Applications

Alexander Schuster, Raphael Hagmanns, Iman Sonji, Andreas Löcklin*, Janko Petereit, Christof Ebert and Michael Weyrich

Synthetic data generation for the continuous development and testing of autonomous construction machinery

Synthetische Datengenerierung für die kontinuierliche Entwicklung von autonomen Baumaschinen

https://doi.org/10.1515/auto-2023-0026
Received February 28, 2023; accepted July 24, 2023

Abstract: The development and testing of autonomous systems require sufficient meaningful data. However, generating suitable scenario data is a challenging task. In particular, it raises the question of how to narrow down what kind of data should be considered meaningful. Autonomous systems are characterized by their ability to cope with uncertain situations, i.e. complex and unknown environmental conditions. Due to this openness, the definition of training and test scenarios cannot be easily specified. Not all relevant influences can be sufficiently specified with requirements in advance, especially for unknown scenarios and corner cases, and therefore the "right" data, balancing quality and efficiency, is hard to generate. This article discusses the challenges of automated generation of 3D scenario data. We present a training and testing loop that provides a way to generate synthetic camera and Lidar data using 3D simulated environments. Those can be automatically varied and modified to support a closed-loop system for deriving and generating datasets that can be used for continuous development and testing of autonomous systems.

Keywords: autonomous construction machinery; machine learning; requirements engineering; simulation; synthetic data generation

Zusammenfassung: Die Entwicklung und Erprobung von autonomen Systemen erfordern ausreichend aussagekräftige Daten. Die Generierung geeigneter Szenarien und Daten ist jedoch eine anspruchsvolle Aufgabe. Insbesondere stellt sich die Frage, wie man die Art der Daten eingrenzen kann, die als aussagekräftig gelten sollen. Autonome Systeme zeichnen sich dadurch aus, dass sie mit unsicheren Situationen, d.h. komplexen und unbekannten Umweltbedingungen, umgehen können. Aufgrund dieser Offenheit ist die Definition von Trainings- und Testszenarien nicht einfach zu spezifizieren. Insbesondere für unbekannte Szenarien und Corner-Cases können nicht alle relevanten Einflüsse im Voraus ausreichend mit Anforderungen spezifiziert werden, so dass es schwierig ist, die "richtigen" Daten mit einem ausgewogenen Verhältnis zwischen Qualität und Effizienz zu generieren. In diesem Artikel werden die Herausforderungen der automatischen Generierung von 3D-Szenariodaten diskutiert. Wir stellen eine Trainings- und Testschleife vor, die es ermöglicht, synthetische Kamera- und Lidardaten in simulierten 3D-Umgebungen zu erzeugen. Diese können automatisch variiert und modifiziert werden, um ein Closed-Loop-System für die Ableitung und Erzeugung von Datensätzen zu unterstützen, die für die kontinuierliche Entwicklung und das Testen von autonomen Systemen verwendet werden können.

Schlagwörter: autonome Baumaschinen; maschinelles Lernen; Requirements Engineering; Simulation; synthetische Datengenerierung

***Corresponding author: Andreas Löcklin**, University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany, E-mail: andreas.loecklin@ias.uni-stuttgart.de
Alexander Schuster, **Iman Sonji and Michael Weyrich**, University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany, E-mail: alexander.schuster@ias.uni-stuttgart.de (A. Schuster), iman.sonji@ias.uni-stuttgart.de (I. Sonji), michael.weyrich@ias.uni-stuttgart.de (M. Weyrich)
Raphael Hagmanns and Janko Petereit, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany, E-mail: raphael.hagmanns@iosb.fraunhofer.de (R. Hagmanns), janko.petereit@iosb.fraunhofer.de (J. Petereit)
Christof Ebert, Vector Consulting Services GmbH, Stuttgart, Germany; and Robo-Test, Stuttgart, Germany,
E-mail: christof.ebert@vector.com

1 Introduction

When developing autonomous systems, not all system requirements, such as unknown situations, corner cases, and difficult conditions, can be identified in advance due to the high complexity of the environment and the autonomous system itself [1, 2]. This can result in unwanted behaviors and critical defects, ranging from cybersecurity vulnerabilities [3] to issues from incomplete or corrupted perception [1]. In all cases, it is necessary to resolve the faulty behavior as quickly as possible by continuously improving the system. This requires rethinking traditional product development cycles. Continuous development and lifelong learning are required to guarantee adequate behavior in previously unseen scenarios. Such continuous development and testing, especially of complex autonomous systems with AI components, requires a lot of data for training and testing. The data must not only be available in sufficient quantity but must also match the situations, where the system previously failed to operate. Ultimately, such data-based automation solutions must be developed to prove the safe operation of autonomous systems. One of the major challenges in this continuous development lifecycle is to provide the "right" data for each system [4].

This work addresses how to efficiently and transparently generate test data and thus validate an automated system and in parallel optimize its requirements. This data generation pipeline is demonstrated using an autonomous excavator as an exemplary robotic platform. The platform and its sensor equipment are depicted in Figure 1. In this work, we focus on the generation and usage of synthetic camera and Lidar data, since our use cases focus on environment perception and object detection. The operation of autonomous construction machinery is not only highly relevant in different domains but also provides an ideal example for the identification of safety concerns due to its harmful nature. Perceiving the environment is the first essential step to enable such complex robotic systems to sense, perceive, and act autonomously within their intended surroundings. This environment perception is exceptionally hard for the typical operating environment of an excavator: It is complex, unstructured, and continuously evolving. Unsafe situations cannot be collected or even predicted beforehand [5].

This makes it hard to apply classical approaches of collecting real-world data or testing in a real operational environment. The use of *Digital Twins* for verification and validation [6], in particular the simulation of an autonomy stack in a realistically modeled operating environment, is one way to tackle this problem. However, this involves an additional development effort, which could be compensated for in later phases by benefits such as accelerated future development and the provision of error-free updates for users. Further advantages of synthetic-data-based systems arise with respect to data efficiency and data-driven identification of particularly hard-to-cover corner cases [7].

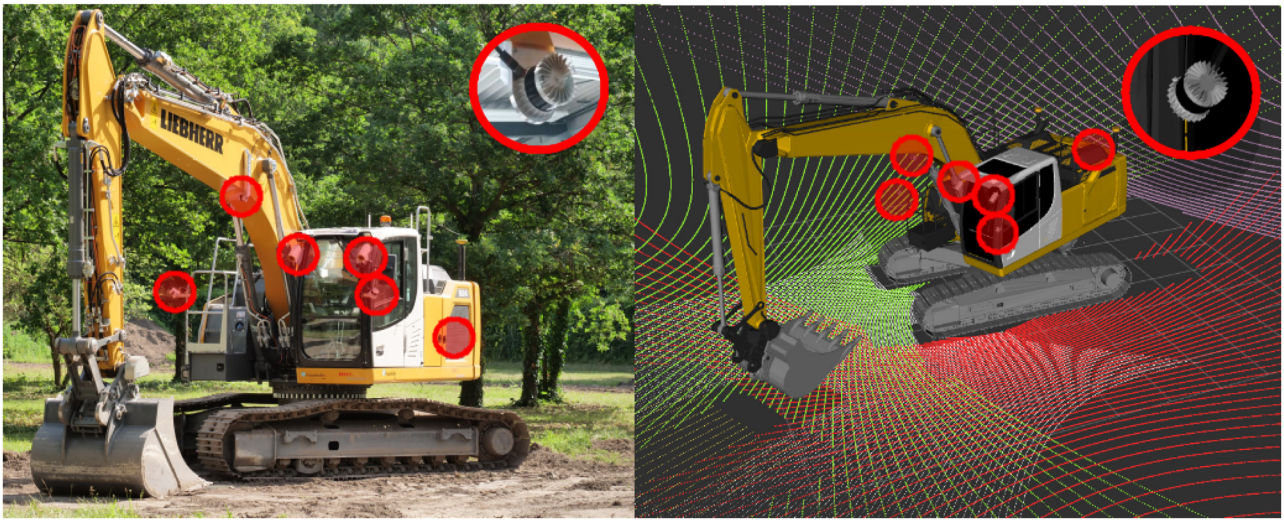


Figure 1: Overview on the sensors of the autonomous excavator used for experiments. On the left side, the real excavator is depicted with four Lidar sensors and two cameras highlighted with red circles. The simulated excavator is shown on the right side of the image, again with the virtual sensors highlighted in red. In addition, simulated points of the different Lidar sensors are shown in different colors to highlight the coverage of the environment.

Therefore, this work identifies the main challenges to the continuous development process of autonomous systems and discusses the suitability of synthetic data generation within a closed feedback loop of training and testing. The resulting main contributions are as follows:

- Introduction of a *Training and Testing Loop* scheme, which allows for continuous test-driven development of AI modules.
- Provision of a *Synthetic Dataset* in a rendered environment which allows to train and test autonomous capabilities of an excavator.
- *Simulation and Bridging* of the perception stack of an excavator within an Unreal-based simulation environment as well as an extensive evaluation of its capabilities.
- Introduction of an *Assistant System*, which allows guiding the user through the *Training and Testing Loop.*

The remainder of this paper is structured as follows: The following Section 2 presents current solution approaches from the literature, Section 3 introduces and analyzes an automated Training and Testing Loop that is based on the automated generation of synthetic data. Section 4 presents the concrete implementation of the proposed architecture in a prototyped fashion. The suitability of the generated synthetic data for improving the perception capabilities of an autonomous excavator is discussed. Finally, Section 5 concludes the work and presents further research directions.

2 Related work

An industrial autonomous system is defined in Ref. [8] as follows:

> An industrial autonomous system is a de-limited technical system, which systematically and without external intervention, achieves its set objectives despite uncertain environmental conditions.

Ref. [8, p. 5]. This applies in particular to autonomous mobile robots in outdoor environments, where the system complexity is exceptionally high. The main reason for this is the uncertain, complex, and often unstructured operating environment which necessitates the use of various sensors. The *Algorithm Toolbox* developed by Fraunhofer IOSB [9] serves as an autonomy provider for multiple complex robotic systems. In this work, we are using an autonomous excavator as a primary showcase. The excavator has been equipped with certain capabilities to perform decontamination tasks autonomously, as described in further detail in Petereit et al. [10]. The following Section 2.1 will give some context for safety considerations for such complex systems, before existing approaches to overcome these issues are presented.

2.1 Training and test of autonomous systems

For the verification and validation of an autonomous system, requirements and test cases must be specified. These test cases are then executed and evaluated. Covering all the requirements with at least one test case, for example during the initial product release, is labor intensive. After changes have been made to a system, regression testing is required to ensure that overall system performance has been improved.

In addition, an autonomous system must also be able to deal with unknown situations that have not yet been specified in requirements. Therefore, it is not sufficient to merely cover all requirements. An autonomous system must also react appropriately to unseen scenarios, with the system taking sole responsibility like a human being. In practice, this means that the development and testing of autonomous systems is a complex process that must be reconciled with a continuous improvement character. Development and test processes, as well as the associated supporting technologies, must therefore be scalable and repeatable during the lifecycle. And since development is never complete, it cannot be guaranteed that an autonomous system is able to successfully handle all tasks in all environments. Therefore, instead of exhaustively testing for correct behavior within a predefined set of scenarios, the focus should be on automatically identifying critical cases.

Those cases can be derived by evaluating requirements systematically, also known as *requirements-based* testing [11]. However, in many cases, not all requirements are known beforehand. Therefore, requirements-based testing is not adequate to identify all risks and test for all real-world scenarios. This is why *scenario-based* testing is used to test autonomous systems, such as autonomous vehicles [2]. Scenario-based testing is a testing method that extends requirements-based testing [12]. In this methodology, a *scenario* is a formal construct in a computer-readable description format that maps potentially complex and realistic work progress. The definition of a scenario is extended into three layers: functional, logical, and concrete scenarios. A functional scenario is described using natural language, which is formalized into a logical scenario by adding relevant parameters along with their value ranges and dependencies. Specific scenarios are derived from logical scenarios by assigning single values to the parameters.

The development of autonomous mobile systems is strongly driven by the automotive industry. Our methodology for testing autonomous systems is therefore based on the PEGASUS methodology [13] used in the automotive sector. The methodology shows how the structuring and formalization of the problem area can be used to test autonomous systems with a reasonable amount of effort. Relevant scenarios are developed based on the actual use of vehicles and the environment is formalized along different dimensions such as lighting conditions, weather, traffic situation, etc. The probability of occurrence of different scenarios is derived from empirical values. PEGASUS is used because classic test drives are not suitable for generating sufficient data for extremely rare and at the same time extremely dangerous scenarios. In PEGASUS, the test data can either be real recorded data that represent a specific scenario, or it can be synthetically generated data. The formalization makes it possible to use simulators in a targeted manner, since all relevant scenario parameters are described.

Similar challenges apply to autonomous construction machinery. Here too, test drives in reality are very expensive and only affordable late in the development process with very mature prototypes. In contrast to the automotive sector, off-road environments with few traffic rules have to be covered in a structured and formalized way. In addition, unlike cars, there is usually no meaningful data for the development of machine-learning based software available, as there are much less data sets in this area available that can be used for development purposes. Therefore, the scenario-based approach fulfills two tasks: Enabling the structured, synthetic generation of data that can be used for both development and testing.

2.2 Synthetic training data for the development of autonomous systems

Obtaining training, testing, and validation data for autonomous systems that operate in unstructured off-road environments such as construction sites and landfills, is still a challenging task. Therefore, datasets in these environments are very rare. Currently, only the RELLIS-3D [14] and the Semantic-USL [15] datasets record unstructured environments with modern sensor hardware. Those datasets cover both RGB-image data as well as Lidar data, which are typical sensor modalities used for perception in unstructured environments. However, there is an increasing availability of datasets captured in structured environments, such as the KITTI dataset [16]. For certain perception methods, data from structured environments can serve as a useful starting point. The same applies to synthetic training and testing data: While urban simulation environments have gained increasing popularity in recent years, simulation environments and digital twins are very rare for mobile robots in unstructured environments. The main reasons for this are the increased complexity of realistic assets as well as the smaller scope compared to the large research field of autonomous driving in structured environments.

The use of synthetically generated data helps to compensate for the lack of real data [17]. For the generation of synthetic data, the following techniques are distinguished:

Generic augmentation: Augmentation of small data sets by applying simple data transformations, for example, rotations of image data or random scaling of point cloud data [18, 19].

Complex augmentation: Creation of new data by changing parameters of real data, for example using a generative-adversarial network to create new image data that look similar to other image data [20].

Background substitution: Creation of new data by representing real target objects in front of different backgrounds. For example, cutting out an object and placing it in front of a bright, uniform background and a multicolored background [21].

Rendered target: Represents an extension of background substitution, the backgrounds are still based on real data, but the target object now exists as a virtual model and is inserted as a rendered target object. For example, the lighting of the target object can then be adjusted to the backgrounds in order to obtain a realistic overall image [22, 23]. Such modifications are also possible for point clouds where new ground truth models can be included into existing scenes [19].

Fully simulated synthetic data: In this approach, both the background and the target object exist as virtual models. To generate synthetic data, the background and target object are rendered together. For example, to generate photorealistic image and accurate Lidar data, appropriate simulation engines are used that simulate the entire environment. The advantages of this approach are that no real data is needed and data for unseen complicated or dangerous scenarios can be generated [24, 25].

Given the available approaches for generating synthetic data and the requirements for testing autonomous systems, only fully simulated synthetic data can meet all the challenges in terms of data availability. Particularly advantageous for the use of scenario-based testing is the possibility of generating data for previously unseen scenarios.

3 Improving the perception of autonomous systems using synthetic data

For the development and testing of autonomous systems in unstructured environments, a scenario-based testing method based on fully simulated synthetic data is proposed. Our work is focused on construction machines, that move autonomously, more specifically we use an autonomous excavator as a case study. The use case is to train the perception module of the autonomous excavator shown in Figure 1 using synthetic data to operate safely within its environment. To do this, the perception module requires camera and Lidar sensor data as inputs. The overall structure is comparable to a Software-in-the-Loop (SIL) setup, the perception module represents the device under test. However, our work is not limited to excavators and can be transferred to other machines that are to be operated autonomously in unstructured environments such as construction sites or landfills. For example, our work can also be applied to the development and testing of autonomous dozers, dumpers, tractors, or wheeled loaders.

The provision of synthetic data by means of simulation requires the use of sensor simulation tools on the one hand. On the other hand, it must be determined what exactly is to be simulated. For this purpose, requirements must be analyzed and the operational environment formally modeled. Figure 2 shows the proposed architecture for focused generation and utilization of synthetic data for the development and testing of an autonomous excavator.

The proposed training and testing loop shown in Figure 2 enables scenario-based testing for autonomous systems. As discussed, this requires a scalable and repeatable development and testing process that minimizes manual effort. The goal of our approach is to achieve the highest possible level of automation with regard to system development, i.e. the highest possible level of automation in the processing of requirements and the generation and use of synthetic data. To achieve this, it is necessary to automatically infer scenarios from requirements, then derive test cases from scenarios and generate synthetic data, and after that use this data to develop and test the autonomy stack. The test results in turn show the strengths and weaknesses. We use the term closed loop as the analysis of the test results then drives the next iteration cycle, i.e. the generation of synthetic data focused on the actual weaknesses as well as the optimization of requirements.

To avoid ambiguity, the terminologies and their relations used in the context of the proposed scenario-based testing method are shown in Figure 3. Use cases provide a high-level view of the system's desired behavior. A scenario, which is a specific instance of a use case, details a particular set of events and conditions that occur in the system's environment. A test case is then defined as a concrete scenario. A test case consists of a scene, measurable Key-Performance-Indicator (KPI), and suitable pass/fail evaluation criteria. A scene is a concrete description of what is to be simulated, i.e. all scenario parameters are set to a fixed value and are documented in the scene description. KPIs can be measured in a scene, for instance, the Euclidean distance between an autonomous system and obstacles. Pass/Fail criteria are linked to requirements and they enable the identification of problematic test cases based on the KPIs. A test case can be executed directly as a simulation, whereas a scenario comprises a set of test cases. Requirements state how the autonomous system should behave and are validated by test cases. Since test cases are derived from a scenario, requirements can also be linked to scenarios.

3.1 Case study barrel detection

Accurate perception of the environment is particularly difficult if not all types of appearing objects and situations are known beforehand. As a case study, this paper considers the operation of the autonomous excavator shown in Figure 1

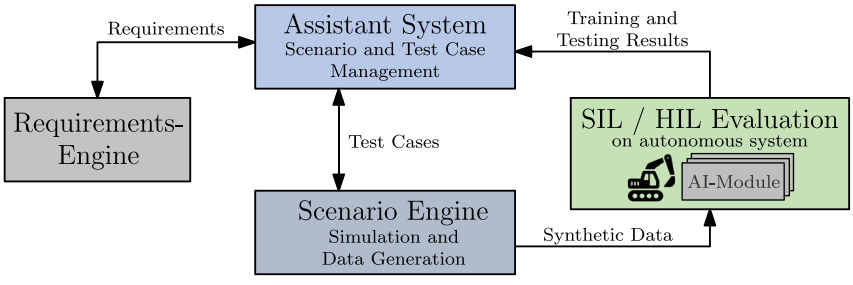


Figure 2: Architecture for generation and utilization of synthetic data for the development and testing of an autonomous excavator.

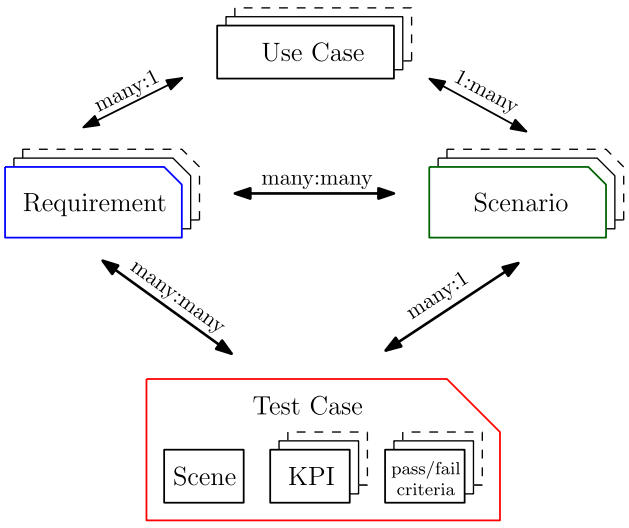


Figure 3: Overview of the terminology used in this work and their connection.

at a landfill site. Here, the autonomous excavator must be able to reliably detect barrels (see also Figure 8). Barrels used for waste disposal may differ in size, shape, and color. In addition, barrels are exposed to weathering in a landfill. The landfill environment can be classified as a special off-road environment, and weather effects in particular have an impact on the reliable detection of barrels. We use the example of "barrel detection" in the following sections to derive requirements and explain how to programmatically generate "missing" data.

3.2 Requirements engine and assistant system – identifying requirements and scenarios

Due to the lack of real data in the domain of autonomous systems in off-road environments, domain knowledge should be first identified by experts. Through interviews with domain experts, relevant situations and environmental parameters and their properties are identified in the context of offroad domain. This domain knowledge is then formalized as functional requirements to elicit the expected behavior for the autonomous system and provide a basis for scenario-based testing. Safety requirements for perception systems are also identified based on ISO 17757 (safety standard for heavy machinery) and in compliance with SOTIF methods to specify risks and hazards. Considering our ongoing example, the availability of different types of barrels and the typical handling of situations with barrels involved would first be assessed by an expert. In the architecture shown in Figure 2, these tasks are allocated to the "Requirements Engine".

Following on that, requirements are analyzed and clustered to further derive base scenarios, create the parameter space, and identify the evaluation criteria. This ensures traceability and later validates that the requirements are tested and fulfilled. The base scenarios also serve as a starting point for identifying corner cases in later iterations. During the later explained result analysis, identified critical parameter combinations can be used to specifically address corner cases. The base scenarios are then used to create a database for the logical scenarios by adding the relevant parameters and their possible values and correlations between some of these ranges.

To generate the test cases, the concrete scenarios are first generated by varying different combinations of the parameters' values. The combinations can be generated manually or automatically using the equivalence partitioning technique. This approach reduces the number of test cases by splitting the parameter space into equivalence classes and then choosing a representative value for each class. As there are currently no available standards to describe scenarios in an offroad context, ISO 21448 was used as a reference to create a description that fits the testing methodology. Following this standard, the scenario is detailed using scenes where each scene represents a snapshot of the environment consisting of the operation area (e.g. slope, pit), environmental conditions, static elements, and dynamic elements. A suitable evaluation metric is then added to the scenarios to form the test cases descriptions. The test cases are then converted into simulation-ready data for synthetic data generation. In the architecture shown in Figure 2, these tasks are allocated to the "Assistant System".

3.3 Scenario engine and autonomy stack – synthetic data generation for perception systems

The "Scenario Engine" shown in Figure 2 is responsible for the actual generation of synthetic data. For this purpose, the test cases created in the Assistance System must be simulated. To do this, the operational environment of the autonomous excavator must be modeled accordingly in a suitable sensor simulation tool that supports high fidelity of the generated sensor data when compared with the real sensor output. The test cases then must be executed and the generated synthetic sensor data must be extracted.

A detailed test case description serves as the main input for the Scenario Engine. The Scenario Engine constructs all

inputs required by the sensor and the environment simulation tool, such as a model of the environment and a model of the simulated sensor. Usually, the same model of the simulated sensor is used in all test cases since the sensor is an integral part of the autonomous excavator. But the model of the environment, such as the position of barrels, the landscape, the weather conditions, etc. changes with each test case. As the construction of all inputs required by the sensor simulation tool as well as the simulation itself comprises several steps, we refer to this process as the "Data Generation Pipeline".

When designing the Data Generation Pipeline, we focus on the following challenges:

1. How to produce simulation environments with high accuracy (e.g. realistic landfill environment)?
2. How to automate the process of data generation?

To achieve a minimal so-called "Reality-Gap" related problems during the development of the autonomous excavator, the simulated environments need to be as realistic as possible [26]. Four main aspects influence the realism of the data:

- Accurate sensor representations
- Detailed and high-resolution assets
- Photo-realistic rendering capabilities from a simulator
- Assessment of assets and environment models by domain experts

Accurate sensor representations in a simulated system are important to later allow for inference on the real system. Especially deep learning models are sensitive to different sensor placements or configurations. Even though there exist approaches to adapt between these configurations, it must be ensured that the actual configuration can be used. Therefore we directly interface the simulator with the autonomous system to ensure up-to-date placements and calibration. This way, sensor properties may be modeled as close to reality as possible. For perception, we mainly focus on camera and Lidar sensors in this work. Accurate and extrinsic calibration of the sensors is only one aspect. It is also important to model the sensor functionality after their real-world counterparts. For cameras, this is highly related to the third aspect of photo-realistic rendering as the image is directly read from the virtual view frustum. Considering Lidar functionality, we use a raycasting-based scheme to model the functionality similar to the one of a mechanical sensor. Even though raycasting is typically slightly more expensive than reading values from a depth buffer, it allows to add noise to the sensor and makes it sensitive to refractive indices of objects in the scene. Photo-realistic rendering capabilities are mainly defined by the choice of the simulation engine and will be discussed in Section 4. In addition, the best models, equipment, and tools are useless if the environment models are unrealistic. To counteract this, the environment models and assets must therefore be assessed by experts for their degree of realism, at least on a sample basis. For assessment by experts, it is important to note that the simulation does not provide a holistic representation of the scene. Instead, the focus is on the fields of view of the camera and Lidar sensors used. Therefore, for example, a realistic representation of the landscape located at a greater distance is not required and therefore not simulated, as this is beyond the field of view of the sensors.

To rapidly simulate newly defined test cases, the Data Generation Pipeline must process as automated as possible. Therefore, it is necessary to decouple the instances directly related to the device-under-test from the environment instances. In the case study, the sensors are strongly related to the perception module of the autonomous excavator and are decoupled from environmental aspects such as for example vegetation or weather conditions. This allows us to easily reuse models, e.g. for each test case, the same sensor models are used. Adaptability is another feature which is achieved using automation patterns. Already existing simulated and evaluated scenes should be adaptable to reevaluate the same scene with certain modifications. Considering the barrel example again, we want to be able to only change the appearance of the barrel instance without the need of recreating the whole scene with only a small modification to the translation, rotation, or color. We implement this by introducing special *adaption points* in our formal scenario description. This is a requirement to allow the Scenario Engine to iteratively change minor aspects of a single scene and evaluate the difference. This reduces the effort to generate customized scenes and also constitutes an important step towards explainability.

3.4 Closing the loop – feedback loop for continuous testing

To ensure continuous development, a feedback loop is necessary to analyze the results from the training and testing data and to identify the test cases that lead to unusual or unexpected system behavior. Artificial intelligence algorithms are used to accomplish this automatically. The first step to achieving that is to identify pass/fail criteria for test cases. These pass/fail criteria are based on Key Performance Indicators (KPIs), which can be measured within a test case or over several test cases. In the context of the perception

module, various KPIs are used to analyze the system performance such as accuracy, recall, precision, detection rate, false negatives, and false positives. These KPIs are used as guidance for tracking the source of the problem. For example, if the false negatives rate is high, this can be an indication of inadequate training or low visibility of the object that can be caused by several factors (e.g. weather obscurants, material deterioration, etc.). A high rate of false positives can indicate anomalies in the real-world map that are tricking the classifier.

Following on that, different approaches can be used to identify certain test case conditions or a combination of multiple conditions that lead the system to not or barely meet the pass/fail criteria of a test case. In our defined example of barrel identification, several scenario conditions that influence the perception model performance can be identified:

Barrel variants: As barrels come in different shapes, sizes, colors, and materials, it can be hard to train for all combinations. Moreover, material rustiness and damages (e.g. dents) in the barrel should be considered.

Barrel visibility: Due to the operating environment of excavators, barrels can be partially hidden by being buried in the ground or in a heap of earth or covered with sand or mud.

Other obscurants: Multiple factors such as rain, fog, dust, smoke, or poor lighting conditions can obscure the barrels.

The correlation analysis between the influential factors defined above and the KPI-based performance analysis of the detection model can for instance be tackled using decision trees. However, this is not part of our pipeline yet but should be considered for future work. To later allow this kind of performance analysis, we need to map the varied parameters of the concrete scenarios (input) to the results from the KPIs (output), which eventually constitutes the loop closure in Figure 2.

4 Realization and evaluation based on an autonomous excavator

The architecture shown in Figure 2 is comprised of the parts Requirements Engine, Assistant System, Scenario Engine, and SIL/HIL evaluation with the system under test (SUT). To evaluate the proposed approach for generating and using synthetic data for the development and testing of an autonomous excavator, the individual parts of the architecture are realized. This section is dedicated to the discussion of the realization details. First, we give an overview on the reference platform utilized for the implementation and experiments.

4.1 Platform description: autonomous excavator

The underlying case study for this publication is the development and testing of the perception module of an autonomous excavator. The autonomous excavator operates in a landfill environment and must be able to reliably detect barrels. The excavator under consideration is a Liebherr R924 24-ton excavator, which is modified with sensors and a control system for autonomous operation. The control system is based on the Robotic Operating System (ROS) [27]. Environment perception of the excavator is provided by a total of four 3D Lidar sensors and a multispectral stereo camera system. Figure 1 shows the real and the modeled excavator with the sensor positions of laser scanners and cameras highlighted. All sensors are calibrated in order to produce a multimodal environment representation as input for the autonomy functions. In this evaluation, we focus on autonomy capabilities in the context of environment perception. Ultimately, perception is the essential basic building block for more advanced autonomy skills such as navigation and task planning.

The excavator exists both in reality and as a virtual model. For the evaluation of the presented approach for generation and utilization of synthetic data, the control system, more precisely its perception module, is trained and tested using synthetic data.

4.2 Requirements engine and assistant system

To manage requirements and generate test cases, a user interface was created using Django and PostgreSQL. The user interface is responsible for defining the requirements and functional scenarios and generating the test cases. Multiple functional and safety requirements for the autonomous excavator are identified and the scenario parameters possible in the simulation framework are derived. The parameters, their varying attributes, and dependencies are then modeled in the database to form a parameter space. To generate the test cases, a uniform distribution is assumed for the varying attributes, and the concrete values are then randomly generated for each test case. The test cases are generated in JSON format and then transferred to the Scenario Engine for simulation and

synthetic data generation. The Assistant System is discussed in more detail in our previous paper [28].

4.3 Scenario engine – realization of the data generation pipeline and utilizing the data in autonomy stack

The detailed information flow of the implemented Data Generation Pipeline is depicted in Figure 4. We first describe the Scenario Engine with its data generation process as the main building block in further detail, before we evaluate the resulting data qualitatively and quantitatively.

4.3.1 Translate test cases to renderable simulations

The main task of the Scenario Engine is to convert the passed test case descriptions of the "Assistant System" into renderable scenes. Essentially, we parse a JSON file containing the structured test case scene description and put it into a format that can be interpreted by our environment simulation tool. For this format, we use the Universal Scene Description (USD), published by Pixar. USD is an open-source format explicitly designed for the description of three-dimensional assets environments. It was designed to be an interchange format and is therefore supported by a variety of simulation as well as design tools, which are eligible for asset creation and simulation-based testing. This procedure provides major advantages compared to creating proprietary assets for a specific simulation tool. It allows us to swap the simulation tool without losing compatibility with the elaborately generated assets. In addition, it does not require direct interaction with a simulation tool to adapt a test case scene it can be done using USD as a proxy. The conversion of a test case description into a simulator readable USD format is performed requires two preparation steps:

Preparation step 1 – Creation of 3D assets as "atomic" components. First, all assets that will be used for scene generation must be created. In this work, all assets were either created manually or bought from exchange platforms. As USD is gaining more popularity, a future extension of the current asset base using tools such as Autodesk Maya and Blender is straightforward. To increase usability, we utilize variant sets for similar assets which allow bundling multiple possible appearances of an asset into a single instance. That way, we are able to create different sets for color and texture. The underlying USD implementation allows to fuse these variant sets to obtain desired combinations. Figure 5 shows an example of such a variant set, where a single barrel instance appears in different shapes and colors. Since the 3D assets are self-contained and will only be referenced

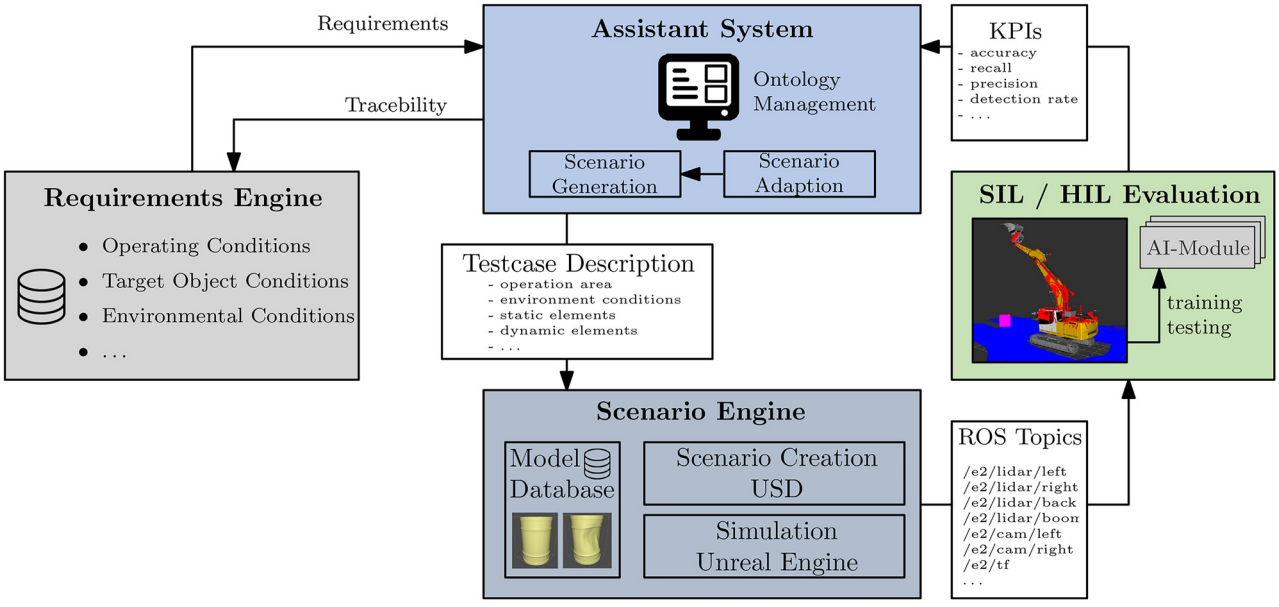


Figure 4: Overview on the dataflow in the prototyped system. The *Requirements Engine* maintains all user-defined requirements for a specific task and platform. The *Assistant System* serves as user interface and allows to configure specific test scenarios. Once a scenario is created, the involved *Testcases* are forwarded to the *Scenario Engine* which is the entrypoint for the simulation. The unreal based simulation is interlinked with the autonomy stack of the excavator and thus provides the generated data for immediate evaluation.

Figure 5: Variant set of a USD barrel instance. Different appearances can be produced by simply adjusting different properties of the barrel.

and parameterized within the scene, this implements the core aspect of reusability.

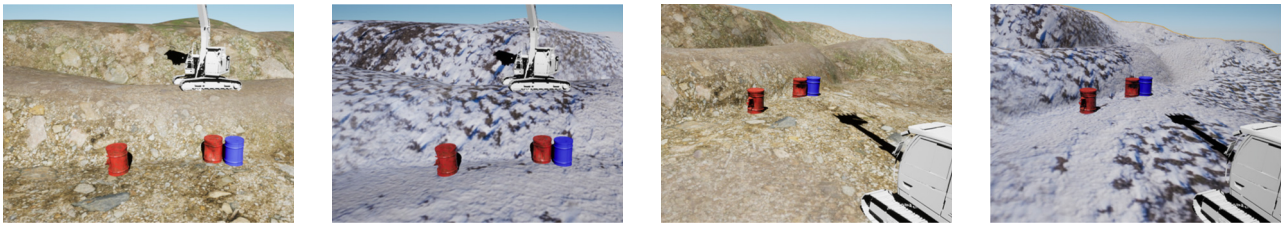
Preparation step 2 – Asset and Scene Management: Assets are maintained by using a dedicated filesystem structure. This allows us to easily maintain and update available assets even without developer knowledge. In addition, validation and adaption using inbuilt USD tools are enabled by using a plain filesystem structure. A set of additional tools provide additional database-like access mechanisms. For instance, all currently available assets and their variant sets can be requested via API call. Scenes combine multiple assets and are likewise maintained as a single folder. By using the API, relevant information about a scene (e.g. number of assets, metadata) may be obtained.

The two preparatory steps only need to be performed if new assets are to be included in the creation of renderable scenes. With 3D assets available as atomic components and well managed in a dedicated filesystem, renderable simulations can be automatically derived from test case descriptions. Therefore, assets and environment information are combined to form a scene described in a USD file. We utilize a Python-based implementation for the automatic creation of USD files. As discussed, the input is the JSON-based abstract scene description of the Assistance System, and we first extract this information and propagate it into a USD file. We can do this by referencing assets in our model database, which fit the description. After the references are established, assets can be transformed by applying the transformations from the scene description to their reference-container. By adapting the container of the selected asset, existing scenes may be reused in future iterations. Since USD supports the layering of opinions, these adjustments can be performed non-destructively and with minimal memory footprint.

4.3.2 Generating synthetic training data for perception systems

As sensor and environment simulation tool, we utilize Unreal Engine Version 4 (UE4) with integrated USD support. This allows for a wide range of asset support on the one side and gains photorealistic and highly efficient rendering capabilities on the other side [29]. The main downside is the interfacing of UE4 from outside the engine, which is why we implemented our architecture based on containers with headless rendering support for UE4. This allows us to run the costly rendering as a separate component on dedicated hardware without interfering with the user interface or the SIL component of our ROS-based autonomy stack. The simulation process itself requires loading USD scenes from USD files before simulating them.

Step 1 – Loading of USD scenes: USD Scenes can be loaded programmatically at runtime without the need for rebuilding. That way, we can load new scenes by leveraging a simple sequential job control. This may be extended to parallel executions on multiple simulation containers in the future. We provide API interfaces to start, interrupt and stop the simulation procedure. Figure 6 shows some example scenes generated with the pipeline. All scenes have a typical landfill environment and contain barrels as the main detection target. However, minor parameter changes led to clearly visible effects in the adapted scenes Figure 6(b)–(d).



(a) Scene A **(b)** Scene B **(c)** Scene C **(d)** Scene D

Figure 6: Different scenes in unreal simulation. All parameters such as textures, colors and modes of different assets can be modified using the USD structure as single source of truth.

Step 2 – Simulating a scene: As the scene description is self-containing, the autonomous excavator's pose and state are also defined within the description. However, the functionality of the available sensors is not part of the scene description. Therefore, after successfully loading a scene, all sensors are placed and configured according to the status defined in the scene description. They can be easily attached to the respective joints within the USD asset. For the autonomous excavator, the sensors are placed in the same locations as mounted on the real excavator (see highlighted sensor positions in Figure 1). We use two sensors with ground truth generation support:

Camera: We use UE4's SceneCapture2D component to implement a stereo camera set. The camera wrapper provide specific camera calibration. For each camera, we add a separate ground truth camera, which uses a different render pass, where vertex colors instead of textures are captured. In a previous step, all assets in the scene are colored using a unique mapping, which is global and therefore persistent in all scenes.

Lidar: We implement a raycasting Lidar using UE4's enhanced raytracing techniques. All of its parameters such as horizontal and vertical angles, length, frequency, noise, etc. are configurable. This allows to model it as close to the real Lidar as possible. The target object is determined by checking simplified hit boxes, which correspond to USD assets in the scene.

Most perception systems work either based on cameras or based on Lidar or similar pointcloud-generating sensors. As we primarily target the perception of unknown objects, we focus on these two sensor types in this work.

4.3.3 Data extraction – utilizing the data in autonomy stack

We directly interface the simulation with the Robot Operating System (ROS) [27], which is based on a publisher/subscriber architecture. This allows utilization of generated data immediately for training and testing purposes. To accomplish this, we extend the ROSIntegration plugin[1] for UE4 to transfer all sensor and status information. As the interface of ROSIntegration transfers the data to a websocket-based ROS bridge, simulation, and autonomy stack of the excavator may reside on different hardware platforms. While this allows to conveniently scale the simulation, it comes with the downside of costly data transfer rates. However, we found the transfer rates to be sufficient for experiments when using BSON serialization format. The transfer rates of sensor data are shown in Table 1.

Table 1: Framerate and throughput measurements for the proposed simulation pipeline.

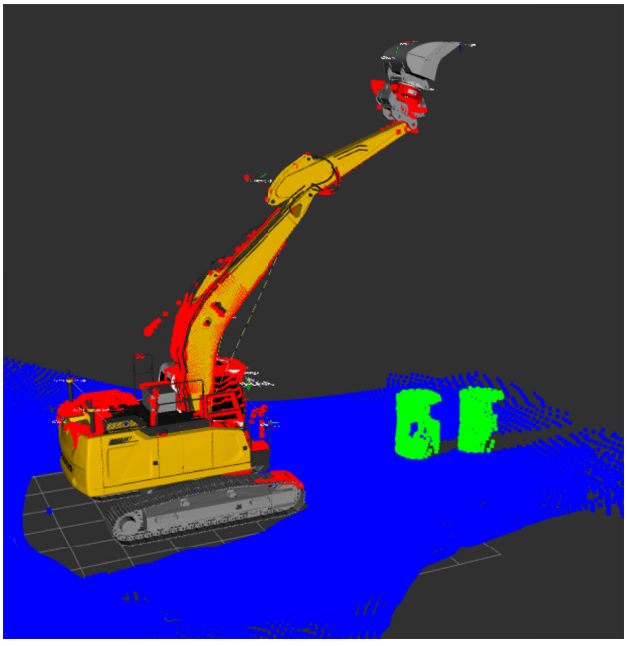
	PCL segmented	Images	Images segmented
Unreal			
Frames/s	7.86	20.0	20.0
ROS			
Frames/s	7.86	4.94	4.76
Throughput/mb/s	2.74	7.42	2.87

We define one topic for each sensor modality and map it to the respective topics of the ROS-based autonomy stack. This way, we can use the synthetic data in the same way as real sensors would provide data. Sensor outputs can be visualized in real-time using common tools such as Rviz, an example of the 3D view and some segmented barrels is provided in Figure 7.

The extracted data is then used to train and test deep learning models of the perception module. This module is part of the Autonomy Stack of the excavator. The task investigated in this publication is the reliable detection of barrels on the basis of camera and Lidar data. For this purpose, the perception module must use a bounding box to mark the barrels on the camera images and also use the Lidar data to determine the relative position of the detected barrels. Humans usually find it easy to recognize barrels, no matter what color they are, how damaged they are, or whether they are partially covered by soil or partially buried. With the help of the synthetic data, the automated detection of barrels is improved. The effort to create the synthetic data with the help of the presented training and testing loop is significantly lower than to create corresponding data sets with real barrels in real operational environments in all possible variations. For the evaluation of our approach, scenarios with different colored barrels were tested with the real excavator, as shown in Figure 8.

The performance of synthetically trained deep learning models on real images is often negatively impacted by the reality gap. One possible way to cope with it is to continue training with a small amount of real data. We qualitatively evaluate the performance of an object detection model trained on different combinations of real and synthetic data and tested on different kinds of barrels in a real scenario. A central component of the ROS-based detection node, we use

1 https://github.com/code-iai/ROSIntegrationVision.



(a) Segmented PCL

(b) Camera Image Left

(c) Segmented Image Left

Figure 7: The visualization shows the sensor perspective of the excavator within the ROS-based autonomy stack. Sensor data coming from the simulation can be inspected in real time. Figure (a) shows the fused point cloud with the segmented barrels, while (b) and (c) show the RGB and segmented camera views from the left camera of the stereo set.

a MaskRCNN [30] implementation with Resnet50 backbone. We initialize the weights from the COCO dataset [31] and continue to train the network on a set of real images of red barrels. As we can observe from the qualitative evaluation in Figure 8, the detection of barrels in another color is not working robustly, as the network does not generalize well on different kinds of the new class. However, when we extend the train data with a set of images explicitly generated by the scenario engine, barrels of different shapes and colors can be detected in a more robust way with improved prediction scores.

4.4 Feedback loop – guided scene creation and optimization of requirements

The results obtained on the basis of the synthetic data for the detection of barrels show the strengths and weaknesses of the perception module of the autonomy stack of the excavator. To close the proposed training and testing loop, the results of previous iterations of training and testing are analyzed and used to plan the next iteration.

For this purpose, the KPIs achieved in the test cases are fed back from the autonomy stack to the Assistance System. There, values are then determined for the pass/fail criteria of the test cases. Based on decision trees, the system then searches for commonalities in failed test cases. If problematic scene parameters can be identified, the next run of the training and testing loop focuses on the problematic scene parameters and examines them in greater detail. If problematic requirements are identified, they are highlighted in the requirements engine and proposed for review. These functionalities are currently undergoing evaluation and are part of the Assistance System implemented using the Django framework.

4.5 Evaluation

The evaluation of the presented approach is coupled with the creation and use of synthetic data based on the discussed case study of the development and testing of the perception module of an autonomous excavator with the task of barrel detection.

For an efficient and transparent generation of test data, requirements are needed to describe the problem. Deriving test cases from requirements is supported via automatic identification of requirements, that are not covered yet by a single test case as well as automated variation of scene parameters. Finally, the test case description is automatically exported as JSON file that serves as input to the Data Generation Pipeline. Here, the scene descriptions are transferred into simulatable scenes using Python and USD. The USD files can then be simulated in the sensor and environment simulator, in our case the Unreal Engine 4. Sensor data for training and testing of the perception module is then extracted using a ROS-interface. Instead of a real camera and Lidar data, the Autonomy Stack of the excavator is stimulated with synthetic data of a simulated camera and Lidar sensor that sense a fully simulated environment.

For evaluation purposes, the barrel detection node of the Autonomy Stack has been evaluated as shown in Figure 8. Therefore, various test cases had been performed at Fraunhofer IOSB at the competence center ROBDEKON.

In terms of efficiency, the generation of synthetic data is superior to the generation of real data, in case there is no available database that can be used. This entire process of the data generation pipeline takes less than a second on our synthetic data generation platform (Ryzen 5800X processor,

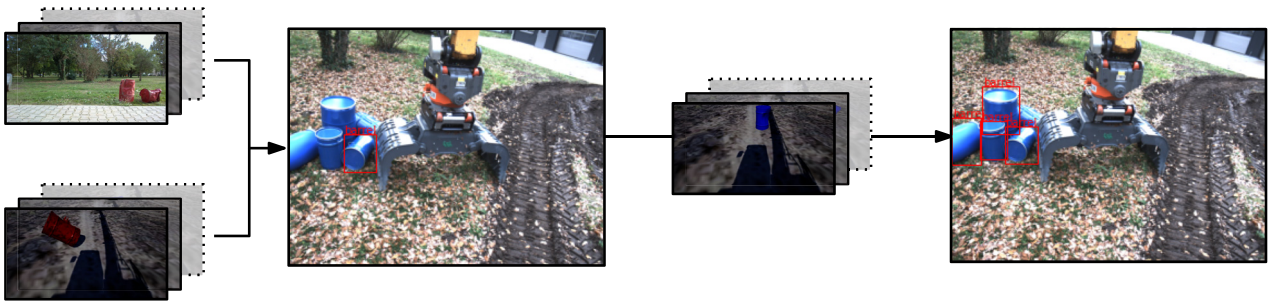


Figure 8: Evaluation of the barrel detection node before and after training with additional synthetic images of blue barrels. The average score on successfully detected barrels improved from 0.81 on the barrel on the left side to an average score of 0.94 on the right test image.

Nvidia 3090 RTX graphics card, 64 GB Ram). The preparation of a new scene with the real excavator and real assets took about 5–30 min. Excavators are already special construction vehicles and excavators equipped with cameras and Lidar sensors are even more special. Thus it is not possible to fall back on image data that has already been collected, as there is simply no comparable data available.

In terms of barrel detection performance, the usage of synthetic data increased performance for the test cases performed with the real excavator of 13 percent averaged on all performed test cases with the real excavator. For this purpose, the test cases carried out with the real excavator were each performed twice. First, the perception module of the excavator had not received any training with synthetic barrels and then showed a recognition probability in the range of 80 percent. Training with 50 synthetically generated data sets increased this to a recognition probability of up to 94 percent.

Overall, this demonstrated that synthetic data was successfully used to improve an autonomous excavator. For this purpose, an approach shown for continuous development, the training and testing loop shown in Figure 4, was used. This makes it possible to react quickly to new requirements and scenarios. Scalability is ensured by an implementation based on containers. Currently, the complete closing of the loop is being worked on, and KPIs are already being fed back for analysis.

5 Conclusion and outlook

Generating and using synthetic data to develop and test an autonomous excavator requires a systematic and sophisticated methodology. As discussed in this publication, this results in a high initial complexity when all components have to be aligned with each other. In (lifecycle-long) use, the following advantages arise:

- Data can be generated in a structured and transparent manner.
- Any scenarios can be efficiently implemented for data generation.
- All parameters are adjustable, thus critical scenarios can be formed and tested in a prioritized manner.
- Direct evaluation capabilities using a ROS-based interface to the autonomy stack of a real autonomous system.

The proposed training and testing loop supports the continuous development of the autonomous excavator used in our experiments. With the emergence of complex autonomous systems, such new development methods become necessary. The use of synthetic data enables the focused and traceable development of autonomous and self-learning systems.

As shown in our experimental evaluation, synthetic data can be successfully used to augment data sets, which are lacking specific scenarios inside a use case. With the proposed architecture, we are able to adjust specific parameters to provide the needed data to improve the datasets and reduce gaps inside the datasets.

The next steps in the further development of the training and testing loop is the complete “loop-closure”. This will enable automated refinement of the synthetic data generation with each run. Furthermore, we plan to use real data as scenes for test cases to reduce the reality gap. In future work, we also plan to investigate new models such as pose detectors and pointcloud-based segmentation models to further explore the capabilities of the data generator.

Research ethics: Not applicable.
Author contributions: The authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Competing interests: The authors state no conflict of interest.
Research funding: This research has been funded by the Federal Ministry of Education and Research of Germany in the framework of "SynDAB", grant number 01IS21057A. Responsibility for content lies with the authors.
Data availability: The raw data can be obtained on request from the corresponding author.

References

[1] C. Ebert and R. Ray, "Test-driven requirements engineering," *IEEE Softw.*, vol. 38, no. 1, pp. 16—24, 2021.
[2] C. Ebert, M. Weyrich, B. Lindemann, and S. Chandrasekar, "Systematic testing for autonomous driving," *ATZ Electron Worldw*, vol. 16, no. 3, pp. 18—23, 2021.
[3] C. Ebert and J. John, "Practical cybersecurity with iso 21434," *ATZ Electron Worldw*, vol. 17, pp. 3—4, 2022.
[4] S. Garg, P. Pundir, G. Rathee, P. Gupta, S. Garg, and S. Ahlawat, "On continuous integration/continuous delivery for automated deployment of machine learning models using mlops," in *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Laguna Hills, CA, USA, 2021, pp. 25—28.
[5] H. Vietz, T. Rauch, and M. Weyrich, "Synthetic training data generation for convolutional neural networks in vision applications," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.
[6] A. Löcklin, M. Müller, T. Jung, N. Jazdi, D. White, and M. Weyrich, "Digital twin for verification and validation of industrial automation systems — a survey," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Vienna, Austria, 2020, pp. 851—858.
[7] H. Vietz, T. Rauch, A. Löcklin, N. Jazdi, and M. Weyrich, "A methodology to identify cognition gaps in visual recognition applications based on convolutional neural networks," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE),* Lyon, France, 2021, pp. 2045—2050.
[8] M. Müller, T. Müller, B. Talkhestani, P. Marks, N. Jazdi, and M. Weyrich, "Industrial autonomous systems: a survey on definitions, characteristics and abilities," *Automatisierungstechnik*, vol. 69, no. 1, pp. 3—13, 2021.
[9] T. Emter, C. Frese, A. Zube, and J. Petereit, "Algorithm toolbox for autonomous mobile robotic systems," *ATZ offhighw worldw*, vol. 10, no. 3, pp. 48—53, 2017.
[10] J. Petereit, J. Beyerer, T. Asfour, et al., "ROBDEKON: robotic systems for decontamination in hazardous environments," in *IEEE SSRR*, 2019.
[11] C. Ebert, D. Bajaj, and M. Weyrich, "Testing software systems," *IEEE Softw.*, vol. 39, no. 4, pp. 8—17, 2022.
[12] D. J. Fremont, E. Kim, Y. V. Pant, et al., "Formal scenario-based testing of autonomous vehicles: from simulation to the real world," 2020 [Online]. Available at: https://arxiv.org/abs/2003.07739.
[13] J. Mazzega and H.-P. Schöener, "Wie PEGASUS die Lücke im Bereich Testen und Freigabe von automatisierten Fahrzeugen schließt," in *Methodenentwicklung für Aktive Sicherheit und Automatisiertes Fahren*, vol. 144, 2016, pp. 163—176.
[14] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "RELLIS-3D dataset: data, benchmarks and analysis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1110—1116.
[15] P. Jiang and S. Saripalli, "LiDARNet: a boundary-aware domain adaptation model for point cloud semantic segmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2457—2464.
[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the KITTI dataset," *Int. J. Robot Res.*, vol. 32, no. 11, pp. 1231—1237, 2013.
[17] H. Vietz, A. Löcklin, H. Ben Haj Ammar, and M. Weyrich, "Deep learning-based 5g indoor positioning in a manufacturing environment," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), September 2022*, 2022.
[18] H. Achicanoy, D. Chaves, and M. Trujillo, "Stylegans and transfer learning for generating synthetic images in industrial applications," *Symmetry*, vol. 13, no. 8, p. 1497, 2021.
[19] J. Fang, X. Zuo, D. Zhou, S. Jin, S. Wang, and L. Zhang, "Lidar-aug: a general rendering-based augmentation framework for 3d object detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4708—4718.
[20] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.
[21] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1301—1310.
[22] M. Z. Wong, K. Kunii, M. Baylis, W. H. Ong, P. Kroupa, and S. Koller, "Synthetic dataset generation for object-to-model deep learning in industrial applications," *PeerJ. Comput. Sci.*, vol. 5, p. e222, 2019.
[23] C. Mayershofer, T. Ge, and J. Fottner, "Towards fully-synthetic training for industrial applications," in *LISS 2020: Proceedings of the 10th International Conference on Logistics, Informatics and Service Sciences*, Springer, 2021, pp. 765—782.
[24] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234—3243.
[25] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: can virtual worlds replace human-generated annotations for real world tasks?" *arXiv preprint arXiv:1610.01983*, 2016.
[26] F. Reway, A. Hoffmann, D. Wachtel, W. Huber, A. Knoll, and E. Ribeiro, "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)*, 2020, pp. 1249—1256.

[27] Stanford Artificial Intelligence Laboratory, et al., "Robotic operating system — ROS Melodic Morenia," 2018 [Online]. Available at: https://www.ros.org.
[28] I. Sonji, H. Vietz, C. Ebert, and M. Weyrich, "An approach to automatically generate test cases for AI-based autonomous heavy machinery," in *9. AutoTest Fachkonferenz*, 2022 [Online]. Available at: https://www.researchgate.net/publication/363536300_An_approach_to_automatically_generate_test_cases_for_AI-based_autonomous_heavy_machinery.
[29] B. Alvey, D. T. Anderson, A. Buck, M. Deardorff, G. Scott, and J. M. Keller, "Simulated photorealistic deep learning framework and workflows to accelerate computer vision and unmanned aerial vehicle research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021, pp. 3889—3898.
[30] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980—2988.
[31] T.-Y. Lin, M. Maire, S. Belongie, et al., "Microsoft COCO: common objects in context," in *Computer Vision — ECCV 2014*, vol. 869, 2014, pp. 740—755.

Bionotes

Alexander Schuster
University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany
alexander.schuster@ias.uni-stuttgart.de

Alexander Schuster studied electrical engineering and information technology at University of Stuttgartand. Currently, he is a member of the IAS academic staff. In the publicly funded project SynDAB, Alexander Schuster is the Scenario Engine expert.

Raphael Hagmanns
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany
raphael.hagmanns@iosb.fraunhofer.de

Raphael Hagmanns studied computer science at Karlsruhe Institute of Technology (KIT). Currently, his research within the "Vision and Fusion Laboratory" at KIT focuses on perception and mapping systems for different kinds of autonomous robots. He is strongly involved in the research group of Autonomous Robotic Systems at Fraunhofer IOSB.

Iman Sonji
University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany
iman.sonji@ias.uni-stuttgart.de

Iman Sonji studied electrical engineering at the University of Stuttgart. Currently, she is a member of the IAS academic staff. In the publicly funded project SynDAB, Iman Sonji is the Requirements Engine and Assistant System expert.

Andreas Löcklin
University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany
andreas.loecklin@ias.uni-stuttgart.de

Andreas Löcklin studied electrical engineering and information technology at University of Stuttgart. Currently, he is a PhD student supervised by Professor Michael Weyrich and Dr. Nasser Jazdi. As team spokesmen, he is leading the team complexity handling in automation technology. Andreas Löcklin is the project manager of the publicly funded project SynDAB.

Janko Petereit
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany
janko.petereit@iosb.fraunhofer.de

Janko Petereit received his degree in electrical engineering and information technology from the Karlsruhe Institute of Technology (KIT) in 2009, where he also received his PhD in informatics in 2016. He now manages the Multi-Sensor Systems research group at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB in Karlsruhe, Germany.

Christof Ebert
Vector Consulting Services GmbH, Stuttgart, Germany; and Robo-Test, Stuttgart, Germany
christof.ebert@vector.com

Christof Ebert is the managing director of Vector Consulting Services in Stuttgart, Germany. He holds a PhD from University of Stuttgart, is a Senior Member of the IEEE and a professor at University of Stuttgart. He is co-founder of Robo-Test, an incubator with focus on validating autonomous systems.

Professor Weyrich
University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Stuttgart, Germany
michael.weyrich@ias.uni-stuttgart.de

Professor Weyrich is head of the Institute for Automation Technology and Software Engineering at the University of Stuttgart. He is the chairman of the board and advisory board of the VDI/VDE Society for Measurement and Automation Technology and heads the Technical Committee on testing networked systems.