Requirements for AI-technology for DT-based AI-training and for system optimization/configuration

[WP03; T3.1; Deliverable: D3.1; Version 2.1 M32 (Update to Version M6)]

AI training using Simulated Instruments for Machine Optimization and Verification

Version	Status	Date	Page
2.0 M32	Internal	2023.11.21	1/32

Document Information

Project	ASIMOV
Grant Agreement No.	20216 ASIMOV - ITEA
Deliverable No.	D3.1
Deliverable No. in WP	WP03; T3.1
Deliverable Title	Requirements for AI-technology for DT-based AI-training and for system optimisation/configuration
Dissemination Level	Public
Document Version	2.1 M32
Date	2023.11.21
Contact	Ezra Tampubolon
Organization	NorCom
E-Mail	ezra.tampubolon@norcom.de

The ASIMOV-project was submitted in the Eureka Cluster AI Call 2021
https://eureka-clusters-ai.eu/

Task Team (Contributors to this deliverable)

Name	Partner	E-Mail
Ilona Armengol	TNO	ilona.armengolthijs@tno.nl
Niklas Braun	AVL	Niklas.Braun@avl.com
Ville Lämsä	VTT	Ville.S.Lamsa@vtt.fi
Duarte Guerreiro Tomé Antunes	TU/e	d.antunes@tue.nl
Maurits Diephuis	TFS	maurits.diephuis@thermofisher.com
Markus Kelanti	Univ. Oulu	markus.kelanti@oulu.fi
Sebastian Moritz	TrianGraphics	sebastian.moritz@triangraphics.de
Thomas Kotschenreuther	RA Consulting	t.kotschenreuther@rac.de
Ezra Tampubolon	NorCom	ezra.tampubolon@norcom.de
Mustafa Majar	TrianGraphics	Mustafa.Majar@triangraphics.de
Andreas Eich	LiangDao	andreas.eich@liangdao.de

Formal Reviewers

Version	Date	Reviewer
0.2,0.6,0.8	2021.11.23	Faruk Caglar (TFS)
0.8	2021.12.09	Stephan Kussmaul (TrianGraphics), Sebastian Moritz (TrianGraphics)
0.8	2021.12.14	Elias Modrakowski (OFFIS), Eike Moehlmann (OFFIS), Tabea Henning(OFFIS), Michael Wild (OFFIS), Mehrnoush Hajnorouzi (OFFIS)
0.8	2021.12.16	Maurice Hemels (TU/e)
1.0	2022.01.07	Remco Schoenmakers (TFS), Jacco Wesselius (TNO)
2.0	2023.11.27	Elias Modrakowski (DLR) and Mehrnoush Hajnorouzi (DLR)

Change History

Version	Date	Reason for Change
0.0	2021.11.16	Ilona Armengol (TNO); Initial version drafted.
0.2	2021.11.22	Niklas Braun (AVL); Added information related to UUV use case.
0.6	2021.11.29	Ville Lamsa(VTT); Ville added an introduction on methodologies/frameworks to set up requirements which is more general and precedes the currently described situation based on discussions.

		Ilona Armengol (TNO) and Nikas Braun (AVL) added requirements in tabular form for oversight and clarity based on discussions and reworked feedback from contributors of deliverable, including the comments of Faruk Caglar (TFS).
0.8	2021.12.02	Maurits Diephuis (TFS) adds information on Reinforcement Learning and Ilona Armengol (TNO) reworks feedback from task participants.
1.0	2021.12.20	Ilona Armengol (TNO) reworks comments and additions from official reviewers.
1.2	2023.10.31	Ezra Tampubolon (NorCom) updates the documents based on the status of other deliverables. New sections are added: Data Pre- and Postprocessing and Requirements for Digital Twin and AI-Training
1.4	2023.11.13	Ezra Tampubolon (Norcom) updates the UUV Use Case part. Requirement table is updated based on the recent version of D1.3. Moreover, ET updates the infrastructure requirements for UUV Use Case
1.5	2023.11.17	Mustafa Majar (TrianGraphics) updates the UUV Use Case Part regarding to the environment simulation
1.8	2023.11.17	Andreas Eich (LiangDao) updates the UUV Use Case 2 Part.
2.1	2023.12.04	Ezra Tampubolon (NorCom) reworks comments and Feedback from official reviewer.

Abstract

This document proposes requirements for AI-technology for DT-based AI-training, and for DT-supported system optimization. The requirements cover general practical data collection methods and practices, and data quality measurement related aspects to ensure AI training will be done with valid input data. Extraction of physics-based thresholds for data validation and boundary conditions for optimization will be covered. The specified requirements are aligned with digital twinning solutions for WP2.

Table of Contents

1 Context and objective of the task 8
2 Data: Basic Requirements 8
3 Reinforcement Learning: basic components and requirements 11
4 Data: Pre- and Postprocessing 13
4.1 Preprocessing 13
4.2 Postprocessing 13
5 Digital-Twin Requirements for AI-Training 14
6 Specifics per use case 15
6.1 Use case STEM 15
6.2 Use case UUV 17
6.2.1 Scenario Generation for Digital Twin based UUV-Tests 17
6.2.2 Environment Interface for UUV 26
7 Conclusion and Future Work 30
8 Terms, Abbreviations and Definitions 30
9 Bibliography 32

Table of Figures

Figure 1: General conceptual view of elements in RL 11
Figure 2: Electron Microscope operator control panel view with buttons of interest. 16
Figure 3: Architecture of UUV Use Case 1 20
Figure 4: Logging Architecture 22
Figure 5: Data Management and ASIMOV Solution 23
Figure 6: LiDAR 3D Point cloud of an Autobahn scene. The scene was analyzed by a perception software. Identified objects have been marked with bounding boxes, which got annotated with additional information (class, e.g. car, and speed). 28
Figure 7: LiDAR Measurement of a Target plate in Carla (left) and overlayed with the measurement in the physical system. 28
Figure 8: Scheme of physical system part of the LiDAR experiment 29
Figure 9: Figure 6 Flow diagram of the LiDAR validation experiment 30

Table of Tables

Table 1 - Requirements use case STEM 15
Table 2 - Requirements use case UUV 1 16
Table 3 - Requirements use case UUV 2 17
Table 4 - Requirements use case STEM working progress 18
Table 5 - Terms, Abbreviations and Definitions 19

1 Context and objective of the task

This document proposes requirements for AI-technology for Digital Twin (DT) based AI-training, and for DT-supported system optimization, where DT stands for Digital Twin. The requirements cover general practical data collection methods and practices, and data quality measurement related aspects to ensure AI training will be done with valid input data. Extraction of physics-based thresholds for data validation and boundary conditions for optimization will be covered. The specified requirements are aligned with digital twinning solutions for WP2.

The documentation of requirements for AI-technology will not cover issues related to the essential technological methodologies to facilitate the data collection. Therefore, the topics of standardized AI-model descriptions such as the ones given with the Predictive Model Markup Language (PMML), standards for machine learning interoperability such as Open Neural Network Exchange (ONNX) or standards for statistical models and data transformation engines such as Portable Format for Analytics (PFA) will not be addressed. Moreover, possible technical issues inherited from messaging technologies such as the MQTT, the Advanced Message Queuing Protocol (AMQP), the Extensible Messaging and Presence Protocol (XMPP) or the Simple Text Oriented Messaging Protocol (STOMP) are not covered. Additionally, novel concepts of meta data related technologies such as Web of Things (WoT) Description, or development platform type of implementations such as FIWARE or EdgeXFoundry are not covered either. Anyhow, the DT-based AI-training will be executed eventually by following the standards and good practices as given e.g., in [1].

In Section 2 an overview is given of the basic general practicalities regarding the data collection without the detailed specification to the use cases or technological methodologies. A digital twin, as a surrogate system, can ensure extensive system-level trials, model tunings, and adaptations. With a digital twin, an infinite number of repetitions and scenarios can be executed in virtual environment effectively. In this context, the basic formulation of the reinforcement learning (RL) [2] is that the model learns from rewards when taking actions in a virtual environment. The couplings between digital twins and reinforcement learning are fully justified by the intrinsic nature of both methodologies. Thus, the considered training requirements are given for reinforcement learning, but can be generalized for other methodologies as well. The basic components and requirements for the reinforcement learning are given in Section 3.

Based on basic general requirements of the ASIMOV-solution, and included in it, the general RL-algorithm, we provide in Section 4 and Section 5 further general requirements on the data processing for the RL-solution and on the digital twin with focus on ASIMOV-solution.

The remaining of this document is devoted to the discussions on the requirements for the specific use-cases within the ASIMOVs project.

2 Data: Basic Requirements

Data collection is the first step into problem solving with RL after the goals are defined. Without valid data any effort in reaching the desired goal will be in vain. Even though a few iterations might be needed before understanding the data requirements for a specific problem, there are good practices that can be followed from the first iteration on. In this section the focus lays on those basic practices that any AI dataset needs to satisfy before taking up an implementation.

The nature of the requirement set for the data can have a diversity of aspects that will be covered.
But to be clear, there is no right or wrong data without a clear goal to be followed. The following data checks and definitions should later also be examined in the light of the goal, in all iterations followed for all goals.

These points are generated from experience in the field by members of the project:

1. <u>Assessment of the data that is already available, or definition of desired data if no prior dataset is available.</u>

2. Assessment of tools that can generate data and limitations.

   A list of tools and parties delivering the tools is highly recommended to understand data generation possibilities and capabilities available to generate data, as well as to understand the limitations. Standards to which the tools comply could be of added value.

3. ype of data available.

   Knowing the structure, form and shape of the data will limit or open possibilities in the next implementation steps, especially of the software components.

4. Examine data with domain knowledge and statistically

   Representativeness: Is the data covering the many aspects that are needed to find a solution (e.g., diversity of parameters, frequencies, diversity of errors)? Rule of thumb dictates, if all information is available that the human expert would need to solve the problem, representativeness is covered.
   There are no (software) tools available to cover this point. Definition of what representativeness means for each goal is needed. Domain experts are required to fill the definition.

   Volume: Any specific data-driven solution will need a minimum of data samples to be able to solve the problem. There is no golden rule, even though having in mind the data input, the complexity of the problem, the given state space and the solution are guides to determine minimum amounts. Some algorithms are much more data intensive than others. Reinforcement learning is known to be (extremely) sample inefficient, e. g, the current SOTA algorithm for Atari games requires 18 million frames or 83 hours of equivalent human play for training.

   Bias: Are some aspects overrepresented o underrepresented in the data? s there more granularity or detail in one or more aspects of the problem? These problems should be avoided at the input to avoid bias at the output.
   Many programming languages such as Python or R provide numerical computation and statistical libraries where numbers and figures can be created to be sure to avoid bias at the input.
   It is good practice to incorporate software which makes these basic checks to avoid problems at the solution.

   Completeness: Checking the data distributions at the different aspects of the problem, together with getting the other data aspects right will make the dataset complete.

   Basic Quality Control for Data: In the data acquisition process, it's crucial to be aware of potential systematic errors. Common examples of such errors include data duplication and data corruption. Data duplication can introduce bias into the creation of RL models, while data corruption can lead to errors in the learning process, often due to mismatches in data formats. To address these issues, several steps should be taken. First and foremost, it's essential to regularly inspect the data to identify any errors. Once identified, efforts should be made to pinpoint the sources of these errors and rectify them. If fixing the source is not feasible, basic operations can be employed to mitigate the error. For instance, duplicates in the data can be removed, and corrupted data can be replaced with default values such as the statistical mean or median. Alternatively, interpolation methods can be employed to estimate values based on neighboring data points, or input from informed users can be used to populate missing values. However, it's important to note that employing these techniques can potentially result in certain aspects of the data being overrepresented. Therefore, caution must be exercised when applying such methods to ensure the integrity of the data.

5. ccessibility and ownership of the data

   Who is responsible for the data? Who will allow access to the data?

6. hysical location of the data.

   Which company or entity is saving or hosting the data or tools that generate data? Knowing the answers to these questions would help to back trace possible issues occurring in building the RL-model.

Except for point 4, 5, and 6, there are no wrong or right answers to be given, being informed of all the points above is paramount before starting to develop other steps further to operate having the right information and avoiding missteps.

In Section 4 the specific data requirements per use case in hindsight will be defined.

3 Reinforcement Learning: basic components and requirements

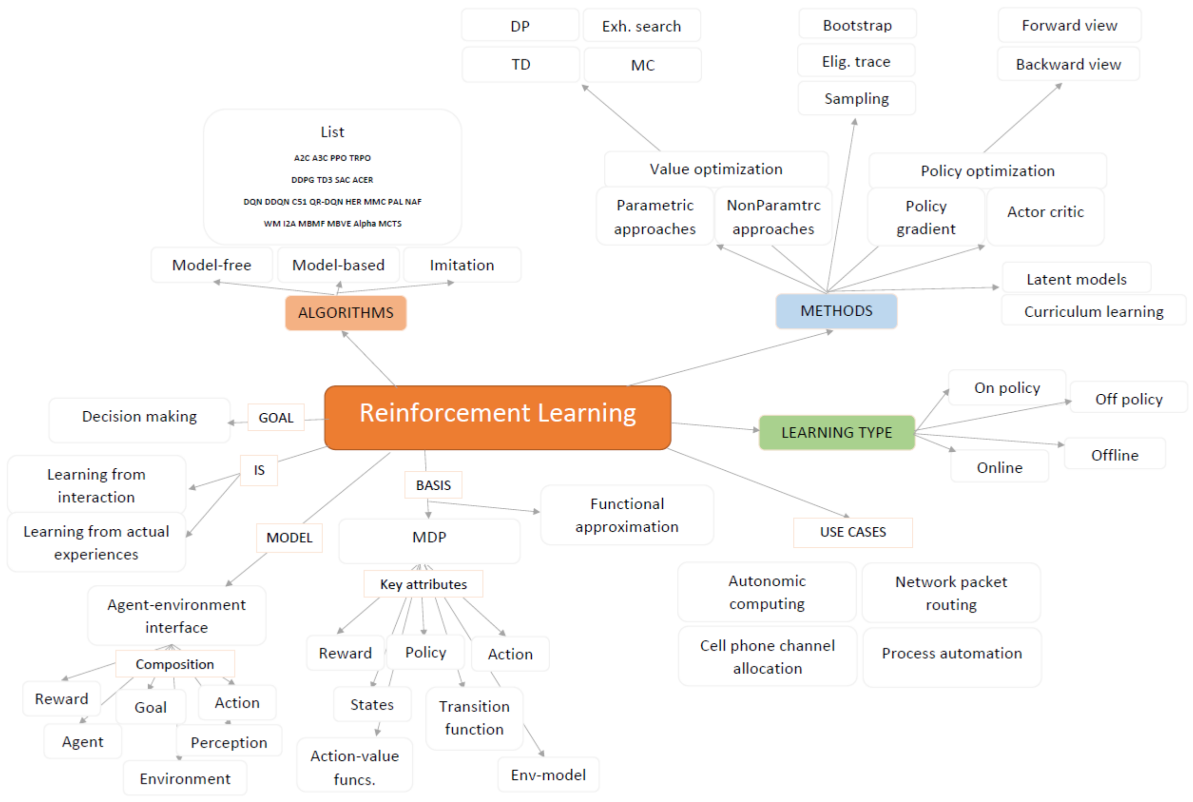

Figure 1: General conceptual view of elements in RL

Reinforcement learning solves a particular type of problem where decision making is sequential, and the final goal is typically long term. Examples of such problems include computer games, robotics, or supply chain logistics. RL aims to learn good strategies from experimental trials and relatively simple feedback from the environment.

To provide a more formal description, in RL, an RL agent engages with an environment by taking a series of sequential actions. Each action taken by the agent leads to a specific state within the environment. The precise way the environment responds to these actions is determined by a model, which may or may not be known to the agent. Moreover, what adds complexity to RL is that this response from the environment is generally characterized by stochastic behavior. Once the agent finds itself in a particular state following an action, the environment provides feedback in the form of a reward. The agent's objective is to maximize the cumulative sum of future rewards, considering the concept of discounting for rewards occurring in the future. RL has been extensively studied resulting in rich varieties of algorithms and methods, as well as applications (see Figure 1).

In the following, we outline the key challenges within RL that must be tackled by the ASIMOV solution. Additionally, these challenges can serve as a framework for establishing the requirements for the ASIMOV solution. Details on the points below is also given in D2.2 [1] and D3.2/3.3 (the deliverables are in the current state of this work in progress and will be published in the future).

The exploration-exploitation dilemma: This issue arises because the RL agent typically does not have prior knowledge of how the environment will react to its actions. Therefore, it's crucial for the RL agent not only to maximize rewards based on the information it currently possesses but also to actively explore the unknown aspects of the environment, especially in terms of how states change when specific actions are taken. Without a balance between exploration and exploitation, the agent could become trapped in suboptimal solutions, as it would rely solely on exploiting known information and miss out on the potential for discovering more favorable strategies through exploration.

Sample Efficiency: In addition to the need for a balance between exploration and exploitation, another challenge in RL arises from the inherent stochasticity of unknown environments. Within the RL algorithm, there exists a sampling component, and executing this component can often be resource intensive. While RL algorithms can demonstrate impressive capabilities, such as learning to play complex Atari video games without prior knowledge of the game's rules or mechanics (the model), it's noteworthy that even state-of-the-art algorithms like "rainbow" [2] demand a considerable amount of training time. To put it in perspective, this algorithm requires roughly 83 hours of training, equivalent to the amount of time a human player might spend on the same game, to achieve proficiency [3].

RL Capability Compared to Domain-Specific Algorithms: In various scenarios, domain-specific algorithms often demonstrate superior performance compared to RL. Examples of these scenarios include applications like optimal control or the use of Monte Carlo Tree Search (see, e.g., [4]) in gaming contexts. Furthermore, leveraging any domain-specific knowledge or information at hand can often result in better outcomes than relying solely on RL algorithms. The reason behind this performance difference lies in the fundamental nature of RL problems, which operate without making any specific assumptions except for the abstract structure defining the interaction between the environment and the optimizing agent. This structure must follow the principles of the Markov decision process (MDP). Therefore, before applying an RL algorithm, it is imperative to assess whether domain-specific algorithms tailored to the specific use case are available or if any priori or domain-specific information can be used to enhance the RL algorithm's performance.

Reward Function Design: In the realm of RL, the primary objective of RL agents is to devise an optimal strategy for maximizing a long-term reward function. This objective must be closely aligned with the goal of the use case. However, it's essential to approach this task with caution because reward functions that appear suitable at first glance may not necessarily lead to the desired behavior in an agent. Comprehensive discussions and illustrative examples regarding this challenge can be found in documents D3.2 and D3.3 (the deliverables are in the current state of this work in progress and will be published in the future). Additionally, D2.2 [1] outlines specific overarching requirements pertaining to the structure and characteristics of the reward function:

- The reward function should quantify performance on a predefined task using a single scalar value.
- The reward function should encompass the overall objective and should not grant rewards for intermediate steps that might seem reasonable to the user.
- The reward function should define what needs to be achieved without specifying how to achieve it, focusing solely on the desired outcomes.

Stability: A notable challenge within the realm of RL is that trained agents tend to become highly specialized, and their acquired knowledge often does not generalize to other environments. To illustrate the extent of this challenge, even when using identical agents in identical environments but initializing them with different random seeds, they may end up learning entirely distinct strategies. This characteristic underscores a significant degree of instability during the training of RL algorithms. Consequently, achieving stability and consistency in RL training is a formidable task. Tuning hyperparameters (parameters controlling the learning process) and ensuring the reproducibility of results become exceedingly difficult in such an environment. One approach to addressing this issue is to implement a sophisticated documentation and logging system that supports the training process.

Dealing with Continuous-Valued Data: In certain scenarios like UUV [5], specific environmental data exhibits continuous characteristics. However, traditional RL methods are typically designed to operate with discrete-valued data. Apart from the approach of discretizing the data, another effective way to address this challenge is to employ a sophisticated RL algorithm capable of effectively handling continuous-valued data. Moreover, in case of UUV, we require that the RL agent can both handle continuous- and discrete-valued data simultaneously.

Markov Property: The theoretical foundation of RL heavily depends on the Markov Property inherent in the underlying environment. In essence, all theoretical guarantees, including concepts like optimality and convergence, assume that the underlying environment conforms to the principles of an MDP. Consequently, it is crucial to establish and maintain this property within the environment, as the absence of the Markov Property can potentially lead to failures.

A digital twin will act like the environment for the RL agent in ASIMOV framework. Next to the above-mentioned challenges, the mayor hurdle will be formed by the differences in data distribution generated by the digital twin and the actual machine. If they don't match, learned behavior will not generalize.

4 Data: Pre- and Postprocessing

Besides the basic requirements on the data discussed in the Section 2, further requirements on the data are also needed for ensuring the functionality and success of RL-Solutions. Especially on the aspect of how the raw data needs to be processed in order that the RL-model can work efficiently with it. Specific aspects have been discussed thoroughly in ASIMOV deliverables 2.2 M16 [1]. We summarized the findings in the following.

4.1 Preprocessing

To begin with, it is essential to preprocess the raw data in a manner that reduces the complexity of the learning problem. Various methods can be employed for this purpose:

Discretization of Continuous-valued data: In the realm of RL, input spaces may not always be discrete, whereas traditional RL algorithms are designed for discrete data. Situations where RL problems involve continuous-valued data arise, as seen in the UUV Use Case (as detailed in D1.3). The solution here could involve either discretizing the data or utilizing RL algorithms capable of handling continuous-valued data, as discussed in the previous section. The latter sort of algorithms is discussed in D1.3 [5]

Parametrization of Input Data: To streamline the training of AI models, one can simplify the problem by parametrizing the input space. This technique reduces the underlying dimensionality to match that of the parameter space. For instance, dynamic inputs like movements can be represented using parametrized trajectories. Another approach is to use basis functions, such as wavelets for processing image data.
Data Encoding: Frequently, the data that the RL agent needs to be trained with that an AI agent needs to perform to achieve system objectives are inherently complex. However, it is often possible to group these complex signals into families with similar characteristics. These groups operate at a higher level, typically having lower dimensions and reduced complexity. This approach aids in mitigating the overall complexity of the AI training problem by simplifying the action space.

Constraint Handling: Systems operating in the real world have physical limitations that must be considered in their control architecture. These limitations may encompass constraints on feasible input ranges or constraints on internal states/observations. Examples of input constraints include physical limits on valve positions or voltage domains. One basic requirement for ensuring a functioning RL-algorithm is that data, both training and input, must be processed in order that the given constraints are fulfilled. For detailed discussions, we refer to D2.2 [1].

4.2 Postprocessing

To ensure the functionality and efficiency of an RL algorithm, additional data processing steps may be necessary. Most of the requirements are aimed for the data used in the training phase of the RL-agent. These steps include:

Data Cleanup: After the initial preprocessing steps, it's important to perform basic data processing tasks, such as removing duplicate entries and repairing corrupted data, as discussed in the "Data: Basic Checks" section.

Noise Reduction: Noise can disrupt the learning process of the RL agent, and it's essential to eliminate or minimize such disturbances whenever possible. For instance, addressing high-frequency noise can be crucial.

Data Normalization: RL algorithms rely on optimization methods, and the effectiveness of these methods often depends on the scaling of the optimization space. Therefore, normalizing the data can be beneficial to enhance the performance of the RL solution.

State Shaping: The state in RL contains all the information available to the RL agent at a given moment, and the Markov property is a fundamental concept in this context. The section initially discusses the Markov property and its various types. Once the type of process is identified, further steps in state shaping can be undertaken. These steps may involve restoring the Markov property if necessary, extracting features from the process output, or developing observers to estimate hidden information.

Reward Function Design: Part of the post-processing involves formulating the reward function, which generates a reward signal based on the system's outputs. The primary objective of the reward signal is to express the performance on a predefined task as a single value. The RL agent then aims to maximize cumulative rewards over time to optimize its long-term performance. It's important to note that the reward function should focus on defining the overall objective and should not be used to guide the trajectory to the goal based on a priori knowledge. Instead, it should specify what needs to be achieved without prescribing how to achieve it.

5 Digital-Twin Requirements for AI-Training

ASIMOV's approach incorporates the use of DTS as a valuable resource for training RL solutions. In this context, DTs serve as a simulated environment or playground for training the RL agent. This approach offers several benefits, including enhanced agent availability and control, rapid training capabilities, cost reduction in operation, and the ability to explore extreme or hazardous scenarios, among others.

The concept of employing DTs for RL training has been explored in depth in documents D2.1 and D2.3 within the ASIMOV project. In the following sections, we provide a summary of the key findings from those documents, with a specific focus on the requirements that have emerged from the discussions in those tasks.

Execution Efficiency of DT: To ensure low divergence between DT and Physical system and therefore the effectiveness of AI Training on DT, DT should possibly be updated in real-time using the input from the real system, and vice versa.

Decision for Trade-Off between accuracy and execution efficiency: In many practical cases there is a trade-off between accuracy and execution efficiency, keeping in mind that both must meet the requirements of the system.

Robustness of DT-Model: The digital model used in a DT should be robust, generate reproducible data, be comprehensible, but most importantly it should be accurate and deliver results fast.

Minimal DT-Model parameter choices: When creating a digital model, parameters are determined which can be used to control the model. To reduce the complexity of the model, it is advantageous to use the smallest possible number of parameters which are required to produce a valid model. The minimal number of parameters that still meet the requirements of the use cases is referred to as relevant. One objective when creating a model is to determine the relevant parameters.

6 Specifics per use case

6.1 Use case STEM

One of the critical measures to have the desired final image on an electron microscope (EM), is to output a Ronchigram with the desired pattern that indicates that beam is calibrated and aligned. In the Dutch use case automating the process of outputting this Ronchigram result of a calibrated EM is the goal.

From the RL perspective: the input image (Ronchigram) is the current state, the environment is the DT of the EM column (consisting of subparts condenser system, deflectors, stigmator, upper objective system, electron-beam interaction), the actions the RL agent can take are the interface controls (e.g. current intensities). The actions taken at the interface controls are also the output provided to the DT, in order the DT can generate a next state (Ronchigram image).

The desired Ronchigram:
The Ronchigram represents the diffraction pattern of the electron beam which is obtained through the interaction with the sample. The beam is influenced by components in the column and by other external factors such as temperature. When the Ronchigram shows no patterns in the relevant part of the image, we can consider that the column of the EM is calibrated, so we can consider that the goal is accomplished.

That is to say that the desired metric for this goal could be a white noise-alike metric, at the desired part of the image. In image processing, constant power spectral density can be calculated. Other known metrics as Average Peak Signal-to-Noise-Ratio (PSNR) and Structural Similarity Index Measure (SSIM) could also be used. Thinking and defining the reward function in this direction is an important step for the RL module.

Initial situation:
The initial Ronchigram generated by the DT reflects the state of the beam at a given moment, and once the machine is functioning, realignment of the electron beamneeds to be performed eventually; the EM is a very sensitive device, where any minimal change in the state of a component or the environment can cause unacceptable performance problems. An uncalibrated column will result in distorted end images (and aberrated Ronchigrams); the Ronchigram will then show undesirable patterns. There are many types of aberrations that need automatic correction within the project's lifetime. The initial state of the beam and the elements in the EM column are influenced by many factors that vary. Those variations are simulated in the DT and are in the scope of work package 2.

The result of having many initial combinations of variables is that many aberrations of different degrees can be present, outputting a very large range of possible states (i.e., Ronchigram images). Deciding which aberrations will be simulated in the DT, and to which degree, is of great importance to decide how the initial dataset will look like. More information about the types of aberrations can be found in the main case description in Deliverable 1.2 [6]. The objective of the RL is to turn those aberrated (patterned) images into constant non-patterned regions as just mentioned.

Initially the aberration correction of choice to be automated is defocus and astigmatism (two-fold astigmatism A1), which is the first case of subcase 1.1. In this first instance the initial Ronchigrams will show patterns only reflecting two-fold astigmatism A1. That also means that the first solution will be an RL agent that only needs to correct for an A1 astigmatism, given the interface controls that will be given (see Subpart “Interface and Actions”).

Ronchigram image requirements:
In any case, the images generated by the DT need to be Ronchigrams of a certain size and resolution. The minimum requirement will be images of size 512x512. The images are per definition grayscale, with a bit depth of 16 minimal. Ideally all data generated is saved with the same extension, as different extensions could generate different compression styles messing with the RL algorithm understanding (even though not visible for our eyes). Further iterations and initial implementation will guide the process further.

Those images are in first instance generated by a single-slice simulator that generates Ronchigram images from an amorphous carbon sample statistically generated. The next tool that will be used is a multi-slice simulator; this tool simulates the electron beam and sample interaction between different slices of the sample. The multi-slice simulator assumes the sample contains multiple slices as opposed to the single slice in single-slice simulator. In the latter tool, other image processing techniques may be used to improve the initial state of the image.

In a later stage, not in the scope of this document, the images will be generated by the end DT, simulating the actual machine in more detail, and generating images closely matching the 'real' Ronchigram.

Images are per definition non-structured data, and quantitative pixel by pixel. Generalization is this kind of environment is known to be a hard problem for state-of-the-art RL approaches.

For each dataset generated a statistical analysis as indicated in the fourth requirement stated in Section 2 should be done, and report as to if the data requirements are met should be done before undertaking next steps. An important fact is that for each goal the dataset requirements are met. Generating a dataset requirement per goal will be a hard requirement that can be found in the table of requirements below.

The high dimension of data and states at the input for any AI system requires systems that can cope with that. This kind of input already limits the approaches that can be used at the RL module.

<u>Interface and actions:</u>
In the bigger use case scenario, no information is currently available in the main document about the interface simulating the action possibilities to be taken by the agent, even though requirement EM_SUC1_FR_DT_002 in the main use case description indicates controls that are available in the real machine.

The current intensities that can be tweaked are represented in the microscope control panel as 'Multi-function knobs X and Y' as it can be seen in Figure 2. Those knobs regulate stigmatism values in the lens, specifically they regulate Stigmator X and Stigmator Y. For now, it is assumed that there will be some initial float values (e.g. 1e-6) and that the knobs increase by certain tolerance (e.g. 1e-10). A drawing on the real interface can be found below, and red circles around the knobs of choice that need to be available in the first implementation step can be seen.

The intensities to adjust astigmatism are also the knobs found by the EM operator when fixing the astigmatism problem. Relevant for the interface is that the implementation concentrates on direct controls available to the machine operator, and no other underlaying mechanisms that would not be used normally. Also, simulating other underlaying mechanisms is out of scope for parameters considered for this task. The objective is to automate this task away from the operator so that the time can be focused on the experiment instead of the calibration of the machine. Validating that RL technologies can be used to solve such a task is also in the goals.

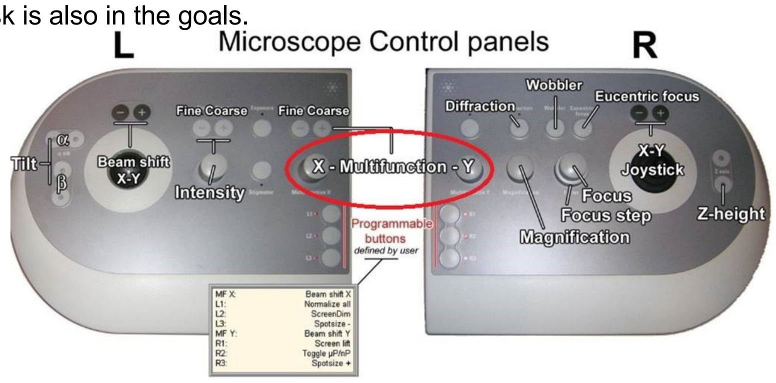

Figure 2: Electron Microscope operator control panel view with buttons of interest.

Table 1: Requirements Use Case STEM

RID	Component	Requirement type	Requirement description	Priority
Rq_EM_1_01	RL-input	Non-func	The input for the RL module will be an image of minimal dimension 512x512, grayscale and minimal 16 bit depth.	H
Rq_ EM_1_02	RL-input	Non-func	All images generated will be of the same size and grayscale depth.	H
Rq_ EM_1_03	RL-input	Non-func	All images generated will be generated in the same image extension.	H
Rq_ EM_1_04	RL-input	Func	Images generated will have read and (write) permissions.	H
Rq_ EM_1_06	RL-input	Non-func	All datasets generated will be accompanied with a .csv file where all metadata information about the dataset generated will be available, and metadata about the generator tool will be available (e.g. setting with which datapoint was produced).	H
Rq_ EM_2_01	Infrastructure	Func	All relevant datasets generated will be published and hosted at an available location for partners. Tools could be shared too.	M
Rq_ EM_3_01	RL-interface	Func	The actions that the agent will be able to perform to fix Astigmatism A1 and where the DT will be able to react to because modelling allows are: 1.Multifunction X and 2.Multifunction Y.	H
Rq_ EM_4_01	RL-output	Non-func	The RL will output a set of parameters (actions taken), in the first sub use case consistent of value for 'Multifunction X' and 'Multifunction Y.'	H
Rq_EM_5_01	DT-tooling	Non-func	Document describing simulated parameters in DT tooling, and ranges expected. Combinations possible on simulator should also be present.	H
Rq_EM_6_02	Ronchigram state	Func	A function measuring how well the Ronchigram is looking is needed to stop the optimization by the RL agent when column is calibrated. Defining metric to reach this point is necessary eg SNR, probability density function etc.	H
Rq_EM_7_01	Data generation	Non-func	A description of what representative data means needs to be filled out by an expert-matter topic for each goal, including astigmatism A1.	M
Rq_EM_8_01	Data_generation	Non-func	Output numbers and figures to assure data distribution is right, and to avoid bias and incomplete datasets.	M

Other Requirements

Further requirements for the TEM Use Case are given below:

Table 2: Requirements use case STEM (Other Requirements).

RID	Component	Requirement type	Requirement description	Priority
Rq_ EM_1_05	RL-input	Func	All images will be generated with label nomenclature 'Ronch_*number_aberrationtype.extension*'. This is currently a technical limitation of the DT that needs to be fixed	L
Rq_ EM_2_01	Infrastructure	Func	All relevant datasets generated will be published and hosted at an available location for partners. When the site is available also renaming of files in a more coherent manner will be possible too.	M
Rq_EM_5_01	DT-tooling	Non-func	Document describing simulated parameters in DT tooling, and ranges expected. Combinations possible on simulator should also be present.	H
Rq_EM_6_01	RL agent performance	Non-func	The RL agent should be able to perform with high requirement, given that RL problem solving should cover the most complex scenarios. Metrics should be defined to measure the RL agent success. Defining the metrics of success is a hard problem highly dependent on the solution to be chosen, and at this stage the information is not available yet.	H

Some requirements important but not mentioned in the requirement table, are momentarily working progress and will be given in other deliverables.

6.2 Use case UUV

6.2.1 Scenario Generation for Digital Twin based UUV-Tests

In "Unmanned Utility Vehicle" (UUV), the first goal is to test components of a vehicle (e.g., chassis, Sensors, Driving function) in the most effective manner possible. The output of this use case would be a Test Plan that describes a sequence of scenarios that need to be tested on a vehicle testbed, to gain the most amount of information about the component under test. The main goal for this use-case is to provide test scenarios ensuring safety coverage of the autonomous vehicle.

Together, the factors mentioned above deliver a rough range of test scenarios. However, during RL training, to verify and validate the test scenario coverage to ensure safety coverage, component level simulation in this sub use case could be considered as the foundation for higher level simulation such as system level simulation and on road driving in the later phase of RL training.

Inputs

The reinforcement learning agent employs KPIs to assess the actions it generates. In the context of RL, these KPIs serve as the agent's rewards. Since RL algorithms typically operate with a single numerical value as reward, the KPIs must undergo aggregation into a single value through a weighted sum. The selection of weights is specific to the use case and dependent on the objectives encoded within these metrics.

The choice of KPIs is driven by the goals of achieving safety in critical UUV scenarios and introducing diversity in environmental conditions. Within this framework, the KPIs can be divided into two categories: criticality KPIs and diversity KPIs. In the initial phase of the prototype, the focus lies on achieving critical scenarios. Once this objective is met, attention will shift towards incorporating diversity into the environment. A detailed list of criticality KPIs is provided in D1.3 [5]. Whenever feasible, these KPIs are supplemented with a reference value to facilitate the normalization of the resulting value.

The criticality furthermore needs to be split up into different categories to pinpoint the type of criticality that was present in the scenario. This could be, e.g., a poorly tuned lane keep assistant influences different criticality types compared to a poorly tuned adaptive cruise control. Also, an important aspect is to link the criticality with the underlying driving function of the UUV. KPIs in this aspect are, e.g., metrics measuring the reaction time of the UUV.

The information about the component under test, as well as the actual information about the component's state, serve the input to the reinforcement learning agent.

The exact type of information that will describe the state of a component will require careful feature engineering (for further details see D1.3 [5]). The data streams from vehicle sensors, e.g., for accelerations, wheels speed, ground speed, yaw, pitch, roll, wheel travel, can be compared with the data from the sensors for AD functions, such as camera images and LiDAR point clouds, to evaluate their similarity in various scenarios. That way, the reinforcement learning agent can generate parameters for follow up scenarios, that will yield data with as little similarity to already collected data as possible.

Actions

The reinforcement learning agent control the scenario generation. Therefore, it should output a set of parameters, which describe scenarios and their variations. A scenario can be seen as an instance of a traffic situation, in which different traffic objects interact with each other. One prominent example being a pedestrian, that suddenly walks on the street and is therefore intersecting with the vehicle's trajectory. To describe such a scenario, multiple parameters are required in the static and dynamic part of the scenario.

The static part of the scenario describes the surroundings of the scene. This could be the density of trees next to the road, weather information, the type and visibility of lane markings on the street and so on. By varying these parameters, a wider operation window regarding the perception (e.g., LiDAR, camera) can be tested.

Most of the static scenario is no subject for optimization. However, after receiving the variation information for the 3D environment from the RL agent and the creation of the 3D environment, a validation check process is initiated. This involves evaluating whether objects are logically and probably located in the scenario, considering factors such as size, position, and count. The validation mechanism includes the following criteria:

1. Validation of object placement:

   After the creation process of the 3D environment, Trian Builder [7] generates a validation text file with information about the placed objects. The system compares the intended number of objects (from JSON data) with the number of resulting objects in the generated variation. The system provides feedback based on the comparison and indicates conformity or deviation from the intended placement.

2. Analysis of overlaps and collisions:

   The generated text file also contains information about overlapping or colliding elements in the environment. A numerical value describes the extent to which these elements overlap. This information is communicated to the RL agent as a penalty and influences its behavior in subsequent iterations.

3. Concrete implementation criteria:

   The "overlap value" is expressed as a percentage and indicates the extent to which the bounding boxes of certain objects overlap. This percentage allows the generated value to be calculated independently of differences in the size and shape of the overlapping scene elements. For more detailed information, we refer to T2.5. In the status of this deliverable D2.5 is not yet published. However, that deliverable will be published soon in [8].

The resulting action defines then critical cases which can be used for testing the UUV behavior. The dynamic part of the scenario describes how the different traffic participants interact with each other. Velocities, routes, as well as parameters that control when a critical situation should occur and which traffic object types are involved, are part of that category. Defining the current list of parameters that aim to define these scenarios is the task of T2.1-Workgroup whose deliverable will be published in [8] soon.

Infrastructure

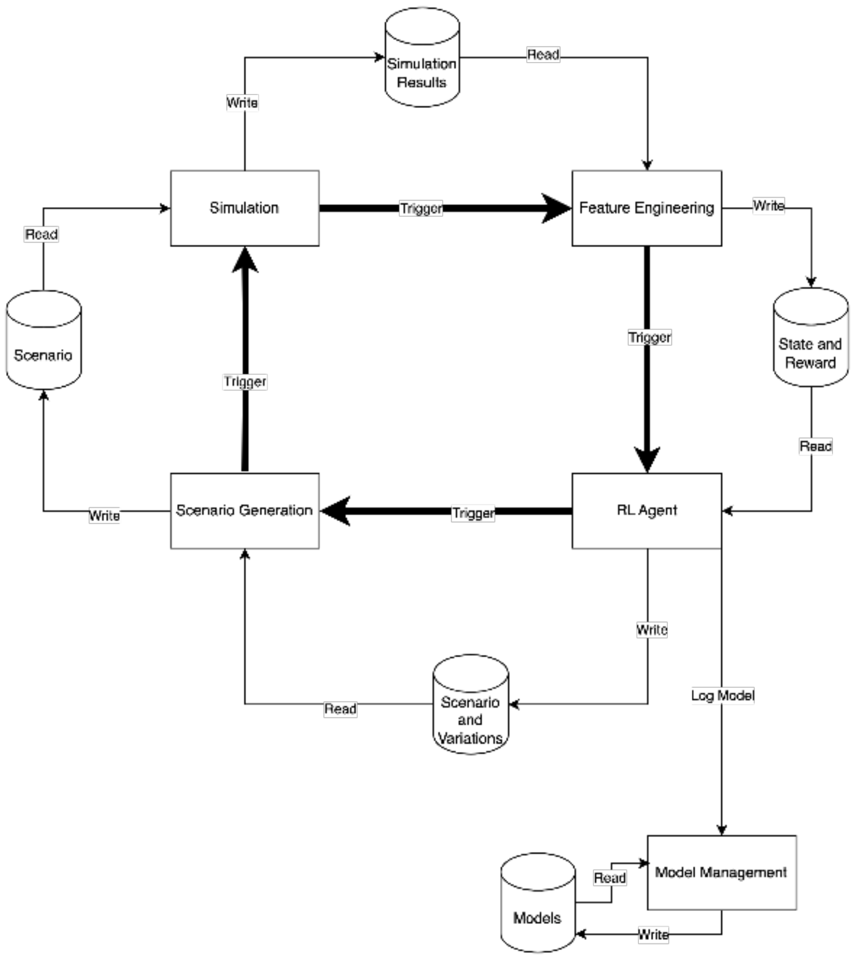

Figure 3: Architecture of UUV Use Case 1

Micro-Service Oriented Structure: For the UUV use case, we require that the RL solution is structured into manageable components that are loosely connected through communications. The rationale behind this approach is as follows:

- Independence of Development: By isolating the components, developers can work on each component independently without causing conflicts or dependencies on other parts of the system. This modularity enhances the development process's efficiency and flexibility.
- Simplified Debugging, Testing, and Validation: With distinct, well-defined components, debugging, testing, and validating the functionality solution becomes more straightforward. Issues and bugs can be identified and addressed within specific components without affecting the entire system, making the development process more manageable and efficient.
- Scalability Advantage: Isolation provides a significant advantage when it comes to scaling the application. Developers can focus on scaling individual components as needed, rather than attempting to scale the entire system at once. This targeted approach to scaling enhances the system's overall efficiency and performance.
- Efficient Communication: The loose coupling of these isolated components allows for more efficient communication between them. Developers can implement communication methods that are tailored to the specific needs of each component, optimizing data exchange and overall system performance.

In the context of the UUV 1 use case, our system comprises four key components as shown in Figure 3: the RL Agent, Scenario Generation, Simulation, and Feature Engineering. To achieve our defined objectives, we need to establish a specific organizational structure among these components. This organization also includes the order in which these components operate.

Containerization of the Components: To effectively deploy our components, we find it essential to employ containerization techniques. This approach enables each component to operate within its own encapsulated and self-contained runtime environment, thoughtfully defined by our developers. Through this method, we can ensure consistent deployment of our components across various environments, leading to enhanced portability and the ability to seamlessly adapt to different deployment settings.

Data-Storage Infrastructure for the Component Interfaces: Additionally, we emphasize the need for a well-defined data-storage infrastructure to facilitate data interchange among these components. Access rights to the database are granted based on necessity, ensuring that each component can access the data it requires without unnecessary access to unrelated information. Furthermore, we maintain a clear separation between different databases, ensuring that they are distinct and independent to prevent data overlap or interference among the components. This separation enhances data integrity and the overall efficiency of our system. Such databases can be realized as folders mounted in components container. Other alternative would be the hierarchical labelling structure (facets) discussed in the next part (see the data management part)

Task-Processing Structure: In the initial phase of the project, we establish a structured sequence, meaning that the components are ordered in such a way that each component triggers the start of the subsequent one once it has completed its task. In the future stages of our project, we intend to implement a more efficient communication structure, specifically a queuing system. In this improved setup, tasks will be organized and executed within designated queues, eliminating the need for direct triggering between components. Instead, each component will independently process and complete tasks from its respective queue. This shift to a queuing structure enhances system performance and streamlines the workflow by allowing components to work autonomously, responding to tasks as they become available in their queues.

Alignment of data with the usual standard: For the practical use of the ASIMOVs solution, we require that the data is available in the usual industry standard. The OpenDRIVE [9] and Carla World file formats [10] align with usual standards and ensure the integrity and reliability of the simulation process within the ASIMOV project. OpenDRIVE, developed by the ASAM organization [11] for realistic simulations of virtual road networks, offers elements such as lanes and pedestrian crossings. This ensures safe testing of vehicles in simulated environments. Carla World files are integral to the Carla (Car Learning to Act) simulation environment. Carla features highly developed sensor models such as lidar or radar and serves as a simulation platform in UUV use case. Both data formats are designed to meet standards for accurately representing 3D environments and are thoroughly discussed in the document of new sections of D2.2. For the time-series data, we suggest that such data is available in the so-called ASAM-ODS format [12], which is suited for Big-Data processing.

Scenario Engine: For simulation purposes, Scenario Engine which can playback the provided scenario in the given standard OpenSCENARIO is required, developed, and maintained by the ASAM organization, is an established standard in the automotive industry. It defines the traffic scenarios and focuses on the dynamics and interaction of road participants within the road networks provided by OpenDRIVE in context of UUV. OpenSCENARIO files are then simulated in a varied 3D environment. Details on this aspect is treated in the workgroup for T2.2 [1].

<u>Data and Training Management</u>

To achieve complete control over the optimization process in the UUV use case and to have reproducible results for ASIMOV's solution, it is essential to employ advanced frameworks specifically designed for documentation, data, and resource management purposes.

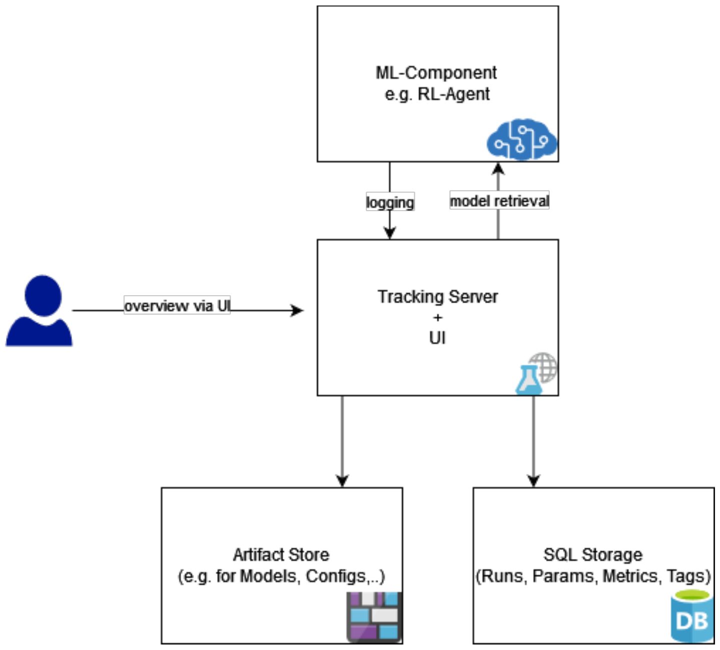

Figure 4: Logging Architecture

Model Logging Framework: To begin, our first requirement in this hindsight is the implementation of logging and tracking components within the RL process. This component should possess the capability to securely store the RL model, including its associated parameters, during the training phase. The incorporation of robust logging and tracking of RL experiments provides several distinct advantages, as it empowers us to:

- **Monitor Experiment Parameters:** We can easily observe the specific parameters utilized in each experiment, allowing for a detailed examination of the configuration settings.
- **Retain Historical Models:** The system retains previous models employed in experiments, providing a valuable reference for understanding the evolution of the RL model.
- **Optimize Hyperparameters:** Through comprehensive tracking, we can pinpoint the best-performing hyperparameters throughout the entire experiment, enabling us to refine and enhance our models effectively.
- **Analyze Learning Progress Over Time:** Continuous monitoring of learning experiments facilitates the comparison of results over time, revealing trends and progress.

The presence of this functionality not only opens the potential for transferring learning results between different timeframes and various learning agents but also enables domain experts to engage in in-depth analysis of the learning outcomes. Additionally, it offers the capacity to reproduce learning experiments when further analysis or validation is required, contributing to the robustness and transparency of the RL process. A suitable framework for such purposes is given by MLFlow [13] illustrated in Figure 4. Essentially, MLFlow provides a framework for logging the learning experiments, both, the parameter in a structured database (here SQL-Database), and the created model in a separate database suited for storing files.

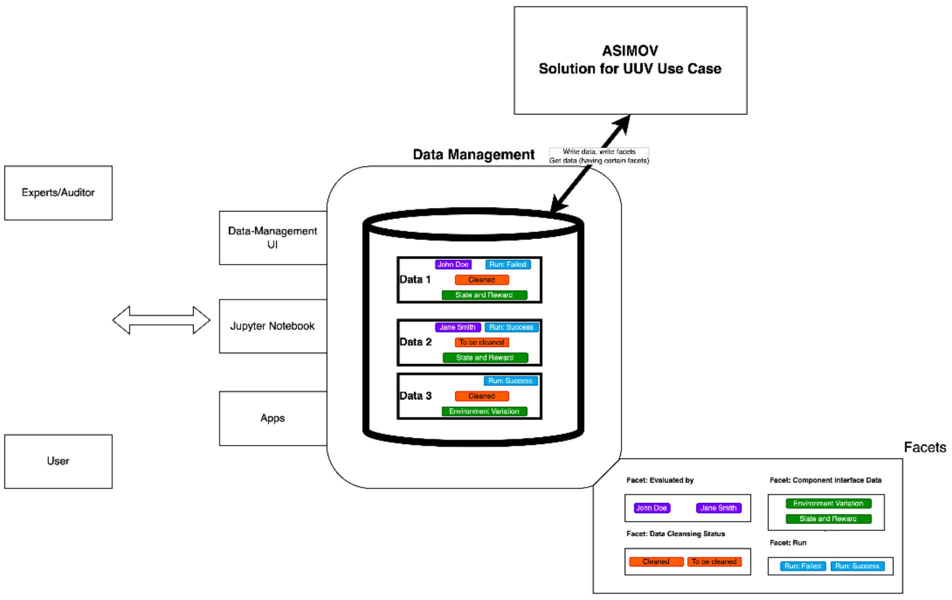

Figure 5: Data Management and ASIMOV Solution

Data Management Platform: Furthermore, to optimize the utility of ASIMOV's solution, it proves advantageous to establish a comprehensive management and documentation framework that extends beyond the training phase and encompasses the data management aspect as well. This holistic approach ensures that the RL solution is seamlessly integrated into the entire data lifecycle, covering further critical stages such as data preprocessing, ingestion, curation, discovery, and subsequent utilization across a spectrum of analytical applications.
When the RL solution is intricately woven into the data management system, it stands to benefit from enhanced database query accuracy and performance. Moreover, as the foundational data platforms evolve to better support emerging initiatives, such as providing direct compatibility with tools like Jupyter notebooks, it becomes feasible to facilitate the development of AI-driven applications and the creation of further intricate data models.
As for the specific requirements in the realm of data management:

- **Rapid Application Development:** There's a need for expeditious application creation based on the available data, with the added capability to design user interfaces for no-code deployment of ASIMOV's solution. This streamlines the process of building and deploying applications that leverage the data effectively.
- **Hierarchical Data Labeling (Facets):** It's advantageous to have the ability to assign hierarchical labels to the data. This hierarchical labeling system facilitates structured organization and categorization of data, making it easier to access and utilize effectively. Such labels can be seen as facets of the data. Furthermore, this label can be also used for realizing the component data interface replacing folders allowing flexibility in handling.
- **Integration with Applications:** The data management system should seamlessly integrate with various applications, allowing for a cohesive connection between data and applications, i.e., based on the assigned labels. This integration enhances the accessibility and utility of ASIMOV's solution within the broader ecosystem.

By fulfilling these data management requirements, ASIMOV's solution can unlock its full potential, enabling efficient application development, structured data organization, and seamless integration with analytical tools and models.

Further Requirements for Solution Optimization

Parallelization Concepts and Advanced Resource Management for the Solution Creation and Operation: If additional performance gain is needed in term of execution speed in the training.

and operation of RL solution, one can make use of parallelization. The first step to be taken for this is the division of the solution into smaller components which is one the first requirements for the solution architecture. Then parallelized processing can be realized as already discussed previously by queueing architecture. Besides the processing paradigm, we require efficient processing suitable resource management which e.g. organize the components and assign resources to them. Such a resource management should ideally fulfill the following properties:

- **Scalability:** The resource manager's primary role is to facilitate the effortless scaling of applications, either up or down, as the situation demands. This can be achieved by dynamically adding or removing resources for the respective components when necessary. Such adaptability enables the solution to autonomously respond to fluctuations in demand, guaranteeing its capability to efficiently handle substantial workloads and datasets.
- **High availability:** The resource manager should possess the capability to automatically maintain high availability for applications. This is achieved by proficiently scheduling task execution containers across numerous computational nodes within a computing cluster and swiftly substituting any failed executing containers. This proactive approach serves to prevent downtime and assures that applications remain continuously accessible to users.
- **Improved resource utilization:** The resource manager should possess the capability to autonomously arrange task scheduling in accordance with the currently available computational resources. This proactive allocation strategy contributes to enhanced resource utilization while minimizing wastage. Consequently, it not only leads to cost reduction but also enhances the overall efficiency of applications.
- **Easy deployment, updates, and version control:** The resource manager should streamline the deployment and updating of applications through the adoption of a declarative configuration approach. This empowers developers to articulate their application's desired state, and the resource manager will automatically align the actual state with this specification. Additionally, it is essential to incorporate version control functionality within the resource manager, facilitating a comprehensive understanding of its configuration and simplifying deployment across the consortium's member environments.

Parallelization Concept for Handling Big Data Analysis: When dealing with the analysis of Big Data in the context of RL solutions for UUV, it's crucial to acknowledge that a substantial amount of data will be generated and necessitates thorough analysis and validation. Furthermore, users of the ASIMOV solution for UUVs may want to utilize the processed data for various purposes, including integration with other systems. To efficiently manage this task, it becomes essential to explore strategies for data handling, and one promising avenue is the adoption of parallelization techniques. However, it's important to recognize that implementing parallelization methods requires data to be structured in a specific format, which introduces a certain time cost that must be factored into the overall cost and benefit analysis of the parallelization solution. At present, the state-of-the-art approach in this regard is the utilization of Spark, which necessitates data to be presented in the Parquet format. For even swifter data processing in the automotive use case, an alternative option involves transforming the data into the ASAM ODS format. This can lead to significantly enhanced processing speed and efficiency.

Summary

We summarized the requirements given above in this table:

Table 3: Requirements Use Case UUV

RID	Component	Name	Type	Func-tional?	Requirement description	Priority
Rq_ UUV_1_01	Infrastructure	Containerization of Components	Appli-cation	Func	Solution is realized as containerized components. Specific components in the first step: Feature Extraction, Scenario Generator, Simulation, and Reinforcement Learning.	H
Rq_ UUV_1_02	Infrastructure	Ordering and Organization of the Components	Appli-cation	Func	The components are well-ordered according to the specified workflow. In the first step each component triggers after the completion of its task the subsequent component. In the next prototype, queueing infrastructure might be realized instead the subsequent trigger structure.	H
Rq_ UUV_1_03	Infrastructure	Data Infrastructure	Appli-cation	Func	Database structure for the interaction between the components should be well-defined. Access right to the databases should be provided regarding to the necessity	H
Rq_ UUV_1_04	Infrastructure	Task Status	Appli-cation	Func	Every task done by the components is labelled by the corresponding status. Status should reflect whether the task is initialized, running, failed, or succeeded.	H
Rq_ UUV_1_05	Infrastructure	UI	Appli-cation	Non-Func	Learning process can be controlled and adjusted by user via UI in a no-code environment.	M
Rq_ UUV_1_06	Infrastructure	Framework for RL Logging	Log-ging	Non-Func	Framework for documenting the learning process should be given. The Framework should be able to store RL model and further information created in the training step. Moreover, the information should be retrievable and ergonomic possibilites to analyze the generated data should be given.	M
Rq_ UUV_1_07	Infrastructure	Logging	Log-ging	Non-Func	Every parameter set and RL input will be saved in the storage platform for tracking.	H
Rq_ UUV_1_08	Infrastructure	Data Management Platform	Appli-cation	Non-Func	A platform which allows the interaction of user with the data arises in the ASIMOVs solution, e.g., for analysis and validation purposes	M
Rq_ UUV_1_09	Infrastructure	Data Availability	Data Stor-age	Func	All relevant datasets generated will be published and hosted at an available location for partners. When the site is available also renaming of files in a more coherent manner will be possible too.	M
Rq_UUV_2_ 01	RL-interface	RL-input	Re-ward/S tate	Func	The Inputs for the RL agent come out of the Feature Engineering Component, which provides information about the observed crticality of the scenario.	H
Rq_UUV_ 2_02	RL-interface	RL-input	Re-ward/S tate	Func	The inputs must be normalized to improve learning of the Agent.	H
Rq_ UUV_2_03	RL-interface	Dimension of States	State	Func	The outputs contain discrete and continuous parameter values.	H
Rq_ UUV_2_04	RL-interface	Output Feasibility	Output	Func	The outputs must be in the respective range for each parameter.	H
Rq_ UUV_2_05	RL-interface	State Definition	State	Func	The state shall consist of criticality metrics (Rq_UUV_3_05), and basic vehicle configurations (Rq_UUV_3_06).	H
Rq_ UUV_2_06	RL-interface	State Transition	State	Func	In each iteration, the criticality values of the new scenario are created.	M
Rq_ UUV_2_07	RL-interface	Criticality Values	State	Non-Func	If multiple critical values are available, it must be ensured that each of the critical values represents one unique criticality metric. The specific critical values are given by the outcome of FE workgroup [5].	M

RID	Component	Name	Type	Func-tional?	Requirement description	Priority
Rq_ UUV_2_08	RL-interface	Basic Vehicle Configurations	State	Non-Func	Basic vehicle configurations are used to give the RLA some basic knowledge about the vehicle, e.g., vehicle weight. This information shall be as restricted as possible.	H
Rq_ UUV_2_09	RL-interface	Initial State	State	Func	The first state is blank (values = 0) except basic vehicle configurations.	H
Rq_ UUV_2_10	RL-interface	Episodicity	Training	Non-Func	The application of the RLA is structured in episodes. The characteristics of the vehicle change slightly, but we stay inside the operational design domain (ODD.	H
Rq_ UUV_2_11	RL-interface	Complex Environment	Training	Non-Func	We need to assume that the system the RLA shall optimize is highly complex. Thus, the algorithm needs to be capable to deal with this level of complexity.	H
Rq_ UUV_2_12	RL-interface	Distributed Learning	Training	Non-Func	Ideally, the algorithm is capable of being trained in parallel.	M
Rq_ UUV_2_13	RL-Interface	Input	Training	Func	The RLA shall be able to take one numerical variable as the reward.	H
Rq_ UUV_2_14	RL-Interface	Weighted Sum	Training	Func	The reward shall be a weighted sum out of all KPIs.	H
Rq_ UUV_2_15	RL-Interface	KPIs	Training	Func	The KPIs are aligned with the goal of providing safety critical scenarios and diverse environment variations.	H
Rq_ UUV_2_16	RL-Interface	KPI and driving function	Training	Func	The set of KPIs should include metrics linked with the driving function of the UUV	H
Rq_ UUV_3_01	RL-output		Output	Func	The RL outputs parameters for the scenario generation, which afterwards get translated into the ASAM OpenSCENARIO format. Additionally, some parameters of the campus environment are variated	H
Rq_ UUV_3_02	RL-Output	One Action – One Scenario	Output	Func	In the first step, one action of the RLA shall be the input for one scenario.	H
Rq_ UUV_3_03	RL-Output	Systematic Scenario Errors	Output	Func	The possible actions shall be designed in such a way that any parameter combination produces a valid scenario.	H
Rq_ UUV_3_04	RL-Output	Output Parameters	Output	Non-Func	In the first step the output parameters shall be static environment parameters, e.g., time of day. In the later step the output parameters shall include dynamic environment parameter.	H
Rq_ UUV_3_05	RL-Output	Output Values	Output	Func	The actions shall support both discrete and continuous values.	H
Rq_ UUV_4_01	Other Requirements	KPI-Logging	Diagnostics	Non-Func	The implementation should be capable of logging the main KPIs to track training progress.	M
Rq_ UUV_4_02	Other Requirements	Language	Implementation	Non-Func	The RLA shall be written in python to support the latest RL advancements.	M
Rq_ UUV_4_03	Other Requirements	First Step I/O	Implementation	Non-Func	The RLA shall store a file with its actions and receive the state and reward as files in a folder as well.	H
Rq_ UUV_4_04	Other Requirements	Data Efficiency	Implementation	Non-Func	The algorithm should be data efficient.	M

6.2.2 Environment Interface for UUV

In contrast to the discussions made in Subsubsection 6.2.1 regarding the scenario generation for UUV, here we address a more basic question concerning the requirements of AI training data used for autonomous driving, i.e. how realistic artificially derived training data has to be even in small details.

Background: Autonomous driving functions not only need to be able to identify and classify objects around the ego-vehicle, but to identify traffic situations, so called scenarios, and assess if the ego-vehicle needs to react. A typical scenario is a cut-in, e.g. on a highway with several lanes in one direction: A faster car takes over the ego-vehicle and cuts-in on the ego vehicles lane. Already here several sub scenarios are possible: the other car cuts-in too close, or breaks during the cut-in, one, two or three other

vehicles are present and interact, the weather is rainy or windy, the street is dry, wet or covered with leaves, the scenario occurs in Germany or the United States, which have different traffics laws, or in the United Kingdom, where vehicles drive on the left not the right side of the road.
It becomes obvious that there are thousands, if not millions, of scenarios AI based driving functions need to be capable to recognize. Collecting all this data for training has been proven highly inefficient. Especially since some scenarios are relevant, but do not occur often.
A possible solution is to generate scenario data by utilizing digital twins, consisting of an environment simulation, a simulation of the used sensor, and, for validation purposes, the object perception software itself.

Drawback of Carla Sensor Model: There are several tools available to simulate both and that can stream raw data to the perception software. One is Carla, which is used in the described UUV use case in Scenario Generation for Digital Twin based UUV-Tests Scenario Generation for Digital Twin based UUV-Tests. However, each setup does not include all factors needed for a detailed digital twin of the components. E.g., Carla uses ray-casting to determine what a specific sensor detects of the environment. Unlike ray-tracing the technique does not consider multiple reflections, therefore the result will be a simpler representation of reality. Implementing other factors, such as the exact roughness of the road to rock car and sensor as in the physical system, would consume a considerate number of resources (collecting the information, building the model, running the model). Resulting in the question, if such a level of detail can be achieved economically. But also, if it is needed. Except for corner cases, most of the mentioned perturbations would only cause a minimal variation in the data stream.

Requirement for more Sophisticated Environment Sensor (LiDAR): This use case evaluates the question how detailed a digital twin environment for LiDAR sensors must be, one of the integrals sensor types used to realize autonomous driving, to produce good enough scenario data for AI training without consuming to many resources. LiDAR is an abbreviation for *Light Detecting and Ranging*. Laser beams of a specific wavelength are emitted by the LiDAR head. These are reflected by objects in the environment and finally propagate to the detector. Because the speed of light is constant for every given medium and the time of the emission is known, the system can calculate for every detected burst of light where it got reflected[1]. Therefore, each frame of information the sensor issues is a three-dimensional cloud of reflective spots of different intensities, called point cloud, see Figure 6. The result is in principle not only influenced by the position of objects that caused the reflections but also by their reflectivity in the wavelength range of the laser, the local air density, weather (especially fog, but also rain), vibrations perturbing the measurement, e.g. smaller and larger road bumps that rock the vehicle together with the vehicle suspension, possible multi-reflections and other factors. A precise representation of the physical system would have to regard all these factors. But is such a detailed simulation needed?

[1] The distance is calculated based on the time of flight, the detection of the angular orientation of the reflective spot depends on the LiDAR type.

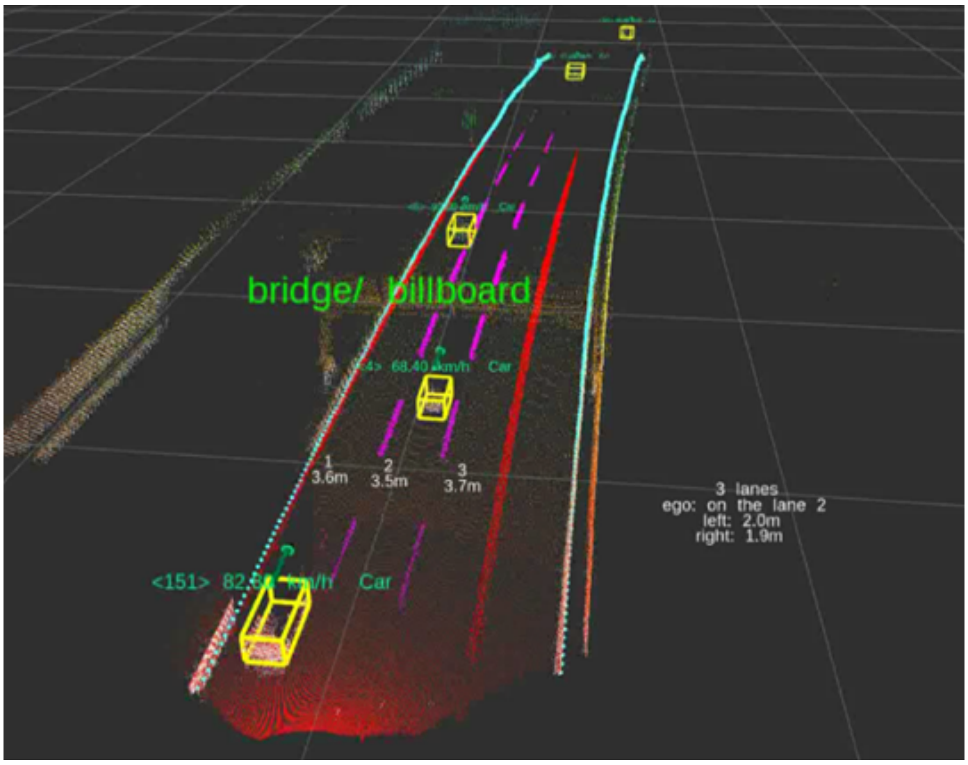

Figure 6: LiDAR 3D Point cloud of an Autobahn scene. The scene was analyzed by a perception software. Identified objects have been marked with bounding boxes, which have been annotated with additional information (class, e.g. car, and speed).

Figure 7 shows a LiDAR image of a target plate (usually used for calibration) generated in CARLA (left) and overlayed with data from the measurement of the physical system in a lab with a fixed LiDAR position is shown. The basic properties of the physical LiDAR have been reproduced, additionally a 0.05 per cent of noise where added. Still the resulting points reflected from the target appear very ordered and with a continuous change in intensity from top to bottom (left). The points of the physical measurement do not show such a distinctive intensity change and especially in the top part of the target the pattern is significantly disturbed (right). However, the target is well recognizable in both instances. Also, the edge measurements of the target taken in the digital system fit the edges of the target of the physical system well. Compared to the absolute dimensions the deviations are large on point level and small on object level.

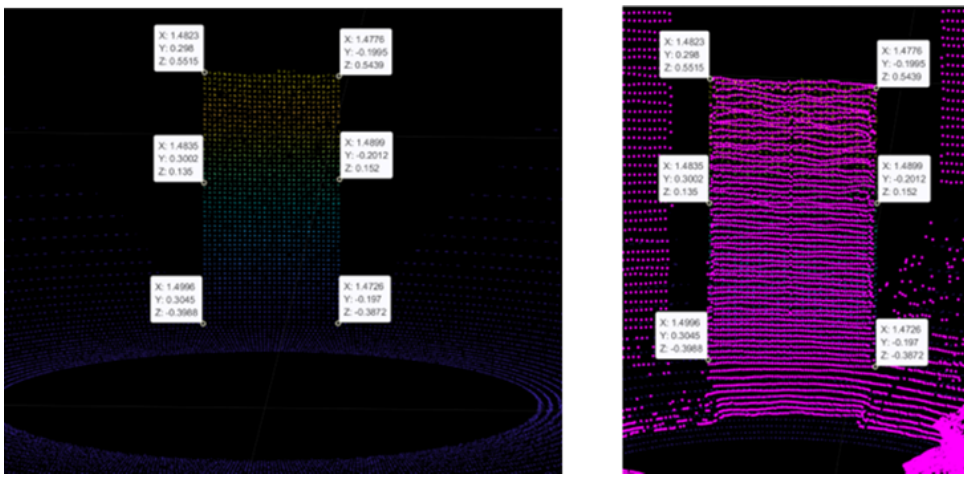
X: 1.4823 Y: 0.298 Z: 0.5515
X: 1.4776 Y: -0.1995 Z: 0.5439
X: 1.4835 Y: 0.3002 Z: 0.135
X: 1.4899 Y: -0.2012 Z: 0.152
X: 1.4996 Y: 0.3045 Z: -0.3988
X: 1.4726 Y: -0.197 Z: -0.3872

Figure 7: LiDAR Measurement of a Target plate in Carla (left) and overlayed with the measurement in the physical system.

Version	Status	Date	Page
2.0 M32	Internal	2023.11.21	28/32

Because AI-based object perception is based on point groups (shape etc.), therefore on object level, and works despite different noise levels on a variation of typical roads and (mild) weather conditions, the deviations caused by a simplified framework as CARLA uses it, should not alter the result, i.e., the object still gets recognized, classed, and positioned as in the physical system. Of course, for AI training purposes the noise level of the digital twin should be of similar type and strength as in the physical counterpart. But in a time-dependent data stream, the noise pattern, caused by several factors, appears random, which can be easily modelled.

Practices for Sensor Validation: To test this hypothesis, LiangDao designed a hunter/target experiment with a physical and digital counterpart. In the physical system, see Figure 8, the target is represented by an autonomous shuttle. The hunter is represented by a mobile roadside unit (RSU) consisting of two LiDAR and the hard- and software for object detection[2]. While the shuttle propagates on its course, the RSU records its environment. The raw data is analyzed by the object detection, the shuttle gets identified, positioned, and tracked. The object data is recorded and saved in an object list.

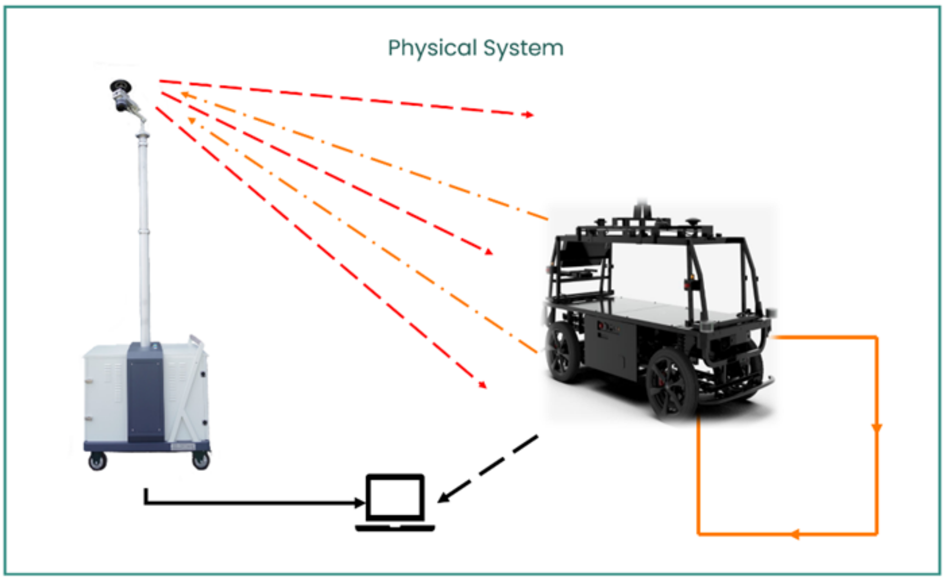

Figure 8: Scheme of physical system part of the LiDAR experiment

The experiment is then rebuilt and repeated within CARLA. The basic properties of the RSU, especially the used LiDAR, the shuttle and the environment are modelled to match the physical counterparts. The general steps of the experiment are depicted in Figure 9. In both counterparts the shuttle is identified, positioned, and tracked. The digital twin of the LiDAR is positively validated if the shuttle gets identified and the position of both shuttles (physical and digital) is matched within an error margin given by the intrinsic measurement error of the physical system. At the current stage, the experiment is set up and first runs have been carried out. These indicate that in principle a digital twin of LiDAR sensor and environment with reduced fidelity and properties produces usable data for LiDAR perception software, which in principle can be used for training purposes. Therefore, at least for some scenarios the used tools can be used to generate training data. For a more precise evaluation additional results are needed.

[2] The object perception of the RSU is based on the autonomous driving function perception software. The results can be translated. However, using the mobile RSU allows a greater freedom positioning.

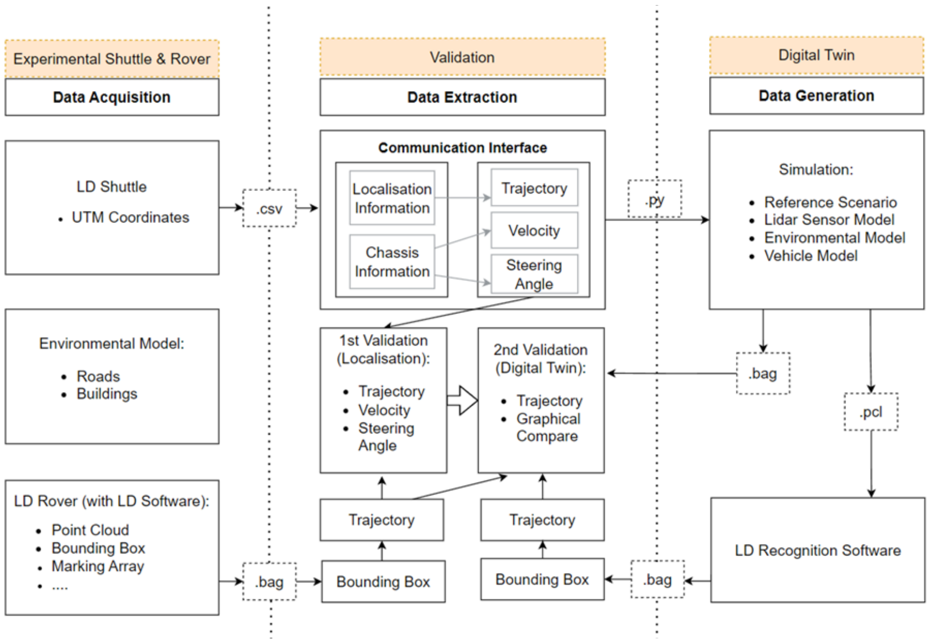

Figure 9: Flow diagram of the LiDAR validation experiment

The development of the use case has highlighted several points. Data derived from digital twins can be used for AI training purposes, if gathering data in the physical counterpart is too complex. To ensure efficient use of resources, the number of reproduced properties and their fidelity can be reduced. However, a very good understanding of the requirements is needed. In case of the described use case the reproducibility on point group level, not on single point level. Also, the limitations of the digital twin need to be regarded. For example, in case of this use case the limitations of CARLA concerning ray-casting: certain weather conditions with a drastic change in reflective properties, such as fog or heavy rain and snow, will not lead to realistic results. Also, corner cases, e.g., distant objects represented only by a small group of points, where the noise level on point level becomes critical, should be avoided.
If such limitations are kept in mind, using data derived from digital twins for AI training purposes can speed up the process, reduce costs, and can even be used to reduce bias that may be introduced by training data gathered in the physical world.

7 Conclusion and Future Work

This document outlines the present progress of our efforts in defining the prerequisites for DT-based RL learning. To outline these prerequisites, we initially offer a high-level perspective on Asimov's solution and then proceed to specify requirements for individual use cases.
This document is to be considered as a condensed summary of the prerequisites featured in the ASIMOV project's various deliverables. Hence, we advise diligent readers to refer to the published deliverables for more comprehensive information.
Requirements form an integral part of an ongoing project and are intended to be incorporated into most feature deliverables delivered by the ASIMOV consortium.

8 Terms, Abbreviations and Definitions

Table 5 - Terms, Abbreviations and Definitions

ABBREVIATION	EXPLANATION
ACC	Active Cruise Control
AD	Autonomous Driving
AEB	Autonomous Emergency Braking
AI	Artificial Intelligence
ASAM	Association for Standardization of Automation and Measuring Systems
DT	Digital Twin

EM	Electron Microscopy
KPI	Key Performance Indicator
LKA	Lane Keep Assist
LiDAR	Light Detection and Ranging
MDP	Markov Decision Process
RL	Reinforcement Learning
RSU	Mobile Roadside Unit
SEM	Scanning Electron Microscopy
STEM	Scanning Transmission Electron Microscopy
TEM	Transmission Electron Microscopy
UUV	Unmanned Utility Vehicle

9 Bibliography

[1] ASIMOV-Consortium, "Asimov-Project EU: D2.2 Methods and Tools for Training AI with Digital Twin M16," 30 11 2022. [Online]. Available: https://www.asimov-project.eu/news-and-publications.

[2] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[3] W. v. Heeswijk, "Rainbow DQN - The Best Reinforcement Learning Has to Offer," 08 12 2022. [Online]. Available: https://towardsdatascience.com/rainbow-dqn-the-best-reinforcement-learning-has-to-offer-166cb8ed2f86.

[4] M. Swiechowski, K. Godlewski and J. Mandziuk, "Monte Carlo Tree Search: A Review of Recent Modifications and Applications," *Artif. Intell. Rev.,* vol. 56, pp. 2497-2562, 2023.

[5] ASIMOV-Consortium, "Asimov-Project EU: D1.3 UUV Proof of Concept Demonstration M18," December 2022. [Online]. Available: https://www.asimov-project.eu/news-and-publications.

[6] ASIMOV-Consortium, "Asimov-Project EU: D1.2 STEM Proof of Concept Demonstration M18," March 2023. [Online]. Available: https://www.asimov-project.eu/news-and-publications.

[7] Triangraphics GmbH, "Trian3DBuilder," Triangraphics GmbH, [Online]. Available: https://trian3dbuilder.de.

[8] ASIMOV-Consortium, "ASIMOV Publication," [Online]. Available: https://www.asimov-project.eu/news-and-publications.

[9] Association for Standardization of Automation and Measuring Systems (ASAM), "ASAM OpenDRIVE," ASAM, 22 11 2023. [Online]. Available: https://www.asam.net/standards/detail/opendrive/.

[10] Open Source Community, "Documentation: CARLA Simulator," [Online]. Available: https://carla.readthedocs.io/en/latest.

[11] Association for Standardization of Automation and Measuring Systems (ASAM), "ASAM Organization," [Online]. Available: https://www.asam.net/about-asam/organization.

[12] Association for Standardization of Automation and Measuring Systems (ASAM), "ASAM ODS," 31 December 2022. [Online]. Available: https://www.asam.net/standards/detail/ods.

[13] Open Source Community, "MLFlow," [Online]. Available: https://mlflow.org/.

[14] ASIMOV-consortium, *ASIMOV - Full Project Proposal,* 2020.