Small Scale, Big Impact: Experiences from a Miniature ViL Testbed and Digital Twin Development

Elias Modrakowski(✉), Niklas Rahenbrock, Eike Möhlmann, and Henning Schlender

Institute of Systems Engineering for Future Mobility, German Aerospace Center (DLR e.V.), Oldenburg, Germany
Elias.Modrakowski@dlr.de

Abstract. The concept of Digital Twin (DT) has gained enormous momentum over the past years in many fields with a variety of purposes. We investigated the usage of DTs for the development and testing of automated driving functions. In this context, we wanted to train an agent to challenge the automated driving function of a vehicle via reinforcement learning (RL). For this, we build both, a miniature Vehicle in a loop (ViL) testbed and its digital shadow. The idea is to use the digital shadow as the training environment for the agent resulting in reduced cost and time for training. We decided specifically to build a miniature version of a testbed to accelerate development, reduce resource consumption and increase adaptability.

This paper contributed to the engineering of DTs by reporting our approach regarding the development of a digital shadow of a miniature ViL testbed and the lessons learned. First, we motivate the decision for a miniature testbed. Secondly, we describe the high-level architecture and the technical implementation including both the digital shadow and its physical counterpart. Third, we describe the application of the DT for RL and the experiments enabled by our setup. We conclude with the lessons learned. The main takeaway is that DTs are an excellent means to develop, disseminate and present new methods for the validation of automated vehicles. Here the benefits outweigh the effort of DT construction.

Keywords: Digital Twin · Digital Twin Development · Experience report · Miniature testbed · Vehicle-in-the-loop

1 Introduction

In the last decade, the topic of Digital Twin (DT) has been attracted by many fields [15,19]. DTs are a digital replica of a physical asset (the so-called physical twin (PT)) [13,15] or even the link between the digital and physical world [44]. They are regarded as a solution to reduce both, the time-to-revenue as well as

T. Margaria and B. Steffen (Eds.): ISoLA 2024, LNCS 15223, pp. 83–106, 2025.
https://doi.org/10.1007/978-3-031-75390-9_6

the costs for development, deployment, and operation [29]. This is because the impacts of changes on products can be analysed without bringing them into the actual operation. Even better, different changes can be analysed in a parallel and automated way (see "Experimental Digital Twin" in [37]). This allows us to find the best configuration efficiently. As the approach can also handle variants of products a configuration can even be customer-specific and consider individual preferences, requirements, and country-specific regulations.

Cyber-physical systems (CPSs) are often complex and costly to operate. On one hand, this makes automated optimization of their configuration expensive and challenging. On the other hand, manual optimization is typically slow and error-prone. Examples are electron microscopes or x-ray machines in hospitals, assembly lines in factories, transport systems, and automotive testbeds in research or certification facilities. In this paper, we focus on the operation of such automotive testbeds for the certification of automated vehicles. The approach to increase the efficiency of the testbed operation is to combine DT and reinforcement learning (RL).

In essence, we build a testbed and its digital shadow (DS), combining both with an RL agent to tackle the following research question:

How to build a digital shadow applicable to train, understand and evaluate RL for an agent operating the physical system?

In this paper, the targeted application of the agent is to operate an automotive testbed in a way that challenges an automated vehicle. This challenger will operate a testbed – the physical system – for automated vehicles but is trained on the DS of the testbed as shown in Fig. 1. The physical twin consists of the real system and the agent as (part of) its control software. Note that we apply the concept of various integration levels as proposed by [1], where different stages exist, progressing from a basic digital model to a comprehensive DT. In consequence, the software system developed and described in this paper is primarily the DS of the testbed with direct physical-to-digital twinning. A full feedback (digital-to-physical twinning) is given by the RL agent's addition by transferring its configuration such as weights (see Fig. 1).

With this paper, we share our experience in creating a physical test setup and its DS: how an adequate subject was identified, how it was implemented and synchronized, how it was applied to RL-based optimization for automated vehicle testing and what we learned along the way. In particular, we describe the decision process for selecting the right system for the respective use case in Sect. 2 and the high-level functionalities and identify the DS in Sect. 4. A literature review on comparable setups can be found in Sect. 3. The PT and DT development and technical implementation are described in Sect. 5 and Sect. 6, respectively. Finally, we give a small overview of the applications enabled by this setup in Sect. 7, describe the performed experiments in Sect. 8 and discuss the learnings in Sect. 9.

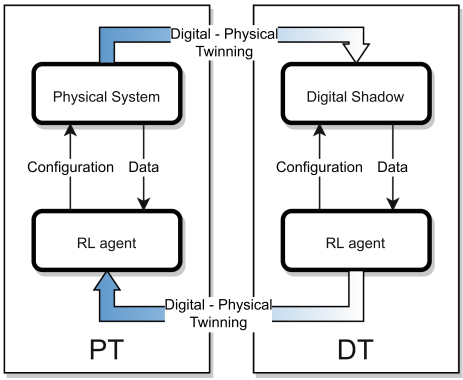


Fig. 1. Simplified twinning interaction of the DT and PT.

2 Requirements and System Selection

To create a DS, a physical system is needed. We call the combination of the DS and physical system a *setup*. In this section, we describe how we selected the setup for our application. First, the purpose that this setup (i.e., the physical system and its DS) shall provide is considered. Our target application is DT-based RL. In particular, we are interested in investigating methods for DT-based RL. For this, we need a setup that allows us to understand, test and evaluate RL methods. To obtain a suitable setup, we (I) followed the development process of DTs including the identification of relevant parameters, (II) developed a DT architecture in the RL training context and (III) applied methods of DT validation. Our desired setup had to satisfy the following minimal requirements.

1. **Adaptability** The setup must be highly configurable and expandable w.r.t. functionality to enable easy prototyping and adaptability to changing researchers' needs.
2. **Explainability** The system must be non-intractable i.e., it must be decomposable into clearly distinguishable functionalities. These need to be easily understandable as well as the interaction between them. The goal is to minimize the time for others to grasp the concept.
3. **Relevancy** The setup must provide a behaviour that is representative of the use case i.e., to a real testbed. Otherwise, the transfer and alignment of derived methods to the use case are hindered.
4. **Accessibility** The used software and hardware to implement the setup shall be off-the-shelf, freely available and low in complexity to enable faster development and a shallow learning curve for new users.
5. **Efficiency** The setup must be resource efficient. This means the construction and running costs in terms of time, personnel and finances.
6. **Reproducibility** For easier testing (e.g., generating data for validation), tests must be reproducible.

There are multiple options for selecting a suitable reference system as the physical system (see Fig. 2) to satisfy the requirements. Instead of replicating a vehicle on a street or a proving ground which would be out of scope, we were inspired by vehicle-in-the-loop (ViL) testbeds (see [4]). ViL testbeds possess the advantage of having a simulated environment and scenario play-out which enables repeatability of tests. They consist of a full-scale vehicle whose sensors are stimulated using inputs from a virtual environment. The test vehicle's output on steering and acceleration are measured at the wheels and fed back into the simulation. This makes it possible to run virtual scenarios with real hardware. Testbeds already play a vital part in testing and validation of (highly-)automated vehicles in the industry [28,32]. Here, the application of the DT concept comes naturally. That is, instead of having a real, full-scale vehicle and testbed which are resource-intensive and slow (only testing in real-time is possible), a DT could be created. Using this DT allows faster software and hardware configuration testing in earlier development stages as well as using multiple DTs in parallel. It would close the gap between Software-in-the-loop and Hardware-in-the-loop or ViL testing.

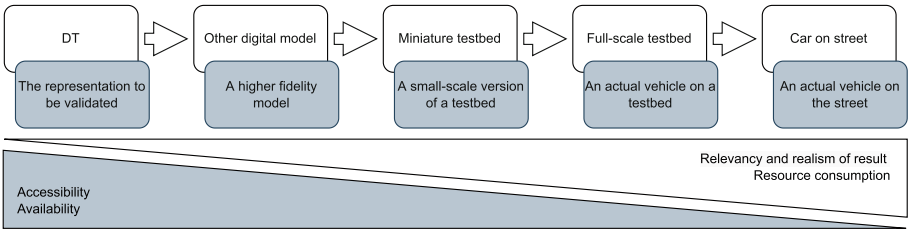


Fig. 2. Conceptual diagram showcasing the order of increasing realism for a vehicle representation. The DT (left) needs to be validated against a representation that has a higher degree of realism (further right).

However, for the creation and especially validation of a DT, access and a detailed understanding of the reference system is needed. Having a full-scale system (see two elements on the right in Fig. 2) as the reference presents multiple problems. First and foremost, due to limited resources like space, budget, and expertise, and also for simplicity and practicality, the usage of a full-scale vehicle in any environment is not viable in our case. In addition, the assessment of whether the virtual environment and scenario in SBT can be seen as valid is still an open research topic [2,26,39] and is out of scope.

One alternative solution to fulfil these requirements for testing e.g., methods for validation of a DT w.r.t. another model is the manufactured universe approach as presented in [41]. The manufactured universe approach argues that validation methods (and extending to development methods as well) can be evaluated even though a real system does not exist but another model serves as the "ground-truth". The applicability of the validation method can then be verified

but no conclusion can be made on the validity of the DT. For the development process, however, this approach is abstract and requires building another model of the system. Due to assumptions that are made during the development of the two models, it is easy to commit systematic errors in terms of omitting phenomena that a researcher might not be familiar with but are obvious to system experts.

Thus, we decided to use a miniature version of a testbed as the physical system because it avoids many of the mentioned drawbacks. One does not face the construction and running cost and effort w.r.t. handling and adaption to changing requirements that the full-scale system may require. However, it gives enough realism to test and evaluate methods related to DT research even though the proof-of-concept DS is not valid in relation to the full-scale vehicle. The secondary benefit of selecting a testbed is that most of the components from the real system are already in digital form, i.e. software, such as the environment simulation and the automated driving function which can be directly reused as part of the DS. This, greatly reduces development time and validation is not required. Furthermore, as it will be later described in detail (see Sect. 4), the actual DS modelling will concentrate on modelling the perception of an automated vehicle. In the context of highly automated driving, perception (especially testing of perception) is a very challenging topic and currently highly relevant for research. Therefore, the third benefit is that our DS may be used for testing methods for validating the perception chain (e.g., RGB camera) in the future. Using the same environmental simulation, scenarios, driving function and vehicle dynamics in both systems, improves controllability and tractability.

3 Related Work

A DT is characterized as a virtual replica of a system created through a combination of models and data [44]. The fundamental purpose of a DT is to mirror the behaviour of a real system by *frequently*[1] adapting (i.e., synchronizing) to real-time conditions, distinguishing it from a mere digital model [3,13,40,44]. DT research is predominantly application-focused in a wide range of fields. For example, a literature review on DT architectures [12] found 86 % of scientific contributions to be solution proposals i.e., studies that are domain-specific solutions to the realisation of DTs. Further, [12] states that 50 % of the publications are validated on simplified use cases or prototypes i.e., full-scale versions but reduced in scope. For evaluating DT methodologies, the authors are only aware of two instances of using scaled-down implementations of a given use case. First, Fraunhofer IESE constructed a setup using a miniature version of a factory to showcase the Basyx-Middleware[2]. Secondly, [16] validated their predictive DT approach using a scaled-down aircraft.

[1] What is considered as "frequent" is however not specified in the literature (see e.g., [15]).

[2] See https://eclipse.dev/basyx/about/.

As for every software system, during and after the development of highly automated driving functions for road vehicles, testing is necessary. There are several approaches available to perform this. As testing environments, [32] distinguishes between Model-, Software-, Hardware- and Vehicle-in-the-Loop as well as proving ground testing and public road testing with increasing integration of software and hardware components. While currently, validation and homologation are done on public roads and proving grounds, with higher levels of vehicle automation, there is a consensus that these tasks can only be achieved meaningfully by pushing it more towards simulation-based testing [32]. A main focus in this field is the validation of such testing environments [9,11,32,33,43]. For a further push towards meaningful, high-fidelity modelling of the virtual components in Software-in-the-Loop (SiL), researchers have provided a number of modelling methodologies [36] and their exemplary implementation tested in proving grounds [31], or with real-world images [7]. In the automotive context, miniature ViL testbeds for evaluating methodologies are uncommon and to the authors best knowledge there are no comparable setups. Focussing just on a sub-system, the perception for highly automated vehicles, [38] created a camera stimulation setup as a testbed also including a camera in front of a monitor as described in more detail in Sect. 4. However, it is limited to perception without a full-feedback loop and is centred on camera model validation. Furthermore, the potential of integrating DTs into vehicle testing has already been investigated by scholars such as [21,34,47]. They argue the benefits of tailor-made virtual testing, and off-loading to the simulation. However, these works only offer a high-level, limited exploration of such a system. In general, the automotive testing domain is not an active field for DT research yet which is more focussed on manufacturing [19].

A review of the related work highlights that using a miniature evaluation system for methods related to DTs and/or automotive testing has not yet received much attention. This is despite the potential the concept offers. Thus, we want to share our experience and achieved benefits.

4 Concept of the Setup's Functional Architecture

In general, the testbed shall work as follows, illustrated in Fig. 3. The vehicle's perception is solely camera-based. Within the virtual environment a camera picture is generated. While it has a position and focal length defined, its image can be seen as "perfect" by not having any deficiencies. It is then shown on a monitor to which the camera is directed. The camera captures an image representing the displayed scene but also naturally includes the impurities of the camera. The image is then processed by the driving function namely a lane-keeping assistant (LKA). Its output is a desired goal steering angle which is then fed back as control input to the virtual vehicle representation within the virtual environment. Both the driving function and the virtual environment of the testbed are software which can be directly reused for the DS (see left side in Figre 3). Hence, the interaction between the screen and the camera needs to be modelled and

parametrized or "twinned" to fit the physical reference (see Fig. 3). Thus, the DS is composed of a virtual environment, camera model, and driving function and can be defined as a parametrizable model of the testbed which can play out scenarios resulting in similar behaviour and results as the actual testbed. Note that, while it would be possible to also create a DS of the vehicle dynamics which, instead of taking desired speed and steering angle, would use a measured actual speed and angle, we decided to focus on screen and camera because of the following reasons.

1. An additional parametrizable model would add complexity to the twinning process as vehicle dynamic models need test scenarios (e.g., for performing step functions) to calculate the parameters of the underlying physics.
2. Realistic force feedback to emulate road conditions on the model car's scale is infeasible due to its complexity and therefore out of scope.
3. For the proof of concept, one DS of one component is sufficient to evaluate and prove our methods.
4. Choosing between a DS of the vehicle dynamics, and a DS of the screen with the camera, the latter is easier to understand by humans and visualizable. This also simplifies presenting the results to an external audience.

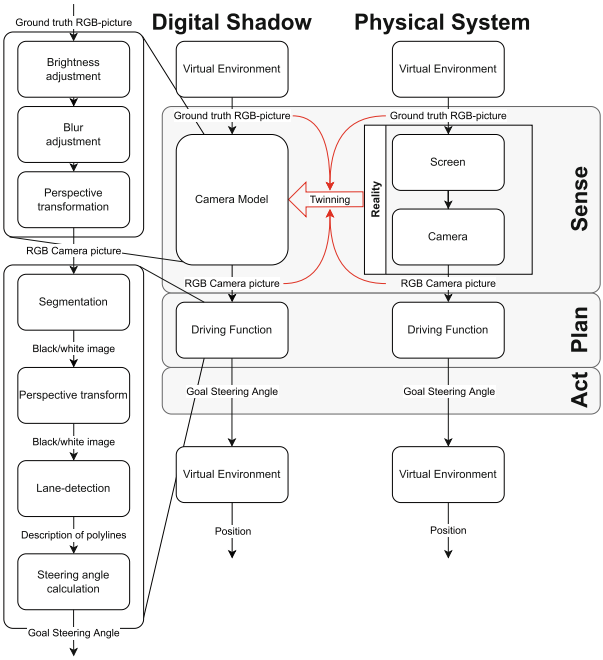


Fig. 3. Information flow through the DS and physical system highlighting the two instances where components are not virtual and which part shall be twinned.

To clarify, the setup follows the notion of [23] for hierarchical DTs: a DT may consist of other DTs and models. In this case, the DT consist of a DS of the testbed as shown in Fig. 1. Further, the DS consists of several models such as the driving function and the virtual environment, but also of a DS of the camera (seen in Fig. 3 as "Camera Model"). In the following, we will refer to the DS of the testbed as "the DS".

5 Technical Implementation Physical System

The general setup of the ViL demonstrator is a miniature vehicle based on the f1tenth-challenge[3] model car adapted and set into a virtual environment by running in front of a monitor and on top of a dynamometer as seen in Fig. 4. The vehicle can move and turn its wheels while only the previous can be captured by the dynamometer. As the scope of this setup has been expanded beyond DT development and is used in multiple projects as a demonstrator, it was given the name AMDEV-bed (abbreviation for "Automotive Model Demonstration Evaluation and Verification Bed".

It must be mentioned that mounting the necessary hardware on a miniature vehicle with actuators does not directly contribute to the fidelity of the setup with respect to a real testbed. In this context, a straightforward camera and monitor configuration may be equally effective in achieving the desired outcome. Here a simple camera and monitor setup may have achieved the same purpose. However, the benefit of doing so is two-fold. First, it improves the adaptability for future demonstrations and reuse of the setup. Second, due to a visually analogous configuration to its full-scale counterpart, it facilitates comprehension and familiarity among observers due to its similarity in appearance. This visual resemblance enables an intuitive understanding of the system's operation and functionality.

In the following, only the functionality relevant to developing DT-based RL-related methods is outlined and described.

On a technical level, the setup consists of a simulation computer and the miniature vehicle platform as it can be seen in Fig. 5. While the computer is an off-the-shelf desktop PC, the vehicle's computing is done on a Nvidia Jetson TX1. Both run on Ubuntu 20.04 LTS. We have developed the software stack in such a way that ROS 2 Foxy[4] [20] is used as the communication middleware between components wherever possible. The reason is that retrieving data at each step for their analysis is straightforward by subscribing to ROS topics. Also, the components' interfaces are standardized in that way, so easy replacement (like a new driving function) is made possible as well as deployment on different devices. As shown in Fig. 5, the PC has to run three main tasks:

CARLA Server. This component is the virtual environment represented by the open-source simulator CARLA [10] in the version 0.9.13[5]. It is based on

[3] See https://f1tenth.org/.
[4] See https://www.ros.org/.
[5] See https://carla.readthedocs.io/en/0.9.13/.

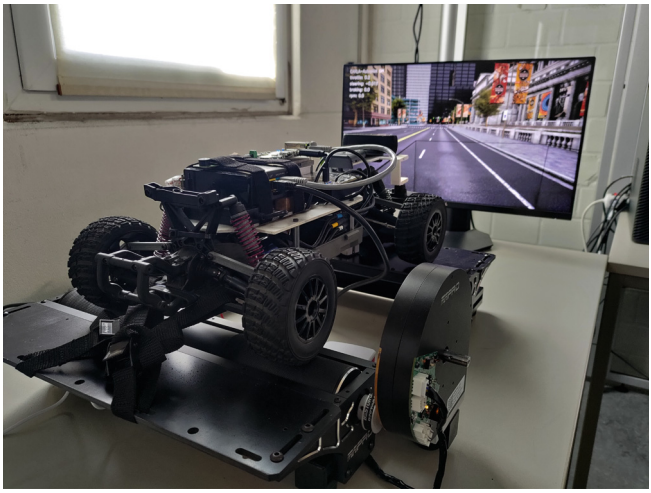


Fig. 4. Image of the miniature ViL testbed setup called AMDEV-bed in an early stage of development. It shows the model car on a roll stand in front of a monitor.

Unreal Engine 4 and has some special features such as pre-build vehicle models and dynamics, street maps, etc. ideal for academic prototyping. CARLA adopted the server-client structure where one can connect through a Python API to a running server instance and is capable of controlling any aspect of the simulation while also retrieving data. The virtual vehicle has a POV camera, and its output is projected onto a monitor which is positioned in front of the physical vehicle, thus mimicking the RGB camera's stimulation as if the vehicle would be driving in the environment itself. The virtual camera within CARLA has the possibility of adjustments[6] the camera configurations and positioning with respect to the origin of the car. However, these settings are left untouched.

ROS 2-CARLA bridge. We created a bridge between the CARLA API and ROS 2 to translate information between the two mediums as a CARLA-client instance. This component receives control inputs via ROS 2 from the vehicle as well as sends out the camera footage from the virtual camera created in CARLA. In addition, the bridge also serves as the simulation control by setting up the simulation (loading maps and configurations) as requested e.g., by keyboard inputs (not shown in Fig. 5).

Camera rendering. As the ROS 2-CARLA bridge published the virtual camera footage via ROS 2, an additional Python script simply displays it on a monitor connected to the PC for it to be captured by the camera of the AMDEV-bed.

[6] see https://carla.readthedocs.io/en/latest/ref_sensors/#rgb-camera.

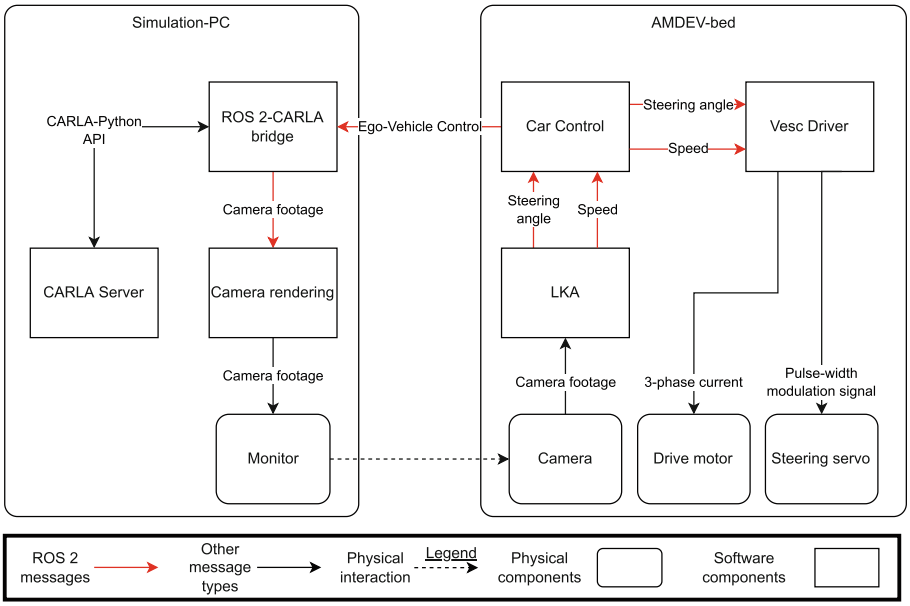


Fig. 5. High-level architecture of the AMDEV-bed. Communication is mainly through the ROS 2 middleware if possible to keep components modular and data reusable.

The vehicle of the AMDEV-bed is composed by three main software components:

LKA. The camera footage that is obtained via the USB webcam connected to the Nvidia Jetson, is processed at the lane-keep assistant (LKA). This driving function is based on the Python version of OpenCV[7] [6]. It consists of four steps as shown in Fig. 3 and is inspired by the work of [18]. First, the lane markings are isolated by using a brightness threshold within a region of interest. This region of interest is situated on the lower part of the footage to only capture the street in front of the vehicle. As the camera's perspective shrinks the dimensions of parts of the lane further away, the image is geometrically transformed to account for this. Next, 2^{nd}-order polynomial curves are fitted through each of the markings using a sliding window algorithm (see [14]) and a fitting function[8]. The middle of the lane further in front of the vehicle can be estimated and a steering angle calculated to reach this point continuously. If one lane is not detected reliably, it falls back to the other with a static offset. Otherwise, it sends out a signal to stop the vehicle. While this algorithm is quite rudimentary, it fulfils its purpose by being able to reliably drive along the given scenario but remains sensitive to image degradation. The output of this function is at one hand a steering angle as well as a throttle position (kept constant unless no markings are detected), both published via ROS 2.

[7] See https://opencv.org/.
[8] See https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html.

Car Control. This software components simply formats and publishes via ROS 2 the output of the LKA to be send to the ROS 2-CARLA bridge on the simulation computer as well as to the VESC driver.

VESC driver. In order to control the motors of the vehicle, Hardware and Software from the VESC Project[9] has been used. The VESC driver is the component taking the control inputs via ROS 2 and translating them to the actuation of the wheels i.e., the drive motor and the steering servo. While at the current stage of development, this is only for illustration purposes, the roll stand has been prepared to measure the wheels' forward motion and send it via a ROS topic to control the simulation in the future.

6 Technical Implementation Digital Shadow

The DS is a simple software-in-the-loop setup as shown in Fig. 6. It is executed on the simulation computer and four main software components:

Sensor model. This component is the implementation of the sensor model. It imports measured configuration parameters from a previously created JSON file. The Twinning process is not captured in Fig. 6.

LKA. The LKA is a copy of the driving function deployed on the PT and is integrated as a function in the "Simulation main script".

Simulation main script This is a script that executes the simulation by starting and configuring the simulation as well as importing the sensor model and LKA to execute the control loop.

CARLA Server. The CARLA server is simply another instance of the simulation environment on the physical system.

It has to be noted, that the AMDEV-bed's sensor stimulation and vehicle dynamics are imperfect and approximate the model car on a street or any real environment only to some certain degree. However, we consider it as the "ground truth". As [36] shows, there is a multitude of approaches to sensor modelling to consider. While low-fidelity and medium-fidelity sensor models consider only geometrical (e.g., obstruction of sight) or physical aspects (detection probabilities), we opted for a high-fidelity approach. According to [36], these offer the most realistic output and are based on rendering the surrounding 3D scene. More specifically, a model is developed based on a post-processing principle where an ideal camera image (that is retrieved from the CARLA) is altered to look more like the realistic raw camera data. Examples of similar approaches have been published in [7,38,46]. Especially [7] structured the model as a pipeline of "filters" as will be done in the following. For this, a "grey-box" modelling approach was followed. This means we leveraged data-driven modelling with prior knowledge about the system. While there are physical phenomena that can be observed and identified in the process, they are not tried to be explained and integrated

[9] See https://vesc-project.com/.

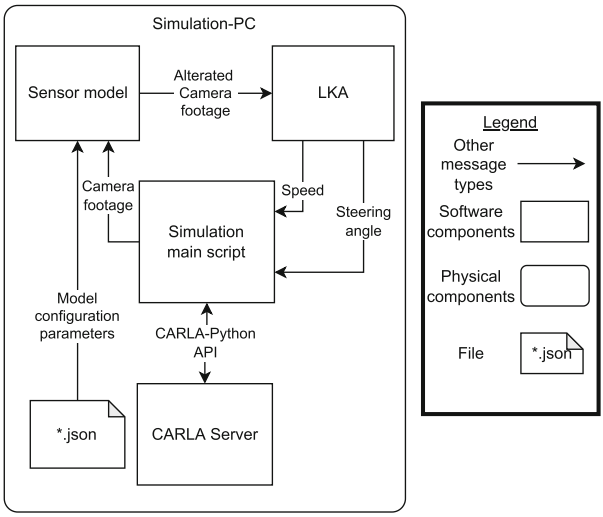
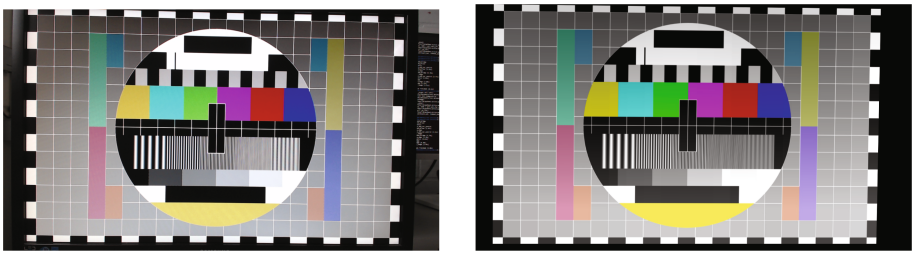


Fig. 6. High-level architecture of the DS.



(a) Test frame from camera feed (b) Adjusted test picture

Fig. 7. Side-by-side comparison of the camera feed's and DS's output to a test picture.

as first-principle models, but algorithms are put into place which replicate the resulting behaviour.

To identify static phenomena, first a test picture (see Fig. 7) is displayed on the monitor and a frame of the camera feed is taken for comparison (see Fig. 7a). In continuation, the captured image is processed and a configuration file (json-file in Fig. 6) is created. In the following, the identified phenomena and their quantization are explained.

Perspective

Due to the imperfect alignment of the camera with the monitor, a shift, scaling and warping effect can be seen. The camera can be both off-axis and at an angle to the monitor. In addition, the monitor only occupies part of the camera's frame (scaling). These phenomena are addressed by first identifying the four corners of the monitor by searching for the closest pixel to the corner of the test frame

which is above a certain brightness threshold. Then, a perspective warp function can be applied to the test picture (see OpenCV's function "warpPerspective"[10]) for it to match the sample test frame. This method can replicate the found perspective phenomena. The limitations are that it is sensitive to finding the right pixel representing the corners of the screen which needs to be visible. It also assumes that they are linear e.g., no barrel or pincushion distortion, which has not been observed in the test frame.

Brightness

When observing the grey grid on the test picture in Fig. 7a an obvious shift in brightness from top to bottom (increasing brightness) can be seen. This is due to the polarization of the monitor where with increasing viewing angle, higher brightness distortion is induced. In addition, the shift can be seen from left to right due to the same reason. To adjust the original test picture in such a way that the brightness mimics the one from the test frame, the test picture is sliced vertically along the grid of grey boxes. It is estimated that the brightness within one box does not change significantly. Thus, for each slice i of $n = 21$ total slices, the brightness of the topmost $G_u[i]$ and undermost box $G_l[i]$ is measured. From the test picture the true brightness B_{true} is known. The necessary brightness adjustment in each slice for each pixel row of the test picture is then linearly interpolated between the aforementioned topmost and undermost boxes. Let $I(x, y)$ represent the pixel brightness value at position (x, y) in the already perspective-adjusted input image and $I'(x, y)$ for the respective output image. In this case, the origin is the upper-left corner. S_i is the set of pixel coordinates in the i-th slice, defined by the slice boundaries. H_l and H_u are the upper and lower measurement height of the brightness, respectively. For each slice i (where $0 \leq i < n$), the slice boundaries are $x \in [B_i, B_{i+1})$, where B_i and B_{i+1} are the slice boundaries. For each pixel row y within the slice, the interpolated brightness adjustment $\Delta B(y, i)$ is calculated as follows:

$$I'(x, y) = \text{clip}\left(I(x, y) + \Delta B(y, i), 0, 255\right), \text{if } x \in [B_i, B_{i+1}) \text{ for each slice } i \quad (1)$$

where

$$\Delta B(y, i) = \frac{(G_l[i] - B_{true}) - (G_u[i] - B_{true})}{H_l - H_u}(y - H_u) + (G_l[i] - B_{true}) \quad (2)$$

Blur

Due to the monitor and camera imperfections, blur is induced, requiring its modelling. The direct measurement of blur is not possible. However, there are methods of measurement for sharpness which is directly reverse related to blur like the calculation of the variance of the picture's Laplacian[11]. In short, a high value of the Laplacian is given where sharp edges between pixel brightness are detected which can be understood as high sharpness and low blur. The overall

[10] See https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87.

[11] See https://pyimagesearch.com/2015/09/07/blur-detection-with-opencv/.

variance gives a reliable metric. However, for reproducible results, the underlying image must be (almost) identical and a link between its Laplacian's variance and Gaussian blur needs to be established. For this, a lookup table is created. The original test picture is iteratively blurred with increasing levels of Gaussian blur in the function of sigma. The sharpness according to the above-mentioned process is calculated and the sigma-sharpness data points are stored. The resulting data can be seen in Fig. 8.

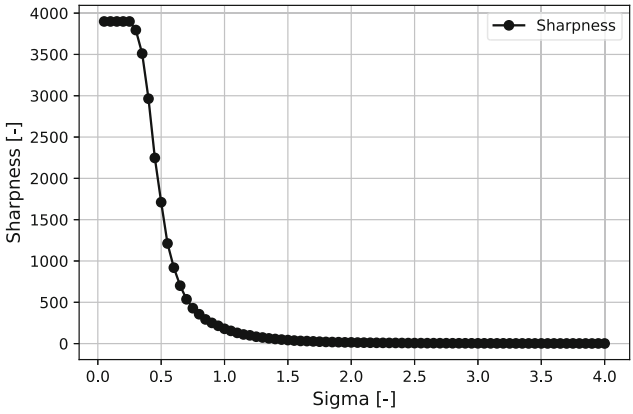


Fig. 8. Gaussian blur's sigma to sharpness relation for the given test image. Increasing sigmas mean increased blurriness of the image.

To reverse the calculation, the test frame from the camera is "reverse-warped" (the test frame looks now like the test picture) according to the already mentioned perspective change and sharpness is measured. Using the sigma-sharpness data as a lookup function, the respective sigma can be calculated and applied to the test picture.

When blur, brightness and perspective adjustments are applied to the test picture, the resulting image is shown in Fig. 7b. It can be seen that the image looks subjectively similar to the Fig. 7a, thus it works as designed. However, slight differences can still be identified. On one hand, there is a difference in colour saturation and effects from outside the screen are not accounted for. However, as validation is one of the research topics the setup shall address, slight but easy-to-identify differences are seen as advantageous.

7 Application

To test and demonstrate the RL environment toolchain for the optimization of CPSs, the setup can now be used. As already stated, the goal of the RL agent's optimization is the alternation of the virtual environment towards critical scenarios given the properties of a vehicle and its testbed.

The use of scenarios is motivated by the fact that it is a major challenge to test and later validate and certify vehicles with highly automated driving functions given the open world they need to navigate in [17]. One solution is the concept of Scenario-based Testing (SBT) (see [5]) with a lengthy history of research [8,25,27,35] where the driving function is tested against a suite of scenarios which shall provide an argument for the function's correctness. This kind of vehicle testing is common and is seen to be promising for the certification of automated driving systems [4,9]. However, the selection of such scenarios that are critical for a given automated vehicle's configuration is more than often cumbersome and requires a lot of time for manual tweaking. Thus, the application of DT-based RL shall optimize concrete testing scenarios derived from an abstract "blueprint" logical scenario [22] by an RL agent (RLA) to maximize criticality.

This concept is automated using the RLA. On the right of Fig. 9, one can see the simplified loop for training and RLA. RL algorithms are trained iteratively as they react to a state the system is in with an action. Based on that action, the system (or environment as it is called in the RL domain [42]) transfers to a new state which is evaluated with a resulting reward. Generally, RL algorithms try to optimize the anticipated cumulated reward. As it can be seen in Fig. 9 on the bottom right, the RL agent proposes a concrete scenario (action) as a list of parameters. These are then processed and translated by a scenario generator to a concrete, executable scenario by alternating the simulated world. It is then loaded into the "environment simulator"' which represents the testbed's DS (more specifically loaded into the simulation environment) and is executed there. The scenario results (e.g., time series or tally statistics) need to be processed by the "Feature engineering" to become an abstract representation (state) and reward, feeding them back to the RLA. We see the physical-digital twinning in line with [15] as the transfer of model configuration parameters to the environment simulator and the transfer of an optimized, altered environment for the scenario for digital-physical twinning.

8 Experiments and Evaluation

With the setup in place, it allows us to perform the experiments to train, understand and evaluate DT-based RL. As the physical system and the DS are entirely independent of each other, simulations may be done on a different machine at a different point in time or the twins can be executed simultaneously. This can be seen in Fig. 10 where both instances are compared regarding a similar behaviour. This was done by executing the same scenario (e.g., driving in circles on CARLA's Town 10). The lower part of Fig. 10 shows the actual camera footage (right) and the sensor model's output (left) including the detected lanes highlighted in green and the steering direction in red.

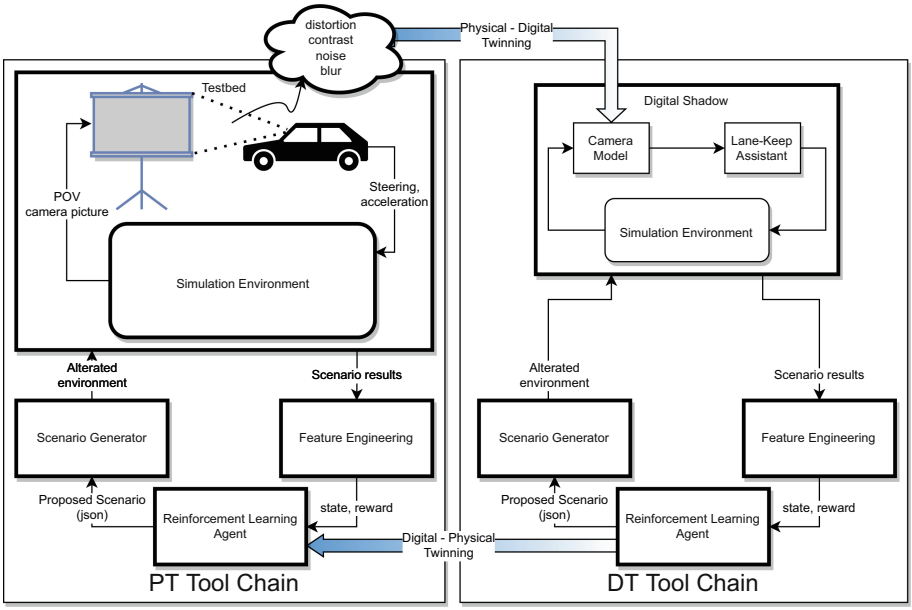


Fig. 9. Overview of the RL toolchain and interaction with the ViL setup. Blue arrows show the twinning. (Color figure online)

As mentioned in Sect. 2, the testbed was used for (I) the development process of DTs in general including identification of relevant parameters, (II) DT architecture in the RL context and (III) DT validation.

Regarding the development process for DTs, the chosen setup helped to get insights into modelling and DT creation. Guidelines for DT development in the context of DT-based RL are presented in [24]. Here, special attention was given to the implications the RL context gave to the DT development process. The modelling process guidelines were successfully validated during the creation of the setup. Also, the simulation environment including the vehicle's LKA and sensor model was used to perform a sensitivity analysis as a method to identify relevant parameters. However, the sensitivity analysis was solely applied to a subset of parameters that the RLA can configure as well as fidelity levels of selected 3D assets. The configuration of the sensor model remained the same across all iterations.

Another activity investigated the DT architecture for RL. For this, a use-case agnostic architecture for DTs using four different viewpoints was successfully developed [23] to fit DT-based RL over the system's lifecycle. As the architecture and the setup were developed at the same time, ideas from the DT architecture could be validated with the help of the setup and the implementation could be compared with the architecture. This included the successful integration into the toolchain as described in Sect. 7 (see Sect. 9).

Fig. 10. Footage taken from DS (left) and AMDEV-bed (right). The upper images show the two representations in their respective virtual environment and the lower images are the modelled and actual footage, respectively.

The third activity is with respect to DT validation summarized in [45]. As the purpose of the DS is to enable the training of the agent, its validity is ultimately linked to the agent's performance in controlling the physical system which is beyond the scope of this paper. While only face-validation and rudimentary comparison of the vehicle's behaviour were implemented to directly test the DS for now, the setup sparked the discussion on new ways of validating vehicle and perception models which will be investigated in the future and evaluated with the setup.

Describing the results of the performed activities is beyond the scope of this paper. However, we conclude that the envisioned and implemented setup (physical system and DS) is well-suited for the desired purpose. The setup enabled and accelerated the research because it helped us to get insights into DT development and use as well as validation tools for DT methods and applications. In essence, it fulfils the requirements described in Sect. 2.

1. **Adaptability** Using standardized interfaces in the PT (in our case ROS 2), the setup has become easily expandable and therefore adaptable to changing needs. E.g, diagnostic and visualization tools were added in later stages with low effort. Also, in a prior iteration of the setup to the one presented in this work, an alternative driving function (emergency braking) and scenario were utilized. The evolution to the current state required minimal to no modifications.
2. **Explainability** The PT and therefore also of the DT are composed of small components with simpler functionalities and clear interfaces. This eases explaining details of the setup to others and, thus, provides a high level of explainability.

3. **Relevancy** While the behaviour of the setup compared to a full-scale testbed was not assessed, during the discussions with automotive experts from the industry, it became clear that the setup and resulting insights allowed in-depth discussions and further enabled and contributed to enhancements in the use case.
4. **Accessibility** The use of well-known and well-documented open-source software (like Python, ROS 2, OpenCV and CARLA) significantly reduced the effort needed to create the setup. This is because of the accessibility and availability of prior knowledge. Additionally, the physical aspect of the setup (due to the use of off-the-shelf components and their arrangement) is easy to grasp and even modify by potential users. This further allows the application e.g., for educational purposes in lectures and trainings. Besides this main advantage, the setup minimizes the time required to get acquainted with the setup.
5. **Efficiency** The setup resulted in minimal costs of construction due to freely available, off-the-shelf hardware and the usage of open-source software. This also reduced the personnel resources required. Once it was developed, there are no running costs.
6. **Reproducibility** Thanks to the virtual environment, scenarios can be replayed whenever needed with closely similar results. This enables high reproducibility. However, there are potential sources of non-determinism in the interaction between the monitor (sensor stimulator) and camera (sensor) due to external factors in the laboratory where the setup is located (like light).

We conclude, that our setup was able to fulfil the requirements and significantly contributes to the application of DT methods and demonstrations of the proofs of concepts. In our experience choosing a scaled-down version as the physical system significantly accelerated the development, application of DT methods and obtention of results.

9 Learnings

In the following, we collected the five main "meta learnings" of the AMDEV-bed and DS development. These are beyond the knowledge gained in terms of the methods developed/evaluated with the setup at hand but may be as relevant.

9.1 Understand the System

Understanding the system thoroughly is vital for the development of such a setup. We achieved this by creating both the testbed and DS simultaneously step by step. Although we did not implement a fully modular architecture, we noticed that achieving full understanding can only be accomplished through a modular approach to setup development. This also enables validation of components which helps with finding sources of errors early during development time and can be used to rate the system's overall validity (see [30]).

9.2 Highly Automated Twinning Is Key for Automation and DTs Should Be Designed as Such

Our overall workflow benefited from the ability to have an automated twinning process. As the state of the physical system needs to be assumed to change constantly, the additional effort of designing the setup and implementing a twinning process[12] was worth it. Especially when the demonstrator is used for dissemination purposes, quick on-site synchronization of the DS makes the setup more convincing as the expected behaviour of third persons is that the physical system and DS are similar.

9.3 Communicating What a DT Is, Is Still Complicated

At first, it was difficult to communicate what the DT of a testbed is, especially as the testbed contained major parts that were "simple" simulation models. While these components are essential to the construct, it has been found that they are not seen as part of a DT by others. Expressed by the architecture developed in [23], we adopted the mindset of hierarchical DTs where DTs can consist of models or software components and DTs of components of the overarching system. This has been found to be a very robust view on the topic and was confirmed during the development of the setup.

9.4 Benefit of in-Sourcing the Experimental Setup

The utilization of a small-scale demonstrator offered several advantages, especially in facilitating on-site experiments which are less resource-intensive in terms of both time and budgetary requirements. This approach proved to be greatly beneficial because we were not dependent on access to external systems or facilities controlled by other stakeholders. It mitigated the risk associated with scheduling meetings (e.g. due to conflicts), logistical challenges and intellectual property protection. It is a common issue in research projects that partners who could provide knowledge may not want to (or can) share their knowledge which is necessary for the construction of a DT and the execution of experiments with the actual system. By employing a compact demonstrator, we were able to conduct local experiments with greater efficiency and autonomy, thereby minimizing the need for extensive coordination with external parties. This independence not only streamlines the evaluation process but also improves the credibility of research outcomes by ensuring a robust, controlled and most important close to realistic testing environment.

9.5 Dissemination Enabler

One interesting side effect of the setup is the potential for dissemination activities. As our miniature testbed is compact, relatively easy to transport and to

[12] We regard twinning as "the act of synchronizing the states of the physical and virtual entity/twin" [15].

set up, we identified the opportunity to showcase it as part of our method dissemination. While posters and presentations are a well-established medium for disseminating academic results. Often such results seem to lack a tangible, real-world application. As DTs inherently have a physical component, a small-scale demonstrator can bridge the gap between a theoretical concept and a practical application in an accessible manner. It is simply more convincing and facilitates understanding and engagement among the audience. Further, it fosters interactive discussions. We noticed during discussions with colleagues that the subject is easier to understand, critical feedback can be made clearer and, thus, progress can be made faster. We think that this comes from the simplified, self-contained problem the miniature test setup portrays.

This also holds for external dissemination of research results. We brought a live demonstration to the ESI Symposium 2024 including a large monitor to show live feeds and statistics from the synchronously running DS and testbed (see Fig. 11). It helped non-domain experts to understand the subject and to quickly grasp the application and even to transfer it to their respective domains. While it is an increased effort to bring a live demonstration in terms of preparation and logistics, the intensified engagement with scholars and non-experts for cross-domain dissemination is worth it.

Fig. 11. Demonstration setup of the AMDEV-bed on the ESI Symposium 2024. The setup includes the AMDEV-bed as well as an interactive digital poster containing live feeds and telemetry from the setup.

10 Conclusion

In this paper, we share our experience in the development and application of a miniature ViL testbed for investigating methods based on DTs. We document and discuss our motivation and choice of a miniature setup including requirements, the integration into the context of the use case in the ASIMOV project,

we present the modular concept of the chosen setup, how we implemented the functionality of the individual modules (reusing existing components as well as building on standardized interfaces) in detail, and how the setup was used. We report on the overall development and the assessment of the collected requirements as well as insights gained during the development.

The setup has been shown to enable training, understanding and evaluating RL for an agent operating the physical system by fulfilling the requirement stated and supporting the creation of academic results regarding DT-based RL. In conclusion, we strongly advocate to use of such scaled-down versions of the reference system for experimenting, learning, and evaluating. It enables one to spend limited resources on developing new applications faster and more efficiently. In addition, disseminating research results on the topic of DTs becomes more intuitive, perceivable and, in turn, effective. It is in line with the idea of "think big, start small" and makes research more agile and convincing. In our research institute, we plan to use the developed setup as a demonstrator and, over time, expand its capabilities to enable other research topics, exploiting the knowledge and experience we have gained.

Acknowledgments. This research is partially funded by the German Federal Ministry of Education and Research (BMBF) within the project ASIMOV-D under grant agreement No. 01IS21022G based on a decision of the German Bundestag.

We would like to thank Marvin Günther for creating the interactive visualization that helped us share our findings at the ESI symposium. His work was pivotal for making our presentation much clearer and more engaging.

References

1. Aheleroff, S., Xu, X., Zhong, R.Y., Lu, Y.: Digital twin as a service (DTaaS) in industry 4.0: an architecture reference model. Adv. Eng. Inform. **47**, 101225 (2021). https://doi.org/10.1016/j.aei.2020.101225
2. Alghodhaifi, H., Lakshmanan, S.: Autonomous vehicle evaluation: a comprehensive survey on modeling and simulation approaches. IEEE Access **9**, 151531–151566 (2021)
3. Barricelli, B.R., Casiraghi, E., Fogli, D.: A survey on digital twin: definitions, characteristics, applications, and design implications. IEEE Access **7**, 167653–167671 (2019). https://doi.org/10.1109/ACCESS.2019.2953499
4. Bock, T., Maurer, M., van Meel, F., Müller, T.: Vehicle in the loop. ATZ-Automobiltechnische Z. **110**(1), 10–16 (2008)
5. Brade, T., Kramer, B., Neurohr, C.: Paradigms in scenario-based testing for automated driving. In: 2021 International Symposium on Electrical, Electronics and Information Engineering, pp. 108–114. ACM, New York, NY, USA (2021). https://doi.org/10.1145/3459104.3459208
6. Bradski, G.: The OpenCV Library. Dr. Dobb's J. Softw. Tools (2000)
7. Carlson, A., Skinner, K.A., Vasudevan, R., Johnson-Roberson, M.: Modeling camera effects to improve visual learning from synthetic data. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision - ECCV 2018 Workshops, pp. 505–520. Springer International Publishing, Cham (2019)

8. Damm, W., Harel, D.: LSCs: breathing life into message sequence charts. Formal methods in system design **19**, 45–80 (2001)
9. Dona, R., Ciuffo, B.: Virtual testing of automated driving systems. a survey on validation methods. IEEE Access **10**, 24349–24367 (2022). https://doi.org/10.1109/ACCESS.2022.3153722
10. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16 (2017)
11. Düser, T.: X-in-the-Loop - an integrated validation framework for vehicle development using powertrain functions and driver assistance systems (2010)
12. Ferko, E., Bucaioni, A., Behnam, M.: Architecting digital twins. IEEE Access **10**, 50335–50350 (2022). https://doi.org/10.1109/ACCESS.2022.3172964
13. Grieves, M., Vickers, J., (None): Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahlen, F.J., Flumerfelt, S., Alves, A. (eds.) Transdisciplinary Perspectives on Complex Systems, pp. 85–113. Springer (2016). https://doi.org/10.1007/978-3-319-38756-7
14. Hanssen, S., Holzinger, K., Stubbe, H.: Towards general sliding window stream analysis. Netw. **47** (2021)
15. Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B.: Characterising the digital twin: a systematic literature review. CIRP J. Manuf. Sci. Technol. **29**, 36–52 (2020). https://doi.org/10.1016/j.cirpj.2020.02.002
16. Kapteyn, M.G., Willcox, K.E.: From physics-based models to predictive digital twins via interpretable machine learning. http://arxiv.org/pdf/2004.11356v3
17. Koopman, P., Wagner, M.: Challenges in autonomous vehicle testing and validation. SAE Int. J. Trans. Saf. **4**(1), 15–24 (2016). http://www.jstor.org/stable/26167741
18. Landwehr, J.: Realisierung einer kamerabasierten Spurerkennung für autonomes Fahren (2023)
19. Liu, X., et al.: A systematic review of digital twin about physical entities, virtual models, twin data, and applications. Adv. Eng. Inform. **55**, 101876 (2023). https://doi.org/10.1016/j.aei.2023.101876
20. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: design, architecture, and uses in the wild. Sci. Rob. **7**(66), eabm6074 (2022). https://doi.org/10.1126/scirobotics.abm6074
21. Meng, Z., Zhao, S., Chen, H., Hu, M., Tang, Y., Song, Y.: The vehicle testing based on digital twins theory for autonomous vehicles. IEEE J. Radio Freq. Identif. **6**, 710–714 (2022). https://doi.org/10.1109/JRFID.2022.3211565
22. Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles: 26-30 june 2018. In: IEEE, Piscataway, NJ (2018). https://doi.org/10.1109/IVS.2018.8500406
23. Modrakowski, E., et al.: Architecture for digital twin-based reinforcement learning optimization of cyber-physical systems. In: Tekinerdoğan, B., Spalazzese, R., Sözer, H., Bonfanti, S., Weyns, D. (eds.) LNCS 14590. Springer Nature Switzerland AG (2024). https://doi.org/10.1007/978-3-031-66326-0_16
24. Moritz, S., et al.: Identification of relevant parameters modelled in DT (Project Deliverable). ASIMOV (2024). https://itea4.org/project/workpackage/document/download/8931/T2.1-Parameter-Identification-Process-v2.0.pdf
25. Nalic, D., Mihalj, T., Bäumler, M., Lehmann, M., Eichberger, A., Bernsteiner, S.: Scenario based testing of automated driving systems: a literature survey. In: FISITA web Congress, vol. 10, pp. 1 (2020)

26. Neurohr, B., de Graaff, T., Eggers, A., Bienmüller, T., Möhlmann, E.: Providing evidence for the validity of the virtual verification of automated driving systems. In: European Dependable Computing Conference, pp. 5–13. Springer (2024). https://doi.org/10.1007/978-3-031-56776-6_1
27.)Neurohr, C., Westhofen, L., Henning, T., de Graaff, T., Möhlmann, E., Böde, E.: Fundamental considerations around scenario-based testing for automated driving. In: Intelligent Vehicles Symposium Proc, pp. 121–127 (2020). https://doi.org/10.1109/IV47402.2020.9304823
28. Pietruch, M., Młyniec, A., Wetula, A.: An overview and review of testing methods for the verification and validation of ADAS, active safety systems, and autonomous driving. Min. Inform. Autom. Electr. Eng. **58**(1) (2020)
29. Rasheed, A., San, O., Kvamsdal, T.: Digital twin: values, challenges and enablers from a modeling perspective. IEEE Access **8**, 21980–22012 (2020). https://doi.org/10.1109/ACCESS.2020.2970143
30. Reeb, D., Patel, K., Barsim, K., Schiegg, M., Gerwinn, S.: Validation of Composite Systems by Discrepancy Propagation. http://arxiv.org/pdf/2210.12061v1
31. Reway, F., Hoffmann, A., Wachtel, D., Huber, W., Knoll, A., Ribeiro, E.: Test method for measuring the simulation-to-reality gap of camera-based object detection algorithms for autonomous driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 1249–1256 (2020). https://doi.org/10.1109/IV47402.2020.9304567
32. Riedmaier, S., Nesensohn, J., Gutenkunst, C., Düser, T., Schick, B., Abdellatif, H.: Validation of x-inthe-loop approaches for virtual homologation of automated driving functions. In: 11th Graz Symposium Virtual Vehicle (2018). https://api.semanticscholar.org/CorpusID:202611286
33. Riedmaier, S., Schneider, D., Watzenig, D., Diermeyer, F., Schick, B.: Model validation and scenario selection for virtual-based homologation of automated vehicles. Appl. Sci. **11**(1), 35 (2021). https://doi.org/10.3390/app11010035
34. Rjabtšikov, V., Rassõlkin, A., Kudelina, K., Kallaste, A., Vaimann, T.: Review of electric vehicle testing procedures for digital twin development: a comprehensive analysis. Energies **16**(19) (2023). https://doi.org/10.3390/en16196952
35. Ryser, J., Glinz, M.: A scenario-based approach to validating and testing software systems using statecharts. In: 12th International Conference on Software and Systems Engineering and their Applications (1999)
36. Schlager, B., et al.: State-of-the-art sensor models for virtual testing of advanced driver assistance systems/autonomous driving functions. SAE Int. J Connected Autom. Veh. **3**(3), 233–261 (2020). https://doi.org/10.4271/12-03-03-0018
37. Schluse, M., Priggemeyer, M., Atorf, L., Rossmann, J.: Experimentable digital twins-streamlining simulation-based systems engineering for industry 4.0. IEEE Trans. Ind. Inf. **14**(4), 1722–1731 (2018). https://doi.org/10.1109/TII.2018.2804917
38. Schneider, S.A., Saad, K.: Camera behavioral model and testbed setups for image-based ADAS functions. e & i Elektrotechnik und Informationstechnik **135**(4-5), 328–334 (2018)
39. Schütt, B., Steimle, M., Kramer, B., Behnecke, D., Sax, E.: A taxonomy for quality in simulation-based development and testing of automated driving systems. IEEE Access **10**, 18631–18644 (2022)
40. Stark, R., Damerau, T.: Digital Twin. In: Chatti, S., Tolio, T. (eds.) CIRP Encyclopedia of Production Engineering, pp. 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg (2019). https://doi.org/10.1007/978-3-642-35950-7_16870-1

41. Stripling, H.F., Adams, M.L., Mcclarren, R.G., Mallick, B.K.: The method of manufactured universes for validating uncertainty quantification methods. Reliab. Eng. Saf. Syst. **96**(9), 1242–1256 (2011)
42. Sutton, R.S., Barto, A.: Reinforcement learning: an introduction. adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts and London, England, second edition edn. (2018)
43. Viehof, M., Winner, H.: Stand der technik und der wissenschaft: modellvalidierung im anwendungsbereich der fahrdynamiksimulation. Tech. rep, TU Darmstadt, Darmstadt, Germany (Juli (2017)
44. Wagg, D.J., Worden, K., Barthorpe, R.J., Gardner, P.: Digital Twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications. ASCE - ASME J. Risk Uncertainty Eng. Syst. **6**(3) (2020). https://doi.org/10.1115/1.4046739
45. Wild, M., et al.: Digital Twin Validation - Methods and Techniques (Project Deliverable). ASIMOV (2024). https://itea4.org/project/workpackage/document/download/8906/ASIMOV_D2_5_M32_version_1-0
46. Wittpahl, C., Zakour, H.B., Lehmann, M., Braun, A.: Realistic image degradation with measured PSF. Electron. Imaging **30**(17), 149–1–149–1 (2018). https://doi.org/10.2352/ISSN.2470-1173.2018.17.AVM-149
47. Xue, D., Cheng, J., Zhao, X., Wang, Z.: A vehicle-in-the-loop simulation test based digital-twin for intelligent vehicles. In: 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 918–922 (2021). https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00153