Identification of relevant parameters modelled in DT

[WP2; T2.1; Deliverable: D2.1 Version 2.0]

Non-Confidential

AI training using Simulated Instruments for Machine Optimization and Verification

Version	Status	Date	Page
Version 2.0	Public	2024.05.01	1/62

Document Information

Project	ASIMOV
Grant Agreement No.	20216 ASIMOV - ITEA
Deliverable No.	D2.1
Deliverable No. in WP	WP2; T2.1
Deliverable Title	Identification of relevant parameters modelled in DT
Dissemination Level	Public
Document Version	Version 2.0
Date	2024.05.01
Contact	Sebastian Moritz
Organization	Triangraphics
E-Mail	Sebastian.Moritz@Triangraphics.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
Niklas Braun	AVL	Niklas.Braun@avl.com
Sacha Kullmann	AVL	Sascha.Kullmann@avl.com
Mario Alberto Martinez Cruz	AVL	MarioAlberto.MartinezCruz@avl.com
Joost van Sambeeck	CQM	Joost.vansambeeck@cqm.nl
Mehrnoush Hajnorouzi	DLR	Mehrnoush.Hajnorouzi@dlr.de
Elias Modrakowski	DLR	Elias.Modrakowski@dlr.de
Andreas Eich	LiangDao	Andreas.Eich@liangdao.de
Narges Javaheri	TFS	Narges.Javaheri@thermofisher.com
Richard Doornbos	TNO	Richard.Doornbos@tno.nl
Christos Kitsanelis	TNO	Christos.Kitsanelis@tno.nl
Sebastian Moritz	TrianGraphics	Sebastian.Moritz@triangraphics.de
Jilles van Hulst	TU/e	J.S.V.Hulst@tue.nl
Roy van Zuijlen	TU/e	R.A.C.Zuijlen@tue.nl

Formal Reviewers

Version	Date	Reviewer
1.0	2022.09.29	Thomas Kotschenreuther (RA Consulting), Pieter Goosen (TNO)
2.0	2024.04.26	Thomas Kotschenreuther (RA Consulting), Lukas Schmidt (NorCom)

Change History

Version	Date	Reason for Change
0.1	2022.08.19	Version for content review.
0.2	2022.09.09	Version for formal review.
1.0	2022.09.29	Intermediate version.
1.1	2024.03.15	Version for content review.
1.2	2024.04.26	Version for formal review.
2.0	2024.05.08	Final version.

Abstract

This document describes fundamental considerations towards a process for identification of relevant parameters for a digital twin. The process includes identifying parameters in the real-world product (physical twin) which are relevant for its digital twin given a specific use case and describing methods to acquire and use these parameters. Despite its importance for the creation of a DT, there is little knowledge about how a parameter identification process must be structured to identify relevant parameters. When developing such a process, solutions must be found for several challenges. On the one hand, it must be determined at which point in the development process methods for identifying relevant parameters can be applied. On the other hand, after all the relevant parameters have been identified, the digital twin that is finally produced must possess certain quality characteristics. The two most crucial quality characteristics that have been identified are accuracy and execution time. Maximizing both of these characteristics typically poses a challenge. Parameter combinations must be found that lead to a digital model that satisfies all these quality characteristics equally. The document can be used as an introduction to the subject of identification of relevant parameters for a DT.

Table of Figures

Table of Tables

1 Introduction

A digital twin (DT) can be defined as a virtual duplicate of a system built from a fusion of models and data [1]. Depending on the requirements of a DT and the system it shall represent, it can include logic, control, sub-DTs and adaptation capabilities (see [2]). Today, DTs are an important part of product development at different stages of the product life cycle [1]. Among other benefits, DTs help to make complex and expensive processes feasible.

In the context of ASIMOV, the DT serves as the data source for training an AI-based agent. The trained agent is then employed to optimize the configuration of the actual system by adjusting the relevant parameters identified for optimization purposes.

The DT should represent the real system (object/process) as comprehensively and realistically as possible. Therefore, the model accuracy is a crucial part of the DT development.
Another aspect is the training of the AI-based agent using a DT. For a reliable result, a high number of training runs is required. To keep the total training time as low as possible, a single run should be significantly faster than real time as long as parallelization is not possible.
Moreover, a DT often represents a complex system, where the status of the system is dynamic and thus its DT also needs to change its configuration over time. This means that the DT should be updated in real-time using the input from the real system, and, vice versa, the real-time control on the real system should be possible by the output from the DT. Therefore, an additional requirement on DT is its execution efficiency. In many practical cases, there is a trade-off between accuracy and execution efficiency, keeping in mind that both must meet the system's requirements.

A DT should be robust, comprehensible, capable of generating reproducible and accurate data, as well as being efficient in its execution. These characteristics should be fulfilled within the scope of the requirements for which the DT is designed. This document mainly focuses on the last two quality characteristics.

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.
For the determination of a model, methods from system identification can be used. One of these methods is parameter estimation. The aim of this method is to determine a model from the data of the system boundaries. The system boundary is defined as the set of inputs, outputs and disturbances acting on the boundary of a system.

The document is structured as follows:

- What we understand as "parameters" is described in chapter 2 in more detail. This includes both the parameters that can control a system and those that a system outputs. In addition, it is defined when parameters are considered relevant.

- Chapter 3 provides an overview of the state of the art. The section on parameter identification begins by emphasizing the relevance of system boundaries for the modelling process. This is followed by a detailed examination of modelling through system identification and parameter estimation, with a focus on structure and data. Different model structures used in the modelling process are discussed in detail. The last section is dedicated to optimization methods. In addition, the trade-off between accuracy and execution efficiency in optimization is discussed.

- Chapter 4 contains the current process of parameter identification for the two use cases in order to set a base line. The current process for identification is based on the knowledge of experts and does not follow any generic process. The parameters identified here can be found in the Appendix.

- Chapter 5 describes at which point in the product development process methods for the identification of parameters for a DT can be applied. The parameter identification process includes methods for system identification (e.g. model selection, data-driven models) and methods for identifying all possible parameters of a known model and subsequently determining the relevant parameters for the purpose for which the DT is built.
  In addition, the process covers methods that optimize the model in terms of execution efficiency. This includes the identification of the most important relevant parameters (e.g. sensitivity analysis) and methods to generate a surrogate model. Finally, the process describes which topics should be addressed or considered in the future. The process has been evaluated during the project, documented in this deliverable.

- Chapter 6 discusses specific topics for the EM and UUV use cases. In the STEM section, which relates to the EM use case, the focus is on fine-tuning of parameters to increase the precision of the microscopy and on parameter identifiability problems. The second part focuses on research related to the UUV use case. Firstly, the influence of the level of detail of the objects in the 3D environment on the criticality metrics is analyzed. In addition, the development of a DT sensor is analyzed in terms of process. Finally, the validation of a LiDAR is analyzed as an example of parameter identification.

2 Definitions

As mentioned in the introduction, the goal of this task is to identify relevant parameters for a valid DT. The word "parameter" is a broad term which needs further explanation. In this section, the word "parameter" is clarified in the context of DT development within the ASIMOV project.

Parametrizing is a mechanism to express the variation of a function, process, or system (using a value). A parameter has a name and a value (and optionally a unit). "Parameter" as a word has several definitions but not an explicit one in this context. INCOSE for example, speaks of "critical system parameters such as weight, speed, accuracy, reliability, and cost" [3].

In DT development (WP2), where a cyber-physical system is modelled, several types of parameters can be distinguished. The goal of DT development is to represent the system as accurately as possible, while keeping the development process as efficient as possible. This is called 'fit-for-purpose'. In order to achieve this goal, all relevant parameters that are important for the representation need to be identified. To explain which parameters are covered in this task, Figure 1 is presented, where the system is placed in the structure used in the ASIMOV project. The way the system is described with input, output and disturbances, is a common method in control engineering, see e.g. [4]. The system (or its DT) is connected to an agent, with in-between pre- and post-processing components (pre- and post-processing are covered in T2.2). The agent determines which optimization action needs to be executed based on the current state. The development of the agent is covered in WP3. The definition of the abbreviations used in the figure (u, d, c, y, h, R, S, A) is as follows:

- **u - Control input**
  The control input u denotes all input parameters that can be changed by the agent (or operator), also called 'knobs' and their values 'knob positions'.
- **d - Disturbances**
  Disturbances are system inputs that change over time, but cannot be influenced by the agent or operator. (e.g., environmental conditions)
- **c - System parameter**
  Parameters that describe the system, which influence the behavior from the control input and disturbances to the output (e.g., masses of components in a mechanical system).
- **y - Output**
  The outputs are all available signals from the system. (e.g., sensor values)
- **h - Hyperparameters**
  Hyperparameters of the agent.
- **R, S, A - Reward, State, and Action**
  Nomenclature in reinforcement learning for the available information to the agent (State), the performance of the previous action(s) (Reward), and the action provided by the agent (Action) [5].

The relevant types of parameters in DT development are the control input, output, disturbances and system parameters (visualized in red in Figure 1). The purpose of T2.1 is to develop methods to identify them. Outputs are included in this list, since they are input for the AI system. The combination of DT and agent has a purpose: optimization, see Figure 1. Therefore the purpose of the combination determines which parameters are relevant for optimization.

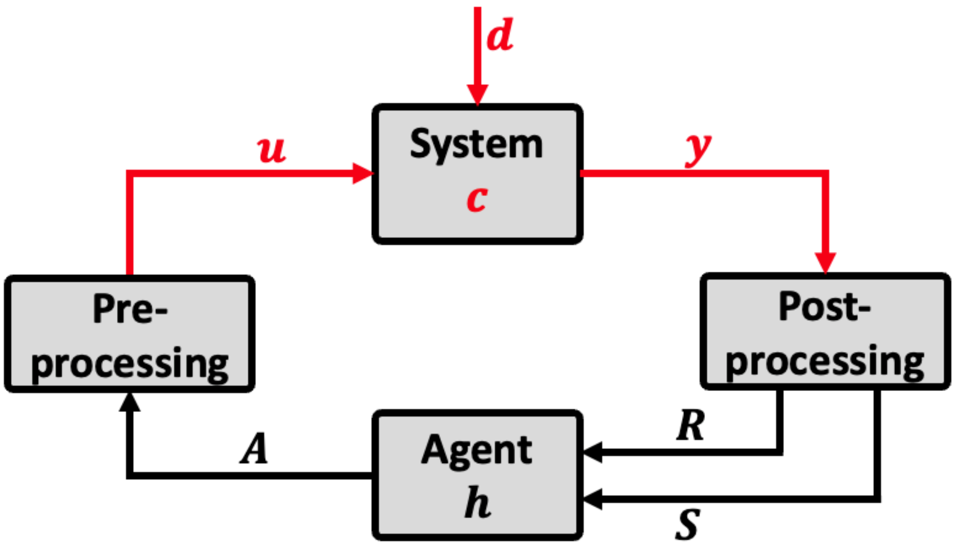


Figure 1 AI-training architecture with the parameters of interest for this document marked in red.

The objectives of T2.1 are visualized in Figure 2. At start there are many parameters, which can be everything one can imagine that can influence the system. Only a subset of these parameters is considered as relevant. What the relevant parameters exactly are depends on the design goal of the DT. Therefore, the relevant parameters are an unbounded subset within the complete parameter space. What is bounded is the subset of candidate parameters. Candidate parameters are an initial list of all the parameters the developer can come up with. It is however still unclear whether these parameters are relevant or not. The goal of T2.1 is to identify the relevant parameters as well as possible, without containing irrelevant parameters.

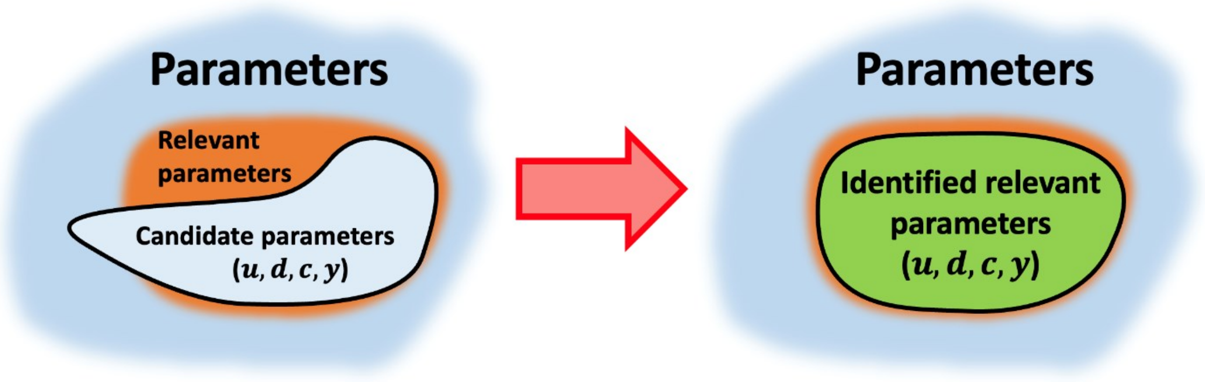


Figure 2 Parameter sets, where the goal of T2.1 is to identify all the relevant parameters.

Besides the parameters mentioned in Figure 1, there is a set of parameters not directly related to the properties of the physical system. Amongst this category are the hyperparameters h of the (configuration parameters specific to the tuning of the agent's behavior) or hyperparameters of the DT (e.g. the step size of the simulation). We will not explore this set of parameters further in this document as they do not map to any physical system parameters.

3 State of the art

3.1 Parameter Identification

In this section, the state of the art regarding modeling from the perspective of the parameter identification process is presented. The majority of the modeling process is detailed in Task 2.3. However, topics like data-driven quantification/estimation of parameter values are part of the parameter identification process, hence they will be included here. These topics are part of the larger field of system identification, which is also covered in this section. The section starts by detailing the procedure of defining the parameters which constitute the system boundaries. Next, some important results in system identification are presented which are relevant to WP2 in ASIMOV.

3.1.1 Defining the System Boundary

The parameter types as established in Figure 1 are an essential first step in the generation of the model. The system boundary is defined as the collection of the inputs (u), the outputs (y) and the disturbances (d), as these are the parameter types that act on the boundary of the system. The system boundary is typically used to perform system identification, as this boundary allows for data-driven model estimation. Naturally, the primary parameters of a model are the inputs and outputs of the system. A good model accurately predicts how the inputs map to the outputs. In the case of a dynamic system, the current output is generally dependent on a history of past outputs and inputs. For a static system, the output is only dependent on the current input.

Choosing the input/output during the system design:
Outputs, typically implemented as sensors which provide data to the AI agent, should generally be chosen as quantities that can be used to measure the performance of the system or as quantities which have a large effect on the correct input, (e.g., in chess) the optimal move choice depends on the current game state, so choosing the game state as the system output greatly improves the attainable performance. For inputs, choose quantities that allow for manipulation of the performance of the systems through the outputs. The choice of input is often limited by physical restrictions. For instance, even though transportation through space is the main objective for a car, one cannot directly change the position of the car with any input. Instead, we can only manipulate the gas pedal which controls the amount of gas being fed into the engine, resulting in more or less torque in the wheels, which in turn results in a change in position over time.

The inputs and outputs follow from the design, and thus are already given during the modeling phase, as the system already exists. However, one can still place additional sensors/actuators to obtain more outputs or inputs, should the system's performance not be sufficient. Additionally, given a set of inputs and outputs, one can analyze which inputs correlate to which outputs using system identification techniques. If there are inputs that do not correlate with any outputs or vice-versa, these might not be relevant to the system behavior and thus can be removed.

Disturbances (d):
What signals affect the way the system behaves? How to find what these are? Can we measure the disturbances? If they can be measured, this effectively turns them into inputs (u) for the purposes of system identification. Getting insight into the disturbances is therefore important, since they could impact the system significantly.

System properties (c):
System properties (physical properties of a system): sizes / weights of components etc. But how can we find which ones are relevant? One can use first principles or existing modeling techniques for similar systems to obtain a preliminary set of system properties.

3.1.2 System Identification

Given a set of system boundaries, how do we come up with a model? An overview of different methods is presented in Figure 3. The basic premise for all is to utilize structure and/or data in order to construct models. The structure creates a bias in the model towards behavior which is expected due to prior knowledge of the system. The inclusion of data biases the model toward behavior that has been experienced and observed. First principles-based modeling is mostly covered in Task 2.3. Instead, this section focuses on system identification and more specifically parameter estimation.

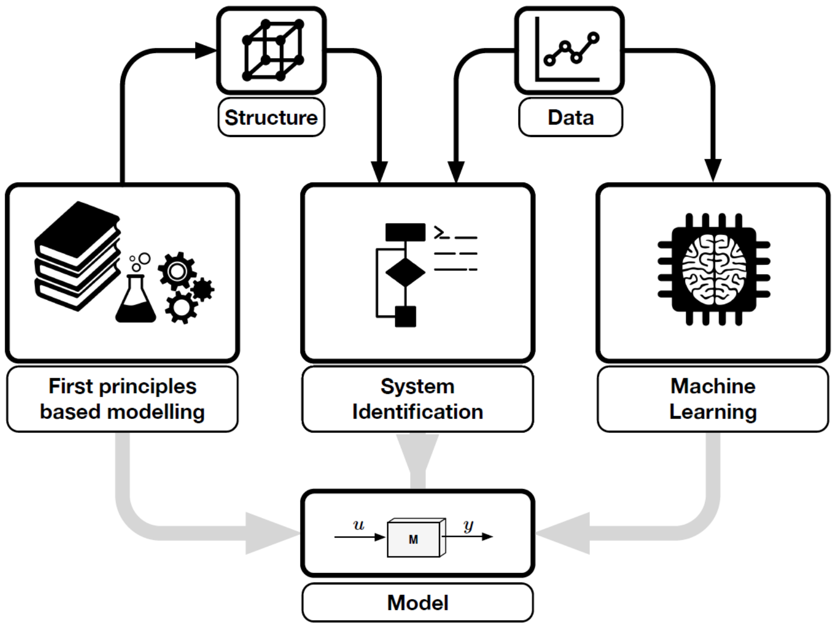


Figure 3 Options for the modelling process of a system [6].

System identification consists of statistical methods to build models for dynamical systems from measured data [7]. Depending on the type of system and the modeling assumptions, different methods are needed.

The following system characteristics have an effect on the choice of system identification method:

- Class: Linear vs. nonlinear
- Dimensionality of the system boundaries: Single-input-single-output (SISO) vs. Multi-input-multi-output (MIMO)
- Behavior: Static vs. Dynamic

While these assumptions influence the accuracy of the applied method:

- Type and prevalence of measurement noise
- Type and prevalence of unmeasured disturbances

System identification focuses on cases where we can make these types of assumptions and characterizations in order to use the available data more effectively. One such example is parameter estimation. In parameter estimation, we ask ourselves; given a (parametric) model structure, how can we fit the model to the available data? This approach is often called grey box system identification, because there is uncertainty in the resulting model due to the unknown parameter values, even though the structure is fixed. These model structures can be for instance: polynomial, basis function expansion. An example of a linear parametrized model structure is given by:

$$y(t) = a_1 y(t - T_s) + \cdots + a_n y(t - nT_s) + b_1 u(t - T_s) + \cdots + b_m u(t - mT_s),$$

where T_s is the time in between measured data samples and a_i, $\forall i \in \{1, \dots, n\}$, b_j, $\forall j \in \{1, \dots, m\}$ are the unknown parameters. This type of model is called an infinite impulse response (IIR) in which the modeled output at time t depends linearly on a history of past input and output values [8]. Quantifying the values for the parameters a_i, $\forall i \in \{1, \dots, n\}$, b_j, $\forall j \in \{1, \dots, m\}$ is commonly performed by finding the least squares fit to an experimental data set. The presence of disturbances or noise on the measurements is then averaged out if the dataset is rich enough.

However, many systems do not fit the model structure of an IIR, such as nonlinear systems. For systems in this class, there also exist nonlinear parametrized model structures, but in general nonlinear system identification is a much more complicated task.

Alternatively, there is black box system identification in which no prior model structure is assumed.

3.1.3 Model Structure

In the modelling process, one of the first steps is selecting the model structure. The model structure depends mainly on the knowledge of the system that is already available. All information about the system that is available a priori, can simplify the modelling process significantly. The types of model structures can be divided into three categories: white-box models, grey-box models, and black-box models, see Figure 4 [9]. This division is made based on prior knowledge and physical insight.

White-box models
A white-box model completely relies on physical insight in the system; there is no data needed to build such a model. Another name for these types of models is first-principle models. The advantage is that there are no assumptions in these models, and the model gives therefore very accurate results. The disadvantage is that it can be very complicated to make such a model when the system is complex (or even impossible).

Grey-box models
These models rely both on physical insight as well as available data. Examples are physical models where the parameters are estimated based on data, or state-space models where the order and structure are known, but the matrix-values have to be determined. The advantage is that the estimated unknowns are still interpretable. The disadvantage is that this method also can become complicated, if there are lots of parameters to be estimated with complex, nonlinear relations.

Black-box models
When the model only relies on data, and no physical insight is added, it is called a black-box model. Black-box models require a flexible model, such that the behavior of the estimated model matches the actual system as close as possible. The advantage is that it doesn't require any prior knowledge, and one can make the structure as simple or complicated as desired, depending on the required accuracy. The disadvantage is that the model is not interpretable anymore; the parameters have no physical meaning.

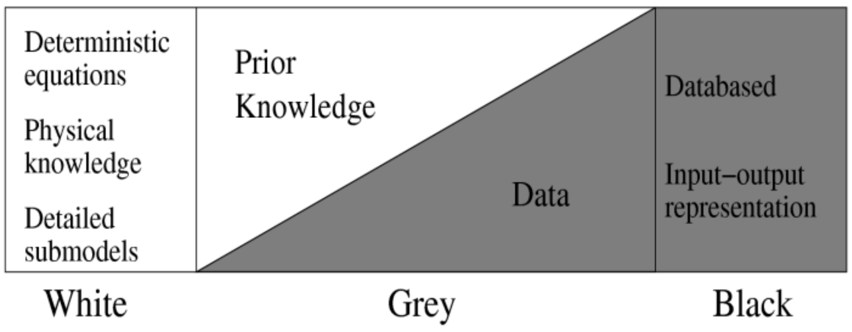


Figure 4 Different types of model structures [10].

3.2 Optimization

Model identification and parameter estimation techniques have been well developed in the past, especially for linear systems. For complex systems with non-linear dynamics, where the linear analytical and numerical approaches are inadequate, there is a need to develop advanced approaches.
Besides overcoming the significant non-linearity of the use cases, in ASIMOV we are looking for models that not only represent the CPS accurately, but they also need to be efficient enough in favor of facilitating their application to train AI with the purpose of optimizing system settings of the real-world counterpart. The objective is to formalize how we can optimize the accuracy-efficiency trade-off during the parameter identification and the parameter estimation to make a DT of the CPS. The main question in this part is to investigate whether the existing methods of parameter identification, and more precisely identifying the relevant parameters, can be used or modified such to meet the requirements of developing DT or we should come up with new methods. We start by reviewing a number of promising approaches that aim to balance the accuracy and execution efficiency level within one model.

Reducing the number of parameters of an existing model can be done in different ways. On one hand, the modeler might apply techniques which help to find the most relevant parameters to adjust the original model. Such a method is primarily a sensitivity analysis. On the other hand, the field of "surrogate modelling" provides Model Reduction (MR) methods [11] which create a new model with a reduced number of parameters.

Sensitivity analysis (SA): Sensitivity analysis quantifies the importance of the model's parameters on the behaviour of the system. It is applied to study how uncertainty in a model output is attributed to different sources of uncertainty in the model input [12]. Such estimation can be used to find the relevant parameters in favour of dimensionality reduction and to ensure the robustness and accuracy of the model across the range of inputs. We can use sensitivity analysis results to rank the parameters according to their influence on the model output (before optimization), and the impacts of their small changes on the key indicators of the optimization that the model is used for (after optimization). The sensitivity analysis can be executed in either global or local manner. While local sensitivity analysis assesses the variation of model output around a fixed, single point in the parameter space, global sensitivity analysis aims to measure the sensitivity of the model output over the entire parameter space. Variance-based sensitivity analysis like "Sobol method" [13] is a form of sensitivity analysis that is common in use to identify the most impactful parameters (see e.g. [14]). In addition, also more sample-efficient methods should be taken into account as well as sampling algorithms. Examples are tree-based techniques which can approximate a global sensitivity analysis (see [15]).

Optimization under uncertainty (finding the most relevant uncertain parameters for the system optimization): In optimization under uncertainty, although the uncertainties of parameters are not always known, but the selection of the relevant uncertain parameters is required. Using the whole set of identifiable or partially identifiable parameters results in unnecessary noise in the parameter estimation.

Therefore, the subset of identifiable parameters should be reduced such to include the most relevant parameters for the optimization objectives. The expected values and variances of the subset parameters are explored during the estimation while other parameters are fixed to their expected values. The prerequisite of this approach is to take the optimization objective function into account while selecting the relevant parameters. Several modelling frameworks have been proposed in the literature for optimization under uncertainty, e.g., stochastic programming and fuzzy programming. Generally, a deep understanding of problem structure and properties is prerequisite for selecting an applicable algorithm [16]. The state-of-the-art algorithm to identify relevant uncertain parameters for optimization purposes, is presented by [17]. The algorithm selects a subset of parameters based on the sensitivity analysis, with respect to the model accuracy as well as the purpose (user)-defined objective function. After estimating the expected values of the identified parameters, the algorithm ranks the uncertain parameters according to their linear-dependencies analysis. The parameters within the selected subset with lower dependencies will be considered uncertain while fixing the rest of the parameters.

Model Reduction (MR): The model reduction target is to reduce the computational load by generating reduced models [11]. These models represent the original system behavior as accurately as possible while their simulation remains fast and cheap. Dedicating time and cost to obtain a reduced model in the development phase can result in cutting the computational cost down in latter phases enormously. The model reduction is realized in different ways, for instance by eliminating the insignificant terms in statistical models, or by finding a low-dimensional approximation for a system of ordinary differential equations (ODEs). The model reduction techniques are classified into two broad categories: (1) Conceptual approaches in the physical-mathematical domain, and (2) Numerical approaches and data driven techniques in systems engineering.

In a mathematical context, the complexity of a dynamic system is characterized by the number of its state variables which is referred to as the dimension of the state space vector. Model order reduction studies properties of dynamical systems in application for reducing their complexity, while preserving the character of the input-output relations as closely to the main system as possible [18]. It is necessary to keep the reduction procedure computationally efficient, while concurrently preserving the main properties of the original system, i.e., the approximation error remains small.
For instance, [19] introduced a model order reduction such that "the knowledge of a smaller number of variables is required to compute and optimize the target variables"(output). They first use spectral clustering to create clusters of input variables that are similar to each other. This is done by creating an adjacency matrix from which a graph Laplacian is calculated, including eigenvalues and eigenvectors. The number of clusters are decided based on the eigenvectors. The clusters are then ranked using a “page rank” algorithm. The cluster(s) with the variables that have the highest page rank is the most important one.

In machine learning context, the number of input variables and features in data is considered as the dimensionality of the learned model. A large number of dimensions in feature space, is likely to result in overfit to the training data and decline the performance. Dimensionality reduction, in return, results in simpler models with improved generalizability and explainability and higher execution performance. Feature selection is one of the most common techniques in this context. It mostly applies scoring or statistical methods to select the irrelevant/unimportant/redundant parameters. Specifically for DTs, MR methods have been applied in order to improve computational efficiency and is identified as a key technology for their development [20]. The authors of [21] show how MR can be applied to DTs in six use cases: Virtual sensors, Predictive maintenance, Operation control, Drivetrain analysis, Lifetime analysis, and Circuit Simulation.

The model compression techniques are useful to reduce the complexity of ensemble models. These types of models are generated by combining multiple other models to represent one original system or process. The disadvantage of ensemble models is that they are mostly large and complex, which increases their application cost concerning the required memory space and computational power, specifically for real-time predictions. The model compression techniques are learned to be convenient solution for this issue, such to gain a simpler yet accurate model. For instance, the proposed compression technique by [22] is to first train the ensemble model with the original data set, and then train the neural net on this ensemble model outcome. The final trained neural net performs similar to the ensemble, and much better than a neural net trained with the original training data, but with lower computational cost compared to the ensemble model.

During modeling optimization, the parameter space is considered as one of the targets to alter in order to reach a higher execution efficiency. Applying MR techniques, for instance, reduces the parameter space in such a way that it generates a less complex but accurate model. Apart from that, the field of surrogate modelling supplies other techniques that can help us to generate the surrogate model of the CPS without necessarily altering the parameter space but changing the structure of the model. In the following we present a short excursion to some other means of optimization.

[23] defines surrogate models in general as an “Algebraic approximation fitted to available data points.” [24] makes the definition more specific by stating: “[…] surrogate models (also known as metamodels, regression models or emulators) […] are mathematically simple models that map or regress the input–output relationships of a more complex, computationally intensive model.” In other words, given sample points generated by the original model which does not meet the execution efficiency requirements, methods can be applied to generate a mathematical model which mimics the input-output relationship without adopting any knowledge about the inner workings of the system itself. Thus, converting white-box or grey-box into black-box models. The main goal is to reduce computational demand while losing only a little accuracy due to the approximation in order to perform applications where high number of samples are required such as optimization, sensitivity analysis or, as in our case, AI training [24]. In order to counter the error arising between the original model and the surrogate model, an additional model can be put in place which simulates the difference between the two models based on the sample sets [25]. [24] performed a literature review on the use of surrogate modelling for the generation of DTs. They list 25 publications where the surrogate modelling methods have been applied and showcased different methods together with the required sampling methods. Examples are Kriging, training neural networks, fitting polynomial functions to name a few.
While the usage of surrogate models is advantageous in many use cases with respect to DTs, [24] points out two difficulties that remain. On one hand, the creation of the sample points required for fitting might not be feasible and annuls the expected saving of resources. [23] suggests here lower-fidelity simulations using e.g., simpler geometry, physics etc. On the other hand, continuous maintenance of those models is required since they were created at one point in time and do not evolve as the original model might do.

Surrogate models are often applied in multi-fidelity approaches. A Multi-Fidelity Model (MFM) is a “model constructed using the information of multiple models with different levels of accuracy” [23]. They differ with respect to two concepts of sub-model relation in MFMs. A Multi-Fidelity Surrogate Model (MFSM) where surrogate models are constructed using the information of multiple models with different levels of accuracy and Multi-Fidelity Hierarchical Models (MFHM) where the fidelity (or surrogate model) is chosen based on a criterion.
The authors of [26] have used a multi-fidelity approach to estimate the crashworthiness of a bus bumper system. It showed that the computational costs were reduced by 33% while the composite objective function value deviated 2% from the original high-fidelity optimization alternative. An example for the application of this framework close to the application of ASIMOV, is from [27] which have applied a multi-fidelity framework, thus using multiple versions of surrogate models, to optimize the training of an Artificial Neural Network (ANN). Their idea is to dynamically increase the fidelity of the models when needed over the course of the training as at the start it is assumed that the AI does not need such accurate results.

A new approach for parameter identification based on Particle Swarm Optimization (PSO) algorithm is proposed in [28] for non-linear model of Permanent Magnet Synchronous Motors (PMSM). They provide simulation as well as experimental results to validate the effectiveness of their method. The approach is developed through finding an efficient model parameter tracking method for PMSM that operates under different conditions, and meanwhile avoiding the more complicated model structures, e.g., Finite Element Analysis (FEA). The summary of their algorithm is illustrated in Figure 5.

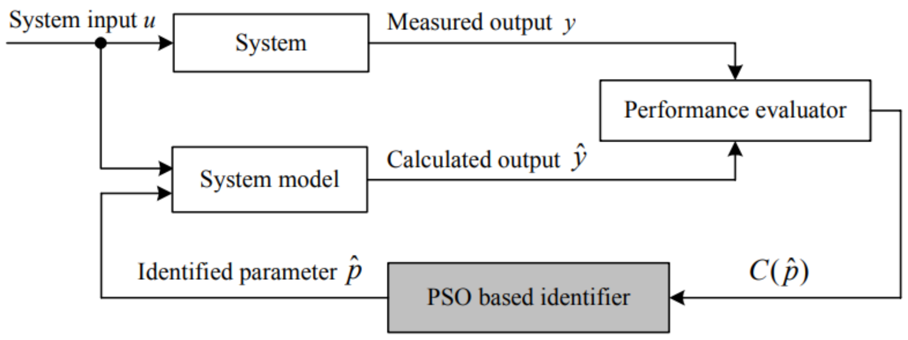


Figure 5 PSO based parameter identification block diagram [28]

An identical system input, *u*, is given to the original system that should be identified and its model. Their outputs are input to the performance evaluator component to output the fitness. The calculated fitness is then input to PSO based identifier to identify the unknown parameters vector. The identification can be applied in both offline and online processes.

In [29], a process is described, which isolates the relevant parameters from the ones with less impact on the result. It is based on providing ranges for parameter values based on expert knowledge, where then individual calculations are performed with discrete samples of parameter combinations inside the limits of their respective ranges. After that, the results are analyzed, in terms of how accurate they show the behavior of the real process. By using a Monte-Carlo process, the relevant parameters can then be separated from the less important ones.

4 Current process in the use cases

In this section, the current process used for the identification of relevant parameters is described for the two use cases. The current procedure for identification is purely based on the knowledge of experts. The description will be used later to evaluate the generic identification process to be developed here.

4.1 Current identification process - UUV

The first step is to identify the components that are available as physical counterparts. The physical parameters of these components can be directly mapped from the real system to the DT. Depending on how accurate the DTs need to be, a high-fidelity simulation based on physical models is necessary. Should it be more important to provide higher execution efficiency, lumped parameters, that approximate the behavior instead of the physics, can also be used for a computationally less intensive model.

Vehicle dynamics models are very mature and therefore the important parameters are already well established in the automotive industry. As the goal of the UUV use case is to identify relevant variations of traffic scenarios, this section will focus on the parameter identification for the scenario generation.

A traffic scenario as defined in OpenSCENARIO and OpenDRIVE, which are part of the ASAM OpenX Standards [30], can be divided into two main parameter blocks. The dynamic part of the scenario describes the behavior and interaction of traffic participants. The static part describes elements, such as road network, buildings, road signs, etc.

The parameters that will be varied by the Reinforcement Learning agent will modify the static part to find critical scenarios that challenge the controlling system. In our case, this includes for example the density of trees on the roadside, positioning of parked cars, time of day, etc., which can be seen as control input u and are derived by expert knowledge or plain common sense. These parameters provide the possibility to identify the influence of changes in the 3D environment on the vehicles' perception and behavior in a given dynamic traffic scenario.
Time of day as well as the density of trees on the roadside both have a direct impact on the perception system of the UUV consisting of LiDAR and camera sensors. Parked vehicles reduce the visibility of road markings and other traffic participants such as pedestrians and are therefore important to induce critical inputs for the vehicles' decision making.

The dynamic scenario definitions were created based on conventional NCAP scenarios for safety ADAS testing [31]. This contains a set of typical critical traffic scenarios, that are used for assessing the safety of autonomous driving functions.
To describe these dynamic scenarios for the UUV use case, the ASAM OpenX Standards are used. They provide a large set of parameters to define vehicles' trajectories, initial positions and speeds, as well as actions that trigger interactions between traffic participants. As these parameters will not be changed during simulation, they can be seen as system parameters c.

A limited number of parameters can therefore already be used to recreate a typical traffic situation. At the start of the traffic scenario, the ego vehicle’s position on a given road network, as well as its orientation needs to be defined. Using global coordinates in combination with a continuous heading value is not suitable, due to the fact that valid positions are referred to the roads according to the ASAM Standards. Instead, meta information from the road network can be used to place the vehicle directly on a valid piece of road. This can be achieved by using the individual road and lane IDs, as well as a value s and t, which define the longitudinal and lateral offset from the start of this road segment in local coordinates relative to the reference line of the lane. An overview can be seen in Figure 6.

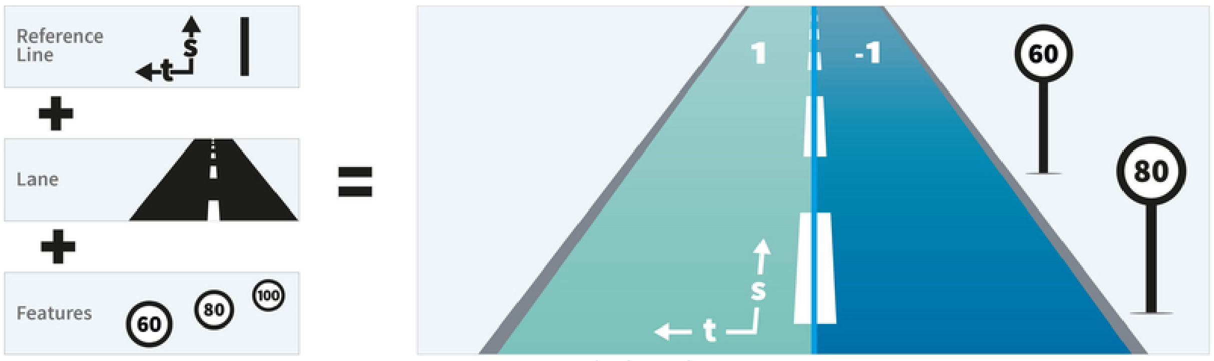


Figure 6 Elements of ASAM OpenDRIVE [32]

A vehicle's initial orientation can be directly linked to the allowed direction of the respective road segment. Additionally, the starting velocity needs to be defined. This needs to be done for all dynamic traffic participants.

The route of every vehicle can be either defined by trajectories, that describe the exact path, or it can be defined by selecting waypoints, the vehicle automatically navigates to.

With these parameters in place, the ego vehicle can move through the environment in a well-defined manner. Critical traffic situations often require interaction with other traffic participants. Such traffic situations can include other vehicles, motorcycles, cyclists, pedestrians, etc. The type of these traffic participants needs to be defined in a parameter according to ASAM OpenSCENARIO. Furthermore, these traffic participants need their routes defined as well. This can be done in a way similar to the ego vehicle.

To align the movement of other traffic participants to the ego vehicle, actions are used. Actions are conditional elements, that measure some value and trigger an instance to do something, when activated. Actions are typically activated when reaching a certain relative distance to an object, position or speed to induce critical scenarios. The resulting action can include multiple elements, such as a lane or speed change. An example would be a scenario where the ego vehicle drives along the street. As soon as it reaches a predefined position, it triggers a pedestrian to move on the street which creates a critical situation.

4.2 Current identification process – STEM

This section describes the parameters for the DT of an electron microscope (EM), which is mainly based on the domain expert knowledge. During the ASIMOV project, we may identify that more parameters must be considered in our DT to sufficiently represent the physical system for the application of interest. On the other hand, we might conclude that some of the parameters can be removed without significant consequence for the accuracy of the results.

The DT of an EM consists of several models, each representing a part of a physical EM. This includes models for electron source, electron beam condenser system, projection system, camera, and interaction of the electron beam with a specimen. Each model includes several parameters. However, to decide which parameters are relevant for the DT, an application/goal should first be defined. Depending on the application, a few models are selected and combined. Afterwards, the relevant parameters can be identified.

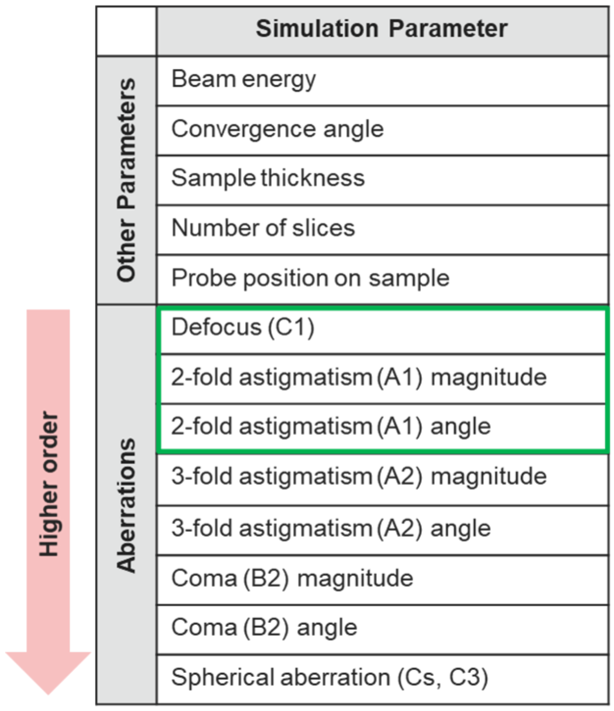
The application selected for the ASIMOV project is the aberration correction in a transmission electron microscope (TEM) system. Achieving the highest resolution in an electron microscope requires correcting aberration of the optical system. The initial focus of the ASIMOV use case is the 1st order aberrations, namely, 2-fold astigmatism and defocus (Although we have started with including 2nd order aberrations as well in later stages of the project). These aberrations drift very rapidly over time and, therefore, they need to be corrected, for almost every image taken. 2-fold astigmatism and defocus are the control parameters for the initial DT of TEM (noted by **u** in Figure 1).

Additionally, higher order aberrations with small values are added to the simulated data to represent the effect of uncertainties in current available aberration correction methods. In other words, when data from a physical TEM is collected small values of higher order aberrations, possibly even unclear to the human eyes, may be presented, and in response to that the simulated data should also include these variations so that the reinforcement learning agent learns about them. Another example of disturbances are camera noise and effects of temperature fluctuations in atomic structure of the specimen. These are considered as disturbance parameters (noted by **d** in Figure 1).

In addition to the aberration coefficients, the properties of an electron beam, such as beam energy and beam-limiting convergence angle determine the status of the system. We consider these as system parameters (noted by **c** in Figure 1). These parameters largely affect the appearance of the Ronchigram images (a type of EM image based on a diffraction pattern, which we use for the aberration estimation), while being constant during the time scale of a particular measurement. The difference to the control parameters (**u**) arises from the application, as we are aiming at changing and controlling aberrations (**u** parameters) but not parameters like the beam energy (**c** parameters) for the chosen application.

Gradually, more computational methods will be added to the DT in order to make it more complete and thus more realistic. In each step, depending on the available models in the DT, a set of parameters for the numerical algorithms will also be present. However, these parameters do not necessarily have physical equivalent in the electron microscope and are considered as internal configuration parameters of the DT. We will not target this type of parameter in the identification process of this document.

Figure 7 shows some of the most important parameters of the current EM DT. This green box shows the control parameters for the first use case within ASIMOV.



	Simulation Parameter
Other Parameters	Beam energy
	Convergence angle
	Sample thickness
	Number of slices
	Probe position on sample
Aberrations	Defocus (C1)
	2-fold astigmatism (A1) magnitude
	2-fold astigmatism (A1) angle
	3-fold astigmatism (A2) magnitude
	3-fold astigmatism (A2) angle
	Coma (B2) magnitude
	Coma (B2) angle
	Spherical aberration (Cs, C3)

Higher order

Figure 7 Most important parameters of the current EM DT

5 Parameter identification process

This chapter aims to understand where in the entire process of the ASIMOV solution, the identification of relevant parameters is done.

5.1 Parameter identification in the context of ASIMOV

We locate the goals of this task at the beginning of the ASIMOV process. Here, a DT is modeled on the basis of a typical system. The purpose is to build a model that can simulate the behavior of the real system so that the RL agent can train to react sufficiently correctly to a real system later on after its training. In addition, within the DT development process, the identification of parameters is one of the first things to be considered. However, due to the iterative process of modelling, topics of T2.1 appear multiple times during the process.

One differentiation that needs to be highlighted is with respect to T4.1. While on first sight task T4.1 deals with the twinning (see [33]), in other words, finding the correct parameter so that the DT and the system are in line with each other, T4.1 evolves more around a continuous validation of the DT during the operational phase where the model is not altered in its nature anymore. T2.1 takes place at the beginning of the ASIMOV solution where a model is first implemented.

Models are always purpose-built. Thus, the fulfillment of the requirements derived from this purpose is crucial (see [34]). Requirements can be categorized in multiple “quality characteristics” as described in ISO/IEC 25010 which is concerned with system and software quality models. Due to the nature of the system of training an RL agent using a DT, two quality characteristics stand out to be of high importance: "Functional Suitability” and “Performance efficiency” and especially their corresponding subcategories “Functional Correctness” and “Resource Utilization”. Despite the standardized naming, we adopt the wording of [35] naming them “Accuracy” and “Execution efficiency” respectively.
We define them as follows:

- Execution efficiency: Given a fixed computer system resource and two models with comparable outcomes, the one whose simulation finishes in a smaller amount of time has higher execution efficiency.
- Accuracy: A model with high accuracy possesses little deviation of its output compared to the real system with equal inputs.

Without achieving both execution efficiency and accuracy, the model is not suitable for its purpose of providing training data to the RL agent. Interestingly, they possess a fundamental and intuitive trade-off. In order to improve accuracy, generally speaking, one needs to do more calculations which decreases the execution efficiency.

We acknowledge that the development of a DT consists of multiple parts (data storage, twinning infrastructure, …) (see WG architecture [36]) but in the phase prior to the training we locate the task, we deal with a “digital model” (see [37]). A digital model is nothing more than a classical model of some system with no inherent initial constraints with respect to its execution efficiency capabilities. In classical models, securing the accuracy of the model is part of the validation process that is abundantly researched already and will be addressed by task 2.5. In the ASIMOV approach, due to the importance of being able to train the AI with a reasonable number of resources and the fact that the digital model only needs to be so accurate that the RL can find good solutions in the real world, performance is more highly valued than in a classical modelling approach. We argue that one cannot model the digital model in a classical way focusing on accuracy and assume that they produce a model that fulfils the requirements of quality characteristics given by the ASIMOV approach. Methods need to be applied which consider both quality characteristics. We cannot assume that execution efficiency is checked during Verification and Validation as they evolve around the accuracy of the model w.r.t. the real system according to the literature. Oberkampf and Trucano [38] state “Verification and validation (V&V) are the primary means to assess the accuracy and reliability of computational simulations.” [35] identifies that “Verification, validation, testing, accreditation, certification and credibility assessment activities primarily deal with the measurement and assessment of accuracy of models and simulations (M&S). Also [39] notices that “A

model is considered valid for a set of experimental conditions if the model's accuracy is within its acceptable range, which is the amount of accuracy required for the model's intended purpose".
This does not mean that methods for evaluating and improving execution efficiency do not exist. In fact, a selection of such methods will be shown later in this document as the aim of the task is to collect them and test them for the ASIMOV solution over the course of the project as they should be an integral part of the development process of the DT.

5.2 Parameter identification during a generic DT development process

In this task we aim to find methods that can find the best trade-off between the two quality characteristics introduced above. Also, the methods applied in the parameter identification process can be divided into two fields. We distinguish between Parameter identification and Optimization:

- **Optimization**: These methods support the maximization of overlap between the set of candidate parameters and the actual relevant parameters (see Figure 2). Simply, by reducing the number of input parameters or setting a variable parameter constant, a simplification of the model happens since the parameter space is reduced as well as fewer interconnections within the model need to be calculated.
- **Parameter identification**: These methods support the determination of the value of the given parameters. Here we assert the concrete value as the system parameters *c* and/or the range the control inputs *u* and disturbances *d* can have. Again, with a tighter range of possible values the parameter space is reduced. In addition, when system parameters influence the accuracy and/or execution efficiency (e.g. 3D model mesh size), selecting the correct value is paramount.

Similar to the accuracy characteristic, the execution efficiency should be improved and evaluated in multiple stages of the DT development process. In order to identify when such methods are applied, a flow chart has been developed as a first iteration of realizing a "relevant parameter identification process" which can be seen in Figure 8 and includes the major considerations one needs to take while modelling a CPS on DT. This implies that the diagram omits some steps of a normal development process while adding additional iterations for the optimization with respect to execution efficiency. Therefore, the DT developing process can differ from the one of conventional modelling. The use cases showed that not every DT's model is created from the ground up but could also be an existing model, adapted to the requirements the DT concept imposes on it. Thus, the first question that may be asked is whether planning and modelling can be omitted and the model can be directly tested to see whether it meets the requirements.
If no model exists, the next question is whether sufficient data about the system's behavior is available.
If not, first-principle are the only option. First-principle models are based the formalization of internal workings of the to-be-modelled artefact and are sometimes also called physics-based models. However, if sufficient data is at hand, it is still possible that the system is numerically intractable and cannot be expressed via a "simple" input-output function. An example of this is the UUV Use Case where the UUV's reaction within the simulated environment to a given scenario is complex, stochastic and cannot be expressed in e.g., differential equations. Here first-principle models are mandatory (or at least need to be decomposed into subsystems). Finally, if there is enough data and one is sure that the system is numerically intractable, one could prefer and choose to use a hybrid modelling approach instead of a fully data driven model. Reasons could be increased explainability and accuracy as it combines the advantages of first-principle and data-driven approaches. Both options produce a model where relevant parameters and their values can be extracted from data.
With the models specified, either built on existing models or using data-driven/first-principle, the DT development proceeds within a validation and optimization loop. The validation asks two questions: (I) "Is the model accurate enough?" and (II) "Is the model fast enough?" If both questions can be answered positively, the development for the DT's model is done.
If the model is not performant enough, it can be optimized. Here one can apply methods again which are concerned with "finding relevant parameters". On one hand, the model can be tweaked by applying methods like sensitivity analysis to find parameters that have small impact on the variance of the output. On the other hand, methods for creating surrogate models of the existing model can be applied.

After the model optimization, it has to be checked whether the model is still accurate enough. This assumes that optimization is always a trade-off between the accuracy of the model and the time of simulation. If the model is not accurate enough there are three options: either the optimization is redone, the discrepancy between the model and the actual system is modelled, or the entire model needs to be reassessed. If the model is accurate enough, the developer would then go on to check if it is as performant as needed, and the cycle continues.

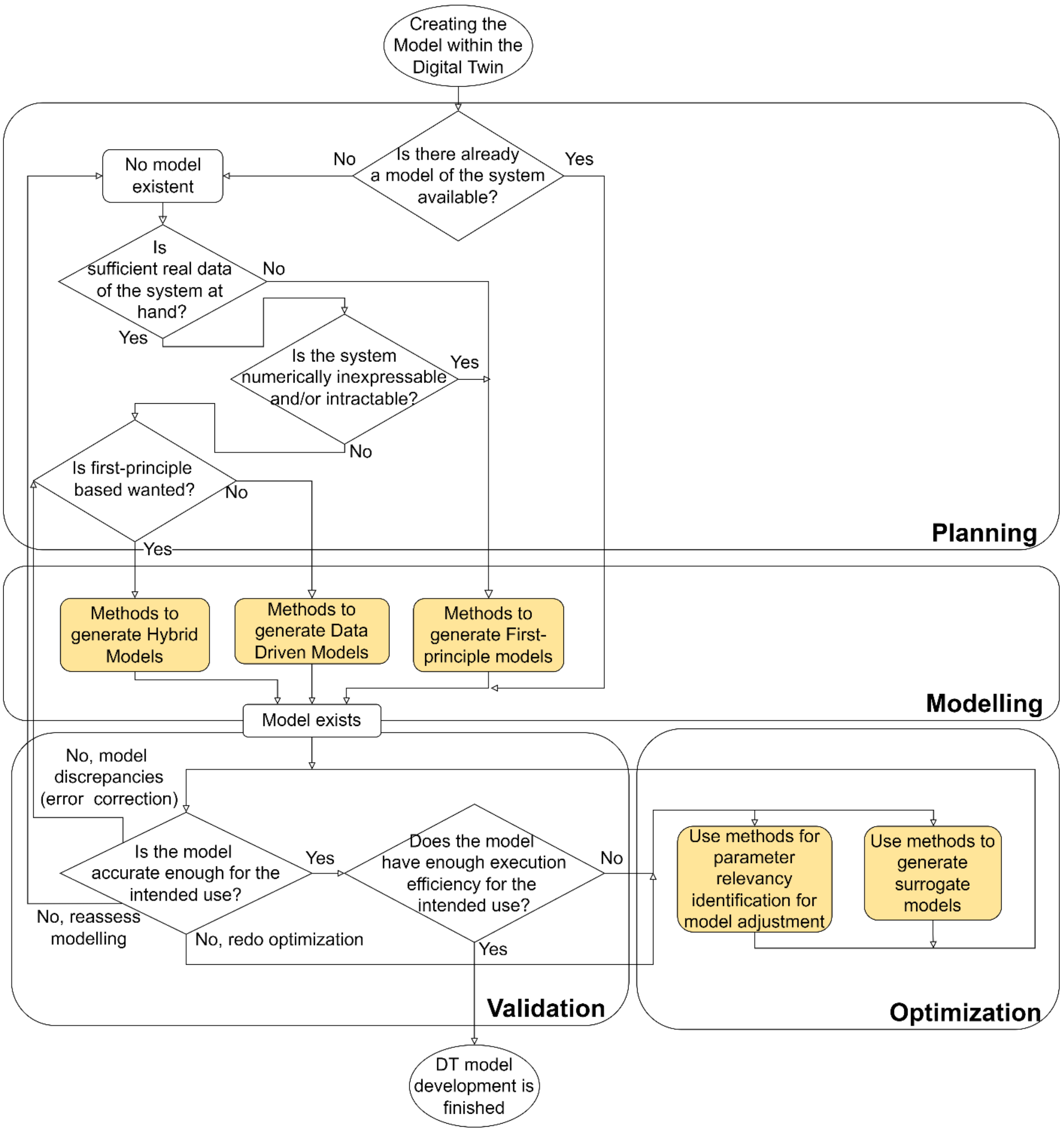

Creating the Model within the Digital Twin
Is there already a model of the system available?
No
Yes
No model existent
Is sufficient real data of the system at hand?
No
Yes
Is the system numerically inexpressable and/or intractable?
Yes
No
Is first-principle based wanted?
No
Yes
Planning
Methods to generate Hybrid Models
Methods to generate Data Driven Models
Methods to generate First-principle models
Modelling
Model exists
No, model discrepancies (error correction)
Is the model accurate enough for the intended use?
Yes
No, reassess modelling
No, redo optimization
Does the model have enough execution efficiency for the intended use?
No
Yes
Validation
Use methods for parameter relevancy identification for model adjustment
Use methods to generate surrogate models
Optimization
DT model development is finished

Figure 8 Process diagram for developing a DT highlighting the steps where relevant parameters are determined.

In Figure 8 we have primarily identified two stages of the development process where methods of “finding relevant parameters” can be applied. On one hand, the first stage is the application of methods that can be done during the modelling prior to integration. In this case, the methods are applied and estimate the execution efficiency, thus shaping modelling decisions. The second stage is after the model or sub-model(s) is implemented. Here methods are applied that are capable of identifying points of improvement. This is summarized in Figure 9.

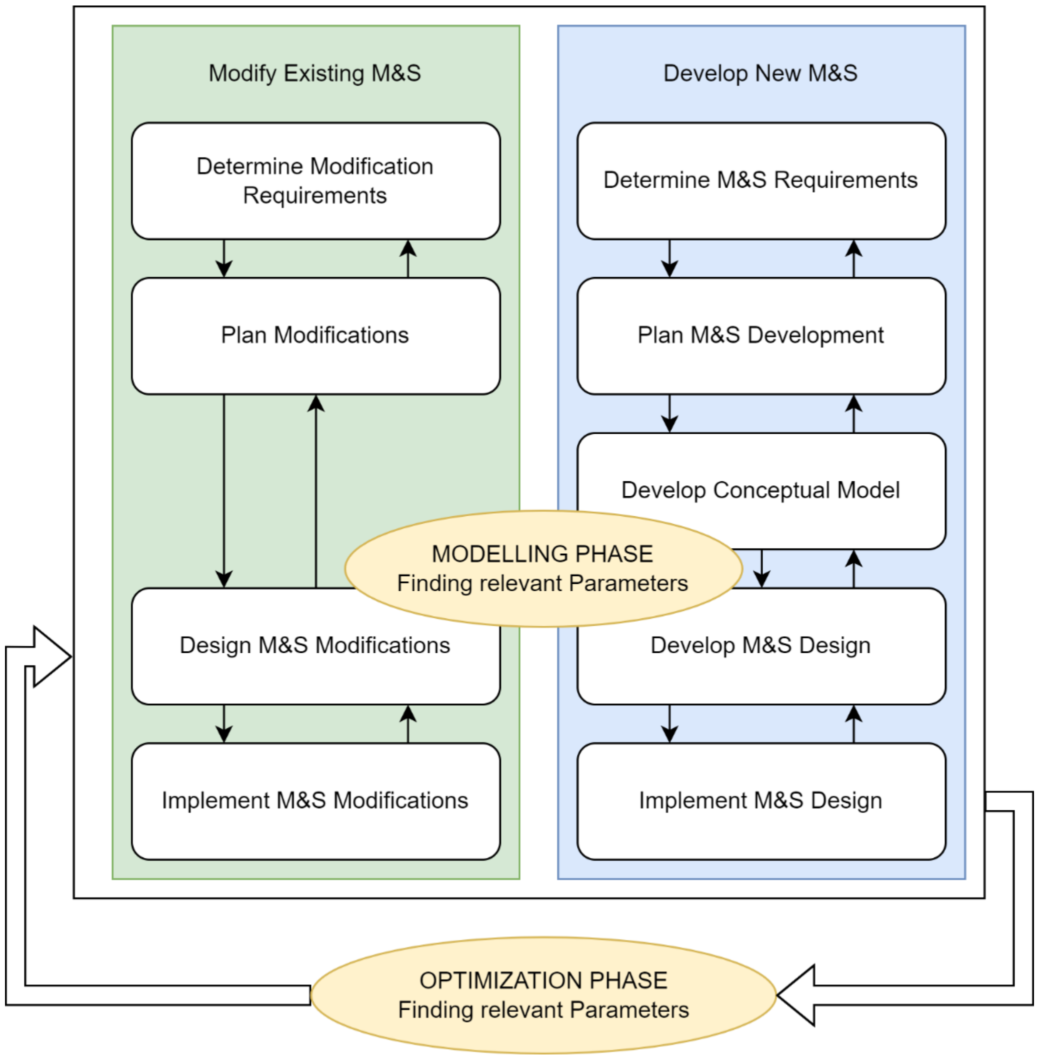


Figure 9 Condensed diagram of phases for finding relevant parameters (Based on DoD (1996) in [35])

The flow chart in Figure 8 highlights those methods that need to be considered for all three natures of models, physics-based, data-driven and hybrid models and also shows that there is an iterative process between operational validation and optimization or modelling. As explained above, validation is checking whether the model meets the quality characteristics for its intended use while optimization and the modelling improve accuracy and execution efficiency.

Since the work in this task revolves around the definition of “relevant” which is tightly linked to the validity of the DT, the final “parameter identification process” must be aligned with the validation process. Thus, close cooperation with T2.5 will be necessary. Central will be how and when the methods to improve execution efficiency can be applied and how and when they are evaluated. In addition, existing and possibly new methods to improve a DT’s execution efficiency will be tested. A selection of these methods from the State of the Art can be found in the following sections.

5.3 Adaptation Model

The parameter identification process also requires a deep understanding of how certain parameters of the model can be estimated beforehand and which parameters might require data from a real system to be sufficiently selected. This is especially true for hybrid models, where parts of the model are designed with a first-principal model in mind, while others are then refined by using real datasets.

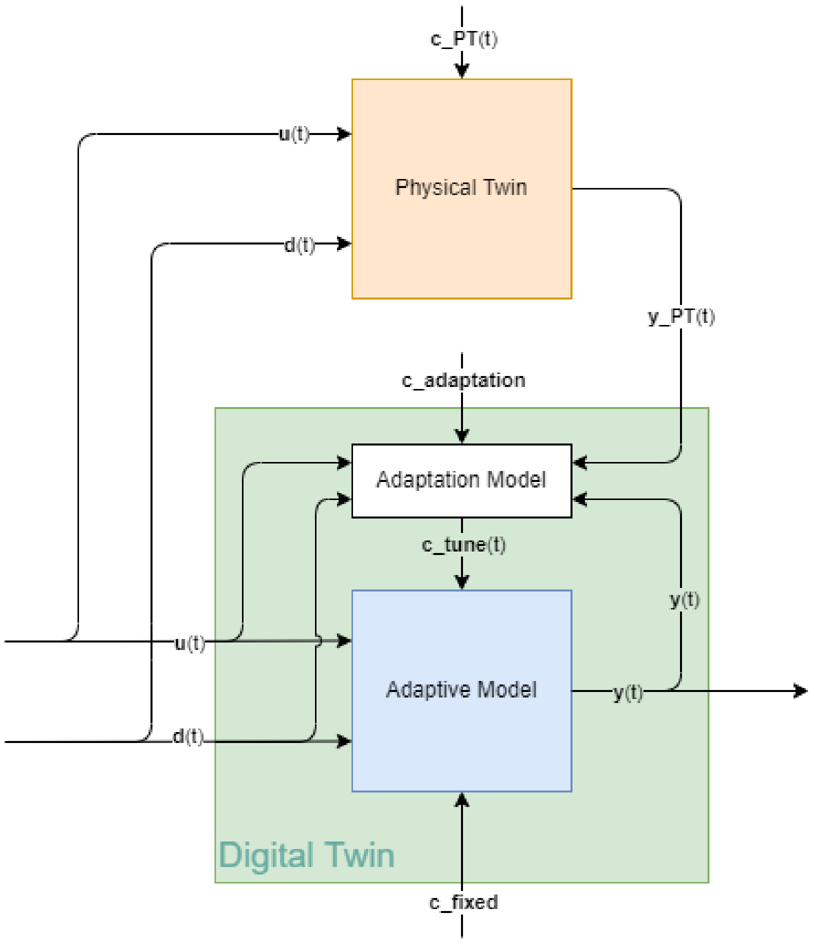


Figure 10 Adaptation Model

The idea of the adaptation model has been introduced in [2]. It is depicted in Figure 10 In short, the idea is that this adaptation model is the main differentiating factor between a digital model and a digital twin. It allows the DT to adapt its behavior, based on inputs of the physical twin. The adaptation model is hereby only the part of the model, that actually changes depending on physical twin feedback. The remaining model then only describes behavior that is not affected by physical twin feedback.

This leads to the fact that two main categories of parameters can be distinguished.

- Parameters that have a constant value
- Parameters that change value depending on PT feedback.

When building a model, it is important to identify these categories for each parameter. Parameters that depend on PT feedback are not completely free to choose by the adaptation model, however. It might be that a certain plausible range or distribution for values can be provided.

Examples for parameters that have a constant value are generally the ones that can be easily measured beforehand. Such parameters could be the suspension geometry or the wheelbase of a vehicle.
Parameters that could be subject to value changes during operation might be the mass of the vehicle. There might be a typical range for that, but each specific vehicle will, also depending on the number and weight of passengers and/or luggage, have a different mass.

6 Use cases specific

This section addresses topic-specific aspects for the STEM and UUV use cases.

6.1 STEM

This section covers topics related to the STEM use case. The “Parameter Tuning” section explains how to determine the optimal configurations of various parameters. The “Parameter Identifiability Problem” section addresses the difficulties that can occur during parameter identification.

6.1.1 Parameter tunning

The TEM use case aims at aberration correction using electron diffraction patterns (CBED, a.k.a. Ronchigram). Naturally, the control parameters are aberrations, but in addition there are system parameters, such as electron energy and aperture size which remain relatively fixed during an experiment.

Figure 11 demonstrates a high-level view of the connections between the digital twin (DT) and the real system. The requested input parameters by a TEM user (either directly or through an application of interest) are not necessarily the inputs that can be directly used in the real physical system or in the DT models. Therefore, there is a translation layer in between. The translated input parameters then are passed to both the real system and the DT.

Although the digital twin core consists only of physics-based models, in order to build an operational digital twin the post processing, optimization and control blocks (all blue blocks in Figure 11) are also crucial. The output of both DT and the real system are passed to the post-processing block where a model, which can vary in complexity, performs a type of feature reduction, e.g., generating a scaler. Subsequently, the optimization model uses the post-processed outputs to find the closest set of parameters in DT which resemble the real system. Once the DT output is sufficiently close to the real system output a control model will change the input parameters. An example of the control model is a trained reinforcement learning model as it is used in the ASIMOV approach.

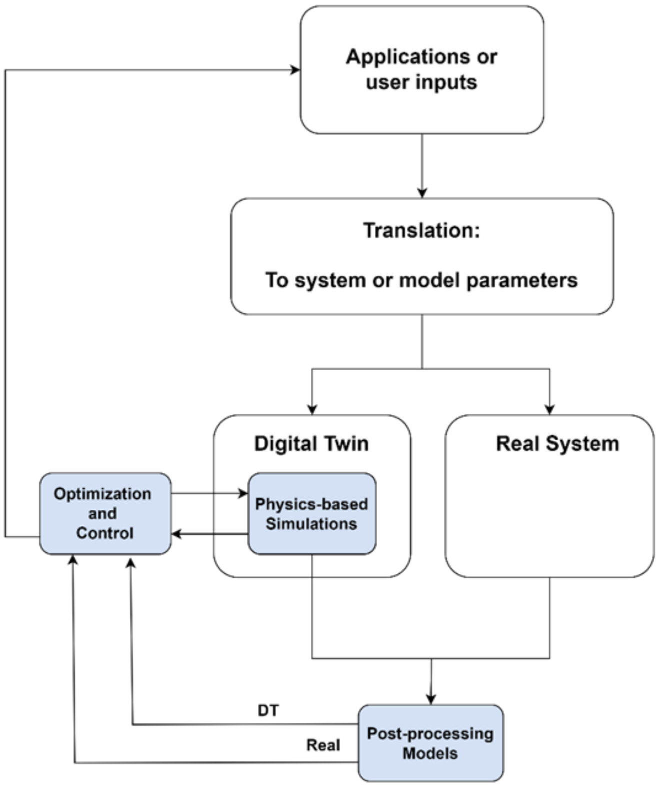


Figure 11 The connections between a real system and it digital twin

The optimization step is also known as DT parameter tunning. This step is necessary as the observed image (CBED pattern in our use case) has a strong dependency to some of the system or disturbance parameters. For example, changing the integration time of acquiring an image on camera affects the amplitude of Fourier transform (FT) of the CBED. In DT, there is an equivalent parameter for the expected number of electrons per pixels which is incorporated into a camera model. This parameter can be tuned to represent the correct integration time in the real system. Figure 12 shows such a relation for the measured data on real TEM. The elongation of the ellipse-like shape in FT is an indication of changes in the control parameters, i.e., the aberrations, defocus and 2-fold astigmatism. However, a similar effect is observed due to different levels of integration time. Therefore, it is important to calibrate the DT camera model for this parameter. The parameter tuning can be extended to more parameters and can become automated which is also currently under development.

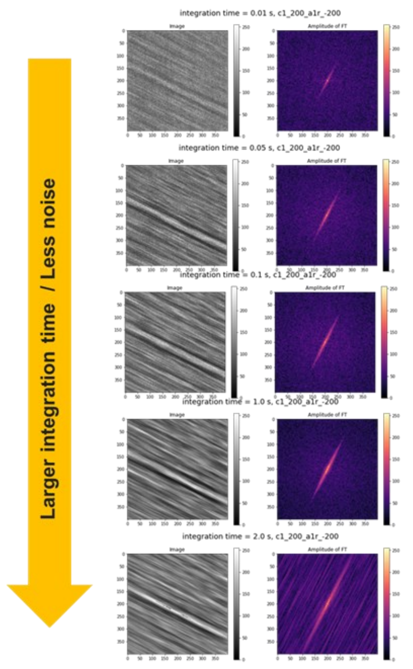


Figure 12 Sensitivity of the observed image to integration time. Data from a real TEM.

6.1.2 Parameter identifiability problem

A model has identifiable parameters if, given infinite number of observations, it is possible to obtain the value of its parameters. In other words, in a model with identifiable parameters it should be theoretically possible to solve the inverse problem. If the observations of a model are the same for more than one set of parameters, then the parameters are not unique and thus the unique parameters cannot be identified. To handle this problem one could use a subset of parameters, a smaller range of parameters, or use a different observation which does not cause this problem.

In case of TEM aberrations, we face the problem with parameter identifiability. This means that the CBED (and its FT) for different set of control parameters can be almost identical. Figure 13 describes this problem. In Figure 13.b and Figure 13.c the images on the grid are generated using different aberration settings (i.e., defocus and astigmatism). The FT of CBED is very similar for multiple sets of parameters as seen in Figure 13.c. in yellow and boxes. This originates from the aberration function definition of the wave that propagates through the optical column of the microscope (Figure 13.b and Figure 13.a). In Figure 13.c also the green boxes, although not theoretically identical, are very similar to the level that it can be impossible to distinguish them in presence of noise in data. Therefore, given one observation (CBED or its FT) finding the unique parameter set is a difficult problem whether it is for classical inverse problem or a deep learning model.

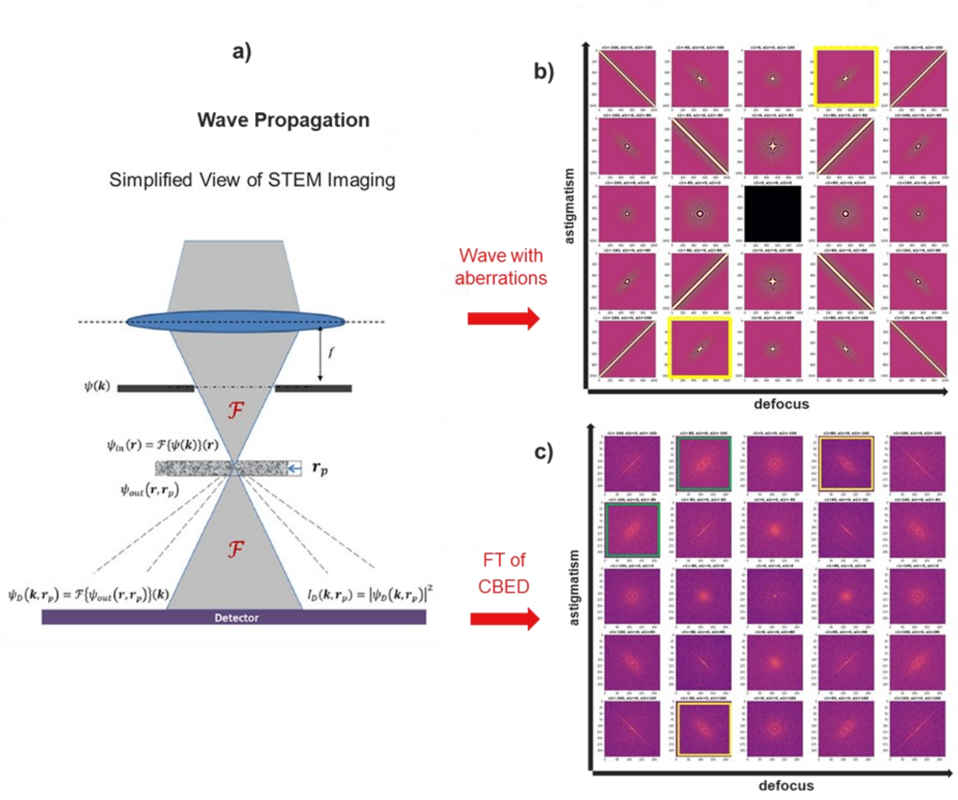

a)
Wave Propagation
Simplified View of STEM Imaging
$\psi(k)$
f
$\mathcal{F}$
$\psi_{in}(r) = \mathcal{F}\{\psi(k)\}(r)$
r_p
$\psi_{out}(r, r_p)$
$\psi_D(k, r_p) = \mathcal{F}\{\psi_{out}(r, r_p)\}(k)$
$I_D(k, r_p) = |\psi_D(k, r_p)|^2$
Detector
Wave with aberrations
FT of CBED
b)
c)
astigmatism
defocus

Figure 13 The parameter identifiability problem observed in FT of CBED and its origin in aberrated wave. 13a, modified from [40]

The solution to the intrinsic problem of the parameters is handled in the control/AI part of the work. Here, we provide a summary of two approaches that were investigated toward this goal. One solution is to change the observation, by including additional information to re-establish the Markov property. Through this approach, the identification problem is reduced, albeit not entirely eliminated. Another approach involves simplifying the identification problem to only finding the location with least aberrations (the middle image in Figure 13.c) by calculating a quantity which reflects a notion of "better", and subsequently, given this new objective, a classical optimization solver can be applied to find location with the minimum value.

6.2 UUV

This section is dedicated to the UUV use case. It is divided into three subsections, each addressing a key aspect of UUV development and application. First, the influence of the level of detail of objects in the 3D environment on the criticality metric is analyzed. Finally, the application of the parameter identification process from Chapter 5 is demonstrated through two examples. The first example shows the development of a sensor and the challenges involved. The second example uses a LiDAR validation case study to illustrate the practical application of the process.

6.2.1 Level of detail in the environment model

6.2.1.1 Introduction

A DT is a virtual copy of a physical object or system that reflects its behaviour. An essential part of this concept is the environment model that defines the environment with which the digital twin interacts. The trade-off between detail and execution efficiency of the environment model is a key challenge.

One performs Scenario-based Testing (SBT) for a purpose: to validate the correct functioning of a specific system. The functioning of the system is assessed in "test cases" by combining a concrete scenario, (criticality) metrics which are selected based on the System under Test (SuT) specific requirements [see e.g. [41]] and associating a pass-or-fail criteria to it in order to have a test measure. The simulator's only purpose is to reproduce the value of this metric in such a way that the resulting pass-or-fail criteria is indistinguishable whether calculated in a real world or corresponding virtual scenario. This is the central assumption of the proposed idea.

A DT relies on a realistic environment model tailored to its intended functions. This can include, for example, the appearance or physical behaviour of objects. An environment model defines the context and boundaries. Through simulation within this environment model, different scenarios and conditions can be explored. The goal is to provide the digital twin with all relevant data and contextual information from its environment that are necessary for a successful simulation. The higher the accuracy of the environment model, the more accurate the simulation results will be.

The state-of-the-art simulators (i.e. CARLA, Hexagon VTD, IPG CarMaker, ...) usually used for such simulations provide with ongoing updates an ever-increasing level of fidelity in every aspect ranging from highly-complex physics models to photo-realistic model assets [e.g., [42]]. Naturally, the pursuit of high-fidelity visual models in every aspect of the simulators, while justifiable as they aim for general-purpose, entails increased computational costs and necessitates additional resources, even though such precision may not be essential when testing only specific functionalities or taking sensor limitations into account.

It is likely that models with a lower level of detail require fewer resources, such as computing power and memory, which means that faster simulation runs can be performed. This is particularly advantageous for the many iterations required in reinforcement learning. Furthermore, simple models can be created faster and require less data to create. For large environments, it may also be useful not to represent all objects at a high resolution to achieve an acceptable simulation time.

However, there are two uncertainties: First, how much does it affect the measured criticality metric? And secondly, it is unclear whether changing the accuracy of the 3D assets within the simulation makes a difference in terms of execution efficiency. The goal of this analysis is to determine what influence the level of detail of certain objects has on the criticality matrix for certain scenarios.

6.2.1.2 On validation and credibility in the automotive context

Central to the utilization of virtual SBT is the simulator's credibility, as without it, the undertaking lacks purpose since not trust in the system can be generated. Liu et al. [43] defines credibility of a model or simulation as "an expression of the degree to which one is convinced that a particular model or simulation are suitable for an intended purpose." Validation as seen also by Sargent [39] and Oberkampf and Trucano [38] refers to "the process of determining the degree to which a model or a simulation is an accurate representation of the real world from the perspective of the intended uses of the M&S [Modelling&Simulation]" [44]. The evaluation of validity is not the only contributing factor to credibility, it is sure the most significant metric [43]. According to these definitions, credibility is fundamentally an informed judgment derived from multiple factors, whereas validation is a (but not exclusively [39]) objective assessment based collected data on predefined metrics.

In the automotive domain, scholars have been researching the credibility or validity of simulators for virtual SBT. For example, Riedmaier et al. [45] present a process that combines model validation and safety assessment for automated vehicles. Stadler et al. [46] introduces an empirical way of assessing the credibility of an automotive simulator by proposing several metrics. Both approaches propose statistical means for a data-driven validation approach by comparing the simulators output at different stages of the simulation chain given a certain scenario to the same scenario measured in reality as the ground-truth. A primary requirement of the system resulting from the proposed work is an optimized simulator and models shall still be credible (therefore the methods can be applied) and ideally "just credible enough".

The initial motivation comes from the fact that numerous studies [47], [48] have been performed to assess the impact of fidelity levels in human driving simulators by taking the highest-fidelity simulation as a reference and compare lower-fidelity variants empirically using participants behaviour for a data-driven evaluation. The research shows that human behaviour directly correlates with the presented fidelity level [49] but beyond a certain point improving the fidelity of models appears to be negligible [50]. Studies like Zhao and Sarasua [51] assess the minimal required visual fidelity of driving simulators based on the perceptual capabilities of humans. However, no similar studies have been done on the impact visual fidelity of simulators for virtual SBT where the human driver is substituted by a highly-automated driving function. Here, the perception algorithms might also have restrictions in its sensory means (e.g., image resolution, maximum range, etc.). Thus, the potential of exploiting different fidelities in virtual SBT is unexplored and analogies could be exploited.

6.2.1.3 Experimental setup

In the following, we outline the parameters relevant to perform Sensitivity Analysis (SA) for the simulation of UUV use case as well as the experimental setup. Note that the sensitivity analysis in this case in not only performed on the levels of fidelities but also on other parameters which are changeable by the RLA. The inputs to the SA's system under test (namely, the entire UUV UC toolchain) can be listed as action and level of fidelity combined. The goal is to analyse the sensitivity of the reward to these inputs.

The input of the environment simulation consists of the parameters listed in Table 1. They are assumed to be discrete/continuous uniformly distributed variables (the schematic representation for placing the objects in simulation environment is introduced in [52]). The following three object classes were identified for the study: Lanterns, cars and trees. These classes are in close distance of the ego vehicle during the simulation. As shown in Figure 14, the level of detail for these object classes can be changed in three levels. The number of polygons varies in the different levels of detail, but this does not necessarily lead to a visual change. However, a low number of polygons results in lower memory usage and shorter loading times, which could lead to a faster simulation run. In addition to the levels of detail already mentioned, there are also different models for some object classes. Six car models from different manufacturers are available. There is one model for lanterns that was modelled after a Berlin lantern. Three different types of trees are also available as models for the study.

Table 1 List of parameters used for the sensitivity analysis

Type of parameter	Parameter	Range
Action parameter (control input)	Visibility of the road markings	0 to 1
	Number of the road patches (e.g. manhole covers or tire tracks)	0 to 1
	Type of the tree	0,1,2
	Number of the trees	0 to 1
	Type of the car parked along the road	0,1,2,3,4,5
	Longitudinal positioning of parked car	0 to 1
	Lateral positioning of parked car	0 to 1
	Orientation of parked car	0 to 1
Simulation parameter	Fidelity level of the model of parked car	1,2,3
	Fidelity level of the model of trees	1,2,3

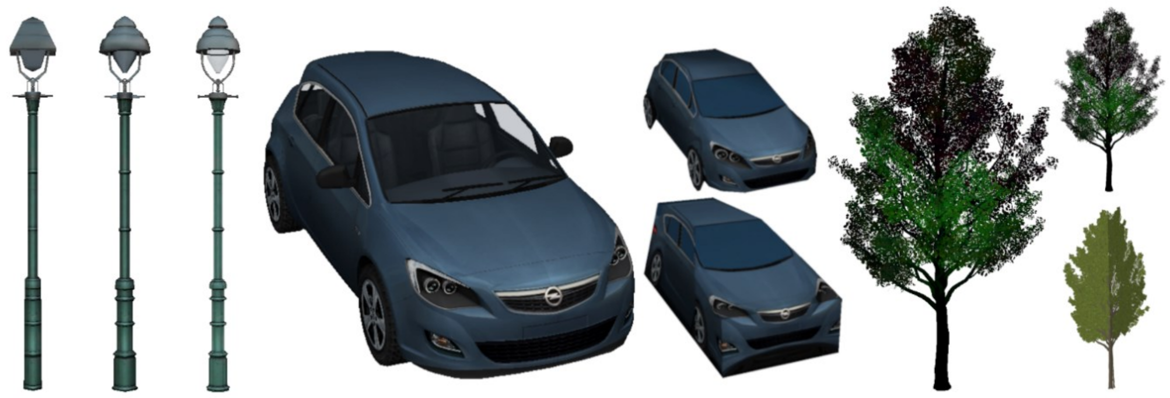


Figure 14 Different level of detail for different object classes

Figure 15 shows examples of three different 3D environments in which the objects have different levels of detail. On the left-hand side, all objects have a low level of detail, in the middle they have a medium level of detail, and on the right-hand side all objects show the highest possible level of detail. The level of detail of the object classes can be changed individually and does not have to be changed uniformly as shown in the image. Different combinations in the level of detail of the object classes are examined during the investigation.

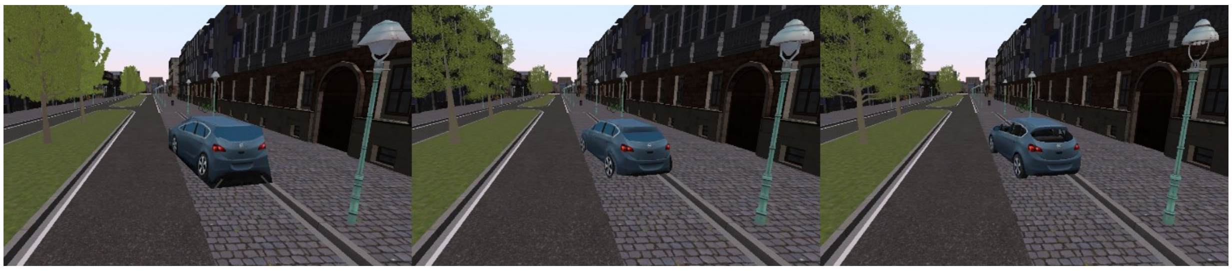


Figure 15 Different level of detail in the environment model (left low, middle medium, right high)

The level of detail is examined based on the process loop that was developed specifically for the UUV use case. The structure of this process loop is shown in Figure 16 above. To train the RL-agent, the following steps are executed each time the loop is run:

1. The RL-agent generates variation parameters that characterize a specific 3D environment and passes them to the scenario generator.
2. The scenario generator creates a detailed 3D environment based on the variation parameters.
3. In the Sensor/Environment Simulations block, the 3D environment is loaded and the ego vehicle with its sensors is created. The simulation is then executed.
4. The result of the simulation is then analysed in Feature Engineering and the values for the criticality metrics are determined.
5. Once the analysis is complete, the RL agent receives the parameters for state and reward, after which the cycle starts again. This process is repeated until the RL agent is fully trained.

In order to analyze the influence of the parameters on the criticality metrics, a modified version of the process loop of the UUV use case is used, which is shown in Figure 16. The RL-agent is removed from the loop and replaced by code that passes variation parameters and fidelity parameters to the scenario generator.

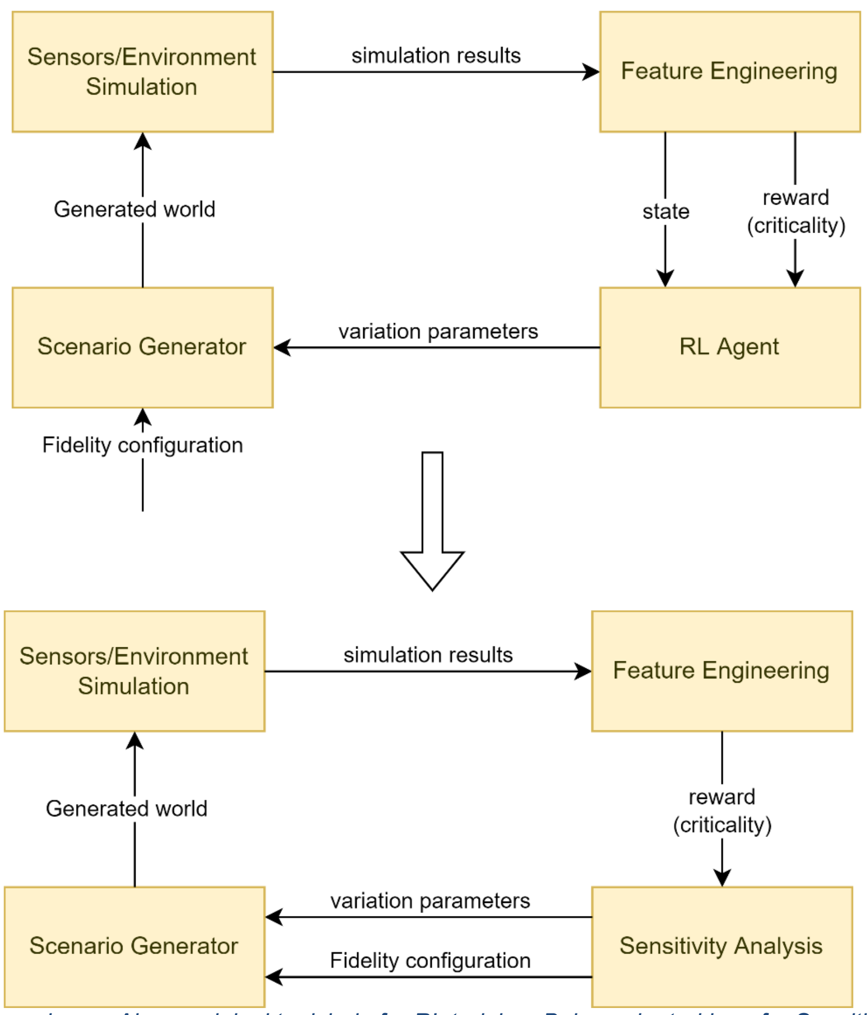


Figure 16 Process loops. Above original toolchain for RL training. Below adapted loop for Sensitivity analysis

A subset of the variation parameters is no longer changed at each iteration but remains constant across all iterations of the investigation. This parameter set is selected so that it has a particularly high influence on the criticality metric to be able to recognize changes more easily.
In addition to the variation parameters, a new set of parameters is used that are not present in the original process loop, namely the fidelity levels. This parameter set changes the level of detail of objects in the 3D environment and is referred to as the fidelity parameter in the image. The level of detail can be set individually for each object class.

The following steps are performed for each iteration:

1. The iteration is started by passing the variation parameters and the fidelity parameters to the scenario generator.
2. The scenario generator creates a 3D environment based on the two parameter sets.
3. In the Sensor/Environment Simulations block, the 3D environment is loaded and the ego vehicle with its sensors is created. The simulation is then executed.
4. The result of the simulation is then analyzed in the Feature Engineering block and the influence on the criticality metric is logged.

6.2.1.4 Experimental execution and results

SA aims to address questions such as the followings [53].

- What inputs cause the largest variation in the output?
- Is there any parameter whose variability has a negligible effect on the output?
- Are there interactions that increase or decrease the variability induced by individual parameters?

In order to select an appropriate SA method to answer these questions, we first explore the purposes of SA for the DEV test bed. The general purposes are distinguished in three groups according to [12].

- *Ranking* aims at generating the ranking of the inputs according to their relative contribution to the output variability.
- *Screening* aims at identifying the inputs that have a small influence on the output variability.
- *Mapping* aims at determining the region of the input variability space that produces significant, e.g. extreme, output values.

To explore influence of above-mentioned parameters, we first focus on the first purpose, i.e. ranking. The methods like One-At-a-Time (OAT) aim at analysing the output variation induced by each input, assuming that the inputs are independent. Instead, we want to explore the interactions between inputs too. Therefore, we should design the SA experiments to extract all plausible impacts of inputs not only on the output but also on the other inputs as well.
In a full factorial sampling, each parameter is treated as discrete, and we consider two or more intervals of its values. If the number of intervals is the same across all the parameters, the number of generated samples for iterating the simulation is estimated by (number of intervals) ^ (number of parameters).
On the other hand, if we can assume that higher-order interactions are negligible, the most significant effects in lower-order interactions can be analysed using a fraction of the full factorial design. Fractional factorial sampling can significantly reduce the number of simulations.

Variance-based method like Sobol sensitivity analysis [12] can identify relevant parameters and helps us select the input variations for the simulation. Typically, Sobol uses a sampling method such as Monte Carlo to generate a set of sample points within the input parameter space. These sample points are often generated using a quasi-random sequence. Within a probabilistic framework, Sobol decomposes the total variance of the model output into contributions from individual input parameters or combinations of parameters. This decomposition provides information about the relative importance of input parameters. Sobol indices (numerical measures) are calculated to quantify the contribution of the individual input parameters or parameter combinations to the total variance of the output. There are different types of indices for the further analyses. For instance, the first-order indices measure the contribution of individual parameters and total-order indices measure the contribution of individual parameters plus their interactions with other parameters. The calculated value of indices is used to identify the most influential input parameters. Higher value indices indicate parameters with significant influence on the output variability, while low indices indicate parameters with less influence.

For 10 dimensions (see Table 1) the Sobol sensitivity analysis was performed including 50 iterations per combination resulting in 1100 simulation runs. The results can be seen in Figure 17.

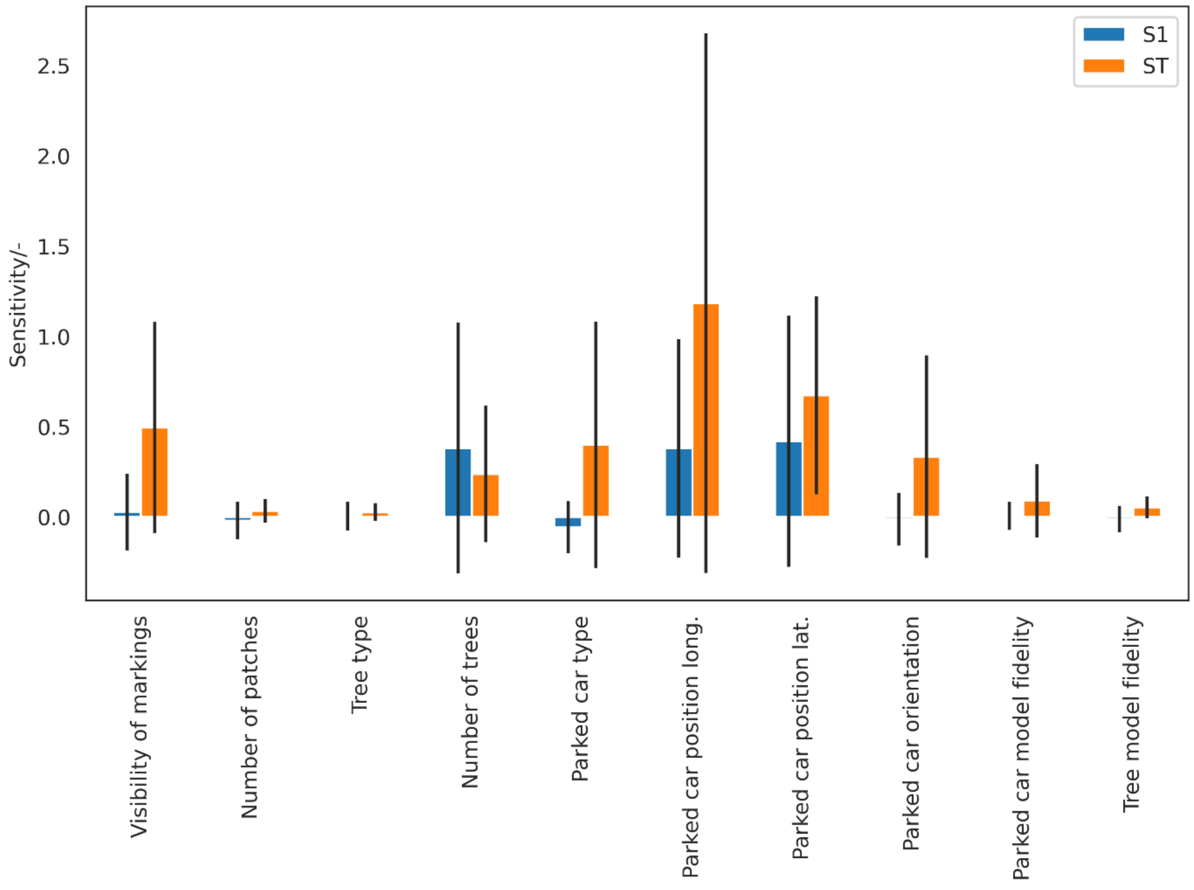


Figure 17 Results of the variance-based sensitivity analysis for 10 selected parameters showcasing first-order indices (S1) and total-effect indices (ST).

6.2.1.5 Discussion

When analysing Figure 17, one needs to acknowledge the broad 95%-confidence intervals on all parameters. This is mainly due to still too low number of samples but could also result of some emergent behaviour. However, the results can still serve as an indicator and trends can be inferred.

Regarding the fidelity, we see that the fidelity of the parked car and tree models on their own have little to no impact on the simulation and only marginally contribute to a change in the simulation's outcome in conjunction with other parameter changes. Thus, any fidelity type could be selected.

Similarly, number of patches on the street as well as the tree model type do not influence the scenario, thus they may be fixed or omitted entirely. The visibility of the road markings and the parked car orientation and the parked car model type on their own may have no influence on the outcome but may be relevant in conjunction to other parameter variation. Interestingly, the number of trees have a relatively high influence on the reward variance but does not influence the variance of other parameters that much.

The most influential object in the scenario is the parked car, more specifically its position. This is intuitive as the car can be placed to overlap the markings right before the intersection, making it harder for the ego vehicle to make it across. Interestingly, the longitudinal positioning has a higher impact on the results than the lateral.

On a meta-level, the experiment showed that due to the necessity of N*(2D+1) runs (in this case D=10 dimensions and N=50 iterations per combination), this approach is highly computationally expensive and scales linearly with both the parameters to be investigated (D) and the complexity of the system (required N). As already stated, N=50 iterations were in this particular situation not enough to narrow the confidence intervals to a reasonable amount. As in total the model has 30 possible parameters, an expert-based reduction prior to the parameter selection for the sensitivity analysis is unavoidable.

6.2.1.6 Conclusion

In conclusion, we can say that a sensitivity analysis as performed here is only adequate for a small subset of all available parameters which was preselected by experts. It was shown that the model fidelity of the parked car and the trees in this scenario have very little impact on the outcome of the scenario and may be set to their lowest value to improve execution efficiency. The most influential parameters are the ones describing the parked vehicle. Other parameters such as the road markings do have some influence on the outcome by increasing or decreasing the variability induced by other parameters and should not be left out entirely without careful consideration. However, the number of patches and tree type as well as the model fidelities can be fixed which reduces the space of relevant parameters significantly.

6.2.2 DT Development of a sensor in light of the process

For the prototypical generation of a DT and its integration into the developed tool chain for training the optimization-RLA, a small-scale vehicle-in-a-loop setup was developed. We acknowledge that the DEV-bed's sensor stimulation and vehicle dynamics do not accurately replicate the real RC-car on a street or any real environment. However, we consider it as the “ground-truth”. While some parts of the real system are also virtual such as the virtual environment and the software like the driving function on the vehicle’s ECU (see Figure 18), other such as the camera system and the actuators contain physical interactions and phenomena between components. As shown in Figure 18, for this prototype we concentrate on the camera system and its interaction with a screen and intend to replicate the changes to the virtual camera footage coming from the virtual environment. In the following the development process of finding the relevant parameters and design choices are shown following the process from Figure 8.

Planning
Following the path in Figure 8, there was no model available and therefore a new one needed to be made from scratch. As total access to the setup was given, there was no restriction on data and the system is not numerically inexpressible. Thus, now there was the question of whether to use a fully data-driven or a hybrid approach. It was not reasonable to explain the phenomena in detail on a first-principle basis using measuring data like the distance and angle between the camera and the monitor as these measurements cannot be automated in an easy way. Thus, we intended to measure and mimic the specific phenomena which can be identified. This can be seen as a model-driven approach while we input knowledge from the underlying phenomena to make design considerations of the algorithms.

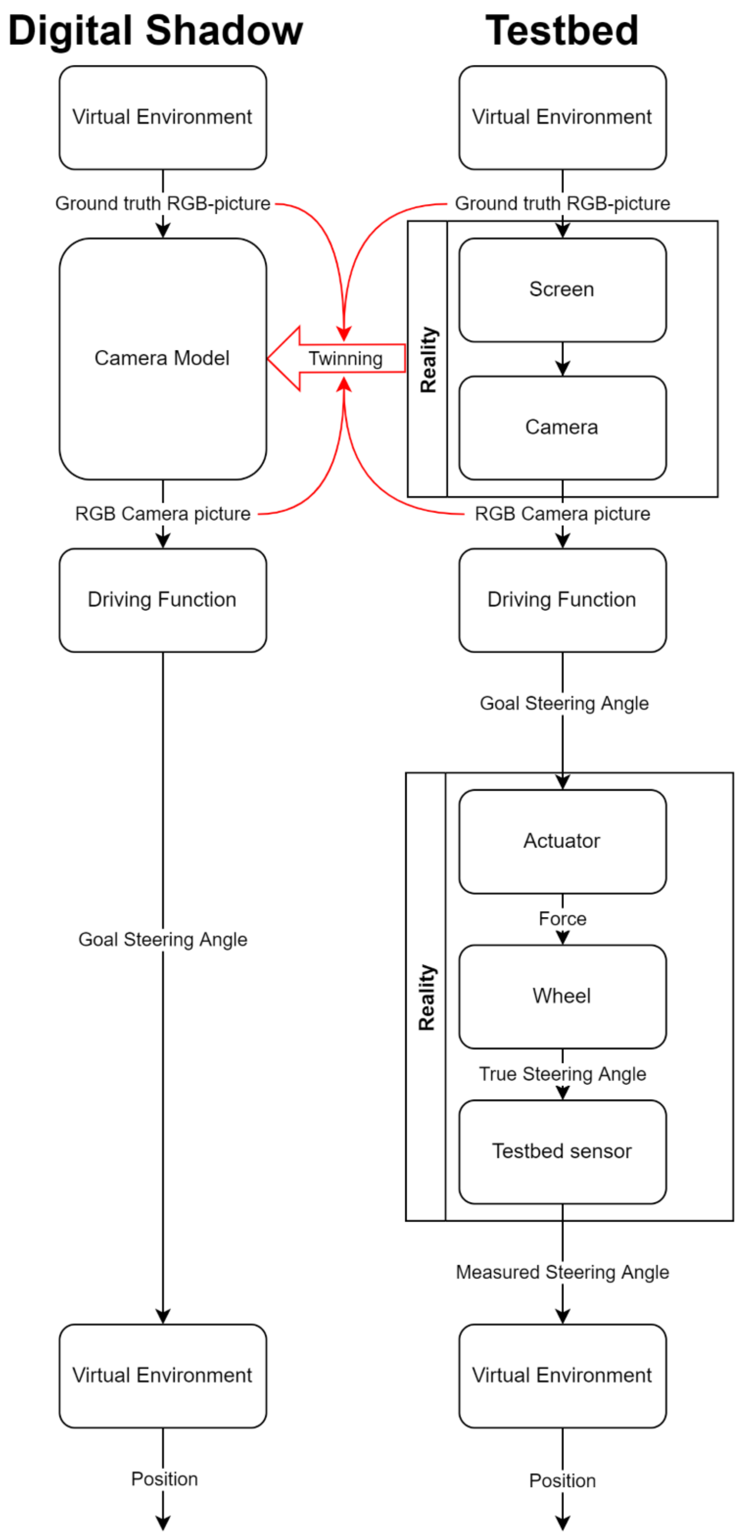


Figure 18 Information flow through the DEV-bed highlighting the two instances where components are not virtual and which part shall be twinned.

Modelling

To identify static phenomena, a test picture (see Figure 19) is displayed on the monitor and a frame of the camera feed is taken for comparison (see Figure 20). In the following the identified phenomena and their quantization is explained.

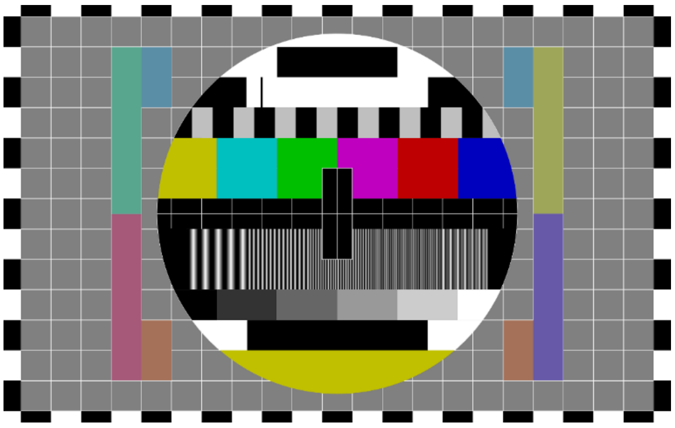


Figure 19 Original test picture

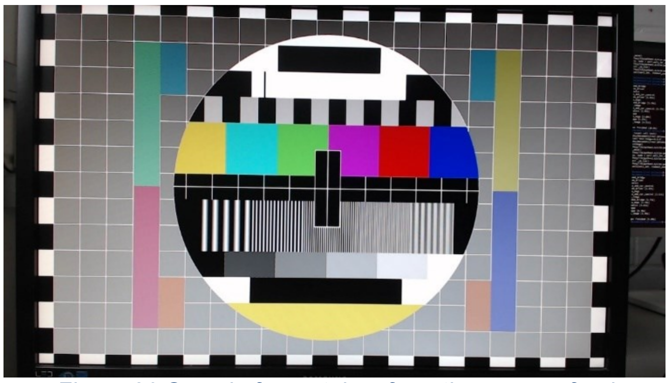


Figure 20 Sample frame taken from the camera feed

Perspective correction is needed which is caused by imperfect camera-to-monitor alignment, including shifts, scaling, and warping effects. To correct these, the method involves identifying the four corners of the monitor by finding the closest pixel to each corner in the test frame. A perspective warp function, specifically warpPerspective() in OpenCV, is then applied to the test picture to replicate the observed perspective phenomena. However, the method has limitations, such as sensitivity to accurately locating the right pixels representing the corners and the assumption that there is no barrel or pincushion distortion, which hasn't been observed in the test frame.

Brightness shifts are observed in the test picture (Figure 20) due to monitor polarization, resulting in an increasing brightness shift from top to bottom and a less pronounced shift from left to right. To align the test picture's brightness with that of a reference frame, the image is vertically sliced along the grid of grey boxes. Within each slice, the brightness of the topmost and undermost boxes is measured. Linear interpolation is then applied to adjust the brightness for each pixel row in the test picture, based on the known true brightness from the reference frame. This method aims to correct the polarization-induced brightness distortions in the test picture.

Blur can be seen being induced by imperfections in both the monitor and the camera. While direct measurement of blur is not feasible, using a method based on the variance of the picture's Laplacian is possible, which is inversely related to blur. A higher Laplacian value indicates sharper edges and lower blur. To establish a reliable metric, a lookup table is created by iteratively blurring the original test picture with increasing levels of Gaussian blur (represented by sigma). The corresponding sigma-sharpness data points are recorded. To apply this to the test picture, a reverse process involves "reverse-warping" the test frame (adjusting for perspective changes), measuring sharpness, and using the lookup function to calculate and apply the appropriate sigma to mitigate blur in the test picture. This method aims to compensate for blur induced by monitor and camera imperfections.

Delay affects the driving function of a virtual vehicle in the ViL-setup. Motion blur is not observed during camera footage recordings, but a noticeable delay influences the driving function's performance. The delay results from various components, including the monitor, webcam, driving function, and network communication. The total delay is a sum of these individual delays, but their specific contributions are unknown due to unsynchronized clocks across the RC car and simulation workstation. To model the delay, the entire process—from a change in the environment's state to an altered control command—is considered. The delay is measured by positioning the RC car without forward movement, instantly rotating the virtual vehicle by 5°, and measuring the time between this event and a step change in the steering command. This measurement is repeated five times and averaged to accommodate fluctuations. The goal is to model the delay as a queue, given the virtual RC car's processing frequency remains at 30 fps.

Additional potential parameters that were identified but intentionally left out was the color saturation. While this is visibly a phenomenon (compare Figure 19 and Figure 20), it was not considered as the driving function is insensible to it. During its image processing, the incoming footage is converted to grey-scale and lines for the lane-keep assistant are identified by converting the image further to black and white with a fixed threshold. While brightness influences the identification of lanes significantly, the color saturation does not.

Validation I

During the first validation cycle, the first question of validation in figure 8: “Is the model accurate enough for the intended use?” was answered using face validation comparing the output from the sensor model (see Figure 21) given the test image (see Figure 19) the image from the camera (see Figure 20). On first sight, the image seemed to subjectively similar.

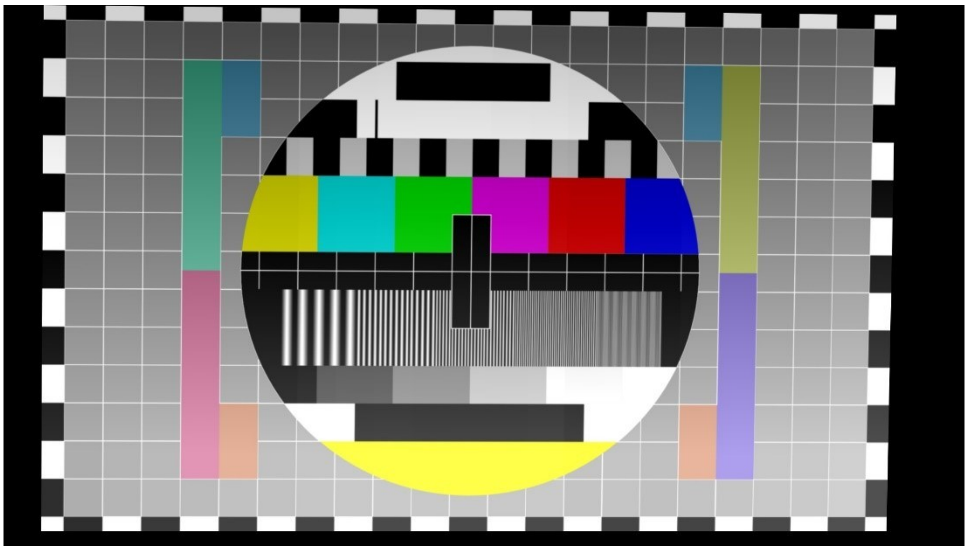


Figure 21 Adjusted test picture

Next, the sensor was integrated in a loop which can be seen as the DT of the test setup as it had the same environment simulation and driving function as the vehicle. Again using face validation, it was easy to see that the virtual vehicle did not follow the lane as the real vehicle. Next, the execution efficiency of the sensor model was measured which was on average 300 – 320ms on the simulation workstation used. This means that the ~3 frames can be processed per second which made the virtual vehicle too slow in reacting to its environment. Thus, it did not pass the “Does the model have enough execution efficiency for the intended use?” question (see Figure 8).

Optimization I
As the face validation resulted positive, first code optimization was performed. By eliminating for-loops and pre-computation of functions, the execution efficiency tripled to 106ms or ~10fps.

Validation II
Again, the sensor model/DT of the camera is integrated in the virtual version of the test setup. Results pending.

6.2.3 LiDAR Validation as an example for parameter identification

To show the usability of the parameter identification process described in Chapter 5, a simple practical example is given where the steps of Figure 8 are followed for a LiDAR validation experiment. The goal is to set up a digital twin of a LiDAR and environment that provides good enough sensor data for an object perception algorithm to identify relevant objects, e.g. for an autonomous driving function of a vehicle.
The general setup is depicted in Figure 22. A LiDAR roadside unit (RSU) scans the environment, an autonomous shuttle propagates along a precisely predefined path. The perception algorithm is fed with the raw data from the LiDAR, detects the shuttle and includes it in an object list with relevant parameters. (The used perception algorithm can be adapted with slight changes for the application in vehicles.) One parameter is the position, which is used for validation of the traced position in the physical system against ground truth and of the traced position in the digital system against ground truth. For the latter the digital twin is created. A more detailed description of the experiment is given in [52] of Asimov.

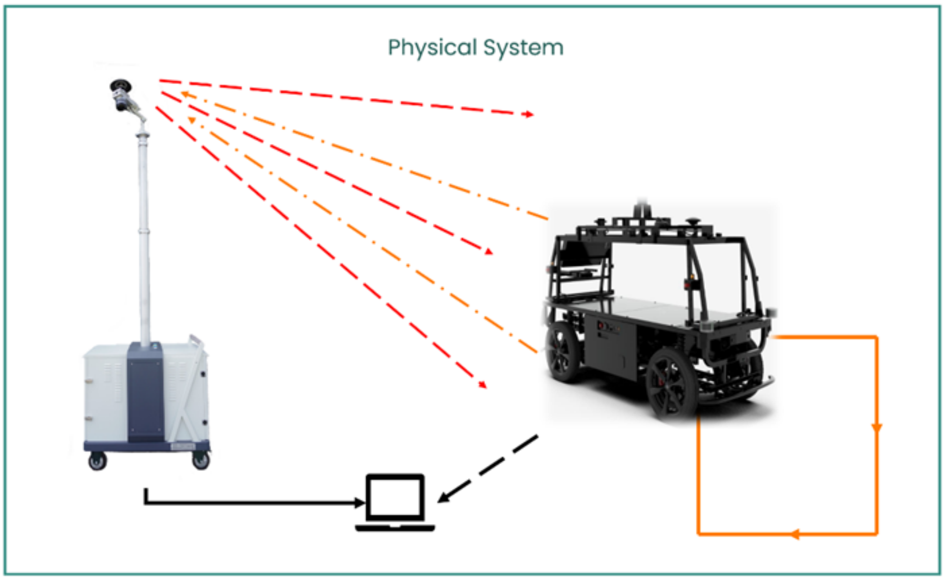


Figure 22 Setup of the Physical System of the LiDAR validation experiment.

To create a sufficient model for the digital twin we follow the model described in Chapter 5, see Figure 23. First, we evaluate if there is already a platform for a model available. The answer is yes. The Sofware Carla, which is used throughout Asimov for the UUV use case, can create typical environments and comes with the ability to model LiDAR-sensors and create LiDAR-sensor-data. Also, Carla is open source and therefore freely available. A major advantage when the resources for a project are limited.
However, is the possible model accuracy enough for the intended use or do the platform limitations not allow for a sufficient model? This of course highly depends on the use case. Here we want to generate good enough data for a perception algorithm connected to a driving function to identify relevant traffic objects with comparable reliability as in the physical system for most traffic scenarios.
To gather LiDAR-sensor-data Carla uses a raycasting-routine to determine which environmental objects are in the field of view of the sensor and reflect the laser back to it. In contrast to raytracing, multiple reflections cannot be considered. A possible major limitation, especially in environments with many reflective surfaces, such as environments with heavy rain or fog.

Also, the exact modeling of perturbations will need many resources. Especially vibrations can disturb LiDAR (if not solid state). For a precise modeling the roughness of the road, the suspension of the vehicle, the stiffness of its frame and many more parameters need to be known and modeled.
In other words, a simulation in Carla will not match reality in every aspect and getting close will already consume many resources.
Is another platform and model needed? Or will a more detailed examination of relevant parameters save the model?

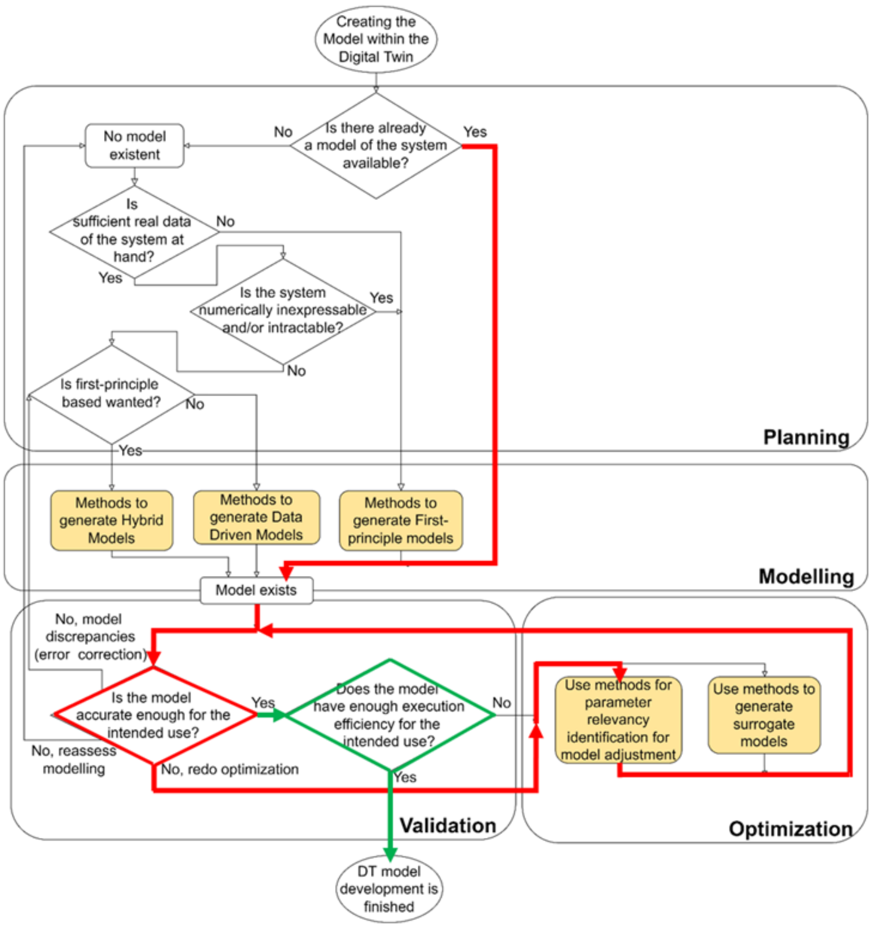
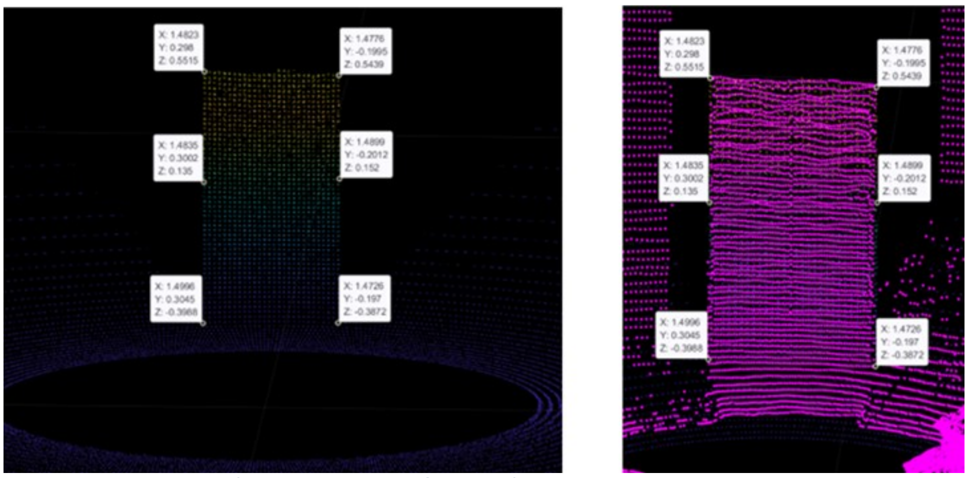

Creating the Model within the Digital Twin
Is there already a model of the system available?
No
Yes
No model existent
Is sufficient real data of the system at hand?
No
Yes
Is the system numerically inexpressable and/or intractable?
Yes
No
Is first-principle based wanted?
No
Yes
Planning
Methods to generate Hybrid Models
Methods to generate Data Driven Models
Methods to generate First-principle models
Modelling
Model exists
No, model discrepancies (error correction)
Is the model accurate enough for the intended use?
Yes
No, reassess modelling
No, redo optimization
Does the model have enough execution efficiency for the intended use?
No
Yes
Validation
Use methods for parameter relevancy identification for model adjustment
Use methods to generate surrogate models
Optimization
DT model development is finished

Figure 23 Scheme for creating a sufficient digital twin model for LiDAR validation.

In Figure 23 we have followed the red path to “Is the model accurate enough for the intended use” in the “Validation” box and keep following the path to “Use methods for parameter relevancy identification for model adjustment” in the “Optimization” box.
We first evaluate the noise issue. Figure 24 shows a LiDAR-lab-measurement of a target plate in the physical system (right) and in Carla (left). For the measurement within the digital system, 0.05 per cent of noise was added to the result. However, the points in the point-grid representing the plate are still relatively equidistant to each other and barely change in intensity. The points representing the plate in the physical system show a relatively strong perturbation along the scan line of the LiDAR, especially in the upper part of the plate.


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 24 LiDAR Measurement of a target plate in Carla (left) and overlayed with the measurement in the physical system (right).

The representation in the digital system is therefore not very good at point level. However, the plate is represented by a large number of points and is defined by the shape of the point-group. On this level the discrepancies are relatively small.

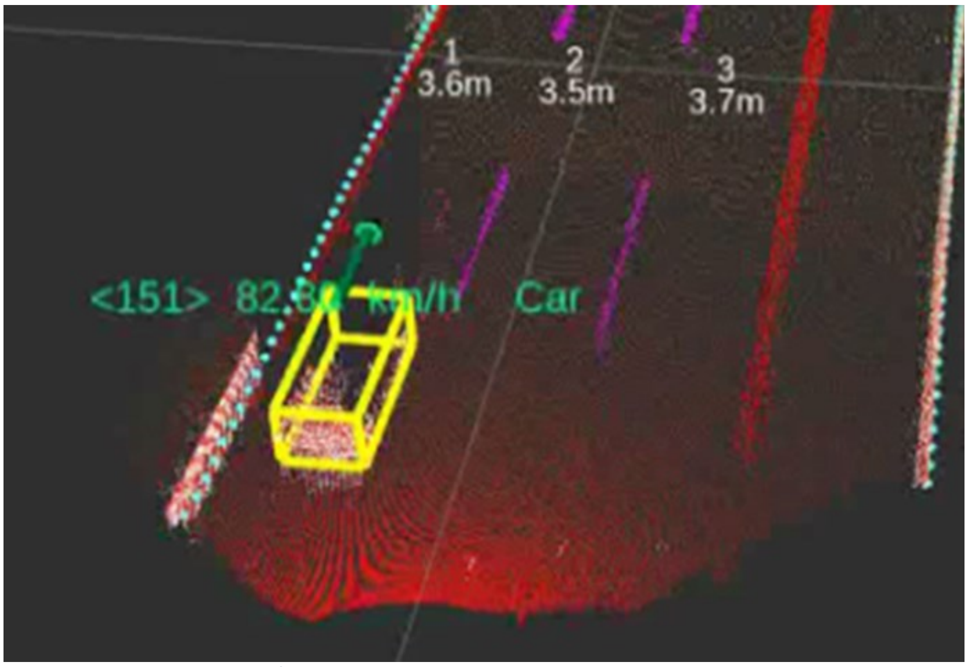


Figure 25 LiDAR 3D Point cloud of an Autobahn scene. The scene was analyzed by a perception software. Identified objects have been marked with bounding boxes, which have been annotated with additional information (class, e.g. car, and speed).

Additionally, the number of perturbations in the lab environment is small. The noise level of data taken in real world environments is quite different. Due to a multitude of influences (road, roughness, suspension, speed…) the pattern appears rather random than distinctive, in point distance as well as in intensity. See Figure 25 for an example. The pattern also changes from frame to frame adding to the random appearance.
In conclusion, the perception of objects does not depend on a perfect representation on point level but depends on the shapes on object level. Small discrepancies on point level will not influence the performance. Especially since the object perception algorithm is confronted and can handle random noise patterns, which appear in practice. Such patterns can be easily added in Carla. (Additionally, the environment in Carla does not need to be modelled to the highest fidelity)

The high and random noise level seen in Figure 25 also limits the influence of multiple reflections, which usually contribute order(s) of magnitude less to the intensity of a LiDAR point than direct reflections.

Therefore, except for corner cases (strong fog, rain, distant objects represented only by a low number of points, -> scenarios where also the physical system LiDAR and the perception algorithm itself will run into limitations) a simplified model as Carla provides it with random noise and raycasting to determine the sensor input will suffice.
In the scheme in Figure 23 we therefore follow the red path back to “Is the model accurate enough for the intended use” in the “Validation” box, answer with “yes” and follow the green path to “Does the model have enough execution efficiency for the intended use?”. The answer here is also “yes”. Due to the limited use of resources in the simplified model, the digital system can run in real time on common hardware, just as the counterpart in the physical system. For the given purpose we have found a sufficient model and the “DT model development is finished”.

7 Conclusion

This document presents the final status of the development of the identification of relevant parameters for the creation of a DT. First in Chapter 2 “Definitions”, explanations were given for key concepts used in the context of DT development and an understanding on what a parameter is, was established.

The state of the art was outlined in two fields. The first part includes the state of the art regarding the identification of parameters, both qualitatively (identifying which parameters should be considered) and quantitatively (identifying which values should be assigned to the parameters). The first type of identification is considered as the definition of system boundaries: methods for constructing an initial set of parameters based on system characteristics. The second type is covered by parameter identification methods, which are a topic in the field of system identification. In describing another type of methods in the second part of the literature review, we emphasized the “relevancy” of the identified parameters to conduct the accuracy-efficiency trade-off by the DT. Existing approaches to adjust or create accurate. yet robust models were reviewed; sensitivity analysis, model reduction and surrogate modeling to name a few.

Next, a baseline was created by describing the current process in the UUV and STEM use cases. The parameter identification process in the UUV use case is mainly based on expert knowledge and published standards such as ASAM OpenX and Euro NCAP. It was the goal to be able to identify parameters that define aspects of critical scenarios which are not covered by standards yet. The section regarding the STEM use case describes the current candidate parameters for the DT of an electron microscope (EM), which is mainly based on the domain expert knowledge. According to the selected application, being the aberration correction in an EM, a digital model is used. The parameters of this digital model are differentiated to categories of control, system and disturbance parameters.

Following, the development process in the light of parameter identification was shown. It provides the motivation for considering finding relevant parameters during the entire development process of the DT and differentiates between two fields: parameter identification and optimization. The section also describes the steps during the DT-development where finding relevant parameters is performed.

Finally, specific topics for the use cases are presented. The first part addresses the EM use case and focuses on the fine-tuning of parameters to increase precision and on problems of parameter identifiability. The second part focuses on research related to the UUV use case. First, the influence of the level of detail of objects in the 3D environment on the criticality metrics was analyzed. In this example, the sensitivity analysis was successfully applied and it was shown that it is a suitable method to identify relevant parameters when the parameter space is already small. Furthermore, the parameter identification process, which was described in detail in Chapter 5, was successfully applied and evaluated on two real-life examples. The first example focuses on the development of a sensor for a DT, while the second example is about the validation of a LiDAR.

This document serves as a summary or guideline for the application of different techniques which improve how well the DT is suitable for the task of training an AI that can optimize a CPS. This will help practitioners to build more efficient and more detailed DTs. It can easily be said that the goal of the task is essential for turning the ASIMOV idea into reality.

8 Terms, Abbreviations and Definitions

Table 2 - Terms, Abbreviations and Definitions

ABBREVIATION	**EXPLANATION**
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASAM	Association for Standardization of Automation and Measuring Systems
ASIMOV	AI training using Simulated Instruments for Machine Optimization and Verification
CPS	Cyber-Physical System
DoD	United States Department of Defense
DT	Digital Twin
EM	Electron Microscopy
FEA	Finite element analysis
LiDAR	Light Detecting and Ranging
MFM	Multi-Fidelity Model
MFHM	Multi-Fidelity Hierarchical Models
MFSM	Multi-Fidelity Surrogate Model
MIMO	Multi-input-multi-output
MR	Model Reduction
M&S	Models and Simulations
NCAP	New Car Assessment Program(me)
ODE	Ordinary differential equations
PMSM	Permanent Magnet Synchronous Motors
PSO	Particle Swarm Optimization
RL	Reinforcement Learning
RLA	Reinforcement Learning Agent
SA	Sensitivity Analysis
SBT	Scenario-based Testing
SISO	Single-input-single-output
STEM	Scanning Transmission Electron Microscopy
SuT	System under Test
TEM	Transmission Electron Microscopy
UUV	Unmanned Utility Vehicle
V&V	Verification and validation
WP2	Work Package 2

9 Bibliography

[1] D. J. Wagg, K. Worden, R. J. Barthorpe and P. Gardner, “Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications.,” *In ASCE - ASME Journal of Risk and Uncertainty in Engineering Systems 6 (3),* 2020.

[2] ASIMOV-consortium, “Architecture of optimized digital twins for AI-based training - ASIMOV Deliverable D2.3,” 2023.

[3] “SEBok,” INCOSE, [Online]. Available: https://www.sebokwiki.org/wiki/Why_Model%3F. [Accessed 9 september 2022].

[4] G. F. Franklin, J. D. Powell and A. Emami-Naeini, Feedback Control of Dynamic Systems, Prentice Hall, 2002.

[5] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (2nd Edition), MIT press, 2018.

[6] M. Schoukens, *Control Systems Section: Machine Learning for Systems and Control course,* Eindhoven University of Technology, Faculty of Electrical Engineering, Spring 2022.

[7] L. Ljung, System Identification: Theory for the User, Prentice Hall PTR, 1999.

[8] A. V. Oppenheim and A. S. Willsky, Signals and Systems, Pearson, 2014.

[9] L. Ljung, „Black-box models from input-output measurements,“ 2001.

[10] H. Madsen, “Grey-Box Modeling; An approach to combined physical and statistical model building,” 2015.

[11] P. Benner, S. Gugercin and K. Willcox, “A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems.,” in *SIAM Review. 57*, 2015.

[12] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana and S. Tarantola, Global Sensitivity Analysis., The Primer, John Wiley and Sons, 2008.

[13] I. M. Sobol, “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates.,” in *In Mathematics and Computers in Simulation*, 2001, p. 271–280.

[14] D. M. Hamby, “A review of techniques for parameter sensitivity analysis of environmental models.,” *Environmental monitoring and assessment,* pp. 135-154, 1994.

[15] M. Jaxa-Rozen and J. Kwakkel, “Tree-based ensemble methods for sensitivity analysis of environmental models: A performance comparison with Sobol and Morris techniques.,” in *Environmental Modelling & Software 107*, 2018, p. 245–266.

[16] N. V. Sahinidis, “Optimization under uncertainty: state-of-the-art and opportunities,” in *Computers & Chemical Engineering Volume 28*, 2004.

[17] D. Müller, E. Esche, L. D. and G. Wozny, “An algorithm for the identification and estimation of relevant parameters for optimization under uncertainty.,” in *Computers & Chemical Engineering. 71*, 2014, p. 94–103.

[18] W. Schilders, Introduction to Model Order Reduction., 2008.

[19] A. Chakraborti, A. Heininen, K. T. Koskinen and V. Lämsä, “Digital Twin: Multi-dimensional Model Reduction Method for Performance Optimization of the Virtual Entity.,” in *Procedia CIRP 93*, 2020.

[20] D. Hartmann, M. Herz and U. Wever, “Model Order Reduction a Key Technology for Digital Twins,” *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing,* pp. 167-179, 2018.

[21] D. Hartmann, M. Herz, M. Paffrath, J. Rommes, T. Tamarozzi, H. Van der Auweraer and U. Wever, “Model order reduction and digital twins.,” in *Model order reduction. Volume 3: Applications*, DE GRUYTER, 2020, p. 379–430.

[22] C. Bucila, R. Caruana and A. Niculescu-Mizil, “Model compression.,” in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[23] M. G. Fernández-Godino, C. Park, N.-H. Kim and R. T. Haftka, “Review of multi-fidelity models.,” *In AIAA Journal 57 (5),* p. 2039–2054, 2019.

[24] Á. Bárkányi, T. Chován, S. Németh and J. Abonyi, “Modelling for Digital Twins—Potential Role of Surrogate Models.,” in *Processes 9 (3)*, 2021.

[25] C. Park, R. T. Haftka and N. H. Kim, “Remarks on multi-fidelity surrogates.,” in *Struct Multidisc Optim 55 (3)*, 2017.

[26] E. Acar, B. Yilmaz, M. A. Güler and M. Altin, “Multi-fidelity crashworthiness optimization of a bus bumper system under frontal impact.,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering 42 (9),* p. 1–17, 2020.

[27] R. C. Aydin, F. A. Braeu and C. J. Cyron, “General Multi-Fidelity Framework for Training Artificial Neural Networks With Computational Models.,” in *In Front. Mater. 6*, 2019.

[28] L. Liu, D. Cartes and W. Liu, “Particle Swarm Optimization Based Parameter Identification Applied to PMSM.,” in *Proceedings of the American Control Conference.*, 2007.

[29] G. M. Hornberger and R. C. Spear, “An Approach to the Preliminary Analysis of Environmental Systems,” *Journal of Environmental Management,* pp. 12, 7–18, 1981.

[30] “ASAM OpenXOntology,” [Online]. Available: https://www.asam.net/project-detail/asam-openxontology/. [Accessed 15 08 2022].

[31] “Euro NCAP,” [Online]. Available: https://www.euroncap.com/en/for-engineers/protocols/. [Accessed 31 08 2022].

[32] “ASAM OpenDRIVE,” [Online]. Available: https://www.asam.net/standards/detail/opendrive/. [Accessed 25 08 2022].

[33] D. Jones, C. Snider, A. Nassehi, J. Yon and B. Hicks, “Characterising the Digital Twin: A systematic,” *CIRP Journal of Manufacturing Science and Technology, vol. 29,* p. 36–52, 2020.

[34] ASIMOV-consortium, “Methods and Tools for Training AI with Digital Twin - ASIMOV Deliverable D2.2,” 2024.

[35] O. Balci, “Verification, Validation, and Accreditation.,” in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[36] D. Richard, B. Niklas, B. Jan-Willem, E. Modrakowski, F. Caglar, I. Ahmad, A. Mooij and S. Moritz, “ASIMOV Reference Architecture,” ASIMOV ITEA Deliverable D4.a, 2022.

[37] S. Aheleroff, X. Xu, R. Y. Zhong and Y. Lu, “Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model,” *Advanced Engineering Informatics, vol. 47,* 2021.

[38] W. Oberkampf and T. Trucano, “Verification and validation benchmarks,” *Nuclear Engineering and Design, vol 238,* pp. 716-743, 2008.

[39] R. G. Sargent, “Verification and validation of simulation models,” in *Proceedings of the 40th Conference on Winter Simulation*, Miami, Florida, 2008.

[40] I. Lazić and E. Bosch, “Analytical Review of Direct Stem Imaging Techniques for Thin Samples,” *Advances in Imaging and Electron Physics,* 2017.

[41] C. Neurohr, L. Westhofen, M. Butz, a. H. Bollmann, U. Eberle and R. Galbas, “Criticality Analysis for the Verification and Validation of Automated Vehicles.”.

[42] D. Gruyer, M. Grapinet and P. de Souza, “Modeling and validation of a new generic virtual optical sensor for ADAS prototyping.,” in *2012 IEEE Intelligent Vehicles Symposium.*, 2012.

[43] F. Liu, M. Yang and Z. Wang, “Study on Simulation Credibility Metrics.,” in *Proceedingsof the Winter Simulation Conference*, 2005.

[44] “NASA. Standard for models and simulation,” 2016.

[45] S. Riedmaier, D. Schneider, D. Watzenig, F. Diermeyer and B. Schick, “Model Validation and Scenario Selection for Virtual-Based Homologation of Automated Vehicles.,” 2021.

[46] C. Stadler, F. Montanari, W. Baron, C. Sippl and A. Djanatliev, “A Credibility Assessment Approach for Scenario-Based Virtual Testing of Automated Driving Functions.,” in *IEEE Open Journal of Intelligent Transportation Systems 3 (2022)*, 2022.

[47] C. Merenda, C. Suga, J. Gabbard and T. Misu, “Effects of Vehicle Simulation Visual Fidelity on Assessing Driver Performance and Behavior: 30th IEEE Intelligent Vehicles Symposium,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[48] R. A. Wynne, V. Beanland and P. M. Salmon, “Systematic review of driving simulator validation studies.,” in *Safety Science 117 (2019)*.

[49] C. Himmels, T. Rock, J. Venrooij and A. Riener, “Simulator Fidelity Influences the Sense of Presence in Driving Simulators.,” in *Adjunct Proceedings of the 14th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. Ed. by Yong Gu Ji and Myounghoon Jeon. New York*, New York, 2022.

[50] P. M. van Leeuwen, C. G. i. Subils, A. R. Jimenez, R. Happee and J. F. de Winter, “Effects of visual fidelity on curve negotiation, gaze behaviour and simulatordiscomfort.,” in *Ergonomics 58.8 (2015)*.

[51] X. Zhao and W. A. Sarasua, “How to Use Driving Simulators Properly: Impacts of Human Sensory and Perceptual Capabilities on Visual Fidelity.,” in *Transportation Research Part C: Emerging Technologies 93 (2018)*, 2018.

[52] ASIMOV-consortium, “UUV proof of concept demonstration - ASIMOV Deliverable D1.3,” 2024.

[53] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson and T. Wagener, “Sensitivity analysis of environmental models: A systematic review with practical workflow,” in *Environmental Modelling & Software*, 2016.

[54] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras and O. A. Oluwatola, Autonomous vehicle technology: A guide for policymakers., Santa Monica CA: Rand Corporation, 2014.

[55] A. Poddey, T. Brade, J. E. Stellet and W. Branz, “On the validation of complexsystems operating in open contexts.,” 2019.

[56] T. Brade, B. Kramer and C. Neurohr, “Paradigms in Scenario-Based Testing for Automated Driving.,” in *2021 International Symposium on Electrical, Electronics and Information Engineering.*, New York, 2021.

[57] C. Amersbach and H. Winner, “Functional decomposition-A contribution to overcomethe parameter space explosion during validation of highly automated driving.,” in *Traffic injury prevention 20.sup1 (2019)*, 2019.

[58] B. Peherstorfer, K. Willcox and M. Gunzburger, “Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization.,” 2018.

[59] S. Andrea, M. Ratto, A. Terry, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana and S. Tarantola, Global sensitivity analysis: the primer, John Wiley & Sons, 2008.

A Reference Parameter

This chapter contains input parameters (u), the output parameters (y) and the disturbance parameters (d) determined on the basis of the use cases.

To determine the parameters, the processes of the use cases were roughly sketched and the parameters were derived from them. The parameters determined are therefore use case-specific and not determined by a generic process. Additional information on how the parameters were determined can be found in chapter 3.

The parameters determined here are then to be compared in a later step with the parameters determined by the generic parameter identification process. The quality of the generic parameter identification process is to be determined by the comparison.

A.1 Use case - Electron Microscopy

A.1.1 General Use Case Information

Name	Electron microscope
Description	The goal is to automatically tune the electron microscope by reducing its aberrations
Company	Thermo Fisher Scientific

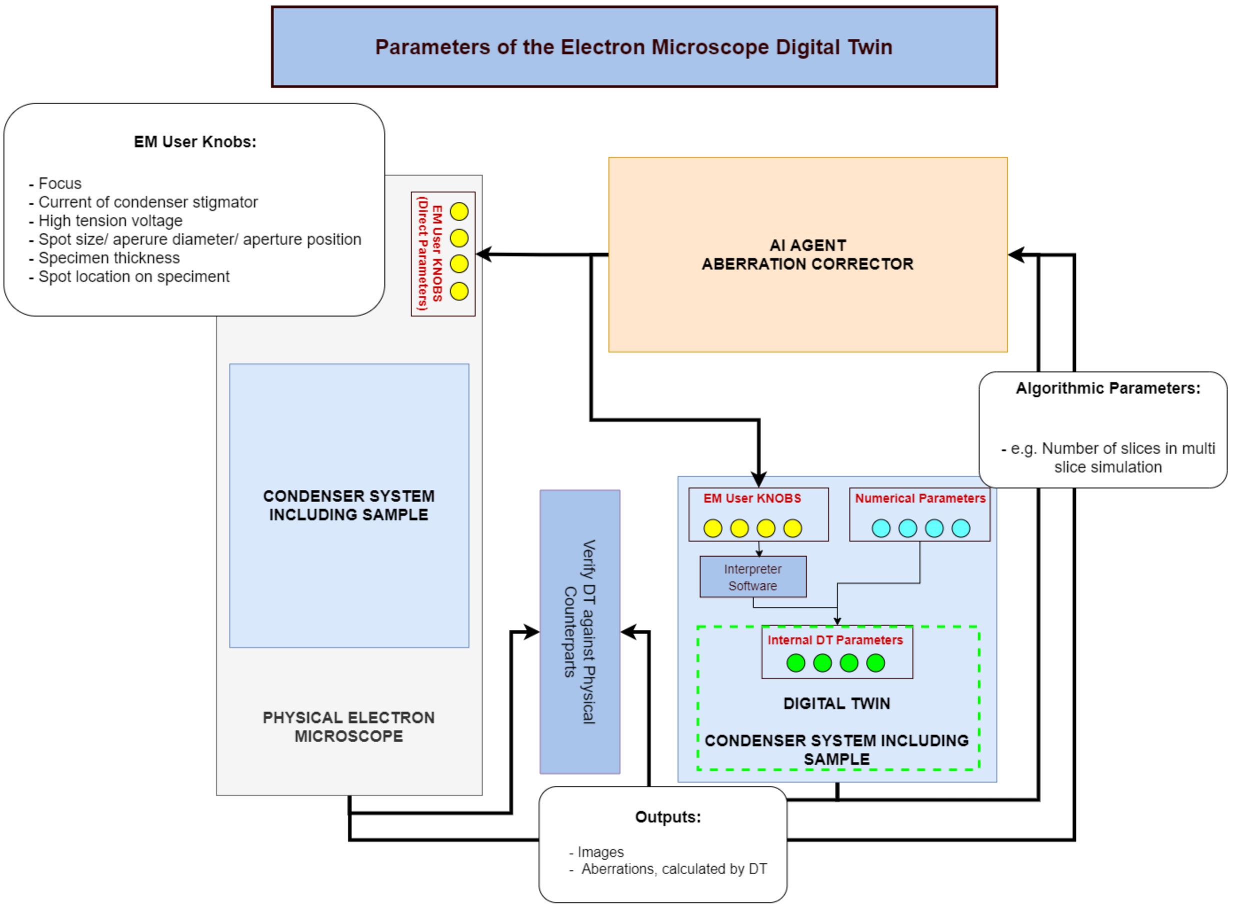

Parameters of the Electron Microscope Digital Twin
EM User Knobs:
- Focus
- Current of condenser stigmator
- High tension voltage
- Spot size/ aperure diameter/ aperture position
- Specimen thickness
- Spot location on speciment
EM User KNOBS (Direct Parameters)
AI AGENT ABERRATION CORRECTOR
CONDENSER SYSTEM INCLUDING SAMPLE
PHYSICAL ELECTRON MICROSCOPE
Verify DT against Physical Counterparts
EM User KNOBS
Numerical Parameters
Interpreter Software
Internal DT Parameters
DIGITAL TWIN
CONDENSER SYSTEM INCLUDING SAMPLE
Algorithmic Parameters:
- e.g. Number of slices in multi slice simulation
Outputs:
- Images
- Aberrations, calculated by DT

Figure 26 Electron Microscopy parameter overview

A.1.2 Parameters available to EM user

Table 3 Details of EM_U_US_001 parameter

PROPERTY	DESCRIPTION
ID	EM_U_US_001
Name	focus (wobble) knob
Description	This control knob is used to change focus
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	Varying

Table 4 Details of EM_U_US_002 parameter

PROPERTY	DESCRIPTION
ID	EM_U_US_002
Name	Condensor stigmator current knob
Description	This control knob is used to change the currents in condensor stigmator lenses
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	Varying

A.1.3 Internal DT parameters

Table 5 Details of EM_C_IN_001 parameter

PROPERTY	DESCRIPTION
ID	EM_C_IN_001
Name	source energy, high tension
Description	Energy of an electron beam entering the condenser system
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	None
Value Range	Varying
Preparation	An interpreter software will determine the value based on EM user Inputs

Table 6 Details of EM_C_IN_002 parameter

PROPERTY	DESCRIPTION
ID	EM_C_IN_002
Name	spot size
Description	Spot size of an electron beam entering the condenser system
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	None
Value Range	Varying
Preparation	An interpreter software will determine the value based on EM user inputs

Table 7 Details of EM_C_IN_003 parameter

PROPERTY	DESCRIPTION
ID	EM_C_IN_003
Name	aperture diameter / aperture position
Description	The diameter and position of apertures (3 apertures for 3 condenser lenses and 1 aperture for the objective lens) in the optics
Value Type	Discrete / Continuous
IO Type	Input
Type	System parameter: c
Unit	None
Value Range	Varying
Preparation	An interpreter software will determine the value based on EM user inputs

Table 8 Details of EM_U_IN_004 parameter

PROPERTY	DESCRIPTION
ID	EM_U_IN_004
Name	individual lens currents (not in use case 1)
Description	The current of individual lenses, 3 condenser lenses and 1 objective lens
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	Varying
Preparation	An interpreter software will determine the value based on EM user inputs

Table 9 Details of EM_D_IN_005 parameter

PROPERTY	DESCRIPTION
ID	EM_D_IN_005
Name	sample thickness / sample position / sample randomness
Description	The thickness and the position of the amorphous sample used
Value Type	Discrete / Continuous
IO Type	Input
Type	Disturbance parameter: d
Unit	None
Value Range	Varying
Preparation	An interpreter software will determine the value based on EM user inputs

Table 10 Details of EM_D_IN_006 parameter

PROPERTY	DESCRIPTION
ID	EM_D_IN_006
Name	Aberrations
Description	The parameters describing optical aberrations in the system. Including defocus, astigmatism, coma and spherical aberration
Value Type	Continuous
IO Type	Output
Type	For aberration of interest: Control parameter: u / for other aberrations: Disturbance parameter: d
Unit	None
Value Range	Varying
How to calculate	The raytracing model in DT will calculate the final aberrations in the image. This will be the output of DT along with Ronchigram images

Table 11 Details of EM_H_IN_007 parameter

PROPERTY	**DESCRIPTION**
ID	EM_CC_IN_007
Name	Algorithmic parameters (to be detailed out later)
Description	The parameters that are only needed for the algorithms and don't have physical equivalents. Example: slice thickness in a multi slice algorithm
Value Type	Discrete or Continuous
IO Type	Input
Type	Hyperparameters: not relevant for this document
Unit	None
Value Range	Varying

A.2 Use case - Unmanned Utility Vehicles

A.2.1 General Use Case Information

Name	Unmanned Utility Vehicles
Description	The parameters for the use case UUV control the resolution of the scene, the level of detail for the environment model and the Adaptive Scenario Generation.
Company	AVL, Liang Dao, Triangraphics

Figure 27 shows an overview of the system and control input parameters for the UUV use case. The system parameters are used to initialize the scene of the environmental simulation.
The control input parameters can be divided into three categories:

- 3D Scenario Generation,
- Simulation,
- Sensor Simulation.

The 3D scenario generation parameters are used to manipulate the generation of the static 3D model and to control the dynamic behavior of objects in the scenes.
The simulation control parameters are used to manage the simulation environment. They can be used to control the simulation time, which indirectly affects the lighting of the scene.
The control parameters of the Sensor Simulation can be used to manage the positions and orientation of the sensors. Furthermore, various sensor properties can be configured.

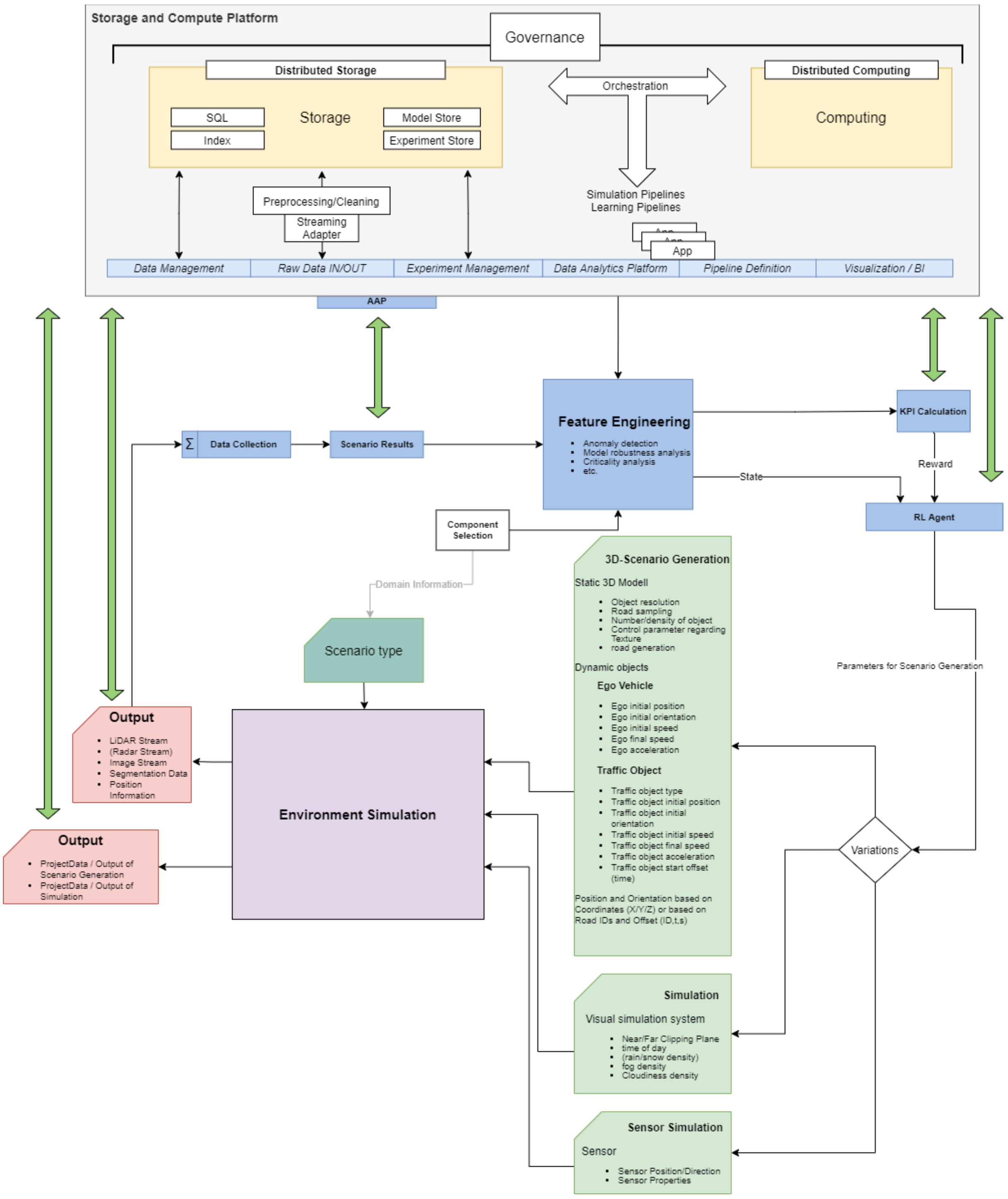


Figure 27 Unmanned Utility Vehicles parameter overview

A.2.2 Parameter

This section gives an overview of the parameters determined at the start of the project and their properties.

A.2.2.1 Captured data output

Table 12 Details of UUV _Y_CA_001 parameter

PROPERTY	DESCRIPTION
ID	UUV _Y_CA_001
Name	LiDAR Stream
Description	The data stream generated by the LiDAR sensor represents the 3D information of the scene captured by the LiDAR sensor. The LiDAR stream can be used for object detection.
Value Type	Continuous
IO Type	Output
Type	Output: y
Unit	m
Value Range	x and y: Typically, these coordinates can vary within the range of the LiDAR system, e.g. from -10,000 to 10,000 meters, depending on the size of the area being scanned and the resolution of the device.

Table 13 Details of UUV_Y_CA_002 parameter

PROPERTY	DESCRIPTION
ID	UUV_Y_CA_002
Name	Radar Stream (optional)
Description	Data Stream that is captured by the Radar sensor.
Value Type	Continuous
IO Type	Output
Type	Output: y
Unit	Angular position
Value Range	Distance to target: The distance range can vary from a few meters to several hundred meters. For example, a vehicle radar typically only covers a few hundred meters.

Table 14 Details of UUV_Y_CA_003 parameter

PROPERTY	DESCRIPTION
ID	UUV_Y_CA_003
Name	Image Stream
Description	The data stream generated by the image sensor represents the visual information of the scene captured by the image sensor. The image stream can be used for object detection.
Value Type	Continuous
IO Type	Output
Type	Output: y
Unit	Pixel (red, green, blue)
Value Range	Each of the three color channels usually has a range of values from 0 to 255, due to the 8-bit color depth often used in digital images.

Table 15 Details of UUV_Y_CA_004 parameter

PROPERTY	DESCRIPTION
ID	UUV_Y_CA_004
Name	Segmentation Stream (Ground Truth)
Description	Segmentation information of the captured scene. In a segmentation screen, each object class is marked with its own color. The segmentation stream can be used for error detection.
Value Type	Continuous
IO Type	Output
Type	Output: y
Unit	Pixel (red, green, blue)
Value Range	Each of the three color channels usually has a range of values from 0 to 255, due to the 8-bit color depth often used in digital images.

Table 16 Details of UUV_Y_CA_005 parameter

PROPERTY	DESCRIPTION
ID	UUV_Y_CA_005
Name	Position data (Ground Truth)
Description	Position information of the captured scene and ego vehicle. • Road id, s and t position, lane id • Ego vehicle, calibration target, sensor
Value Type	Continuous
IO Type	Output
Type	Output: y
Unit	m
Value Range	x, y and y: Typically, these coordinates can vary, e.g. from -10,000 to 10,000 meters, depending on the size of the area.

A.2.2.2 Static 3D Modell

Table 17 Details of UUV_C_ST_001 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_ST_001
Name	Object resolution
Description	This parameter controls the resolution of objects. Depending on the parameter, objects with lower resolution are replaced/generated by objects with higher resolution and vice versa.
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	ID
Value Range	The value range depends on the number of objects with different resolutions. The following range of values results in 5 objects with different resolutions: (1 - lowest detail, 5 - highest detail)

Table 18 Details of UUV_C_ST_002 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_C_ST_002
Name	Road sampling
Description	The resolution of the roads is controlled by a sampling rate parameter. The higher the sampling rate, the finer the resolution of the road. The higher the resolution, the more polygons are used.
Value Type	Discrete
IO Type	Input
Type	System parameter: c (first step), control parameter: u (afterwards, optional)
Unit	Samples per meter
Value Range	(1, Maximum sampling rate)

Table 19 Details of UUV_U_ST_003 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_ST_003
Name	Number/density of object
Description	This parameter controls the number or density of objects. It could be used to determine the density of a forest. A low value is used for a low number/density and a high value for a high number/density.
Value Type	Discrete
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	Min – Max distance between objects (0 – object bounding, 10 – 10 x object bounding)

Table 20 Details of UUV_C_ST_004 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_C_ST_004
Name	Texture parameter
Description	• Texture resolution (pixel, real world size) • Physical based Rendering (PBR) - glossiness, reflectivity, ...
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	Varying
Value Range	• Pixel: 1 - max power of two • Real world size: 0.001 - Max • PBR: Layer count

A.2.2.3 Visual Simulation system

Table 21 Details of UUV_C_VI_001 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_C_VI_001
Name	Near/Far Clipping Plane
Description	In computer graphics, a plane that limits the visible volume across the viewing direction is called a clipping plane. Anything closer to the camera than the near clipping plane and anything farther away than the far clipping plane will not be rendered. This pair of parameters determines the visibility.
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m
Value Range	0.0001 - 10000

Table 22 Details of UUV_U_VI_002 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_VI_002
Name	Time of day
Description	Parameter to change the time of day.
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	hours
Value Range	0 – 24

Table 23 Details of UUV_U_VI_003 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_VI_003
Name	Rain density (optional)
Description	Parameter that determines the rain density
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	0 – 10 (off - high density)

Table 24 Details of UUV_U_VI_004 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_VI_004
Name	Snow density (optional)
Description	Parameter that determines the snow density.
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	0 – 10 (off - high density)

Table 25 Details of UUV_U_VI_005 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_VI_005
Name	Fog density (optional)
Description	Parameter that determines the fog density.
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	0 – 10 (off - high density)

Table 26 Details of UUV_U_VI_006 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_U_VI_006
Name	Cloudiness density (optional)
Description	Parameter that determines the cloud density.
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	None
Value Range	0 – 10 (off - high density)

A.2.2.4 Scenario Description

The Scenario creation is a key element of UUV Sub Use Case 1. With the following parameters we aim to create a large variety of possible scenarios with relatively few parameters.

Table 27 Details of UUV_C_SC_001 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_C_SC_001
Name	Scenario type
Description	It characterizes the choice of template of the driving maneuver. (e.g. 1 => Ego following decelerating target, 2 => Pedestrian crosses street, ...)
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	ID
Value Range	The range of values characterizes the choice of driving maneuvers, with each numerical value associated with a specific scenario, e.g. 1 for 'Ego follows a decelerating target' and 2 for 'Pedestrian crosses the street'.

Table 28 Details of UUV_C_SC_002 parameter

PROPERTY	**DESCRIPTION**
ID	UUV_C_SC_002
Name	Ego vehicle initial speed
Description	It describes the ego vehicle's initial speed at the start of the scenario.
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	km/h
Value Range	0 to 30

Table 29 Details of UUV_C_SC_003 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_003
Name	Ego vehicle final speed
Description	It describes the ego vehicle's final target speed that shall be reached. This is purely a target speed that is then processed by the driving function. If the vehicle e.g. detects objects in front, so it needs to lower the speed, it will temporarily overwrite this target speed.
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	km/h
Value Range	0 to 30

Table 30 Details of UUV_C_SC_004 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_004
Name	Ego vehicle acceleration
Description	It describes the ego vehicle's acceleration from the initial to the final target speed. This acceleration in combination with the initial and final target speed implicitly determines the duration of the scenario.
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m/s^2
Value Range	-5 to 5

Table 31 Details of UUV_C_SC_005 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_005
Name	Traffic Object type
Description	IT characterizes the choice of Traffic Object, the Ego vehicle interacts with. (e.g. 0=> None 1=> Car, 2=> Bicycle, 3=> Pedestrian, etc.?) The goal is to also be able to describe mixed traffic, in which case the Traffic Object type would be a vector.
Value Type	Discrete
IO Type	Input
Type	System parameter: c (and partly control parameter: u)
Unit	None
Value Range	0-3(+)

Table 32 Details of UUV_C_SC_006 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_006
Name	Traffic Object initial speed
Description	It describes the traffic object's initial speed
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	km/h
Value Range	0 to 30

Table 33 Details of UUV_C_SC_007 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_007
Name	Traffic Object final speed
Description	It describes the traffic object’s final speed
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	km/h
Value Range	0 to 30

Table 34 Details of UUV_C_SC_008 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_008
Name	Traffic Object acceleration
Description	It describes the traffic object’s acceleration
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m/s^2
Value Range	-5 to 5

Table 35 Details of UUV_C_SC_009 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_009
Name	Traffic Object start offset
Description	It describes the traffic object’s delayed starting action. This is necessary to create different traffic situations, where the traffic object and the ego vehicle arrive at the same position with different temporal distance.
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m
Value Range	0 to tbd.

Table 36 Details of UUV_C_SC_010 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_010
Name	Traffic Object Road ID
Description	It describes the ID of the road that the traffic object is placed on. Only intersecting or same Road ID as Ego vehicle are valid, as this provokes interaction between the ego vehicle and traffic object.
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	ID
Value Range	Valid Road IDs

Table 37 Details of UUV_C_SC_011 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_011
Name	Traffic Object initial Road position
Description	It describes the traffic object's initial position s on the road it's placed on
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m
Value Range	Valid position s on Road

Table 38 Details of UUV_C_SC_012 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_012
Name	Ego Vehicle initial Road position
Description	It describes the Ego vehicle's initial position s on the road it's placed on
Value Type	Continuous
IO Type	Input
Type	System parameter: c
Unit	m
Value Range	Valid position s on Road

Table 39 Details of UUV_C_SC_013 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_013
Name	Maneuver Orientation Ego
Description	It describes the orientation of the Ego vehicle on the road segment
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	None
Value Range	+1, -1

Table 40 Details of UUV_C_SC_014 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SC_014
Name	Maneuver Orientation Traffic Object
Description	It describes the orientation of the traffic object on the road segment
Value Type	Discrete
IO Type	Input
Type	System parameter: c
Unit	None
Value Range	+1, -1

A.2.2.5 Sensor

Table 41 Details of UUV_U_SE_001 parameter

PROPERTY	DESCRIPTION
ID	UUV_U_SE_001
Name	Sensor Position/Direction
Description	Positioning and alignment of the sensors on the vehicle.
Value Type	Continuous
IO Type	Input
Type	Control parameter: u
Unit	m/None
Value Range	Position – car local space, Direction - normalized

Table 42 Details of UUV_C_SE_002 parameter

PROPERTY	DESCRIPTION
ID	UUV_C_SE_002
Name	Sensor Properties
Description	• Field of view • Sensor specific properties ○ e.g. LiDAR, Radar: Number of rays
Value Type	Continuous/Discrete
IO Type	Input
Type	System parameter: c
Unit	Degree/None
Value Range	Field of View: • Typically, from 0° to 360°, depending on the sensor technology and application. Number of beams: • LiDAR: Can range from hundreds to over a thousand beams. • Radar: Tends to use scan ranges, phased array types can also have hundreds to thousands of steerable elements.

Table 43 Details of UUV_U_SE_003 parameter

PROPERTY	DESCRIPTION
ID	UUV_U_SE_003
Name	Calibration and synchronization parameters
Description	Parameters that determine the position of the sensors both to each other and relative to the vehicle coordinate system
Value Type	Discrete
IO Type	Input
Type	Control parameter: u
Unit	m/degree
Value Range	Translation: From -2 meters to +2 meters in any direction relative to the center of the vehicle. Rotation: From 0 to 360 degrees for a complete rotation, allowing flexible alignment of the sensors.