulation unece r157 in openx," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 174–181.

[11] 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, p. 72, 2022.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] F. Beringhoff, "Towards realizing test conditions for automated vehicles," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2022, pp. 152–157.

[14] H.-P. Schöner and W. Hurich, "Testing with coordinated automated vehicles," in *Handbook of Driver Assistance Systems*, H. Winner, S. Hakuli, F. Lotz, and C. Singer, Eds. Cham: Springer International Publishing, 2016, pp. 261–276.

[15] Institut für Fahrzeugtechnik, TU Braunschweig, "Testmanager - a tool for reproducible test execution," 2021. [Online]. Available: www.ko-haf.de

[16] J. Cremer, J. Kearney, and Y. Papelis, "Hcsm: A framework for behavior and scenario control in virtual environments," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 5, no. 3, pp. 242–267, 1995.

[17] Z. Xiong and J. Olstam, "Orchestration of driving simulator scenarios based on dynamic actor preparation and automated action planning," *Transportation Research Part C: Emerging Technologies*, vol. 56, pp. 120–131, 2015.

[18] H. Weber, L. Eckstein, A. Tenbrock, A. König, J. Bock, and A. Zlocki, "Evidenzorientierte ableitung von sicherheitsrelevanten grundszenarien für die fahrdomäne bundesautobahn," *Berichte der Bundesanstalt für Straßenwesen. Unterreihe Fahrzeugtechnik*, no. 149, 2022.

[19] ASAM e.V., "Asam openscenario xml." [Online]. Available: https://www.asam.net/standards/detail/openscenario-xml/

[20] ASAM e.V., "Asam openscenario dsl." [Online]. Available: https://www.asam.net/standards/detail/openscenario-dsl/

[21] R. Queiroz, T. Berger, and K. Czarnecki, "Geoscenario: An open dsl for autonomous driving scenario representation," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 287–294.

[22] P. Wolf, K. Kurzer, T. Wingert, F. Kuhnt, and J. M. Zollner, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 993–1000.

[23] M. Mukadam, A. Cosgun, A. Nakhaei, and K. Fujimura, "Tactical decision making for lane changing with deep reinforcement learning," *NIPS Workshop on Machine Learning for Intelligent Transportation Systems*, 2017.

[24] X. Wang, H. Krasowski, and M. Althoff, "Commonroad-rl: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 466–472.

[25] A. Karalakou, D. Troullinos, G. Chalkiadakis, and M. Papageorgiou, "Deep reinforcement learning reward function design for autonomous driving in lane-free traffic," *Systems*, vol. 11, no. 3, 2023.

[26] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1326–1333.

[27] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2352–2358.

[28] C. Lu, *et al.*, "Learning configurations of operating environment of autonomous vehicles to maximize their collisions," *IEEE Transactions on Software Engineering*, vol. 49, no. 1, pp. 384–402, 2022.

[29] R. C. Coulter *et al.*, "Implementation of the pure pursuit path tracking algorithm," 1992, technical report of The Robotics Institute, Carnegie Mellon University.