DeLeMIS

Executive Summary

1. About this document

1.1 Scope of the document

This Executive Summary provides a comprehensive overview of the scope of the project DeLeMIS, the methodology, key findings and recommendations.

Name of the Author/ Reviewer	Entity
Dr. Alen Turnwald	e:fs
Niklas Baldauf	e:fs
Bernhard Rebele	DLR
Kristin Lakatos	DLR
Marcus Müller	DLR
Dr. Armin Wedler	DLR

1.2 Table of Contents

1.3 Table of Acronyms

Acronym	Description
AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
MPC	Model-Predictive Control
LBMPC	Learning-Based MPC
CNN	Convolutional Neural Networks

2. Introduction

The project DeLeMIS was funded by the ESA/ESTEC (ESA 4000136972/21/NL/AS), with e:fs TechHub GmbH as prime and DLR institute for robotics and mechatronics as subcontractor, both from Germany.

Main objective of the project is to demonstrate the improvements in a space system by applying Artificial Intelligence (AI) and Machine Learning (ML). For DeLeMIS, planetary exploration missions were chosen as use-case and a rover prototype for testing and investigation of the algorithms and results.

In a planetary exploration mission, the rover autonomy shall be possible even though limited information is available beforehand about the properties of the terrain. Several challenges arise from this problem, especially with regard to the motion control. In general, the physical behaviour of the rover, i.e. the motion and interactions with the environment, is very complex and cannot be fully captured by equations. Furthermore, the environment in which the rover navigates may be unknown and change over time. Therefore, conventional control methods, which require precise models, are more likely to fail. To this end, learning methods enabling improvements of the system model seem very promising. By learning or intelligence in this context, we understand that the rover is able to acquire more information about itself and its environment, change the representation of the knowledge accordingly, adapt the search for proper actions and even re-evaluate situations and actions through the newly acquired knowledge properties with the long-term goal to increase the autonomy of such systems. To obtain more information, the rover is equipped with appropriate sensors.

Figure 1 shows the Phases and the work package breakdown structure within the project. The final documentation will be organized according to the structure.

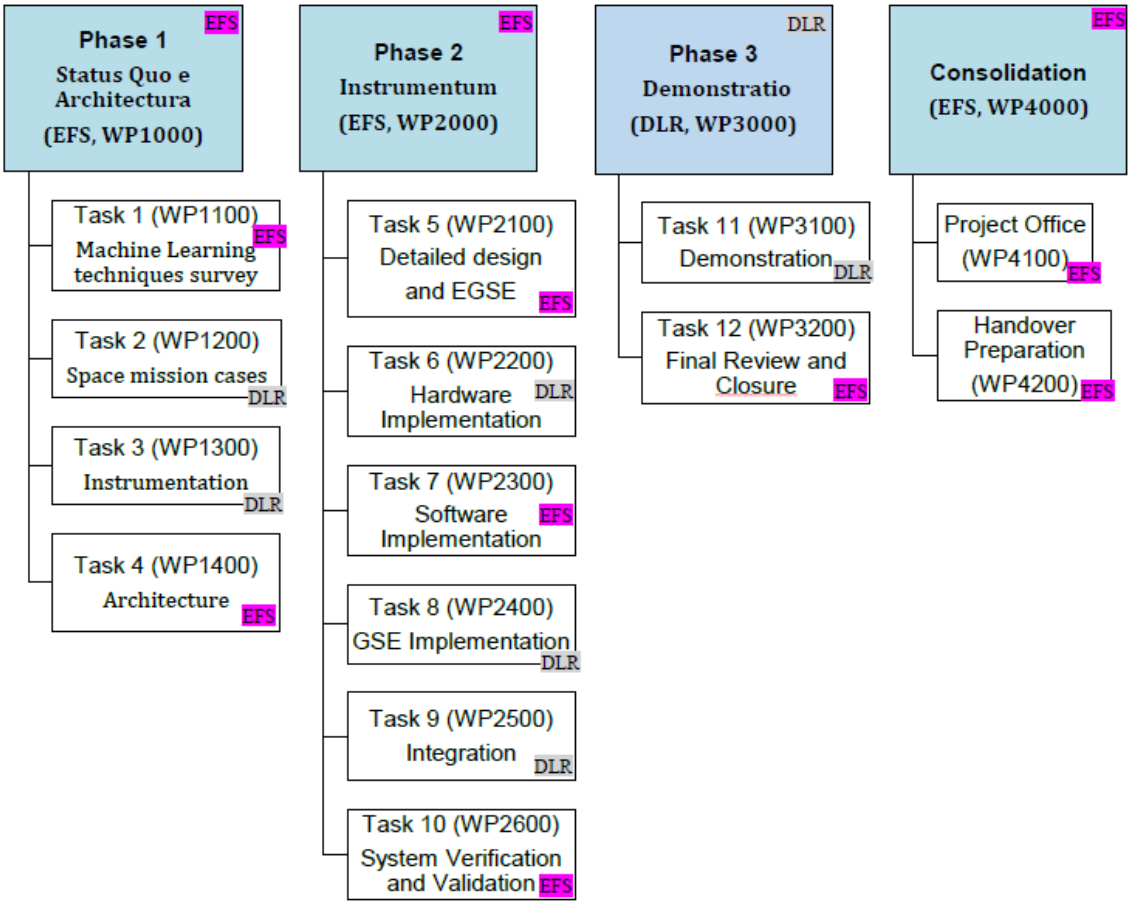


Figure 1: Work breakdown structure and responsibilities.

In the first phase of the project *Status Quo e Architectura*, the state-of-the-art ML approaches were investigated and reviewed. Both methods that were already used in space applications, but also methods that were successfully applied in other domains such as automotive were considered. Furthermore, a survey on current and future space missions was conducted with a focus on the potential application fields for ML for improvement. Based on the results, one or more scenarios were chosen to demonstrate the learning effects on an exemplary space mission. Finally, the overall architecture of the system-to-be-demonstrated was defined and criteria for verification and validation were derived. At the end of this phase, the ML method, the space mission case, the mission scenario, as well as the overall system concept and requirements for measuring the success of the demonstration were defined.

In next phase *Instrumentum*, consolidating the details of the demonstration scenario, the system design, and the implementation concept were defined. Different components (SW, HW, GSE) were provided, developed, and tested on the unit level. Furthermore, the combination of components was tested and validated. After successful testing of the components, the overall system was integrated and prepared for system-level testing. Finally, the overall system was tested regarding the chosen scenario(s). For the development of the overall system including the ML algorithms, hardware, GSE, and component level tests, agile software development approaches were applied.

In last phase *Demonstratio*, the demonstration of the ML algorithm and its application was executed. It was shown with the scenario(s) developed in the previous phases, that the machines were able to learn without human interaction. Proper presentation material was created, and a final presentation was held. Final review and acceptance of the outcome of the project by the agency was initiated.

2.1.1 Goals for using AI and ML in DeLeMIS

The first goal of the project was to investigate the combination of methods from control theory and AI/ML to improve the solutions to perception and control problems arising in the context of planetary rover missions.

Furthermore, improvement potentials by applying AI/ML methods to the navigation software of the rover were supposed to be demonstrated in the runtime.

2.1.2 Assessment and choice of the demonstration use case scenarios for ML application.

Potential demonstration use-cases and scenarios

For the assessment of the demonstration use cases, different rover's architecture layers were considered separately. Possible use cases and relevant scenarios were identified and evaluated regarding certain criteria. Table 1 shows the use cases according to the architecture layer, also discussed in the following subsections. The considered layers are divided with respect to two different aspects. In terms of the abstraction layer the architecture is classified into the three main parts of an autonomous system, namely sensorics, the "brain", and actoric.

Table 1 – Overview of possible mission cases according to application layer in rover structure.

Application Layer	Abstraction Layer	Use cases
Perception	Sensorics	Distinguish between different types of terrain
Localization and estimation	The "brain"	Improve odometry by learning kinetic and kinematic properties, e.g., wheel slip
Motion planning		Avoid disadvantageous terrain Learn unknown terrain properties
Motion control	Actoric	More precise and safe tracking Accomplish challenging trajectories

Proposed mission case scenario for development and demonstration of ML

The scenario involves one of the testing grounds of the. Basically, the testing ground aims to emulate rough terrain. This means, that the terrain features a significant profile of elevation change, impassable obstacles, such as large boulders or rocks, and at least two different kinds of soil, regarding their friction coefficient.

As shown in Figure 2, the rover is located at a defined starting position. In addition, there are several different target locations within the premises of the testing ground. All target locations are physically accessible for the rover by traveling along the a priori determined trajectories from the planning layer.

Regarding its ego localization, the rover utilizes the ground truth positioning system of the testing ground. This means, that the ego localization does not interfere with the evaluation of the improvements for each learning algorithm present below.

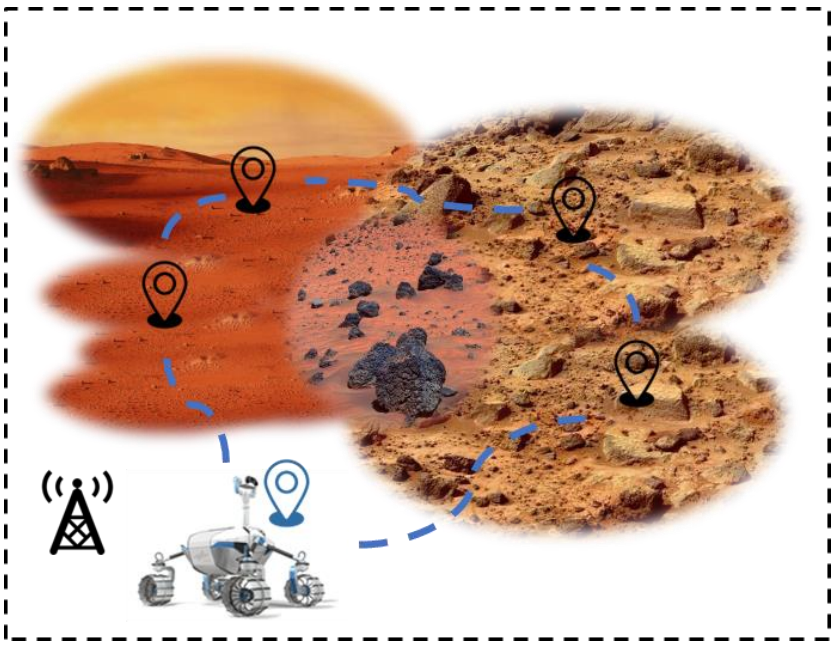


Figure 2: Exemplary structural concept of the scenario on the testing ground. The blue marker indicates the starting position of the rover, the black markers indicate different target locations. The blue dashed lines indicate the paths of the trajectories the rover should follow to reach the desired target location.

Before a scenario with one of the learning algorithms is started, the rover conducts reference measurements by traveling along all available trajectories. Thereby entities such as position, velocity, lateral and longitudinal distance to the trajectories, travel time, as well as entities representing the energy consumption and locomotion robustness will be gathered. By comparing these measurements to the evaluation measurements from the scenario, improvements in the trajectory tracking behavior can be identified, depending on the later to-be-defined measures for the improvement.

How the learning algorithms are implemented in the scenario

The implementation of each learning algorithm can be divided into three parts or phases. However, not every algorithm has to necessarily include all phases. The learning algorithm for the perception layer consists of part one and three, the wheel steering and correction algorithm as well as the learning-based control algorithm consists of all three parts.

The first phase involves offline learning from already available data, for instance, from previous space missions, other experiments, or data from simulation environments. The aim is to generate an educated guess and therefore an initial learning state.

If a learning algorithm contains the second phase, then this phase is conducted right at the start of the scenario. Here, the rover travels within close proximity to the lander to adapt or validate the learning algorithm with respect to the current environment. This might be comparable with an installation run from a real space mission to check if all systems are running correctly and to validate parameters.

The third part is the continuous adaption of the algorithm while following the pre-planned trajectories to the target locations. In this context, this concerns the adaptation of some parameters, units or functional structures inside the algorithms using the collected data within a ride segment. Here, the initial learning state from the offline learning part and the exploration part close to the lander are constantly adapted. In addition, ground information from the perception layer may also be taken into consideration, if available. During this part, the rover is following all trajectories to visit each of the target locations, analogue to the procedure to conduct reference measurements.

Definition of improvement of the trajectory tracking behaviour

As implied in the introduction, the aim of the scenario definition is to provide a reproducible procedure to demonstrate and to evaluate eventually if a learning algorithm can improve the trajectory tracking behaviour of the rover. The evaluation is based on certain criteria which will be defined and detailed during the implementation phase of the project. Such criteria might be the accuracy of the path tracking of the trajectory. Aspects such as lateral deviation from the trajectory, energy consumed for propulsion, or traveling time may be considered. For each learning algorithm, the results of the executed scenario are compared to its previously conducted reference measurements. As mentioned before, the planning layer uses a priori calculated trajectories to the different target locations. This is needed, to ensure that the same *challenging* trajectory is used during the reference and the scenario measurement. For example, a trajectory with less elevation change is likely to be less energy consuming.

3. System Design

3.1. System Requirements

To accomplish the tasks leading to the project objectives properly, a requirements engineering approach will be applied to ensure the fulfilment of all technical requirements, both those from SoW and those agreed on within the Tasks 1-4. A first iteration of the requirements engineering approach as state in the contractor's proposal is given below. Table 2: Definition of requirements' categories gives the definitions of different requirement categories.

Table 2: Definition of requirements' categories

ID	Requirement category	Definition
MSN-xxx	Space Mission Case requirements	Requirements regarding the space mission case for demonstration of the machine learning potentials in space applications. The chosen space mission, or a mission inspired by that, will be taken as the baseline for the demonstration scenario.
SCN-xxx	Demonstration Scenario requirements	Requirements regarding the scenario(s) within a space mission for which machine learning methods demonstrate an improvement in some yet to be defined sense.
SYS-xxx	System and architecture requirements	Requirements regarding the overall system design and architecture of the prototype for the demonstration.

MTD-xxx	Methodical requirements for the demonstrated ML algorithm	Requirements regarding the methodical approaches and algorithms of machine learning to be demonstrated on the prototype hardware.
HDW-xxx	Instrumental and Hardware Requirements	Requirements regarding the hardware and instruments for the proper execution of the demonstration.
GSE-xxx	GSE requirements	Requirements regarding the GSE for the proper execution of the demonstration.

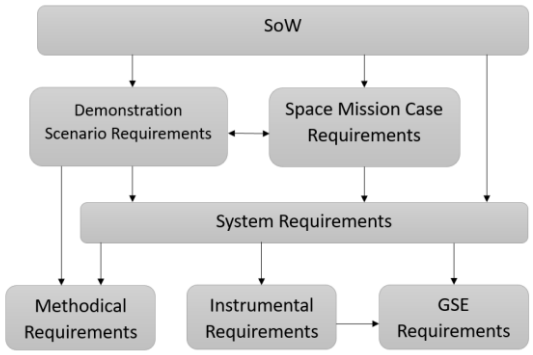


Figure 3: The requirements engineering approach

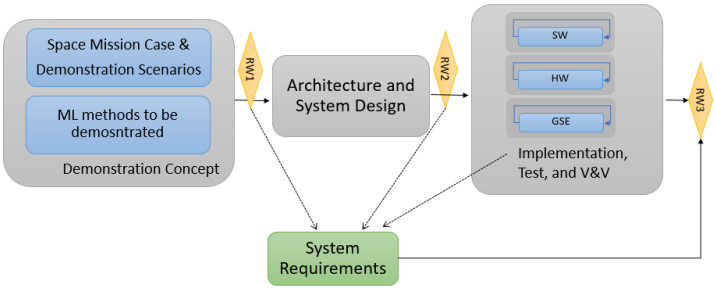


Figure 4: Iterative approach to requirements engineering.

The overall structure of the requirements engineering approach is given in the diagram in Figure 3. The overall approach for the requirements engineering in the project however is designed in an iterative manner. As illustrated Figure 4, the current document will be extended and updated along with the project timeline. The first version of the requirements will be defined and delivered with Review 1. During the detailed design phase, the requirements for the system and the algorithms will be concretised and updated. Furthermore, during the phase of the agile implementation the requirements will be reviewed and updated if necessary. The final version of these documents will be used for the validation and verification at the end of the project. The requirements presented in this documents are in the first phase of the project more on a high level. During the development phase and approaching the final presentation as well as the demonstration phase, more detailed requirements will be added, especially for the purpose of V&V.

3.2. Hardware Architecture

The Lightweight Rover Unit LRU from DLR–RMC is a new innovative rover tailored to the needs of future planetary exploration missions but also for terrestrial applications like search and rescue scenarios. The rover is designed to safely operate on moderate to challenging terrain as often encountered in exploration tasks. It allows the exploration of large areas in a fast and efficient manner. The rover is also capable of manipulating or evaluation objects.

Figure 5: LRU Rover with JACO2 Arm and science cam operating in lunar analogue environment on Mt. Etna

The design of the rover system is based on DLR's long experience in designing lightweight actuator units and robotic systems. The LRU rover locomotion subsystem (LSS) and its potential suitability for space applications have been given highest priority. The LSS and all relevant robotic components (e.g. including sensor setup) have therefore been designed in a space qualifiable manner and a future upgrade to a fully space qualified version has been considered from the very beginning respectively. The four wheeled LSS system is fully steerable. In addition to the four wheel and steering actuators two serial elastic bogie actuators allow to actively control the front and rear bogie joints. This allows e.g. controlling the load distribution to the wheels and the body orientation while maintaining the advantages of passive suspension. The wheels design, also a combination of rigid (e.g. tire tread) and flexible (spokes) elements, is additionally supporting fast and efficient driving in rough terrain. The lightweight design, the advanced kinematics and the unique combination of active and passive chassis elements result in a very high traffic ability, terrainability and overall mobility performance. [1]

The rover body and related remaining subsystems like power electronics, battery and communication are designed for terrestrial applications and based on reliable and cost-efficient commercially available components-off-the-shelf (COTS) as development time and performance have been the main design drivers. In order to fully exploit the capacity of the mobile system, an appropriate navigation algorithm and autonomy is implemented. Such high level control algorithms, e.g. the implemented autonomous way point navigation, are key elements for future exploration missions. As they strongly rely on perception sensors like cameras, the rover system includes a novel PanTilt Unit incorporating a stereo camera. The PanTilt Unit has been designed and optimized for the requirements of this mobile system operation in rough terrain. During the overall design process, the space application of different components and modules has been given highest priority and the related future space qualification process has been considered from the very beginning.

Currently two Light weight rover units (LRU) are available. One unit is equipped with a robotic arm (JACO2 robotic arm Payload 1,5 kg / 2,5 kg) whereas the second unit features a scientific camera and

a landing platform. The science cam is very similar to the one foreseen for the ExoMars rover and include e.g. additional spectral cameras, IR camera and a high-resolution camera.

LRU Sensor integration

The LRU actuators are based on the DLR-RMC Space Drive Unit series based on the robotic joints developed at DLR, a series of Light-weight actuators for space applications have been developed. They have been adapted to the special need of typical planetary rover and mobility systems. Currently the LRU makes use of two actuator sizes are available: The standard ILM38 Space Drive Unit (LSS) for wheel and steering actuators and the smaller LM25 Space Drive Unit is mainly used for the PanTilt actuators. ILM is the shortcut for the german wording of internal-rotor motor. The following number specifies the diameter of the stator (mm). The drive units have been continuously further developed and a space qualifiable unit is available. In general the units comprise the same design structure and are based on a Robodrive ® BLDC motor in combination with a single gear stage formed by a Harmonic Drive ® gear with a gear reduction ratio of 1:100.

System software and interfaces

The architecture of the overall system and the interfaces is given in Figure 6. The overall system from a software point of view consists of four main parts interfacing each other. The block “System Sensors”

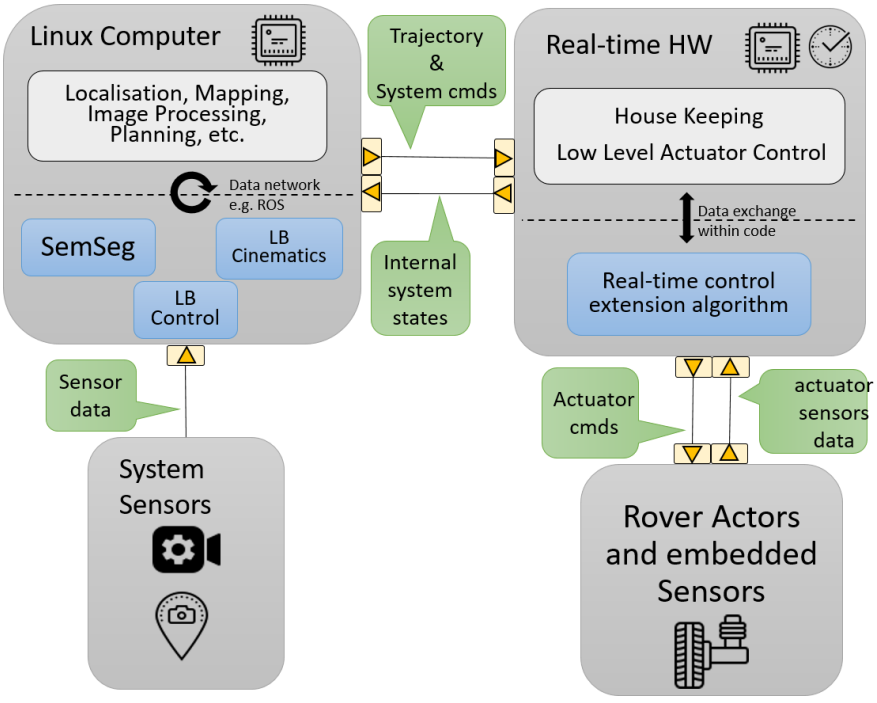


Figure 6: SW architecture: System and interfaces.

contains all sensors used in the scenario. This may contain camera systems, IMUs, any localisation and external positioning system. External sensors send the measurements and raw data to the main computer.

The main computer which uses a Linux OS implements advanced algorithms for localisation, mapping, motion planning etc. A framework, such as ROS, is used to host the algorithms and data exchange among different code components. The light grey box illustrates the components already available on

the rover. Blue boxes show the software components that will be extended or created within DeLeMIS. On one hand, code for semantic segmentation of the ground using image data will be included that uses an artificial neural network. On the other hand, the Learning-Based (LB) control algorithm is applied to the motion of the rover. Furthermore, the cinematics of the overall vehicle is also subject to developments since the cinematics model of the wheel steering affects the motion control. All code units in the main computer use the internal system states determined on the on-board real-time hardware of the rover. The results created by the algorithms are trajectories for the motion as well as system-level commands sent to the rover.

Beside algorithms and code for housekeeping of the rover, low level actuator control algorithms run on the real-time on-boards hardware. Any extension of the real-time and low-level control algorithms for the rover motion will be implemented here and data is exchanged directly through the code structure on the hardware. This unit sends low level actuator commands and receives data from the sensors embedded in the actuators.

Planetary Exploration Lab (PEL): Indoor Soil Testbed

The PEL is located in rooms K315 and K317 in the RMC basement. The facility features a control room overlooking the testing area separated by a windowed wall that faces the soil bin. The room K317 houses the test area (soil bin) and a preparation area next to the soil bin. The control room is located next to the test area people must stay during. During the scenario execution and the operation of the facility (e.g. slope, crane), people must stay in the control room or outside the facility. At any time, people must not stand under suspended loads or the slope.

The DLR-RM PEL indoor test facility consists of a 5.5 m x 10 m indoor soil bin. It is used for testing planetary locomotion systems as well as navigation algorithms independent of weather conditions. The terrain can be composed by different materials, e.g., soft soil or stones and rocks of different size and shape, to investigate the mobility for different terrain set-ups. A wide variety of planetary soil

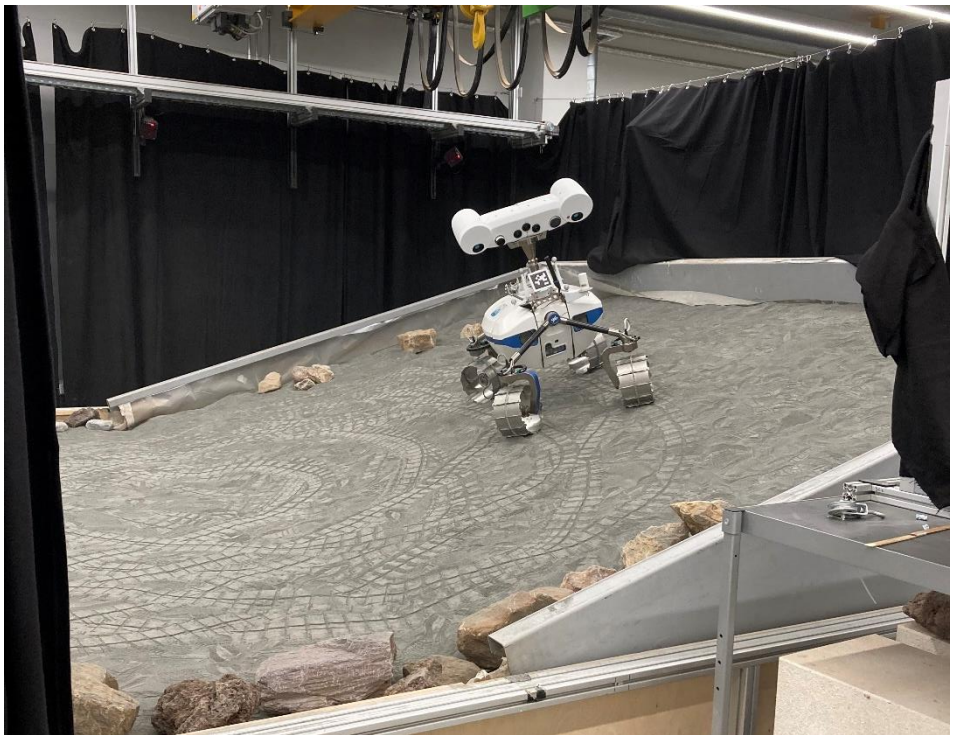


Figure 7: Tilting Platform – PEL with LRU

simulants as well as obstacles and rocks are available. The PEL is currently used for a validation test campaign for a planetary exploration mission (MMX) any may be adapted to the mission needs. This may include exchange of soil or enhancement of sensor setup. The terrain will be remodelled according to the needs of DeLeMIS. It shall be mentioned that it is assumed that exchanging the soil is out of the scope of DeLeMIS project as it is very labour and time intensive and most likely the soil type has to be used as it is. A part of the soil box can be tilted up to 30° to model steep slopes. Furthermore, the facility is equipped with an optical position tracking system and a fully automated Digital Surface Model (DSM) device.

DLR Outdoor Test Facility

The DLR-RM outdoor test facility consists of a large area with a planetary surfaces analogue section. This terrain consists of different surface materials such as sand, gravel, and rocks. Furthermore, a crater formed, as it can be found on lunar surfaces, allows testing, navigation, and planning algorithms in analogue environment. Currently, the testbed is over-worked and enlarged. The new and much larger outdoor test facility will be operational in 2024.

Simulation

The simulator OAISYS is used to generate image data, which can be expected on another planetary body. Therefore, OAISYS needs a set of textures, which can be used to create basic terrains. These tenures were carefully chosen in order to create realistic scenes. For some training procedures it is important to create as many versatile data as possible. In these cases the quality of realistic rendering might be less important than the quantity of used semantic classes. Therefore, we adapted the simulator to also render many terrains, but with a lower quality. With that it is possible to create datasets in the order of multiple thousands in a very fast manner.

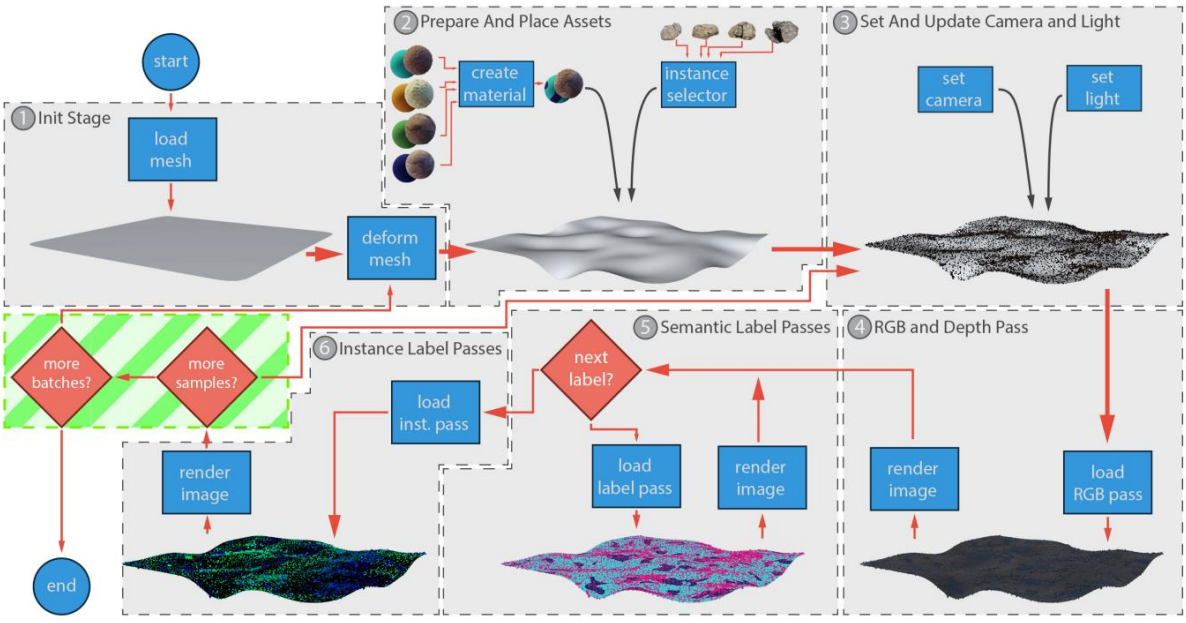


Figure 8:Basic Flow Diagram of OAISYS [2]

A detailed description of the OAISYS simulator and the link to the open-source repository can be found in [2]

Rover Gazebo Simulation

To simplify and accelerate development during the project, a simulation environment has been created that includes a model of the LRU rover, a model of the DLR test environments, and a flexible interface architecture for integrating the of the various algorithms. This gazebo simulator (https://gazebosim.org/), which is integrated into the ROS framework (https://www.ros.org/), also serves as a data generator for the ML-based methods and as a test and verification platform for first-phase algorithm development. Although the simulator is not able to reproduce the exact behaviour of the sand subsurface, a certain degree of sliding some degree of sliding and submergence of the rover can be observed so that nonlinear behaviour can be reproduced. At order to increase the representativeness and reliability of the simulation, data from DLR facilities will be Data from DLR facilities will be collected and used to verify the behaviour of the simulated rover in relevant and comparable scenarios. For this purpose, the rover will follow predefined trajectories in DLR's test facility using a new lunar simulant, the Lunar Ground Simulator. Lunar simulant, which will also be used in the new DLR/ESA lunar test facility "LUNA" in Cologne. "LUNA" in Cologne will be used. The resulting trajectory data are then compared with those of the simulator and the simulation parameters are adjusted accordingly.

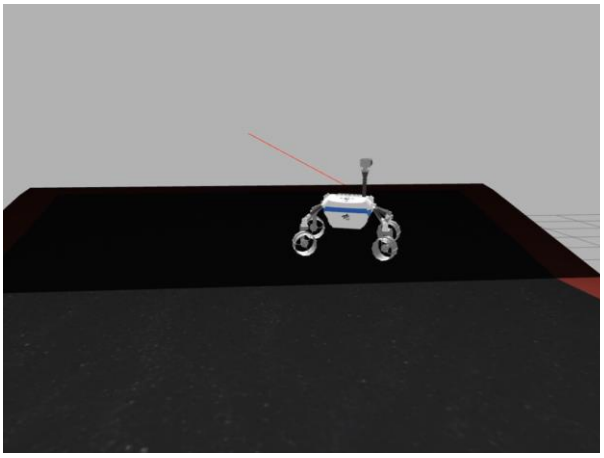


Figure 9: Gazebo simulation of the LRU in the PEL environment

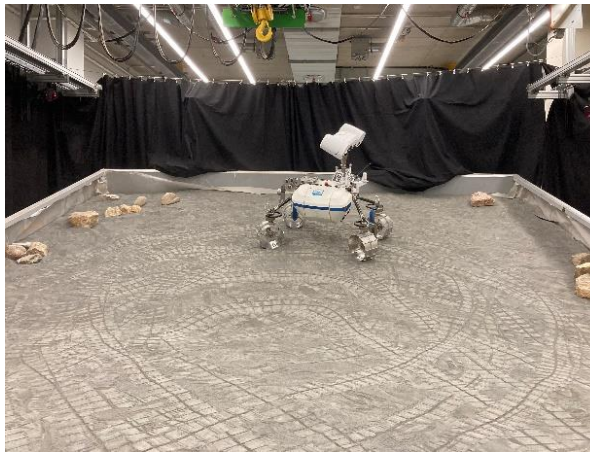


Figure 10: Real LRU rover in the PEL during test drives

3.3. Software Architecture

Learning algorithms investigated in this project are used to approach 3 different problem settings. Each problem is considered isolated as a stand-alone task. This will help isolating the effects of the learning algorithms and enables detailed analysis of the results.

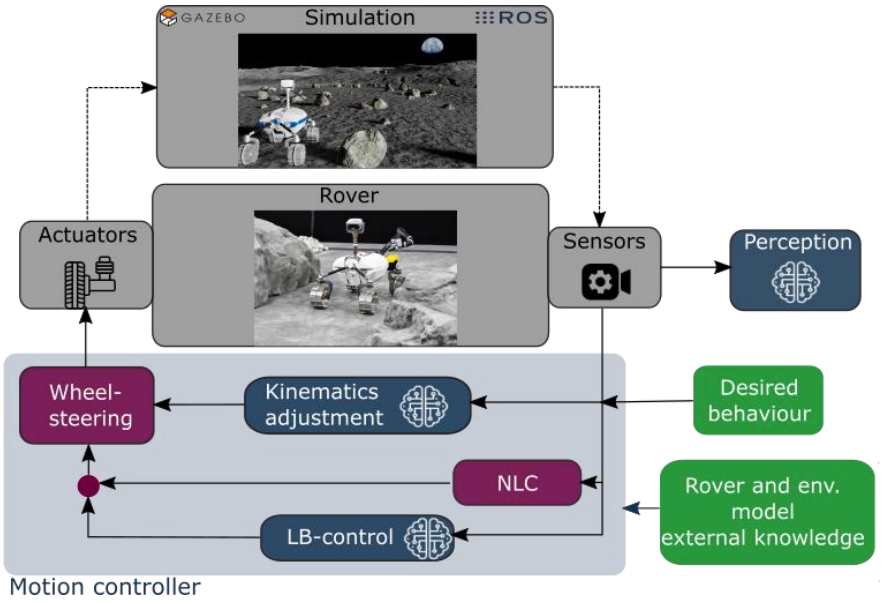


Figure 11: Overall structure of the software.

The blue blocks Figure 11 indicate the learning algorithms:

- ML-based perception,
- ML-based kinematics adjustment, and
- ML-based motion control.

In the following sections, each problem setting, and possible algorithms to solve them are described.

3.3.1 Learning-based perception

For this module, we aim to develop a perception method which can segment an image input based on predefined terrain classes.

- Rendering and generating data sets in OAISYS
- Plug-In OAISYS Object mode

For this module, we aim to develop a perception method which can segment an image input based on predefined terrain classes. The used neural network to perform that task is based on a transformer architecture as illustrated in Figure 12. The developed network can be spitted into an encoder and

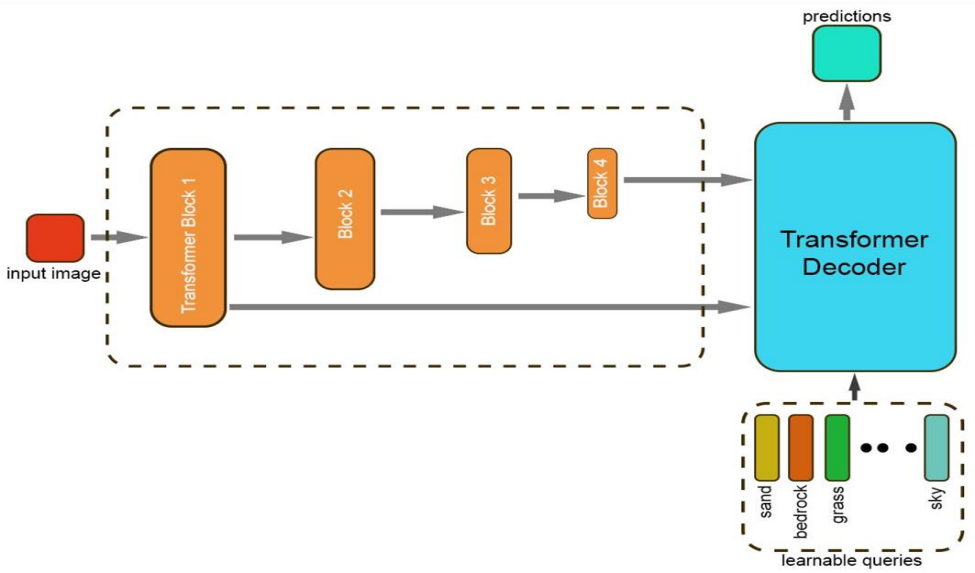


Figure 12: Transformer DNN architecture for image segmentation

decoder part. The encoder part extract semantic features from the provided image. The decoder uses so-called queries, which are attended to these features. The outcome is an image prediction of semantic classes. In the following we give more detail explanation about the structure of the encoder and decoder. In order to get the most information from the provided image the developed network uses a pyramid level transformer encoder. Instead of splitting the image into multiple patches, which are fed into the network, the entire pixel map is provided. After each encoder block the resulting image features are down-scaled and fed to the next encoder block. Transformer are using usually so-called Multi-Head Attention Layers in order to extract image features. However, such layers cannot directly be applied on the entire image, since the needed memory is too large for most hardware. Therefore, spatial reduction attentions (SRA) are used, which can handle such input tensors. The decoder of the network consists of a transformer architecture, which taking the image features from the encoder and learnable queries as input. The learnable queries are vectors, which can be used to store the information of a particular class. In the end of the training procedure each query represents one of the semantic classes. These queries are attended to the input features. If the input features are matching with a particular query, the query will be activated and therefore shows if a semantic class is present in the image. The output of the decoder and with that the network is the semantic segmentation prediction.

3.3.2 Learning-based wheel kinematics adjustment

The wheel kinematics adjustment module implements a learning algorithm, that adapts the control output of the baseline nonlinear controller with an additive term. This changes the input of the wheel steering controller and consequently impacts the rover movement. For this purpose, an approximation function is designed, that is trained online on tracking data to improve the tracking behaviour. Due to implementation reasons, we chose to use Gaussian Process Regression (GPR), which utilizes Gaussian Processes (GP) [3], [4] $g(z) \sim GP\left(m(z),\, k(z,\, z')\right)$. Such a GP model can then be utilized to build a regression model by treating the model as a prior, defined only by a covariance function, also known as kernel function k . In our case, we use the common squared exponential kernel function. By adding observation data,

$$\mathcal{D} = \{\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_\mathrm{N}] \in \mathbb{R}^{n_z \times \mathrm{N}},\ \mathbf{Y} = [\mathrm{y}_1, ..., \mathrm{y}_\mathrm{N}] \in \mathbb{R}^{1 \times \mathrm{N}}\},$$

where n_z denotes the number of features in the observation vector Z and N the number of data points $(z_k,\, y_k)$, the GP model can be used as a posterior distribution. This posterior distribution can then be used to predict output values $\mu^d(z_*)$ and its probability $\Sigma^d(z_*)$ for a desired data points z_* (test points),

$$\mu^{\mathrm{d}}(\mathbf{z}_*) = \mathbf{K}_*^\top [\mathbf{K} + \sigma_\mathrm{n}^2 \mathrm{I}]^{-1} \mathbf{Y},$$
$$\Sigma^{\mathrm{d}}(\mathbf{z}_*) = \mathbf{K}_{*,*} - \mathbf{K}_*^\top [\mathbf{K} + \sigma_\mathrm{n}^2 \mathrm{I}]^{-1} \mathbf{K}_*,$$

where d denotes the "*d-th*" dimension of the output. K , K_*, and $K_{*,*}$ are defined as $k(Z, Z)$, $k(Z, Z_*)$, and $k(Z_*, Z_*)$, respectively.

The features of the observation vector z in the proposed approach consists of the current trajectory dynamics, described by the curvature and the velocity, and the roll angle of the bogie axles of the rover. The corresponding output (additive yaw rate for the nonlinear controller) of the observation is determined by the current tracking performance, the lateral deviation and heading error, according to the trajectory dynamics. The number of data points N of the observation data is limited, due to increasing computational costs with every data point.

The selection of observation data is currently handled with a First In – First Out (FIFO) method. Other methods, such as covariance-based ones, must be further investigated and might be applied during the demonstration as well.

At the current state of the approach, the hyperparameter of the GPR are continuously optimized via the log marginal likelihood method [3] after a new data point is added to the observation data. Unfortunately, this method comes with a high computational cost, which might lead to a fixed set of hyperparameter from the simulation for the demonstration.

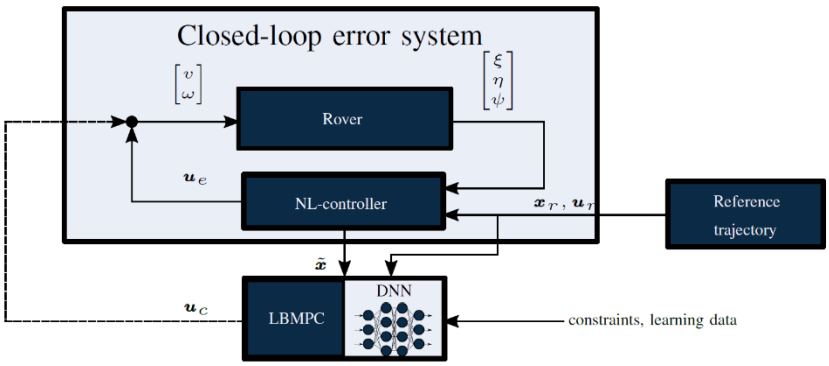
3.3.3 Learning-based motion control

Several learning approaches build on top of Model Predictive Control (MPC), e.g., [5], [6]. MPC provides a set of advanced control methods for the computation of control inputs to a system such that some optimal behaviour, specified by a cost function, is achieved. The basics of the general nonlinear approach of MPC is well explained in [7]. Authors of [8], who are among the top researchers in this area for decades, explain the mathematical concepts and tools required for understanding and

application of MPC and its different variants in detail. The system is usually given by (nonlinear) difference (discrete in time) or differential equations (continuous in time) and may be subject to constraints on its states and inputs. MPC uses this system description to forecast the behaviour of the controlled system. Then it solves an optimization problem, based on current state and these predictions, to find an optimal sequence of inputs that also ensure that the constraints are satisfied.

The desired behaviour and the nominal path-following functionality provided planning layer. The learning-based motion control will be implemented by an iterative learning MPC. Its task is to improve the overall system behaviour despite the unknowns and uncertainties and, at the same time, to make sure that optimality, constraints, and safety criteria are satisfied. A particularly interesting method, that will be investigated, adapted, and implemented was presented in [9]. In [10], the method was used to minimise the lap time of a race car by learning a better model for the car in each lap. In our setting, we will use this method to achieve a better path following than by the nominal system alone. A introduction to this method is given in the following section. It is important to note that the phrase "LBPC" is used commonly in the control engineering literature to describe an data-based online-adaptive control algorithm that uses past measurements and given data to adapt some aspects of the optimization problem embedded in the controller. Below, more details are presented on the approach planned for this project.

This problem setting, we assume that a realizable path is provided by a trajectory planner. We also assume a nonlinear tracking controller and a static wheel steering function that are suited to let the



rover track the given trajectory under ideal circumstances. However, there will be deviations due to

Figure 13: Closed-loop error system framework.

unknown aspects in the physical model as well as uncertain interactions with the environment. To reduce these deviations, an iteratively learning model predictive control (MPC) framework will be employed.

In our approach, the system is first linearised in combination with a non-linear controller and then controlled with a learning-based MPC as illustrated in Figure 13. This approach simplifies the optimization problem and reduces the dependence on an iterative process. To be more specific, the chosen system architecture enables that the MPC only receives the error states of the system, i.e. the deviations from the specified trajectory. Therefore, it is no longer necessary for the learning algorithm to repeatedly run the same trajectory as in many other approaches. To address additional performance goals beyond constraint satisfaction and leverage the advantages of learning-based model error prediction for the trajectory tracking task, a learning-based MPC is synthesized in the outer-loop to gain the optimal control input.

The DNN used for the approximation and prediction of the system dynamics is a Feed Forward Neural Network (FFNN) with ReLU activation features and linear activation features in the output layer. The width of the DNN was three times the number of input features in each layer with a depth of 5 layers. For training the DNN in the Keras framework, datasets with iterations are generated in the Gazebo simulation environment. One iteration consists of a chosen length of 200 time steps. The feature vectors used are the deviations from the reference trajectory, the current and desired velocities, the inclination and pitch of the rover and the last control inputs. The DNN then predicts the expected error in the state prediction for the next time step and is repeated for the whole horizon. For faster prediction and use in optimization, the mesh is stored as a frozen graph, achieving a prediction time of 10 µs for one time step. An even more detailed description of the method can be read in [11].

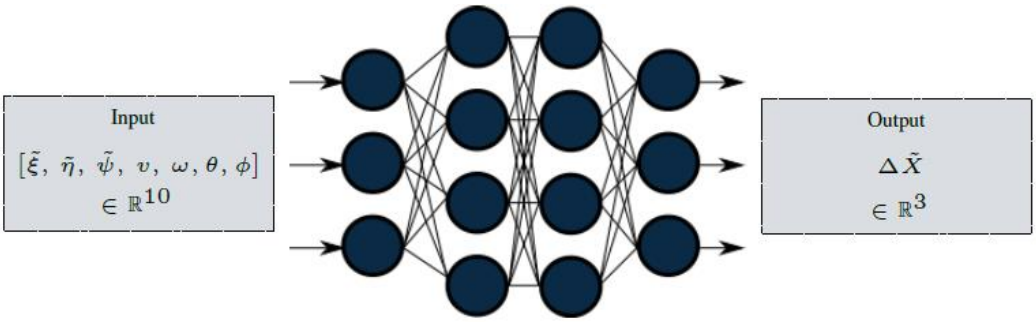


Figure 14: DNN for the system dynamics error prediction.

As an alternative and advanced network architecture, an recurrent neural network (RNN) is used. This is a bi-directional network architecture with internal states. This architecture offers the possibility, for example through Long short-term memory (LTSM) layers, to consider past inputs or sequences in order to make a better prediction of the future system states. The goal of this architecture is to use the RNN part of the network to perform feature selection and prediction of the system behaviour. This part of the network should be trained offline with collected data sets, since the training of this layer takes a lot of time. The second part of the architecture consists of an FFNN and takes over the scaling of the fault prediction to the current environment of the system. This part will then be adapted online during runtime.

4. Testing and demonstration results

4.1. Demonstration: Perception – Semantic Segmentation

To test the trained terrain segmentation network on real data collected by the rover LRU, we collected a dataset of LRU driving in the DLR outdoor lab. The network was than applied on the collected images. The following images show the results of the network. One can see that the terrain segmentation network is achieving quite good results on the collected data. It is also noticeable that different subclasses are correctly distinguished. For instance, different types of gravel, with fine or coarse gravel is separated. One notes that the network has some problems indefinitely the different classes in the distance. In further research it might therefore be interesting to incorporate the depth images or other modalities. Nevertheless, the segmentation network performs quite well in the near field, which is the most important for traversability analysis of a rover unit.

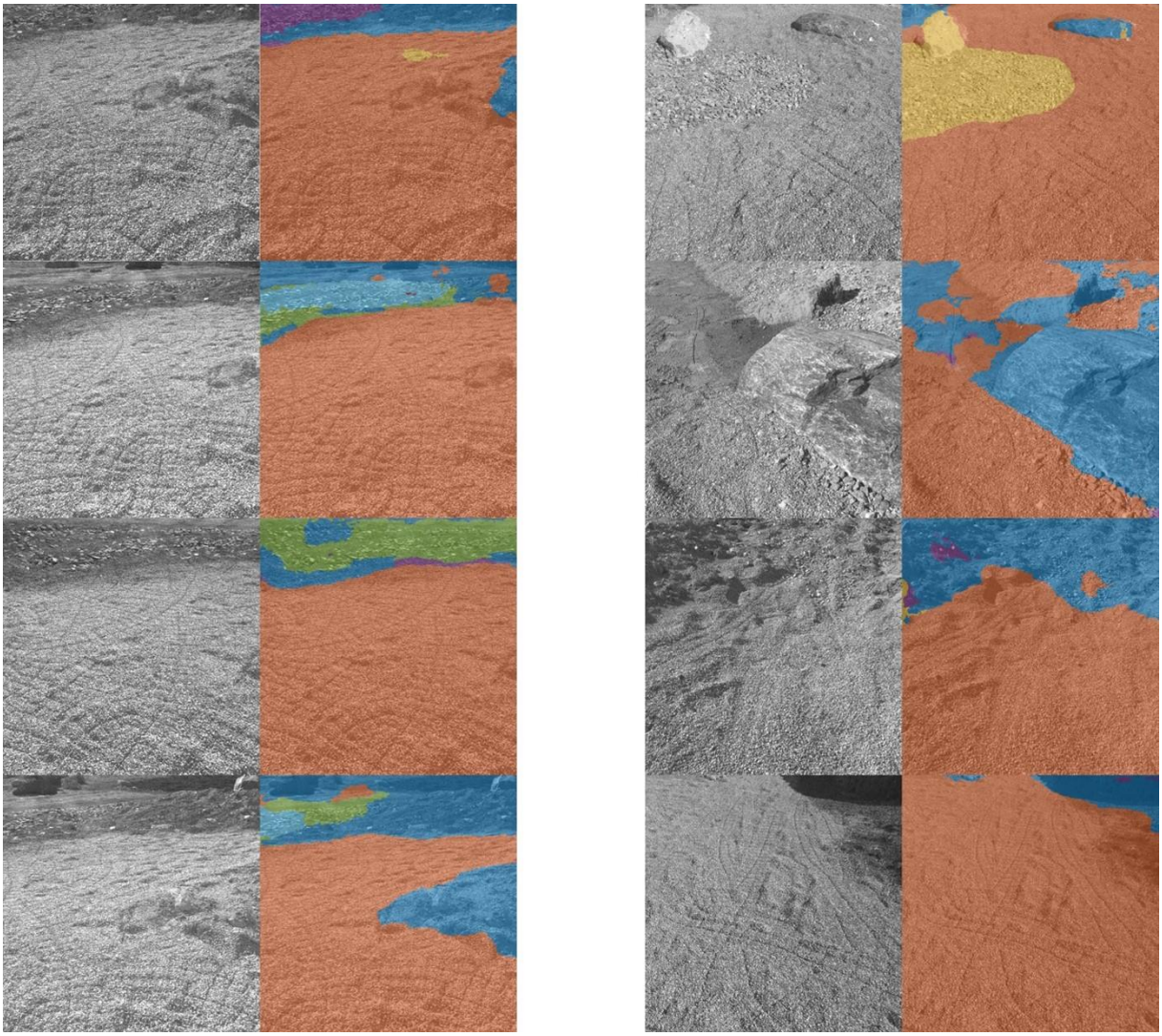


Figure 15: SegNet applied to recorded rover data.

The terrain segmentation was deployed on the ML device of the LRU system. The LRU features a Jetson TX2. The terrain segmentation takes about 2.19 seconds for one inference pass. This value was calculated by averaging 100 forward passes. The network shows that it almost uses the entire GPU efficiently in performance capability as well as usable memory. Taking the velocity of the LRU into account, the segmentation network shows an appropriate performance. Future project may focus on reducing the necessary memory in order to run potential multiple ML tasks next to each other.

4.2. Demonstration: Learning-based MPC

The example Demonstration of the Learning-based MPC already shown in the Detailed System Design takes place in a defined scenario and procedure to isolate the improvement due to the implemented algorithm. The Demonstration is performed according to the setup already used and explained during the test campaign with the same Rover system and test facility configuration.

The demonstration can be subdivided into the following steps:

1. Ground truth drive with *figure of eight* trajectory and nominal MPC (*Drive 1*)

2. Train the RNN in the LB-MPC with data from D*rive 1*
3. **Open loop** LB-MPC drive with a new trajectory (*Drive 2*)
4. **Closed loop** drive with LB-MPC and the new trajectory (*Drive 3*)
5. Train RNN with new data from latest drive
6. **Closed loop** drive with the Newly trained RNN in the LB-MPC (*Drive 4*)

4.2.1 Drive 1: Ground truth drive with *figure of eight* trajectory and nominal MPC.

In the first test drive we used the nominal MPC without the learning-based part in the test facility to drive a simple figure of eight trajectory as reference and logged the data for training of the learning-based module.

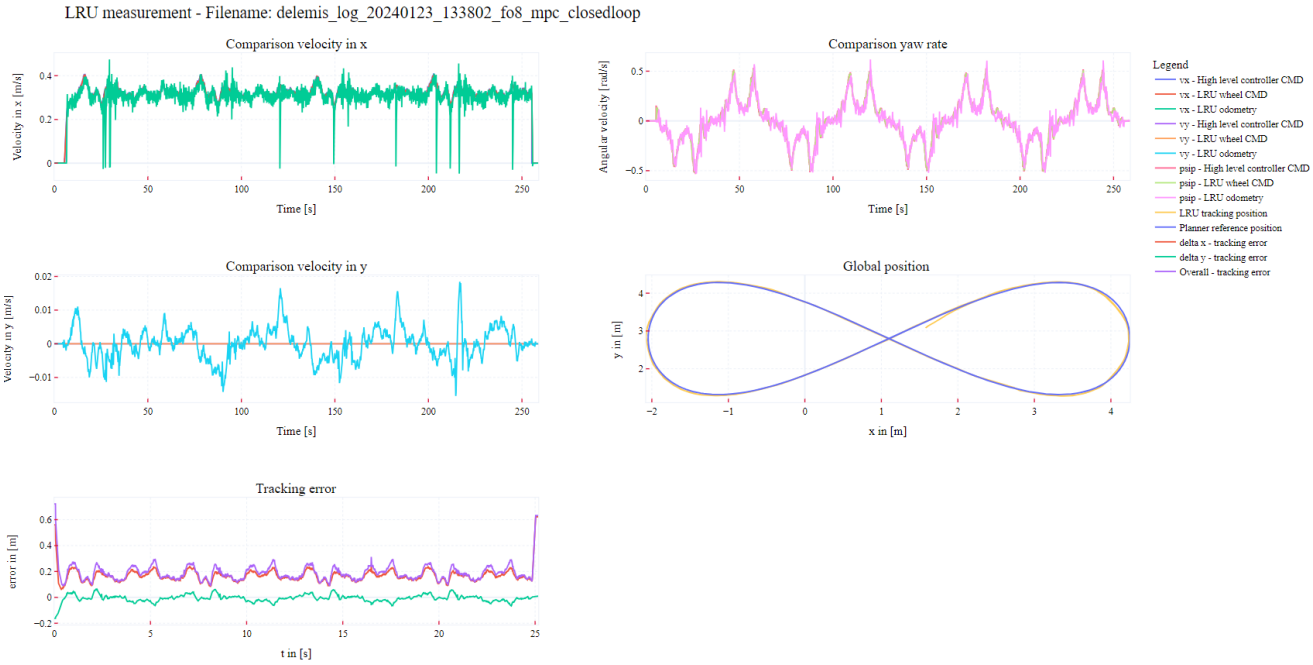


Figure 16: Test of the learning-based motion controller module in overall system. With the commanded high-level velocities from the LB-MPC (blue, pink and purple), the commanded velocities of the low-level controller to the wheels (red, green and orange) and the estimate odometry from the rover (green, pink and cyan). The reference path provided from the planner (blue) and the measured path of the rover by the tracking system (yellow). The lower right figure shows the tracking errors calculated during the test drive, with the error in x-direction (red), in y-direction (green) and the overall absolute error (purple) compared to the reference path.

After the successful test drive of 3 rounds, we trained the RNN with the recorded data. The training history of the RNN in Figure 66 shows the mean squared error loss during the training epochs of the training data set (with dropout) and of the validation dataset. With the R2-Score for the training data set of $R^2 = 0.9149$ and for the validation data set of $R^2 = 0.8967$. This score indicates that the trained model is able to replicate the observed outcomes very well. The training of the RNN took about 3 minutes on the rover hardware.

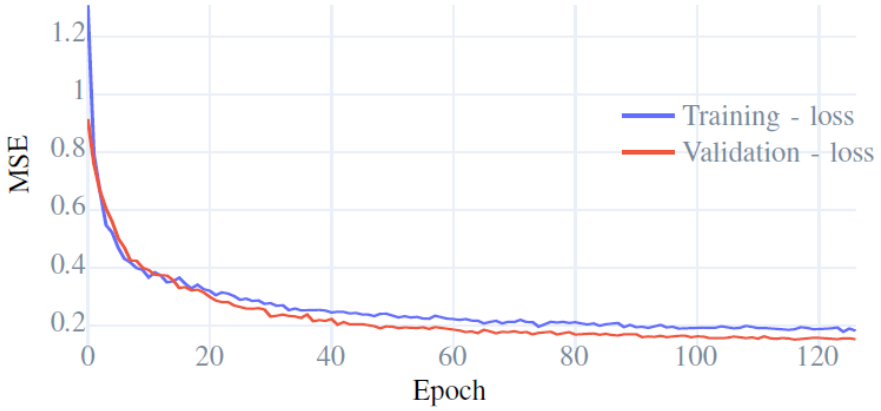


Figure 17: Training history of the RNN showing the mse on the training (with dropout) and validation data set over the training epochs.

4.2.2 Drive 2: Open loop LB-MPC drive with a new trajectory.

In the second test drive we used the learning-based MPC in open-loop configuration to drive another trajectory then in the first drive under same environment conditions. In the open-loop configuration of the controller, the RNN predicts the internal error states $\tilde{x}$ online, but the predictions are not used in optimisation problem.

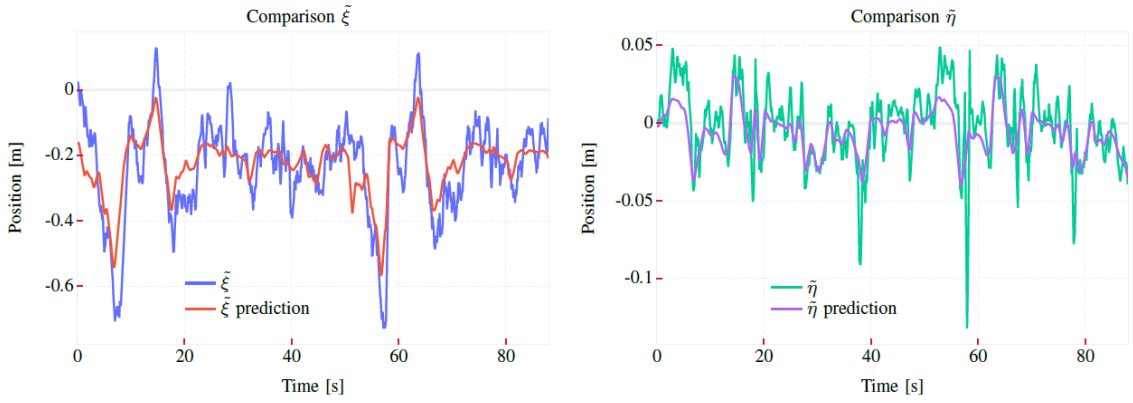


Figure 18: Prediction of the MPC error states $\tilde{x}$. The prediction is computed by the DNN for the next step in the prediction horizon. Including the prediction of the MPC $\tilde{\xi}$ error (red) and the measured $\tilde{\xi}$ error (blue) on the left side and bzw. the prediction of the MPC $\tilde{\eta}$ error (green) and the measured $\tilde{\eta}$ error (purple) on the right side

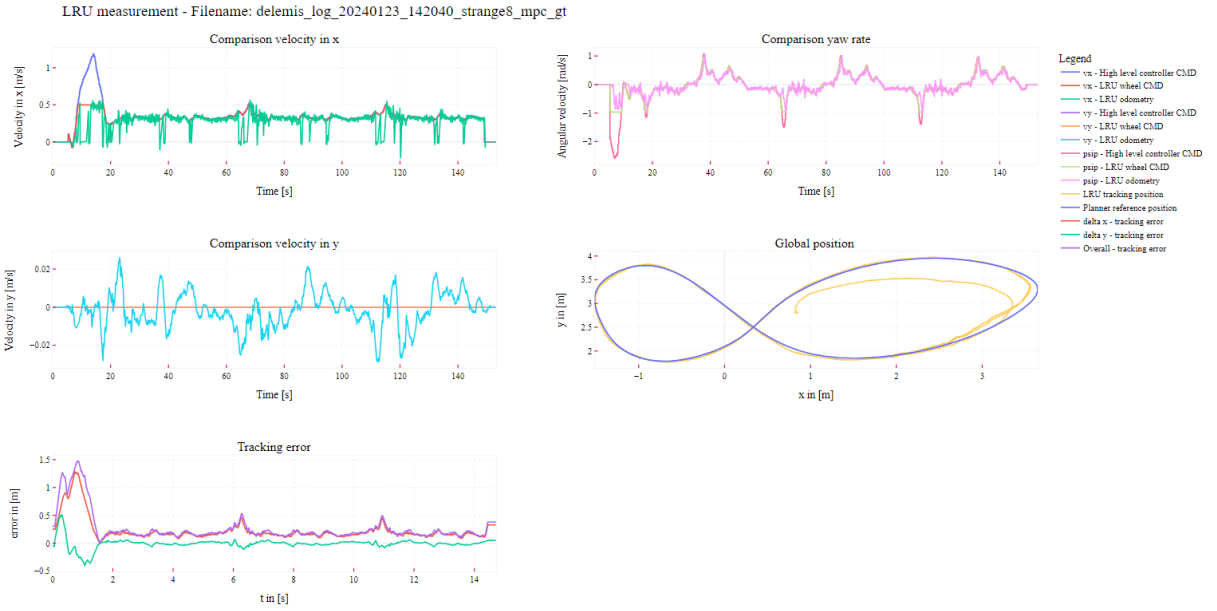


*Figure 19: Test of the learning-based motion controller module in overall system with the **open loop prediction** of the RNN.*

4.2.3 Drive 3: Closed loop drive with LB-MPC and the new trajectory.

In this test drive we used the learning-based MPC in closed-loop configuration to drive another trajectory then in the first drive under same environment conditions. In the closed-loop configuration of the controller, the RNN predicts the internal error states $\tilde{x}$ online and the predictions are used in optimisation problem.

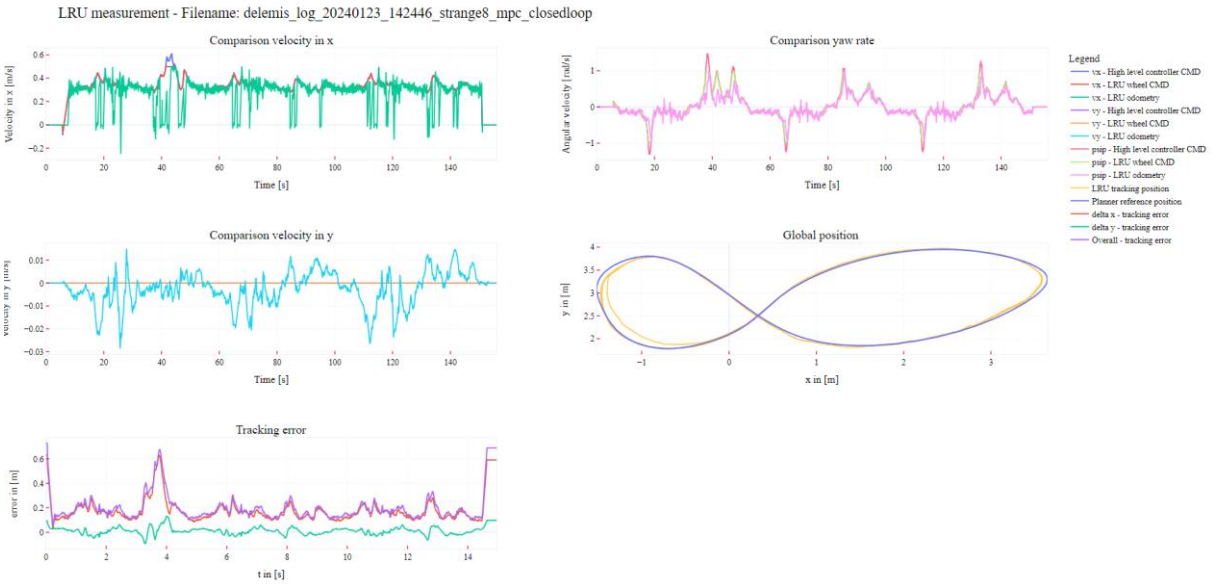


*Figure 20: Test of the learning-based motion controller module in overall system with the **closed loop prediction** of the RNN.*

4.2.4 Drive 4: Closed loop drive with the Newly trained RNN in the LB-MPC

In the last test drive we used the learning-based MPC in the closed-loop configuration to drive the trajectory but with a new trained RNN with the data from drive 3.

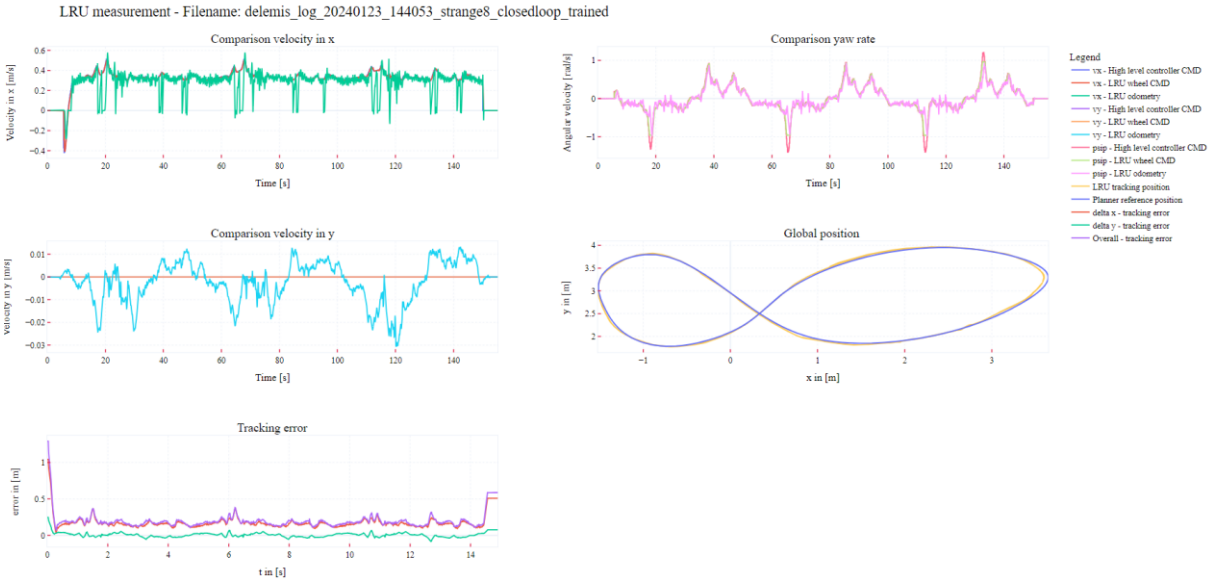


*Figure 21: Test of the learning-based motion controller module in overall system with the **closed loop prediction** of the **new trained RNN**.*

Table 3: Root mean squared error of the test drives 2-4 in [m] with the RMSE calculated from the measured positions by the tracking system. With the RMSE in x-direction (driving direction of the rover, the error in y-direction (lateral deviation of the rover from the desired trajectory and the overall error.

	Ground truth (drive 2)	Closed loop (drive 3)	Closed loop trained (drive 4)
RMSE x	0.3062	0.2075	0.1802
RMSE y	0.0920	0.0335	0.0305
RMSE Overall	0.3197	0.2102	0.2031

In summary, it can be said that the overall trajectory tracking error improves by approx. 34% in relation to the reference (ground truth) drive and improves by 36% with additional online training.

5. Final Review and Closure

5.1. Conclusion and lessons learned.

Robots have a great potential to realize future planetary exploration missions. However, with the use of robotics, new challenges emerge when deployed in distant planets. High communication delays, and direct controllability of the robot is impossible are some of the major problems. Autonomous navigation by using methods from the AI/ML field can be utilized for making a complex robotic system to move on an uncertain terrain win an unpredictable and kaleidoscopic environment.

Main objectives of DeLeMIS were on one hand, Investigating AI/ML techniques for autonomous navigation and control of a planetary Rover on unknown terrain. Another main objective was the

assessment of added value by AI/ML in different modules of the software toolchain of a rover prototype.

Main achievements of the project are listed as

- Implemented developed software on a rover prototype and tested in closed loop including GSE.
- Developed a simulation environment including relevant features such as wheel ground interaction to a satisfying degree.
- Implemented AI/ML algorithms in the simulation as well as on the rover prototype and tested in laboratory environment.
- Minimizing the tracking error despite unknowns and uncertainties by utilization of AI/ML for model knowledge improvement.
- Improved perception and motion control of an autonomous planetary Rover by using advanced control methods combined with AI/ML techniques.

In conclusion, the project was a success and the main objectives were reached. From the results, it can be stated that AI-enabled algorithms have a great potential to increase autonomy for space missions, especially for planetary exploration. It was demonstrated that such complex algorithms are able to be implemented on a relevant hardware setup and contribute to the performance of autonomous systems. It was also demonstrated that the online adjustments of the parameters using data, often referred to as online-learning, is possible and leads to improvements in the robot behaviour. The most significant effect, however, comes from the offline learning that is the tuning of the parameter using known information and simulations. Also, utilizing AI-enabled methods requires more resources onboard and may increase the calculation time for parts of the software. Considering the possible increase in the autonomy for a mission, usage of AI-based algorithms can be highly beneficial. This requires a proper development to higher TRLs, especially involving requirements engineering, testing and V&V, as well as FDIR.

Some of the results created during this project were published in different peer-reviewed conferences [12] [13] [14].

5.2. Outlook and future work

The path to the usage of AI-enabled algorithms in a real space mission to handle the autonomy is al ready outlined. On one hand, algorithms need to be further developed and improved for specific scenarios. Most importantly, methods for formal performance and robustness analysis shall be implemented and used.

On the other hand, existing algorithms and the corresponding software shall be improved, tested, verified, and validated excessively. Analysis of the robustness as well as the V&V for such algorithms requires dedicated methods. There are already different approaches in the literature to testing and V&V for AI algorithms, which need to be investigated.

Furthermore, as a future work and evolutionary progress is recommended introducing causal-AI to achieve a fault-tolerant and robust autonomous robotic system. The modern method of Causal-AI is already gaining attention in many branches of engineering and computer science and could offer a proper tool to approach formal analysis of complex and AI-enabled systems.

Reference

[1] M. J. Schuster, S. G. Brunner, K. Bussmann, S. Büttner, A. Dömel, M. Hellerer, H. Lehner, P. Lehner, O. Porges, J. Reill, R. Sebastian and M. Vayugundla, "Towards Autonomous Planetary Exploration: The Lightweight Rover Unit (LRU), its Success in the SpaceBotCamp Challenge, and Beyond," *Journal of Intelligent & Robotic Systems,* 2017.

[2] M. G. Müller, J. Lim, L. Schmid, H. Blum, W. Stürzl, A. Gawel, R. Siegwart and R. Triebel, "Interactive OAISYS: A photorealistic terrain simulation for robotics research," *ICRA 2022 Workshop on Releasing Robots into the Wild: Simulations, Benchmarks, and Deployment,* May 2022.

[3] C. E. Rasmussen, "Gaussian Processes in Machine Learning," *In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science,* no. vol 3176, 2003.

[4] C. E. Rasmussen and C. K. I. Williams, Gaussian processes for machine learning, Cambridge: MIT Press, 2006.

[5] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica,* vol. 50, no. 12, pp. 2967-2986, 2014.

[6] D. Q. Mayne, "Control of Constrained Dynamic Systems," *European Journal of Control,* vol. 7, no. 2-3, pp. 87-99, 2001.

[7] L. Grüne and J. Pannek, Nonlinear Model Predictive Control, Springer London Dordrecht Heidelberg New York, 2010.

[8] F. Borrelli, A. Bemporad and M. Morari, Predictive Control for linear and hybrid systems, Cambridge, 2011.

[9] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. A data-driven control framework," *IEEE Transactions on Automatic Control,* vol. 63, no. 7, p. 1883–1896, 2018.

[10] U. Rosolia, A. Carvalho and F. Borrelli, "Autonomous racing using learning Model Predictive Control," in *2017 American Control Conference (ACC)*, 2017.

[11] N. Baldauf and A. Turnwald, "Iterative learning-based model predictive control for mobile robots in space applications," in *MMAR*, Międzyzdroje, Poland, 2023.

[12] K. Lakatos, N. Baldauf and a. A. Turnwald, "Towards learning-based trajectory tracking control for a planetary exploration rover: Development and testing," in *International Conference on Space Robotics*, 2024.

[13] N. Panagiotopoulos, T. Lubiniecki, A. Turnwald and N. Baldauf, "Robust Machine Learning Systems For Dependable Space Applications," in *European Data Handling & Data Processing Conference (EDHCP)*, 2023.

[14] N. Baldauf, A. Turnwald, T. Lubiniecki, K. Lakatos and N. Panagiotopoulos, "Learning-based motion control of a rover on unknown ground," in *International Conference on Guidance, Navigation & Control Systems (GNC)*, 2023.