Application to Systems - Methods and Techniques

[WP4; T4.1; Deliverable: D4.1 Version 1.1]

Public

AI training using Simulated Instruments for Machine Optimization and Verification

Document Information

Project	ASIMOV
Grant Agreement No.	20216 ASIMOV - ITEA
Deliverable No.	D4.1
Deliverable No. in WP	WP4; T4.1
Deliverable Title	Application to Systems - Methods and Techniques
Dissemination Level	Public
Document Version	Version 1.1
Date	2022.12.01
Contact	Niklas Braun
Organization	AVL Deutschland GmbH
E-Mail	niklas.braun@avl.com

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
Niklas Braun	AVL	Niklas.Braun@avl.com
Michael Wild	DLR	Michael.Wild@dlr.de
Elias Modrakowski	DLR	Elias.Modrakowski@dlr.de
Andreas Eich	LiangDao	Andreas.Eich@liangdao.de
Lukas Schmidt	Norcom	Lukas.Schmidt@norcom.de
Ezra Tampubolon	Norcom	Ezra.Tampubolon@norcom.de
Thomas Kotschenreuther	RAC	T.Kotschenreuther@rac.de
Faruk Caglar	TFS	Faruk.Caglar@thermofisher.com
Hans Vanrompay	TFS	Hans.Vanrompay@thermofisher.com
Ilona Armengol Thijs	TNO-ESI	Ilona.ArmengolThijs@tno.nl
Arjan Mooij	TNO-ESI	Arjan.Mooij@tno.nl
Roy van Zuijlen	TU/e	R.A.C.Zuijlen@tue.nl

Formal Reviewers

Version	Date	Reviewer
1	2023.01.23	Pieter Goosen (TNO), Remco Schoenmakers (Thermo Fisher Scientific)

Change History

Version	Date	Reason for Change
0.1	2022.10.14	Initial Version
0.2	2022.11.21	Ready to review Version
1.1	2023.03.10	Updated Confidentiality

Abstract

This document aims to provide an overview of what matters when applying the ASIMOV solution to a physical system. Several methods and techniques for aligning the Digital Twin (DT) to its Physical Twin (PT) are discussed and a process for the selection of the right performance measurements is proposed. The focus is on the continuous runtime validation of the DT and how it differs from the initial validation. Furthermore, besides monitoring and aligning the DT and PT, similar concepts are presented to validate and improve AI performance during system use. All the proposed methods are embedded into the broader concept and structure of ASIMOV and examples from use case applications are given.

Contents

Table of Figures

Table of Tables

1 Introduction

One of the core principles of ASIMOV is that the training of the optimization AI mainly relies on the use of digital twins. As the main goal is to control a physical system in the end, it is essential to find methods and techniques to apply this optimization on the physical system. Furthermore, techniques for updating the digital twin are needed. This document introduces methods and techniques of such kind.

The document is separated into a section describing the synchronization of digital twin with its physical counterpart, followed by a section about the AI optimization, which takes on the topic of bringing the optimization to the physical system, as well as developing KPIs to quantify the success of the optimization.

1.1 Nomenclature

Here we give the definition and interpretation of terms that are used in this document. Some definitions are repeated from the ASIMOV Task 2.1 document.

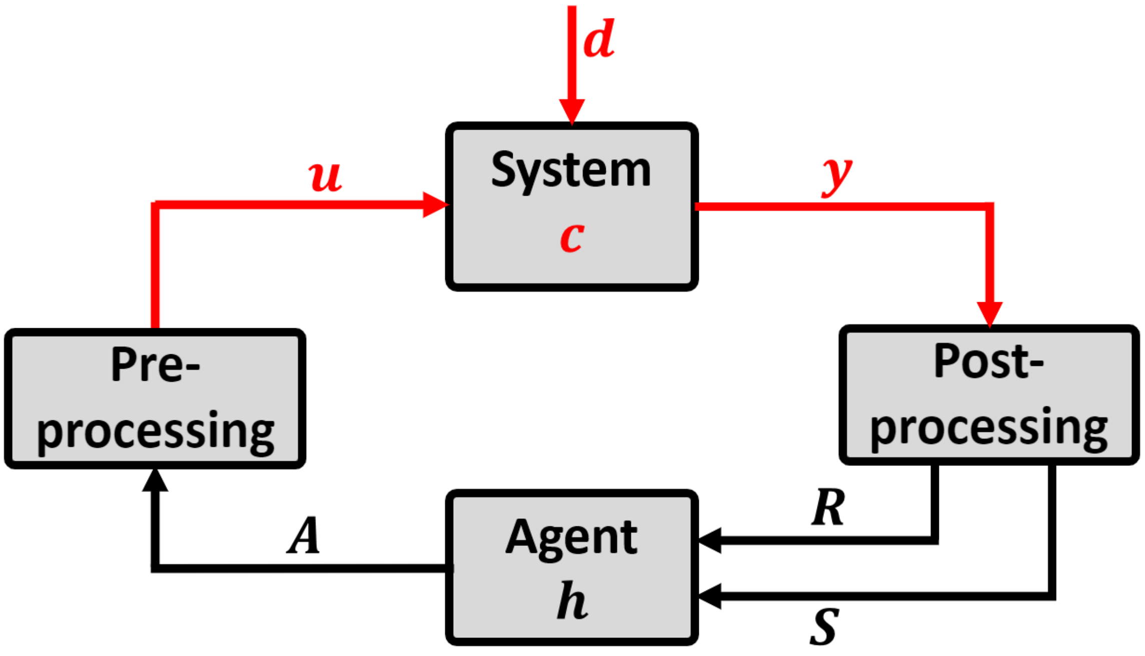

Figure 1 - Architecture of the ASIMOV project with the parameters of interest for this document marked in red. [1]

Control input (u). Parameters that can be changed directly, either by the operator or the RL Agent.
Output (y). Measurement Value / response of the system. "The outputs are all available signals from the system. (e.g., sensor values)" [1]
System parameters (c). Parameters that won't be changed during twinning but were set in previous development phases of the model, e.g., masses of system components.
Disturbances (d). Inputs of the system that cannot be controlled by the RL Agent or the operator but do have an effect on the internal state of the system. In the context of this document, we assume, that a subset of all possible disturbances that influence the PT can be measured and are a disturbance input to the model of the DT
Hyperparameters (h). Configuration of the Agent, including its neural net configuration, e.g., number of layers in an artificial neural net.
Observables. These are all parameters that can be observed directly. They may stem from u, c, and y. They are of special interest here, since they can be used to compare the state of the DT and the PT.
Non-observables. Parameters, whose value cannot be observed directly, but carry information about the state of the system.

Tuning parameters. Tunable parameters, that can be adjusted within certain ranges to make the output of the DT and the PT match during the twinning process. They are a subset of c.
KPI. In this document there are two types of Key Performance Indicators (KPIs). The ones that mirror how well the DT and the PT are aligned, and the ones that quantify the success of optimization (on a DT and/or PT).
Runtime validation. This term means the validation that focuses on parameters that change during runtime, in contrast to the validation from T2.5, which is concerned also with static parameters. It further means that the digital twin of one specific physical asset is validated.

1.2 T4.1 in the ASIMOV Landscape

We assume that the initial validation of the DT and RL was done with the methods detailed in T2.5 and T3.4 respectively. The state and scope of the development of these methods will be made available in D2.5. This especially includes the validation of the following aspects of the DT:

- Conceptual considerations in earlier phases of the lifecycle of the DT
  - Architecture
  - Sub modules
- Static internal parameters, that do not change during runtime
  - System Configurations for DT alignment

Figure 2 illustrates the different aspects of a DT that must be validated for use of the DT in the operational phase. The DT consists of an Adaptive Model, that has some static configuration, and a dynamic part, that can be adapted by the Adaptation Model. The Adaptation Model compares the DT and PT behavior, to adapt the DT.

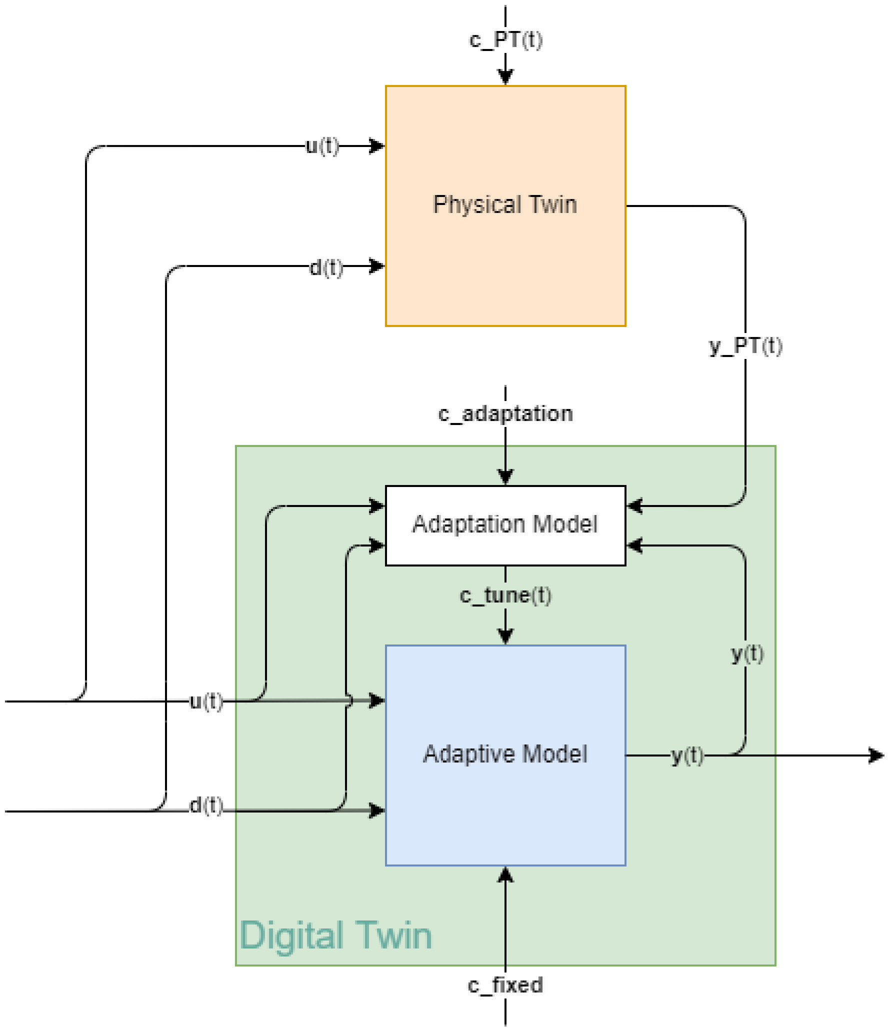

Figure 2 - Inner models of a Digital Twin

The controllable system input u(t) as well as the uncontrollable disturbances d(t) are provided to the DT as well as the PT. It must be noted that although d(t) is provided to PT and DT, the DT will most likely not have sufficient models to use all d(t) that influence the PT. C_PT(t) is the configuration of the PT, which is in general only partly observable and is not necessarily static. y_PT(t) is the resulting output of the system. The DT can be seen as composed of two parts, the Adaptive Model and the Adaptation Model. Ideally, providing the same u(t) and d(t) of the physical system as input to the Adaptive Model, it outputs a similar y(t).

In practice, there will be some difference between the output of these systems and a static fixed initial system configuration c_fixed is unlikely to be found to work immediately in all operational stages of the system. Because of that, there is the possibility to tune the Adaptive Model via the Adaptation Model. It can tweak the parameters c_tune(t) and therefore adapt the DT to the PT by comparing their system behavior and changing the DTs behavior. Using the right methods to adapt the DT during runtime is part of this task.

How such an adaptation can be interpreted can be seen in Figure 3. It shows a schematic of the dynamic behavior of some parameter p, that can be observed in the DT and in the PT. At each twinning (at times $t_{twin,i}$), the DT is tuned to match the value measured in the PT, by adjusting the tunning parameters. Ideally, the intervals between twinnings increase as time progresses, and the deviation between the predicted parameter value and the physical one decreases with the number of twinnings, since the internal dynamic model is being calibrated to match the PT.

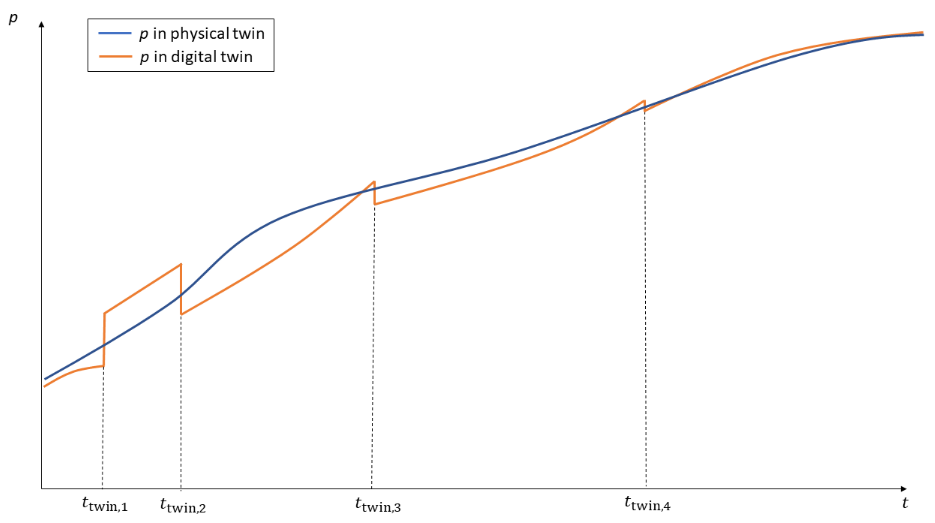

Figure 3 - Twinning Process

In this task we focus on methods to *validate* **changes** of the DT with respect to its physical counterpart, the Physical Twin (PT). Naturally there is some overlap with the methods for the initial validation, but we aim to find methods that allow us to only validate changes in a more efficient way than validating the whole DT with the same methods as the initial validation. These methods will be written down for the use cases STEM and UUV.

2 Digital Twin

2.1 Digital Twins in the ASIMOV Context

The DT, in the context of ASIMOV, can be seen as a virtual playground for the RL agent in which it can be trained without having to interact with the PT. It serves as a safe, always available and fast alternative to the PT. The DT can be used in two different ways depending on the current phase of the ASIMOV approach:

1. During the training phase, system parameter variations can be instantiated that use all the internal structures of the DT prototype, but that are perhaps not connected to a specific PT. Multiple of these variations can be used as a virtual fleet to simulate a diverse field of systems, each with slightly different behavioral properties. Such a virtual fleet can be represented by multiple instances of the same product, with each being slightly different due to virtual manufacturing imperfections or external disturbances. Another way to use virtual fleets is to vary system configurations more substantially and creating a fleet of consisting of different products in a product family. The instantiation of these variations can be done in multiple ways, ranging from a random initialization of a given property distribution all the way to a problem-oriented initialization. Details can be found in [2]. This virtual fleet containing multiple variations is useful in the training phase, but it is only a fictional representation of physical systems and therefore will always behave slightly differently compared to a physical system.
2. As soon as a physical system is available, the virtual fleet can be expanded by adding physical instances. Instead of directly adding the physical systems to the pool of available systems for the RL agent to learn on, their DTs are added. This is the second type of DT, that is used in ASIMOV. It is strictly tied to an instance of a physical system - its PT- and is used during the fine-tuning phase. During that phase, the DT and PT run in parallel while the DT is adapted to match the PTs behavior. This DT also uses the DT prototype structure and therefore also provides the possibility to tune the DT prototype to enhance the behavior of the virtual fleet and bring the variation behavior closer to the physical systems. The more physical systems and therefore PT-connected DTs are added to the fleet, the better the DT prototype becomes. These connected DTs can of course also be used for monitoring purposes and be a useful tool to estimate the state of the physical system based on insights into models that run in its digital sibling. It furthermore enables the possibility to test RL actions on the DT, to see if they are safe to execute on the PT.

The DTs, which are connected to their PTs, as well as the variations - more specifically called the System Configuration Variations in this document - which form the virtual fleet, both originate from the Digital Twin Prototype. Each of these PT-connected DTs as well as the System Configuration Variations are connected to their own RLA Instance, which itself is derived from the Reference RLA. During training, RLA instances connected to variations and PT-connected DTs are trained per instance. The generated data can additionally be stored in a central Experience Storage of the Digital Twin Prototype. Federated learning methods can then be used to integrate the instance specific training progress of the RLA Instances into the Reference RLA. This ensures that the Reference RLA itself is trained with respect to multiple different instances. A newly connected PT-connected DT can then profit from the better zero-shot performance of its RLA instance, which is derived from the Reference RLA. Less required fine-tuning of the RLA instance is the consequence. [2] [3]

Like the Reference RLA, that profits from training progress of RLA instances, the Digital Twin Prototype can be improved by transferring the system behavior observed by PT-connected DTs. It has to be kept in mind however, that the Digital Twin prototype does not only represent a typical DT instance, but also provides the limits of system configuration. When deriving System Configuration Variations from the Digital Twin Prototype, these limits, in combination with distributions derived from the actual behavior of PT-connected DTs can be used to create more meaningful System Configuration Variations.
An overview of these conceptual idea can be found in Figure 4. The concept was initially proposed in [2].

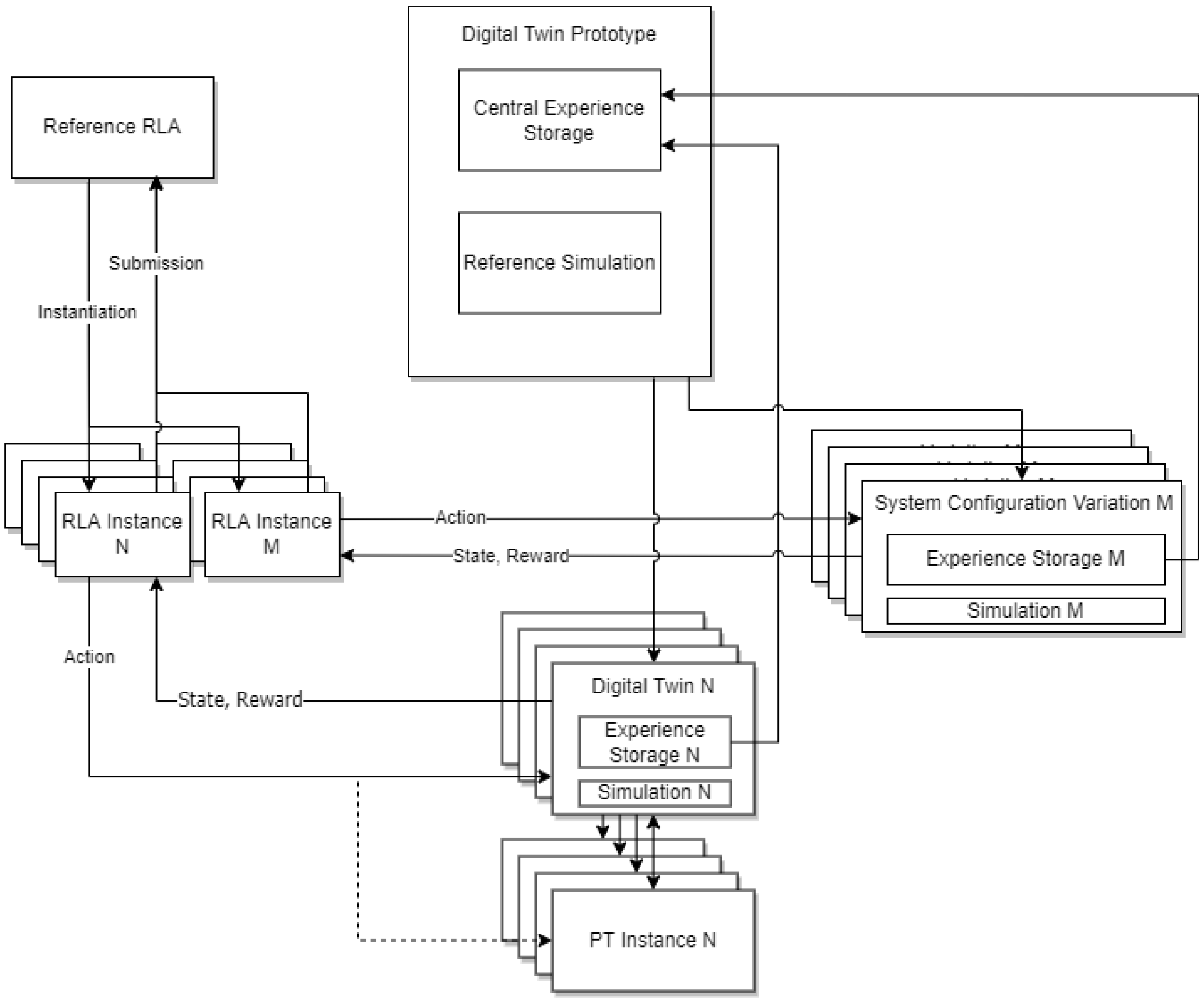

Figure 4 - Different Kind of DTs

2.2 Requirements for Runtime Digital Twin Validation

Validation is always done keeping the purpose of the model in mind. In T2.5 the purpose of the DT is to produce data that can be used to train a RL agent, that can be applied to a PT. Here the purpose of an accurately adapted DT is to train further RL improvements and to monitor the system state as safety checks. This requires that the DT and PT are comparable, i.e., that they do not diverge above defined thresholds during runtime. The allowed differences link back to ASIMOV T2.1 and the accuracy of the parameters. [1]
In general, the methods for comparing DT and PT during runtime, which is the focus of T4.1, will contain a subset of methods available for the initial validation, dealt with in T2.5. With the runtime validation being the focus, there are additional requirements for checking the validity:

- The computational cost of calculating the KPIs must be low enough to make continuous runtime validation feasible.
- The subset of evaluation metrics must be automatically executable.
- The data being used for calculating the KPI must be available during normal operation of the PT.

These limitations will limit the synchronization process to a data-based runtime validation and will not take the structure of the DT into account.

As DTs shall not only be validated during runtime, but also synchronized to match their PTs behavior, an additional requirement must be met. The adaptation of the DT relies on data-driven methods, for which it is essential, that the difference between DT and PT also provides a quantitative comparison of their behavior.

The following sections will focus on the process for synchronizing the DT with its PT instance.

2.3 Comparing Physical and Digital Twins

As the DT should represent its physical counterpart, a central part of the ASIMOV solution is about how to compare the DT with its PT. This does not only help during the creation phase of such a DT, but it also allows for continuous runtime validation in the operational phase. To do this comparison, some techniques are needed to compare and quantify the difference between the data generated by the digital twin and the physical system. As the data output can contain multiple different data types, several different comparison techniques will be applied.
A diagram of the overall process can be seen in Figure 5.

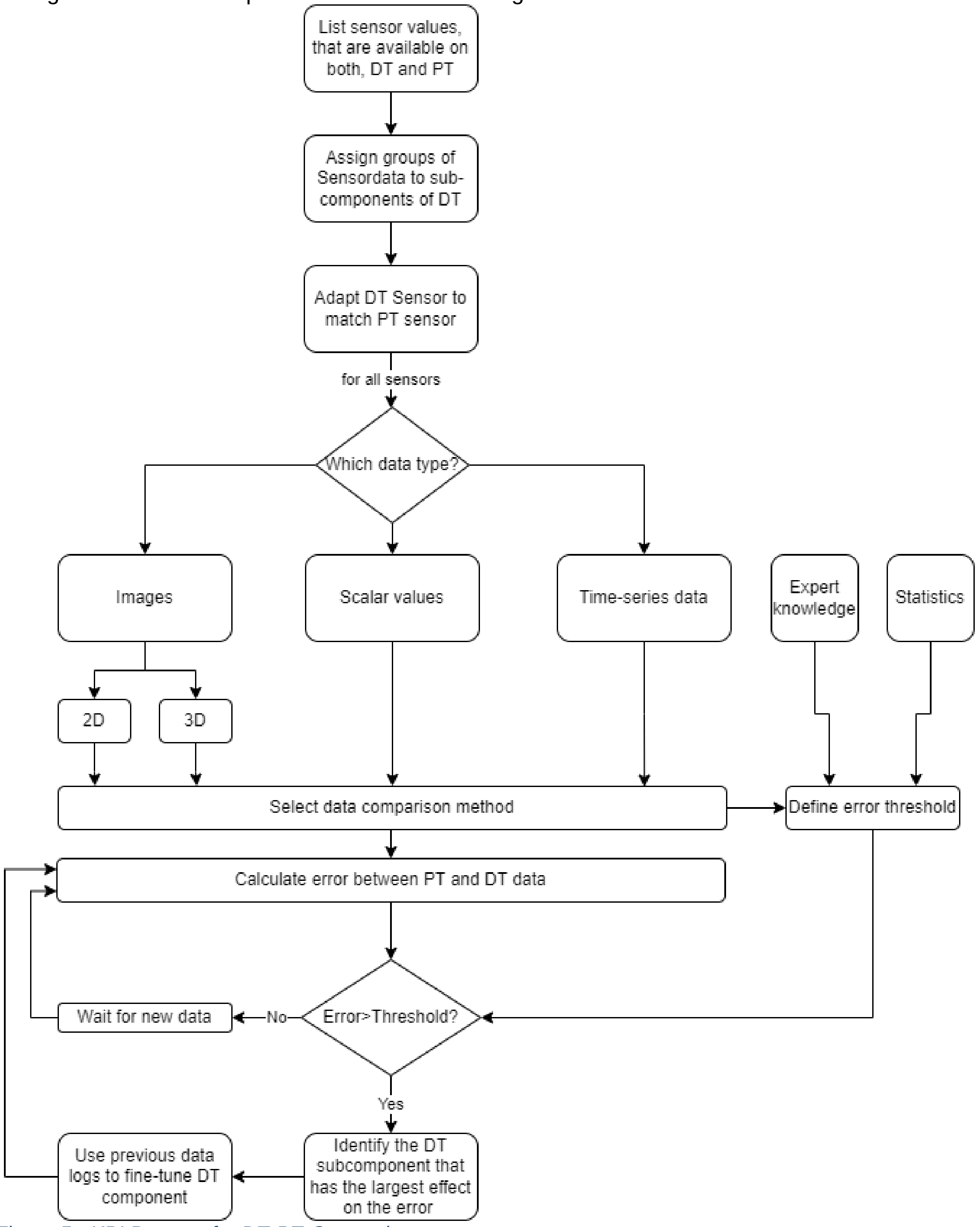
List sensor values, that are available on both, DT and PT
Assign groups of Sensordata to sub-components of DT
Adapt DT Sensor to match PT sensor
for all sensors
Which data type?
Images
Scalar values
Time-series data
Expert knowledge
Statistics
2D
3D
Select data comparison method
Define error threshold
Calculate error between PT and DT data
Wait for new data
No
Error>Threshold?
Yes
Use previous data logs to fine-tune DT component
Identify the DT subcomponent that has the largest effect on the error

Figure 5 - KPI Process for DT-PT-Comparison

The diagram states listing all the available sensor data from DT and PT as the first step. This helps to later identify which are possible values to be used for KPI calculation and therefore limits the subset of applicable KPIs.
Furthermore, it is useful to also assign groups of parameters to sub-components of the DT. This helps identify which sub-component has the greatest effect on a possible mismatch of DT and PT data and therefore enables either automatic adjustment of the respective DT component or helping to diagnose the model's shortcomings, if no automated correction is available. A staged or hierarchical approach for KPI calculation is also possible for better identification of the malfunctioning DT component.
Given the selection of available sensor data, the properties of the measurement data itself must be examined. This includes the measuring frequency, resolution of the sensor, etc. Only if the virtual sensor data and the physical sensor data is comparable in these properties, a suitable comparison of the resulting properties of the virtual and physical components is possible. If this is not the case, the DTs sensor model must be adapted to match the PT. The consistency of datapoints is important, as asynchronous datasets introduce additional errors, not originating from a false model, but from errors of the measurement properties itself.
Next up, the identification of the data types from the dataset to compare DT and PT is relevant. Depending on the data type, there are different suitable KPIs available for comparison, which also require a threshold for acceptance. The acceptance criteria can be defined using experts' knowledge, or statistical methods to provide continuous improvement of the DT. Should the system at hand provide the possibility to assign KPIs to evaluate intermediate results and components, their respective KPIs can be hierarchically linked like the components they refer to. This allows for easier identification of misaligned components. Once the sensor data KPIs have been found and possibly hierarchical KPIs have been defined, the runtime execution of KPI calculation can be executed in multiple parallel instances. Each KPI instance runs in a continuous loop, where new data is being checked in terms of consistency between DT and PT. Should a misalignment between virtual and physical sensor data, indicated by a low KPI value, be detected, a user notification can be sent out for warning. Should multiple KPIs indicate a severe difference between DT and PT, the faulty DT component can be identified using the hierarchical KPI approach.
If automated alignment methods are available, the DT component in question can be tuned to fit the PT sensor data, similar to an approach described in [4]. Depending on the suggested parameters of the DT subcomponent that represent the adjusted behavior, a plausibility check and therefore online monitoring non-directly measurable parameters of the PT can be made available. An example of this could be the measurement of the rolling resistance of the vehicle, which can be measured easily on a testbed. If only for example certain parameters of the vehicle dynamics model and the vehicle dynamics itself are available, the DT model can be used to estimate the non-directly measurable parameters, which can be checked for plausibility. This adds another layer of safety and confidence to running complex vehicles on a testbed.

2.3.1 Image Comparison

In the electron microscope use case, the Ronchigram images are the key to finding the right parameters for the microscope's lens system. A Ronchigram corresponds to the electron probability on the diffraction plane. In simpler words, it is equivalent to the square of the modulus of the Fourier transform of the electron wave, after interaction with the specimen.

Key features in a Ronchigram which can be exploited to measure the aberrations are the Ronchigram's symmetry and magnification. When in focus, the center of the Ronchigram is a high local magnification that represents the aberration-free portion of the electron beam. Further away from the center, thus moving away from the optical axis, aberrations reduce the local magnification. Having an as large as possible magnified central region is a first indicator for an optimal defocus. Figure 6 shows these effects. The presence of asymmetric aberrations breaks the rotational symmetry of a Ronchigram. Two-fold astigmatism unidirectionally stretches the region of high magnification, thereby producing distinctive streaks. Axial coma shifts the center of the Ronchigram. Therefore, the ASIMOV solution should include a system that can compare the images generated by the physical microscope and the DT with focus on the symmetry and structure of the Ronchigram. One representation that highlights the symmetry of the Ronchigram is its Fourier Transform. Comparing the Fourier transform of the DTs and PTs will lower the influence of the (synthetic) specimen on the observed features and provide a more abstract way to compare the PT to the DT in a rigorous manner.

We fully expect that a data representation which allows the AI to train on data from the DT, and generalize on data from the PT, are well suited for evaluating the validity of the DT. Such data representations namely condense all information into a simpler representation and extract the essence of the provided data which is needed for an AI to generalize from the DT to the PT. Even more so, if the AI would extract features itself, these features would be instrumental for comparing the data generated from the DT to the data provided by the PT.

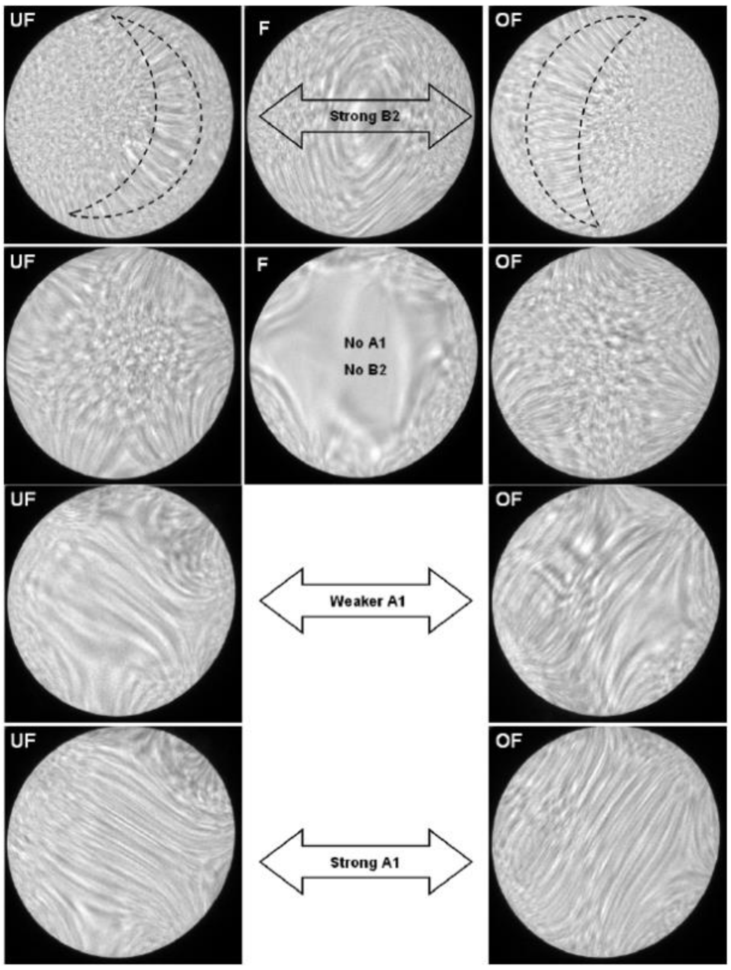

Figure 6 - Aberration in Ronchigrams

2.3.1.1 <u>Histogram Method</u>

Histograms offer a way of comparing images, while also providing the possibility to quantify the difference, which can be used as a quality KPI for the DTs output. To apply the histogram method features of the images are needed. A simple feature would be the brightness value of each pixel. To compare two images, a set of so-called bins will be created. Each bin represents a certain range of brightness. Next, every pixel's brightness value gets assigned to such a bin. After that, the number of assignments to each bin gets normalized by dividing it by the total number of pixels in the image. For comparing multiple images with each other, the normalized bin allocations can be subtracted from each other, which yields their difference.
Obviously one such feature is not that indicative and several of these features need to be combined, to get a more robust comparison. For the electron microscope's stretched lines, edge detection, followed by a histogram of their direction could provide helpful insight. Also, splitting the image into multiple smaller patches and calculating histograms for these regions can serve as useful features.
There are multiple publications around edge detectors as well as the histogram method available, such as [5] or [6].

2.3.2 Universal Data Comparison

The methods mentioned in the previous section focus on comparing images. While this is an essential technique, especially in the Electron Microscopy context, other types of data are generated and need to be compared in other domains. In the Unmanned Utility Vehicle use case, sub-components of the DT, such as the vehicle dynamics model need to be compared to the physical vehicle dynamics by relying on the comparison of sensor values. Such sensor values could be accelerations, measured by IMU sensors, individual wheel speeds or vehicle controls like accelerator, brake pedal and steering angle values. As these are scalar values, other techniques are necessary. Some techniques are described below.

2.3.2.1 Visual Comparison

One easy way to compare data of two different sources is the graphical comparison, which can be done by plotting the datapoints of the datasets in question in one diagram. After a preprocessing step which ensures that the datapoints are time-wise synchronized to each other, a visual inspection of the datapoints can be done. Typically, such a method is used when comparing simulation and physical vehicle tests in terms of speed, forces, etc.
It provides a good indication of the similarity of multiple datasets but relies on the subjective interpretation of a human. Therefore, the interpretation process does not yield quantitative information about the similarities and cannot be automatically processed. This restricts its application in the DT/PT data comparison and generally can be seen as not useful for runtime validation.

2.3.2.2 Scalar data Comparison

In many situations, the comparison of minimum and maximum values can give a good indication of how similar multiple datasets are. The depth of information is obviously limited, as only the extremum values of a signal are compared.
This technique can also be used for the comparison of DT and PT data. It can be automated and is objective. It cannot be used as a sole metric, but it must be combined with other techniques as it is very sensitive to noise and outliers. Furthermore, a similarity in extreme values does not indicate similar system behavior.

2.3.2.3 Timeseries Comparison

While scalar data limits the information received only to a small part of the whole dataset, time-series comparison focuses on the extraction of information from each data element. For that, a comparison of the data at each timestep must be performed and the results aggregated. A comparison technique can be the Euclidean distance of the datapoints at the same time step. The aggregation over the whole dataset can be realized through a sum or an average. A set of popular error metrics or forecasting KPIs are described in [7] and [8]:

- **Mean Absolute Percentage Error (MAPE):** calculated by the division of the error with the actual value, averaged over a timespan.

  $$MAPE = \frac{1}{n}\sum_{T}\frac{|y_{sim}(t) - y_{real}(t)|}{y_{real}(t)}$$

  It has the big disadvantage, that it is not defined for actual values equal 0. Furthermore, errors where the actual value is low, are weighted higher than errors, where the actual value is higher. Especially for datasets, which contain positive as well as negative values, this metric is not applicable. There are variations of MAPE available, such as described in [9], which try to solve these issues.
- **Mean Absolute Error (MAE):** calculated by the absolute error, averaged over a timespan.

  $$MAE = \frac{1}{n}\sum_{T}|y_{sim}(t) - y_{real}(t)|$$

  The MAE is not scaled with the actual value, which makes it less sensitive for upscaling errors near 0 and also applicable in circumstances where negative values can occur. Its values are not linked to the actual value itself, which is why multiple MAEs cannot be compared directly to each other.

- **Mean Absolute Error Percent (MAE%):** takes the MAE and divides it by the average of the actual value.
$$MAE\% = \frac{\sum_T |y_{sim}(t) - y_{real}(t)|}{\sum_T y_{real}(t)}$$
This still makes it inapplicable in cases where the actual value is 0, as long as the average is not, but it eliminates the upscaling of errors in cases where the actual value was low, while still providing some comparability between different MAE%s.
- **Root Mean Squared Error (RMSE):** calculates the squared error, averages that over the whole timespan and then calculates the square root.
$$RMSE = \sqrt{\frac{1}{n}\sum_T \left(y_{sim}(t) - y_{real}(t)\right)^2}$$
The RMSE is sensitive to big error terms, as it squares them. This leads to unequally weighted errors, which overrepresent outliers. It can be used for all value ranges and a modification to make the RMSE of different values comparable, similar to MAE%, is available.
- **Root Mean Squared Error Percent (RMSE%):** takes the RMSE and divides it by the average of the actual value.
$$RMSE\% = \frac{\sqrt{\frac{1}{n}\sum_T \left(y_{sim}(t) - y_{real}(t)\right)^2}}{\frac{\sum_T y_{real}(t)}{n}}$$
Similar to MAE%, RMSE% improves the comparability, but introduces the problem of not being applicable in cases where the average actual value is 0.
- **Normalized Mean Squared Error (NMSE):** An approach, presented in [10] divides the Mean Squared Error by the standard deviation of the actual values, which makes it easier to estimate the model's accuracy with a higher variance dataset. This approach can also help answer the question of how accurate the model has to be, as the physical datasets variation can be used as a target benchmark.
- **Coefficient of determination (R^2):** calculates how good a model explains the observed behavior by a comparison of the variation of the real data with the simulated data [11]:
$$R^2 = \frac{\sum_T (y_{sim}(t) - \bar{y}_{real})^2}{\sum_T \left(y_{real}(t) - \bar{y}_{real}(t)\right)^2}$$

These techniques are applicable when comparing DT and PT data over a time period. It is necessary, however, to either have datapoints from the DT and the PT for the same time steps, I.e., having the same sample frequency with no phase shift, or using countering methods like interpolation to overcome this issue. As the DT, as well as the PT are part of the whole optimization process and it is the goal to mimic the PTs sensors in the DT, such synchronization can be implemented and it can be assured, that the two datasets are synchronized.
The aggregation of the comparison values requires a window or timespan which is used for aggregation. This would lead to a slight delay in perceiving differences between DT and PT.

2.3.2.4 Anomaly Detection

When comparing digital twin and physical systems behavior, the task can be abstracted to identifying datapoints that severely differ between DT and PT. Depending on which system is seen as baseline, the datapoints from the other system can be classified by an anomaly detection.
In [12] multiple different methods for identifying anomalies were considered. This section now focuses on a replicator neural network (RNN) based anomaly detection.
These neural networks are trained in a way that their inputs match their output and therefore represent an autoencoder structure. A risk in this approach is that the neural network learns the identity function and just passes the input values straight to the output. To eliminate that risk, sparsity terms and other regularization techniques can be used. Additionally, by having at least one hidden layer with fewer

neurons than input and output neurons in the outer layers, compression of information can be forced by structure, so that the identity function can no longer be learned by design.
The autoencoder is then trained with representative data from the PT and the reconstruction error, meaning the difference between the input and output of the replicator neural network, is minimized. During the operational phase, the DT's data serves as input for the neural net. A mismatch of DT and PT data can then be identified by evaluating the reconstruction error, which can be seen as the anomaly score. A high reconstruction error indicates a severe difference between DT and PT data, which can be the result of a misaligned DT component. As this score is present for every dimension of the data, the individual contribution of each signal to that anomaly score can be identified as well. This can be used to determine which component of the whole DT needs to be adapted to align the behavior.

2.4 Trigger Digital Twin Update

As soon as a physical system is connected to its DT, a mismatch of DT and PT behavior is inevitable due to effects present in the PT that have not been modeled in the DT. Even a slight drift in system configuration parameters c or disturbances d could lead to a misalignment. Depending on the situation in which this happens, different countermeasures can be taken.

2.4.1 Monitoring

In case the primary use of the DT is currently to monitor the system, it is usually the case that the live data from the physical system is used for assessing the system's state in terms of safety and current operating condition. The DT in that case can be seen as the tool that provides this access and uses its internal model to provide deeper insights compared to the PT alone.
Should the KPIs, linked to either the output of the DT or individual components of the DT, detect an undesired behavior, exceeding a certain predefined limit, an DT update is necessary to realign the twins. A KPI-based monitoring of the alignment of DT and PT offers an easy way to define DT updates. Due to the constant measurement of alignment, a tolerance for deviation is ensured. Depending on the necessary accuracy of the DT, the KPI limits can be set accordingly and used as a trigger.

2.4.2 AI Training

In order for the DT to be aligned with the PT when an optimization cycle is happening, the latter can serve as an additional trigger for the twinning. That way it can be assured that the DT, the AI uses for training is up-to-date, even if the DT would still be inside valid KPI ranges for monitoring.

2.4.3 Event-based

As DT updates require data from the PT to be available, it is useful to trigger DT updates based on the PT data availability. To further limit the number of twinnings required, expert knowledge can be used to define certain events that can trigger an update. One possibility to do that, is to define input and state combinations that are well understood and are modeled with the required accuracy in the DT. If such a scenario occurs, deviations between DT and PT are more meaningful and their alignment offers greater benefits across the relevant operational range, compared to the twinning based on data gathered in less important edge cases. The definition of these twinning-relevant scenarios requires expert knowledge and knowledge about the operational conditions of the system.

Besides entering such an important operational region, predefined inputs, such as the change of an operational mode, can also be used as event-based triggers. This also allows for experts to include their knowledge about which configuration of the system alters its behavior severely, so that it is very likely that retuning of the DT is needed. An example for that could be altering the voltage of the electron source in the STEM use case, or the selection of a different driving scenario in the UUV use case.

2.5 Validating DT Changes

The following section is about how to validate the changes to the DTs, that would occur during the lifecycle of a DT-PT combination.

2.5.1 General Use Case

Validating DT changes can in theory be triggered with every twinning (physical to virtual and virtual to physical). It might however not be necessary to validate the changes every time, especially when we take the adaption model as shown in Figure 2 into account. Consider the situation, that the models that govern the dynamic behavior of the DT (inside the adaptive model) can be updated correctly by the models in the adaptation model, the changes do not need to be validated. In other words, if deviations stem from a phenomenon that is well understood and implemented in the models, the adaptive model can change to account for these deviations.
We need to think about what observables have to be considered when validating changes. One way to derive a set of observables is to first list what is observable in the DT and physical system in principle separately and then find the intersection of these two lists. These observables (or a subset of them) can be compared. As a lower bound for an acceptable difference, the standard deviation from N observations on the physical system could be used. Here, “lower bound” is to be understood as that it would not make sense to check for a difference smaller than this noise of the observable. To construct further observables the monitoring concept could be used. It requires a formal specification of the condition that shall be satisfied.

2.5.2 STEM Use Case

The methods introduced in the previous chapters shall be mapped to the ASIMOV use cases. The STEM use case follows the storyline provided in this chapter.
Story:
1. A RL agent is trained on a digital twin that mimics the behavior of the physical system, to minimize defocus and two-fold astigmatism based on Ronchigram images.
2. When alignment defining parameters are altered (e.g., high tension of the electron source), a long period of time has elapsed or when the position of imaging has been shifted, an update of the PT is warranted.
3. The RL Agent adapts the currents through the lenses, stigmators, etc. within the electron microscope, such that the aberrations are minimized and the desired resolution is reached (virtual to physical twinning).
4. The experimentally acquired Ronchigram can be compared with the simulated one (physical to virtual twinning).
    a. The comparison happens on a set of metrics that were derived with image processing or AI-based techniques.

Ultimately the RL agent should learn to stop, when the system has been optimized.

2.5.2.1 Technical Approach

Validating changes might be more straightforward than in the UUV use case. The PT can be directly connected to the DT. Hence, we foresee to train the RL agent extensively on the DT and only allow fine-tuning on the PT if for example
- The RL agent is connected to a new electron microscope and/or new unseen settings (e.g., high tension of the electron source).
- Drift between the DT and PT has been detected.

By comparing the Ronchigrams obtained from the DT and the PT we can establish whether such recalibration is required.

2.5.2.2 Available Observables

This section gives a short overview of possible observables that are available for comparing DT and PT during operation of the STEM.
- DT and PT
    - Convergence angle (DT, PT)
    - Sample thickness (DT, PT)
    - Beam Energy (DT, PT)
    - Defocus (DT, PT)
    - Two-fold astigmatism value (A1) (DT, PT)
    - Ronchigram Image (DT, PT)

- DT and not PT
    - Sample thickness
    - Sample characteristics
    - Current sensor of the current driving the EM
    - Temperature sensors of the various Optical Elements
    - The electron source specifications
    - The currents through the various optical elements
    - The physical position of the different modules within the electron microscope

2.5.3 UUV.1 Use Case

Similar to the STEM use case, the UUV.1 use case can be formulated as a sequential story.
Story:

1. A newly constructed UUV shall be calibrated for its operational design domain (ODD).
2. The RL Agent proposes a set of scenarios that shall be tested in order to ensure safety. The scenarios were selected to optimize criticality and novelty.
3. The new UUV is calibrated digitally, using these scenarios.
4. For the validation of the changes, the physical UUV has to be put on the testbed. Then the scenarios from 2. are applied. The observables (e.g. steering angle, pixel values of camera sensors, torque, …) of the UUV are observed.
5. The observables from (4.) were observed in the simulation in (2.) as well. Now they can be compared.

2.5.3.1 Technical approach to register changes

The plan is to use anomaly detection to register (and potentially quantify) changes between the DT and the PT. The latter is to be understood as the physical vehicle on a testbed. Keep in mind that it is expensive and cumbersome to put the physical vehicle on a testbed, so it is probably not feasible to validate with each twinning process. As long as the ODD of the UUV does not change, it is not necessary to do a virtual to physical twinning. The physical to virtual twinning ensures that there is no unintended behavior of the UUV. As an example, we can think about *the wear* of the tires, the axles, the steering, or the motor, that lead to a different behavior of the vehicle, when for example the same torque is applied. These kinds of effects need to be monitored. The anomaly detection from above can serve as an indicator that the UUV behaves differently to the way it was intended. The inputs that are given to the RNN can potentially be chosen in a way that they show anomalies during normal operation in reality, so not only when the vehicle is on the testbed. An observation of these anomalies could result in a 'call for calibration' to the UUV. For this it has to be put on the testbed, where the physical to virtual twinning can happen. The RNN is adjusted (retrained) on the DT that now mirrors the state of the UUV again. If wear models are included (and validated) in the DT, this 'call for calibration' might not be necessary. The DT can be adjusted with these 'wear models', which could lead to an explanation of the heightened anomaly value, and respectively to an adjustment of the DT. In reality, effects like the finite measuring frequency have to be considered. A difference in the phase and frequency of those measurements in the DT and the PT can lead to an ostensible difference in observables, which might become obsolete, if the measuring frequency and phase is synchronized prior to the comparison.
As an example, we can think about the wear of the tires. If the model controls the rotational speed of the wheels, but the rubber of the wheels was abraded by let's say one millimeter, the position of the vehicle won't be accurate anymore. This inconsistency might pop up as an anomaly. If the DT would now contain a 'wear' module, that predicts this one-millimeter wear after let's say ten thousand kilometers, a recalibration on the testbed would not be necessary.

2.5.3.2 Available Observables

This section gives a short overview on possible observables that are available on the DT and PT of the Testbed and Vehicle combination, as well as Real World Driving. Such an overview is necessary for identifying, which components of the DT can be compared to their real siblings and which components are available for fine-tuning during the operational phase.

- DT and PT
    - Ego Velocity

  - Ego Acceleration
  - Ego Steering Angle, Brake and Accelerator
  - Traffic Object Positions
  - Distance to Lane
  - Ego Weight(-distribution)
  - Rolling resistance
  - Aerodynamic properties
  - Environment configuration
  - Overall: Every observable describing the environment (as it is identical in both cases), vehicle observables that can be measured from the outside because of their interaction with the testbed. Internal vehicle observables only if they are transmitted via the vehicle bus system and the proper database file is available.
- DT, PT and Real World Driving
  - No ground truth for traffic participants
  - No environment configuration
  - Overall: Same as on the testbed but excluding the environmental configuration parameters and ground truth for the sensor data.
- DT and not PT
  - Internal Sensor Parameters
  - Intermediate Results of the perception
  - Driving function parameters
  - Calibration values to align camera and lidar input
  - Overall: Everything that is directly calculated in the vehicles ECU and is not published via a bus system. Additionally, depending on the available measurement equipment. Data from the vehicle bus is only available to a certain extent.

2.6 Fine Tuning Phase

It is to be expected that there will be some deviation between the DT and PT in terms of their behavior. Reasons could range from wear of the PT, temperature variations within the EM, to uncertainty in the chosen parameter values of the DT during the initial creation of the DT, due to an insufficient amount of data.
During runtime of the whole Optimization system, a refinement of the optimization can be done by using the data gathered on one or even different systems of a fleet. Such a fleet of multiple DT-PT pairs is also proposed in the DT overview provided in Figure 4. Hereby, the optimization of the DT itself can be as valuable as the refinement of the RL agent. Especially when applying the system to multiple instances of the same kind, an accurate DT helps to train the optimization system in the most efficient way.

2.6.1 General View

When refining the DT, it is essential to identify where a deviation between PT and DT comes from. Ideally, the component that is causing this misalignment can be identified by using an ensemble of different KPIs. The ensemble should consist of KPIs that each cover the deviation of individual DT components to their respective counterparts in the PT.
After the relevant component in the DT is identified, the parameters must be adjusted to fit the observed PT behavior. Given previously seen inputs to the PT and the respective reactions from the system, the parameters of the DT can be tweaked to match that behavior. Chapter 2.3 gives an overview about possible methods.

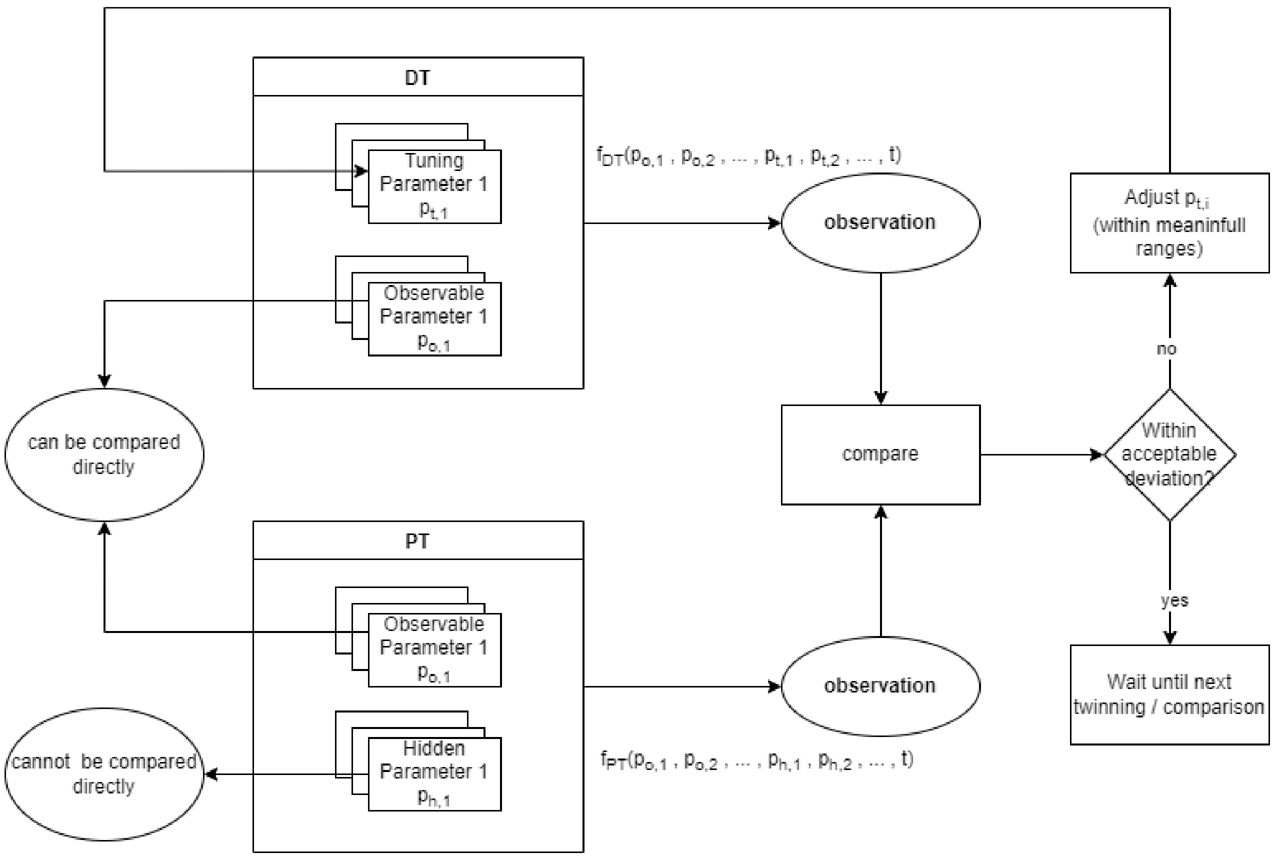
DT
Tuning Parameter 1 $p_{t,1}$
Observable Parameter 1 $p_{o,1}$
$f_{DT}(p_{o,1}, p_{o,2}, \dots, p_{t,1}, p_{t,2}, \dots, t)$
observation
Adjust $p_{t,i}$ (within meaninfull ranges)
no
can be compared directly
compare
Within acceptable deviation?
PT
Observable Parameter 1 $p_{o,1}$
yes
observation
Wait until next twinning / comparison
cannot be compared directly
Hidden Parameter 1 $p_{h,1}$
$f_{PT}(p_{o,1}, p_{o,2}, \dots, p_{h,1}, p_{h,2}, \dots, t)$

Figure 7 - Relevant Parameters during DT alignment

It must be noted that such a process of finding the relevant parameters to change is only necessary when it is not possible to measure them directly. Figure 7 highlights that idea. It shows the DT and the PT, as well as a discrimination of their internal parameters into the set $\{p_{o,i}\}$, which are the parameters that are observable in the DT and the PT, the set $\{p_{h,i}\}$, which are "hidden" parameters, that cannot be measured directly, but still have an influence on the observation, and the set of tunable parameters $\{p_{t,i}\}$, which can be used to tune the DT to match the observation from the PT. Those parameters $\{p_{t,i}\}$ are a subset of the c_tune(t) parameters, introduced in Figure 2. The observable parameters $\{p_{o,i}\}$ are also parameters of the c_tune(t) category. If they are however not describing an internal system configuration but can be treated as some sort of input for the PT, they can be seen as disturbances d.
The process of twinning and therefore the synchronization of PT and DT is only happening during the operational and fine-tuning phase of the system. In the training phase, the Optimization AI is tuned using varied virtual models only, to have the possibility to speed up the process faster than real-time and not block the physical system during the time intensive data gathering process of RL training.

2.6.2 Methods To Adapt Digital Twin

Having found a misalignment of one or multiple observables, we have to think about how to align the PT and the DT. This can be done by changing the tunable parameters in several ways. As mentioned above, it is beneficial to assign groups of parameters to sub-components of the DT. If we assume that the misalignment can be traced to a specific subcomponent, we can concentrate on adjusting the tunable parameters for this subcomponent only.
In principle observations stemming from twinning processes can be treated in two ways: As independent measurements (i.i.d.), or as sequential, making use of a potential sequential pattern in the data. One of the simplest ways to relax the i.i.d. assumption is to consider a *Markov model* [13].

2.6.2.1 Markov Chain

In a first-order Markov Chain of observations $\{x_n\}$, which are our measurements, the distribution $p(x_n, x_{n-1})$ of a measurement is conditioned on the value of the previous measurement.

$$p(\mathbf{x}_n|\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}) = p(\mathbf{x}_n|\mathbf{x}_{n-1})$$

Our model contains tunable parameters, that may change over time (over the measuring steps), which also changes the conditional distributions. If a potential trend in the data is expected to be visible over more than one successive observation, we can move to higher-order Markov Chains, which often becomes a problem, since the number of parameters grows exponentially with the order.

2.6.2.2 Hidden Markov Models

Another way to use the Markov Chain for our measurements $\{x_n\}$is to introduce an additional latent variable z_n for each observation x_n. Then, we can assume that those latent variables form a (first-order) Markov Chain. This graphical structure is known as a *state space model.* Given a state z_n, the previous and following are independent, so that

$$\mathbf{z}_{n+1} \perp\!\!\!\perp \mathbf{z}_{n-1} \mid \mathbf{z}_n.$$

If the latent variables are discrete, then we obtain the *hidden Markov Model* (HMM). If both the latent and the observed parameters are Gaussian, we obtain a *linear dynamical model*.
The connection of the latent variables z_n and the measurements x_n happens via emission probabilities Φ_k which can be represented in the form

$$p(\mathbf{x}_n|\mathbf{z}_n,\boldsymbol{\phi}) = \prod_{k=1}^{K} p(\mathbf{x}_n|\boldsymbol{\phi}_k)^{z_{nk}}.$$

K is the number of states the discrete variable x can have. There are many flavors and extensions of the HMM, out of which the most fitting one shall be chosen based on the characteristics of the measured data and the requirements of the concrete use case. If the latent variables have some meaningful interpretation, the most probable sequence of hidden states for a given observation sequence can be found using the *Viterbi algorithm*.
In order to use methods like the ones described above, the DT needs to be modeled as a probabilistic graphical model. This is proposed in [14], where the model is realized as a dynamic decision network: a dynamic Bayesian network with the addition of decision nodes, like shown in Figure 8. It "shows a dynamic decision network that mathematically represents a physical asset and its digital twin. Nodes in the graph shown with bold outlines are observed quantities (that is, they have a deterministic value), while other quantities are estimated (typically represented by a probability distribution). Directed edges represent conditional dependence. Here, the control nodes are decision nodes." [14]

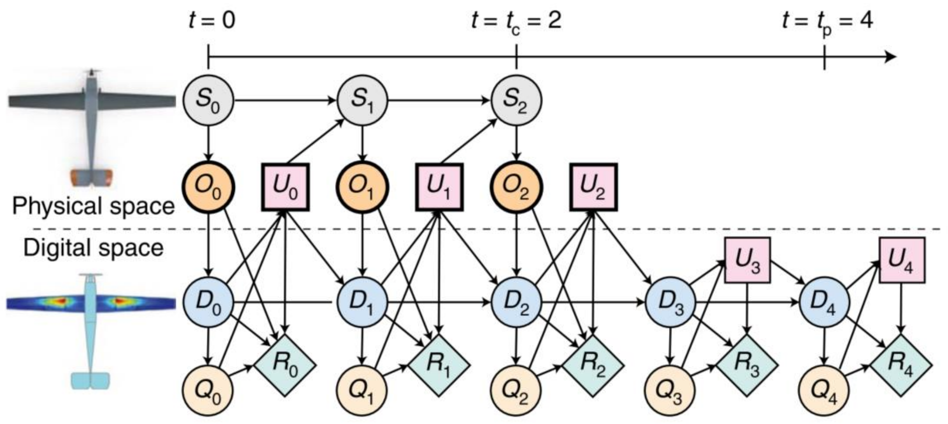

Figure 8 - Probabilistic graphical model for the asset–twin system from [14]

2.6.2.3 Expert Knowledge

Wherever possible, valuable information from experts can be used to update or assist the update of tunable parameters. This does not necessarily require active interaction with an expert during runtime, but can be done in the conceptual phase of the DT creation. Expert knowledge can be used to define appropriate ranges for tunable parameters, possible links to other parameters or even processes on how such parameters would be tuned traditionally. Furthermore, information about the system at hand can provide information about possible drifts of tunable parameters in a specific direction. An example could be the wear-related geometry and therefore behavioral changes, that are not measured directly, but would typically only deviate from their initial parameterization in one direction without maintenance. Another example consists out of the inherent symmetry breaking, certain aberrations induce within the Ronchigram. Astigmatism for instance reduces the radial symmetry to two-fold symmetry, as shown in Figure 6.

2.6.2.4 Gradient Descent

Let's assume, that the space of the tunable parameters in the DT is convex and that a gradient exists that points in the direction of the configuration which results in a prediction that matches the measured observations the best. Then we can apply gradient descent methods to find this optimal configuration of the tunable parameters in an efficient way. Several methods exist in the literature how to do gradient descent. They differ for example in how the step size in the direction of the steepest gradient is varied. If the gradient cannot be calculated, stochastic gradient descent can be applied. There, the gradient is estimated by using data.
If no such gradient exists this leaves us with either stochastic sampling of parameter configurations, or, if the dimension of the space of the tunable parameters is not too high, a grid search approach.

2.6.2.5 Fleet Data

Even though the adaptation of parameters of a DT to fit the behavior of its physical twin is something that is very instance specific, fleet data can still provide useful guidelines for the individual adaptation. In [15] a hierarchical approach is described, that takes into account, that there are many different configurations of systems in a fleet, where some of them share similar assets. The generated models of such assets can then be used to have a more detailed understanding of similar assets in newly added systems, where no or not enough physical data is available.
Such an approach can also be used to support the adaptation of parameters of an individual instance. The instantiated models of the fleet can provide a distribution of likely parameter values for a given subcomponent, I.e., asset and therefore a starting point for further improvements. In [15] a hierarchical parameter structure is described, where an instance is initialized with fleet data and is further individualized as soon as it gathers more data. That way, a suitable parameter estimation can already be done early in the operational phase and can be further fine-tuned by the specialization of the instance based on its own generated data.

2.6.3 STEM Use Case

In the case of the Electron Microscopy Use Case, the DT represents the virtual representation of all the physical electron microscope components that impact image formation. These components include amongst others: electro-magnetic lenses, electro-magnetic correctors, sample-beam interaction, the detector. All together these components contribute to the Ronchigram image formation (DT). It is crucial to generate Ronchigram images that are similar to the experimental Ronchigram images such that the RL Agent is capable of generalizing from simulated data to experimental data. To evaluate the different components within the DT, one could compare the Ronchigram images which result from small parameter changes for certain components in the DT to their physical counterpart. Ideally, identical changes in the input parameters should alter the experimental and synthetic Ronchigram in a comparable manner. In this manner, separate KPIs can be realized for the individual components.

2.6.4 UUV Use Case

In the case of the Unmanned Utility Vehicle Use Case, the DT represents a virtual representation of a vehicle on a testbed. As the DT is used for training, it is important that it can generate realistic output and

resembles the behavior of the physical system as close as possible. Improving the DT therefore has the benefit of being able to improve the optimization AI with realistic data from a DT.
As the whole DT is divided into multiple DT components, separate KPIs for these components can be realized. The interfaces between these individual components can also typically be measured or read out from the physical vehicle via its connection to the testbed.
The adaptation of the DT (component) can then happen using previously logged data. The parameters of the DT (component) can be changed in a way, so that the same input into DT and PT leads to the same output. The selection of parameters to use for adaptation depends on the available measurement equipment. Every parameter that can be measured directly can also be used for the DT directly. Only parameters whose value cannot be measured should be adapted during this process. To ensure validity when changing the parameters, expert knowledge can be used up front to define a valid range.

3 Reinforcement Learning

With respect to the RL agent, we consider three topics:

1. Definition of performance metrics on the RL agent
2. Process to update the (reference) RL agent
3. User control of the system optimization

3.1 Monitoring RL Performance

To compare different iterations of RL agents and evaluate their performance, the following section gives an overview of possible KPIs to quantify the performance and provides a process on how to choose the right KPIs. Based on that, countering methods can be applied, should an RL optimization not lead to the desired outcome. During the proof-of-concept phase and regular service intervals, the performance can also be compared to conventional methods.
As with the DT, the aim of this section is to provide methods to continuously monitor the RL. An initial validation must be done before, which will be discussed in D3.4.

Apart from assuring the consistency between the DT and the PT, we also want to create usable metrics to measure KPIs when the RL is applied on DTs and PTs. The workflow is shown in the following figure. The individual blocks are described in the following sections. It must be kept in mind that the optimization and KPI-based triggering of RL updates based on the evaluation on the DT only, requires a good synchronization of the DT and PT as a prerequisite. A misaligned DT and resulting diverging behavior from its PT in terms of optimization state would make a KPI-based evaluation of this optimization meaningless.

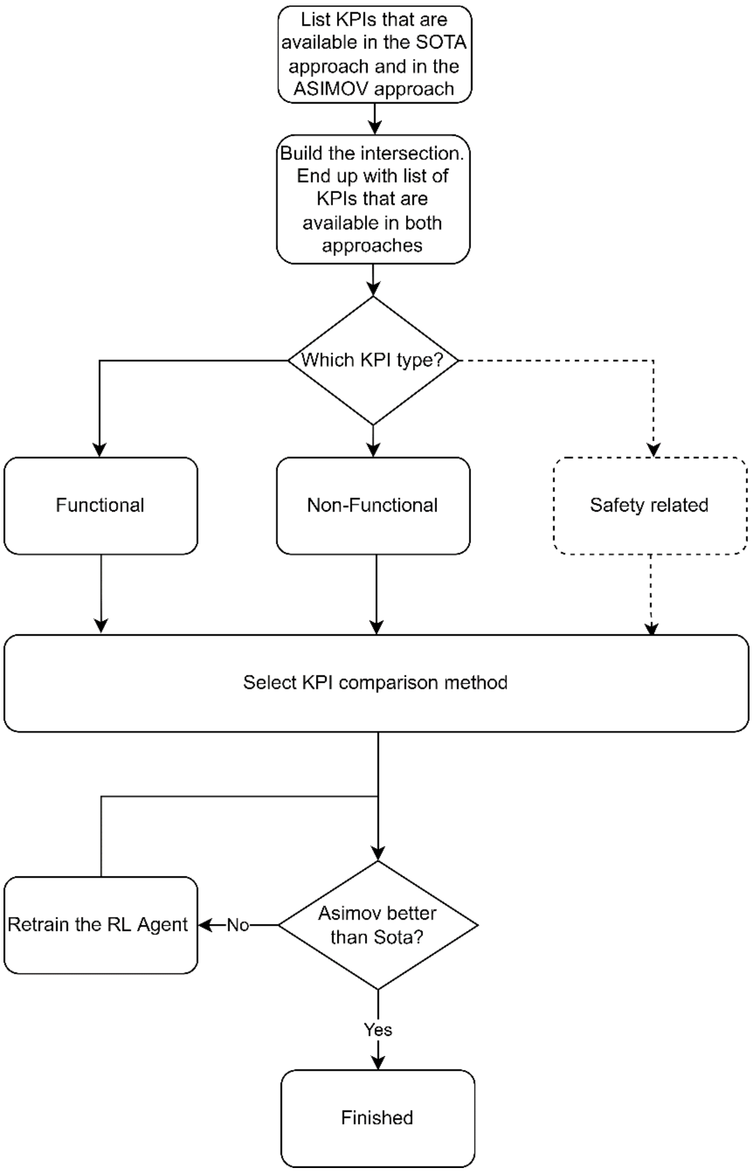
List KPIs that are available in the SOTA approach and in the ASIMOV approach
Build the intersection. End up with list of KPIs that are available in both approaches
Which KPI type?
Functional
Non-Functional
Safety related
Select KPI comparison method
Retrain the RL Agent
No
Asimov better than Sota?
Yes
Finished

Figure 9 - KPI Process for RL Comparison

3.1.1 Listing the KPIs

When listing the available KPIs, we will limit ourselves to ones that are available in previous solutions for the problems at hand (i.e., calibration of STEM / generation of scenarios) as well as for the Asimov solution. As shown in the flow diagram the KPIs will be divided into different categories, which might require different means of comparison. Since the KPIs for the RL Agent are much in line with the KPIs for the Asimov solution itself, they are used here as a base. They are functional, non-functional, and safety related.

- Functional KPIs are a measure for how well the policy works when applied on the PT.
- Non-Functional KPIs are a measure for realization related aspects.
- Safety-related KPIs are KPIs that measure the safety for employees when applying the policy.

3.1.1.1 Functional KPIs

As portrayed in [16], once the RL Agent is trained sufficiently, it needs to be applied to the PT. Several KPIs can be thought of, that mirror how successful this application is:

- Accuracy of result (i.e., the state of the system after applying the optimization)
    - Does the result achieve the required accuracy?
- Robustness (Ability to handle disturbances (external disturbances coming from outside of the system), extreme inputs, etc.)
- Reliability (Ability to always obtain a good result.)
- Reproducibility (Ability to always obtain the same result.)
- Time to result (Execution time in operational phase to reach the result.)
- Explainability (Ability to give insight into how the solution works, and plausibility of the result)

Depending on the termination condition of the optimization process, there may be a trade-off with the Realization-related KPIs (in the next section), especially the ones concerned with operational cost. If we for example only terminate the optimization once a certain quality is reached, it could be that the Asimov-based calibration can always result in a higher quality than the State-of-the-art (Sota) calibration. If we however have a termination condition that is based on a limited budget (time, computations, or storage), we need to compare the quality of the calibration between the Asimov and the Sota approach. For this we need to find KPIs that mirror this quality. These KPIs shall be a good measure for the final output, which differs for the use cases, and shall be discussed separately. An apparent choice for a KPI, which we might want to avoid due to its subjectivity, is to compare the final output with the intuition of an expert.

3.1.1.2 Non-functional KPIs

As outlined in [16], the ASIMOV solution needs to be economically sound, additionally to functionally operational. This leads to KPIs related to the following realization-related aspects:

- Footprint (How much space/ energy/ resources/ human resources does the solution need?)
    - Examples for UUV.1:
        - Necessary hardware:
            - Sota: operation of the testbed | Asimov: Additional GPUs)
        - Externalized costs / amount of (nonrenewable) resources used:
            - Sota: Tons of CO2 produced by driving, running testbed, etc. | Asimov: Tons of CO2 produced by training models, simulating, etc
        - Human recources:
            - Sota: mechanics that operate the testbed, or drivers that steer the vehicle | Asimov: Programmers, Developers
- Maintainability (How well can the solution be upgraded or evolved in the field?)

3.1.1.3 Safety-related KPIs

In the ASIMOV approach, user interaction with the system regarding the initialization and calibration processes is reduced. Assuming a non-zero chance of safety incidents during calibration by a human operator, RL-based calibration can lead to a safety improvement. This topic, however, shall not be further examined in this task.

3.1.2 Build the intersection of the KPIs

Some KPIs might only be available in the Sota approach and others only in the Asimov approach. If those cannot be converted to a comparable number, they will be omitted for the comparison.

3.1.3 Select KPI comparison method

Most of the non-functional KPIs can be evaluated in an economic way. We assume historic data from the controlling / accounting departments to be available for the Sota approach. Running the Asimov solution for some time, this data can be compared to the newly collected data for the Asimov approach. The ecological KPIs, e.g., externalized costs/amounts of non-renewable resources are a bit more complicated to derive, since they necessitate an integrating over the whole value chain of the materials and machines that were used for the respective solution.

As for the functional KPIs, we need to remember, that for the use cases the Sota approach was centered around an expert. As a result, the assessment of the quality of the solution is subjective by nature.

If multiple KPIs need to be optimized, a weighted sum of the KPIs needs to be created with the weights defining the relative importance of the KPI. To make multiple KPIs comparable among each other, a normalization, based on reference values and a scaling between 0 and 1 is required.

3.1.4 Proceed if Asimov is not better than the Sota approach

If the Asimov approach did not lead to an improvement, measured with the defined KPIs, this can be a clue that the training of the RL Agent needs to be improved. This is not part of this work package and will only be briefly touched on in Section 3.2. Further explanations can be found in the deliverables of ASIMOV Work Package 3, but the retraining can be triggered by the results described above.

3.2 Improving RL Performance

With the goal of improving the system's KPIs, training of the RL agents is done using the DT. As a matter of course, the reward of the agents should be chosen to be aligned to the system's KPI in order that the aforementioned goals are achieved. In the process of learning, the RL agents aim to improve the chosen reward. To relieve the computational burden, the mapping to be optimized usually stems from a parametrized function class (such as a neural network in the case of deep reinforcement learning) so that optimization is done by tweaking the corresponding parameters.
In general, it is not possible to optimize all the specified system KPIs as there might be KPIs which contradict each other. Therefore, one needs to make trade-off between the different KPIs. The next best optimality concept is the Pareto-optimality concept which is the usual solution concept in multi-objective optimization. The usual way to reach a Pareto optimal point is to shape the optimization objective/reward as the weighted average of the given objectives/KPIs. The corresponding weight can be assigned in the static manner, for instance proportional to a certain priority of the KPIs, or alternatively, they can be assigned dynamically to yield, e.g., the point such that the worst KPI is optimal. The work [17] gives some general guidelines and broad literature overview on tackling RL problem in hindsight of multi-objectivity.
Training the RL agent using the DT should yield a policy which can directly be applied on the physical system. Therefore, it is important for the performance of the ASIMOV approach to have a DT which aligns well with the PT. Respectively, if there are changes in the PT, retraining of the RL agents is necessary for ensuring that the RL agents still perform well in the new PT environment. One possible process is illustrated in Figure 10.

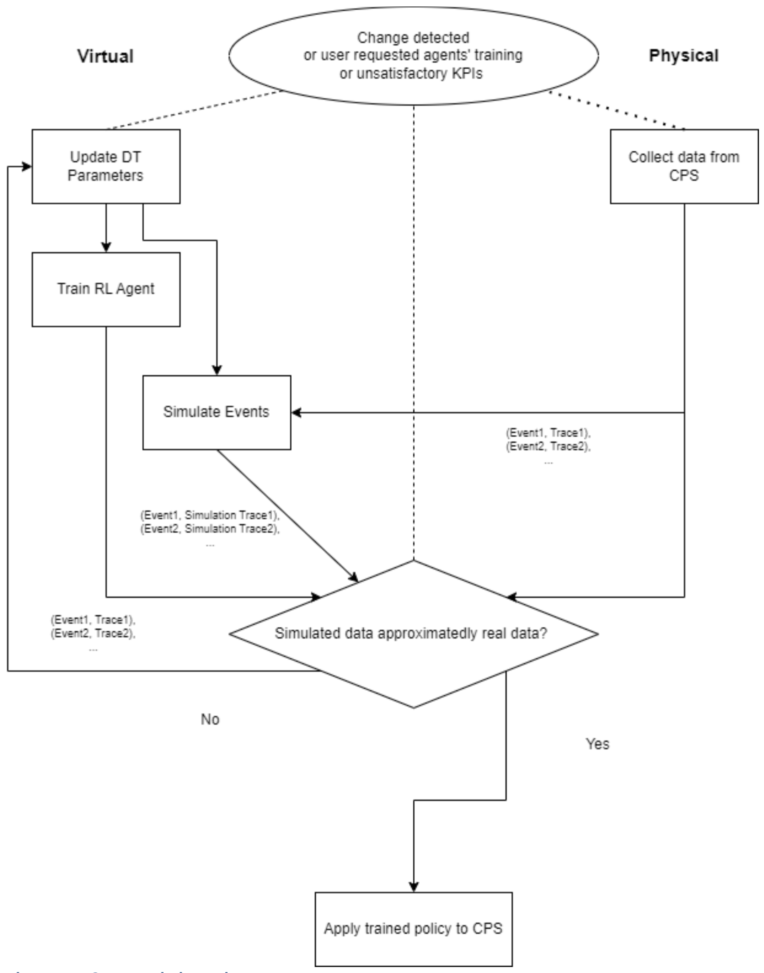

Figure 10 - Training the AI

According to the above, the process is started as soon as changes in the PT are detected. In the beginning of the process, the DT parameters are updated with the aim that PT and DT are aligned. DT then simulates the environment based on the new parameters providing a playground for training the RL agents. In parallel, data in the form of dynamical path is collected from the PT for validating the representation power of the newly aligned DT. Specific methods for detecting changes in PT and methods for aligning DT with PT are given in the previous sections.
Change of RL agents' hyperparameters can also be initiated as soon as one has unsatisfactory KPIs (with unobservable parameters – see Figure 10) and as soon as the system's objective respective to KPIs changes.
Often, the RL agents reside in similar environments. To relieve the computational burden, one may train only one reference agent instead of several agents. Moreover, the training can be done on the centralized experience storage that is built by all the DT instances. This idea is based on the idea of federated learning. After training the reference agent, the solution can then be deployed to the corresponding agents (see also [2]).

When refining the RL, observed and logged data from the physical system can be used to adapt the DT closer to its PT. This refined DT can then be used to interact with the RL agent. That way, the RL agent can train independent of the PTs availability, by just using the DT and provide better actions when applied on the PT again.

3.2.1 UUV Use Case

In the case of the UUV.1 use case, all scenario suggestions based on the current state, which includes information about the novelties as well as criticalities, will be logged as well. Later, these state and reward

combinations in conjunction with the actions as inputs, can be used to train the RL agent offline, not needing to interfere with the actual system, but with recorded data or the interaction with the DT.
This also makes this approach modular, as the datasets for training do not necessarily need to come from the same PT but could be generated by a combination of DTs of (planned) products and physical data gathered from multiple PT in use.

3.2.2 STEM Use Case

In the STEM case, various electron microscopy parameters along with the actions generated by the RL agent could be logged. Below are some of the parameters that could be stored in the logs:

- Timestamp
- Beam energy
- Convergence angle
- Magnification
- Probe current
- Actions (e.g., stigmator x and stigmator y values) generated by RL agent
- Ronchigram images generated by the physical microscope
- Current through the various Optical Elements within the microscope

This logged data will later be analyzed and used to refine the RL agent offline.

3.3 User Interaction with the RL Optimizer

Reinforcement Learning is an optimization consisting of multiple steps, that lead to the desired solution. That’s why the process has a continuous nature. E.g., in the UUV.1 use case, an optimization step does not lead to the final solution. Instead, the RL agent applies a continuous process that terminates when reaching certain termination conditions. The ultimate goal in this use case is defined by finding the right sequence of actions, that define scenarios, that lead to the most valuable test cases. Similarly, the purpose of the EM use-case is to retrieve the best sequence of actions that lead to the desired image resolution. After each step the agent takes, the observations must be processed, and new actions must be taken accordingly.

With a running system in place, it becomes important when the RL optimizes the CPS. This ensures that the operator of the system understands when a change in system behavior is induced by a new optimization step. Two different processes of how such an update could be triggered are described in the following sections.

3.3.1 Continuous Process

An intuitive method on when to start that optimization is to enable an automatic optimization at the startup of the CPS. After this startup, the RL continuously optimizes the system and changes calibration whenever necessary. This has the benefit of minimal interaction needed with the human operator. Assuming a sufficient RL performance, the operator can rely on a well calibrated system all the time.

3.3.2 Event-based Process

The continuous optimization process might come with the disadvantage of a lack of comparability of system outputs depending on the state of optimization. It could hinder the operation of the electron microscope while it is in use and the operator is not actively deciding to recalibrate. Similarly, the adaptation of new sensor configuration parameters during a simulation run in the UUV use case, might also lead to the vehicle behaving differently in the beginning and the end of a simulation, which leads to undesirable behavior, even though the updated sensor configuration might be a valid update.
It is therefore important to trigger only in situations where the user expects such an optimization step or actively requests one. This also allows for expert knowledge to be used to define events in which optimization is necessary. Such events could be a certain change of configuration in the system, switching to a different operation mode, etc. The possibility to couple optimization triggers for the RL to events, that have previously been used to indicate to an expert, that manual optimization is required, can therefore also ensure trust in the system and could lead to better acceptance.

4 Conclusion

Task T4.1 of the ASIMOV project has proposed a process as well as at set of methods to accomplish DT synchronization as well as a runtime monitoring of the RLA. Hereby, multiple requirements and ideas derived from other tasks of the ASIMOV project, as well as the results from the Working Group were taken into account. Multiple methods for data comparison were introduced and their suitability for runtime evaluation was touched on. Additionally, techniques on how to adapt the DT based on that knowledge were considered. A similar process and methods were proposed to encounter the runtime monitoring of the RLA, including techniques on how to use fleet knowledge to leverage the gathered data for improving RL Optimization during runtime. The architectural concept of the ASIMOV solution, including these methods, are subject to T4.3, with the respective tooling aspect being further examined in T4.2.

5 Terms, Abbreviations and Definitions

Table 1 - Terms, Abbreviations and Definitions

AI	Artificial Intelligence
DT	Digital Twin
KPI	Key Performance Indicator
PT	Physical Twin
RL	Reinforcement Learning
RLA	Reinforcement Learning Agent
Sota	State-of-the-art
STEM	Scanning Transmission Electron Microscopy
UUV	Unmanned Utility Vehicle

6 Bibliography

[1] ASIMOV-consortium, “Identification of relevant parameters modelled in DT - ASIMOV Deliverable D2.1,” 2022.

[2] ASIMOV-consortium, “Methods and Tools for Training AI with Digital Twin - ASIMOV Deliverable D2.2,” 2022.

[3] Q. Z. L. L. K. Z. Jiaju Qi, “Federated Reinforcement Learning: Techniques, Applications, and Open Challanges,” *https://arxiv.org/pdf/2108.11887.pdf,* 2021.

[4] G. D. e. a. Maso Talou, “Efficient Ventricular Parameter Estimation Using AI-Surrogate Models ,Frontiers in Physiology,” vol. https://doi.org/10.3389/fphys.2021.732351, 2021.

[5] Rong, Li, Zhang and Sun, “An improved Canny edge detection algorithm,” *IEEE International Conference on Mechatronics and Automation,* no. doi: 10.1109/ICMA.2014.6885761, pp. 577-582, 2014.

[6] D. K. Park, Y. S. Jeon and C. S. Won, “Efficient use of local edge histogram descriptor,” *In Proceedings of the 2000 ACM workshops on Multimedia (MULTIMEDIA '00). Association for Computing Machinery, New York, NY, USA,* no. DOI:https://doi.org/10.1145/357744.357758, p. 51–54, 2000.

[7] N. Vandeput, “Data Science for Supply Chain Forecasting,” *Berlin, Boston: De Gruyter,* no. https://doi.org/10.1515/9783110671124, 2021.

[8] M. V. e. a. Shcherbakov, “A survey of forecast error measures.,” *World applied sciences journal 24.24,* vol. DOI: 10.5829/idosi.wasj.2013.24.itmies.80032, pp. 171-176, 2013.

[9] H. K. Sungil Kim, “A new metric of absolute percentage error for intermittent demand forecasts,” *International Journal of Forecasting,* Vols. Volume 32, Issue 3, no. https://doi.org/10.1016/j.ijforecast.2015.12.003, pp. 669-679, 2016.

[10] H. V. a. K. H. Gupta, “On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics,” *Water Resour. Res., 47, W10601,* vol. doi:10.1029/2011WR010962, 2011.

[11] D. W. M. &. J. G. Chicco, The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation, PeerJ. Computer science 7: e623, 2021.

[12] C. S. L. C. a. A. v. d. H. Guansong Pang, “Deep Learning for Anomaly Detection: A Review,” *ACM Comput. Surv. 1, 1,* vol. Article 1, no. https://doi.org/10.1145/3439950, p. 36, 2020.

[13] C. M. Bishop, Pattern Recognition and, New York: Springer-Verlag, 2006.

[14] M. G. Kapteyn, J. V. R. Pretorius and K. E. Willcox, “A probabilistic graphical model foundation for enabling predictive digital twins at scale.,” *Nature Computational Science 1 (5),* Vols. DOI: 10.1038/s43588-021-00069-0, p. 337–347, 2021.

[15] M. G. M. &. P. A. Dhada, “Anomaly detection in a fleet of industrial assets with hierarchical statistical modeling.,” *Data-Centric Engineering E21,* vol. doi:10.1017/dce.2020.19, 2020.

[16] ASIMOV-consortium, “Working Group Document D4.a,” 2022.

[17] R. e. a. C.F.Hayes, A practical guide to multi-objective reinforcement-learning and planning, Autonomous Agents and Multi-Agents Systems 36, 2022.

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