Towards learning-based trajectory tracking control for a planetaryexploration rover: Development and testing

Lakatos, Kristin; Baldauf, Niklas; Turnwald, Alen

Towards learning-based trajectory tracking control for a planetary exploration rover: Development and testing

Kristin Lakatos[1], Niklas Baldauf[2], and Alen Turnwald[2]

***Abstract*— Slip estimation and compensation are crucial factors for the success of ground-based planetary exploration missions. This work presents the development and implementation of a test campaign for the systematic evaluation of trajectory tracking controllers for a planetary exploration rover. The performed tests focus on machine learning algorithms for wheel-ground interaction. The control approaches are developed and tested both in simulation and on a rover prototype in a space-analog scenario. First results comparing the performance of the classical tracking controllers and their learning-based extensions are presented. The results imply that the use of machine learning techniques can improve the tracking performance on difficult terrain significantly.**

I. INTRODUCTION

The exploration of foreign planets is a goal that bothers humankind since the beginning of its days. Nowadays, manned and unmanned missions are launched in order to explore the nearest celestial bodies. Due to the risks and sometimes insuperable barriers of manned space missions, robotic vehicles have been widely used for scouting and research purposes, as exemplarily shown in the NASA Mars missions [1], [2]. However, one of the pitfalls for ground-based vehicles is the unknown terrain. The danger of undesired slipping and skidding can culminate in getting stuck, as e. g. the *Spirit* Rover experienced in May 2009 in loose, soft terrain [3]. Thus, it is no surprise that the topic of wheel-ground interaction is of great interest [4]–[6]. From a control perspective, it can be divided into the question of slip estimation and slip compensation. An overview of the state of the art focusing on current and past space missions can be found in [7] and the references therein. Still, this problem is not yet solved, especially using conventional methods of model-based slip estimation and compensation. Thus, machine learning techniques have recently come to the fore as potentially promising approaches.

The DeLeMIS project showcases AI methods that enable a planetary rover to autonomously navigate in uncertain terrain without intervention. The long-term goal is to increase the autonomy of such systems, even with limited prior information about the terrain properties. This presents several challenges, particularly in motion control, as the rover's physical behavior and interactions with the environment are complex and not entirely predictable through equations or models. Therefore, learning methods allow for improvements and thus hold significant promise. In this context, "learning" refers to the rover's ability to gather more information about itself and its environment, adjust the representation of this knowledge accordingly, adapt its search for suitable actions, and even reassess situations and actions based on newly acquired knowledge. In this paper, the focus will primarily be on presenting the fundamentals for testing and comparing learning-based control algorithms from the DeLeMIS project. An overview of the software modules developed in the course of the project is given in Fig. 2.

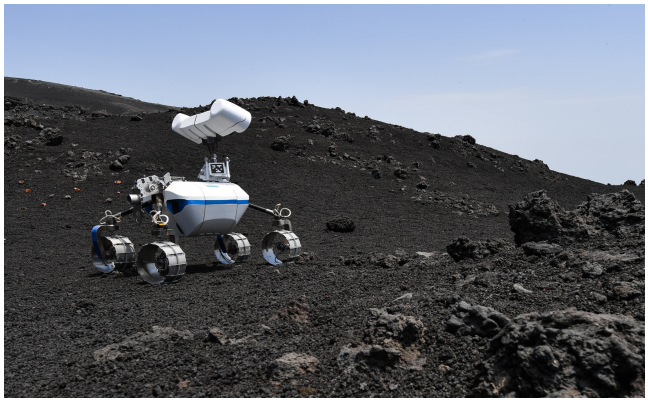


Fig. 1: LRU driving on volcanic soil during a Mars-analog campaign on Mt. Etna in 2022. ©DLR/Oprean

Within this work, hardware tests are performed on the Lightweight Rover Unit (LRU) at the Institute for Robotics and Mechatronics of the German Aerospace Center, see Fig. 1. The rover is a research prototype of a planetary exploration rover, designed according to space concept studies [8], [9]. The platform design features high mobility and versatility in rough terrain. For localization and mapping purposes, it is equipped with a stereo vision system. The software framework running on the rover provides a great level of autonomy [10], which is one of the key requirements for planetary exploration [1]. The rover was part of the ROBEX and ARCHES missions on Mt. Etna in 2017 and 2022 [11]–[14], simulating autonomous Moon and Mars missions in a space-analog terrain [15]. During these missions, the wheel-ground interaction was examined experimentally based on a parametric slip model [16] in order to improve wheel odometry measurements. However, the results show that terrain properties can vary significantly. Thus, a statically calibrated slip model still results in significant errors in wheel odometry computations. To overcome the drawbacks

[1]Kristin Lakatos is with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR). Kristin.Lakatos@dlr.de

[2]Niklas Baldauf and Alen Turnwald are with the department of Space Applications (e:space), e:fs TechHub GmbH. Niklas.Baldauf@efs-techhub.com, Alen.Turnwald@efs-techhub.com

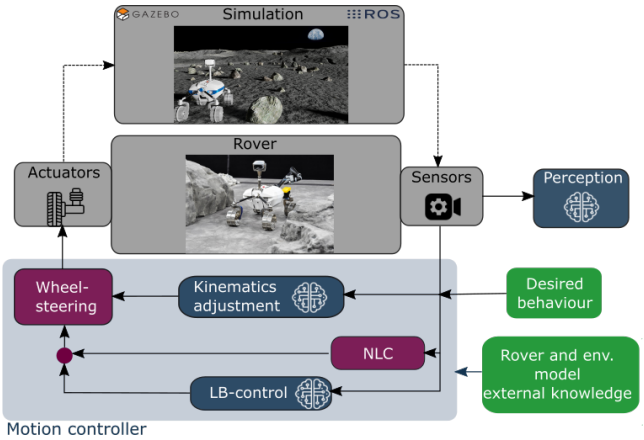


Fig. 2: High-level structure of the developed modules within the DeLeMIS project (a detailed description can be found in [22]). This work focuses on the learning-based control (LB-control) part.

of model-based slip computations, we implement and test an approach based on machine learning. The focus is on a combination of model predictive control (MPC) and learning-based extensions. A detailed overview of the possible variants of learning-based MPC methods is provided in [17] and [18]. A concept which is closely related to our approach was recently presented in [19]. However, our approach only requires a small training data set and does not use a Gaussian process. Other works like [20] or [21] focus on racetrack scenarios and are conceptually iteration-based. In contrast, our approach can also be used beyond lap-based scenarios.

Our main contributions are the implementation of two algorithms for slip-compensated trajectory tracking based on machine learning algorithms, and the setup and implementation of a test campaign which allows to evaluate the performance of the approaches w. r. t. trajectory tracking systematically. Hardware experiments are performed with a real rover in an indoor testbed filled with a Moon-analog soil. In this work, the setup of the test campaign is described in detail. Moreover, first results are shown and discussed, while an extensive publication of the results will follow.

The paper is organized as follows: The test scenario and problem statement is described in Sec. II. Test environment and procedures are sketched in Sec. III, while the used controllers are shown in Sec. IV. Some selected results of the test campaign are shown in Sec. V. A discussion of the experimental setup and the results follows in Sec. VI. Sec. VII concludes the paper.

II. SCENARIO DESCRIPTION

The goal of the test scenarios is to provide a reproducible procedure to demonstrate and evaluate improvement potentials of learning algorithms to the rover's trajectory tracking behavior. The scenarios are defined to be implemented in a real test environment. At the same time, the investigated control algorithms are kept as isolated as possible to ensure precise analysis of the results in absence of external influences.

A. Trajectories

A set of standard trajectories at different levels of difficulty was created considering the geometric specifics of the test facilities. The set consists of ellipsoid, figure-of-eight-shaped, and a set of Lissajous curves of different orders. Moreover, a GUI for the creation of custom trajectories is provided, which allows the selection of an arbitrary number of waypoints, which are then connected by spline interpolation. Additionally, different velocity profiles can be assigned to each trajectory, particularly a constant and a sinusoid velocity profile with a specified maximum translational velocity.

B. Problem statement

The fundamental task of the motion control subsystem is to ensure that the rover adheres closely to the prescribed trajectory, even when the terrain is only partially understood or entirely unknown. Moreover, the objective is that the performance of this subsystem improves over time through iterative refinement. The continual enhancement serves as an indicator of the system's learning capabilities and its ability to navigate more effectively in varying environments. Thereby, the motion control subsystem operates on an intermediate level, situated between the planning layer and the low-level motor control layer. Its primary function is to translate the high-level trajectory instructions from the planning layer into rover commands while accounting for uncertainties and constraints. The low-level motor control layer is then tasked with executing the commanded actions by controlling the individual actuators.

In order to isolate the problem and evaluate targeted algorithms within the framework of the project, several assumptions are made for the motion control subsystem. First, the rover relies on a ground truth positioning system installed within the testing environment for a precise self-localization. This system provides accurate positional information without introducing biases or inaccuracies that could affect the evaluation of learning algorithms. Second, the trajectories provided to the motion control subsystem are always assumed to be reachable with no obstacles. Note that these assumptions are only made w. r. t. the setup of the test campaign, to ensure comparability and significance of the obtained data. None of the assumptions is necessary for the learning-based algorithms tested within the campaign. A discussion of the test setup follows also in Sec. VI.

III. TEST ENVIRONMENT AND TEST PROCESS

The testing of the algorithms is performed both in simulation and on the real rover. In the following, the simulation environment as well as equipment and process of the hardware tests will be described.

A. Simulation

For an initial test of the developed modules, a simulation environment shown in Fig. 3 is set up in Gazebo and ROS. A model of the LRU and the testbed is integrated and the behavior is compared with real measurement data from test drives. The simulations and physics parameters are optimized

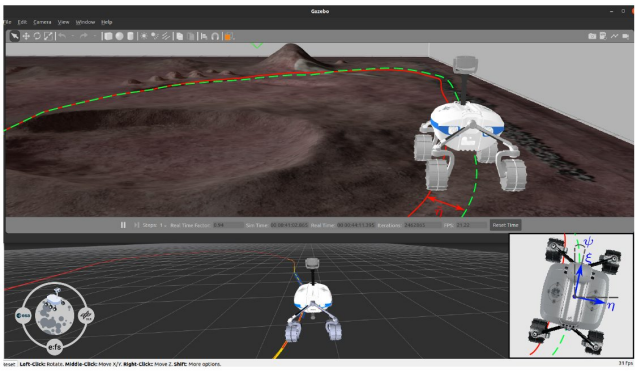


Fig. 3: The Gazebo simulation including the LRU in a modeled landscape [22]. The simulation is designed such that the same software packages can run in simulation and on the rover.

in order to reproduce a driving behavior that is as similar as possible without losing the fast simulation runtime. The selected ROS architecture and distribution of the algorithms into individual ROS nodes make it possible to run the same modules in the simulation and on the real rover. This makes it possible to carry out software in the loop (SIL) tests as a preliminary preparation for hardware tests.

B. Test environment and equipment

The hardware test are performed with the LRU rover at the DLR test facilities in Oberpfaffenhofen. The LRU was built in 2015 at the Robotics and Mechatronics Center of the German Aerospace Center. The research prototype features driving speeds of up to $1.1\,\mathrm{m/s}$. The LRU has a total length of $1140\,\mathrm{mm}$ and a total width of $740\,\mathrm{mm}$, with $230\,\mathrm{mm}$ ground clearance. A pair of stereo cameras is mounted in a height of $940\,\mathrm{mm}$ on a pan-tilt unit and used for navigation purposes. Additionally, the rover is equipped with a complex science camera module. The four individual steering and wheel propulsion actuators are mounted on two bogie axes (front and rear), which are connected to the body via Serial Elastic Actuators (SEA). This allows to control the body roll angle to distribute vertical loads on the wheels or improve tip-over stability in rough terrain, while profiting from the passive damping properties of the SEAs. All actuators (bogie, wheel propulsion and steering) utilize ILM38 drive trains with $5\,\mathrm{Nm}$ rated torque, which were designed for space applications. In total, the weight of the rover is less than $40\,\mathrm{kg}$. In order to support the autonomy requirements of a planetary exploration mission, the computations of all relevant software components are performed on board. Therefore, the LRU comprises a 9th generation Core i7 mobile processor (featuring efficient mechanisms for throttling CPU power) for non-realtime software components. Additionally, a NVIDIA Jetson TX is used for extensive on-board image and navigation processing. The low-level control algorithms are computed on a separate realtime computer, which is currently an *UP Squared* with an Intel® Atom™ processor and a realtime operating system.

The Planetary Exploration Lab (PEL) is an indoor

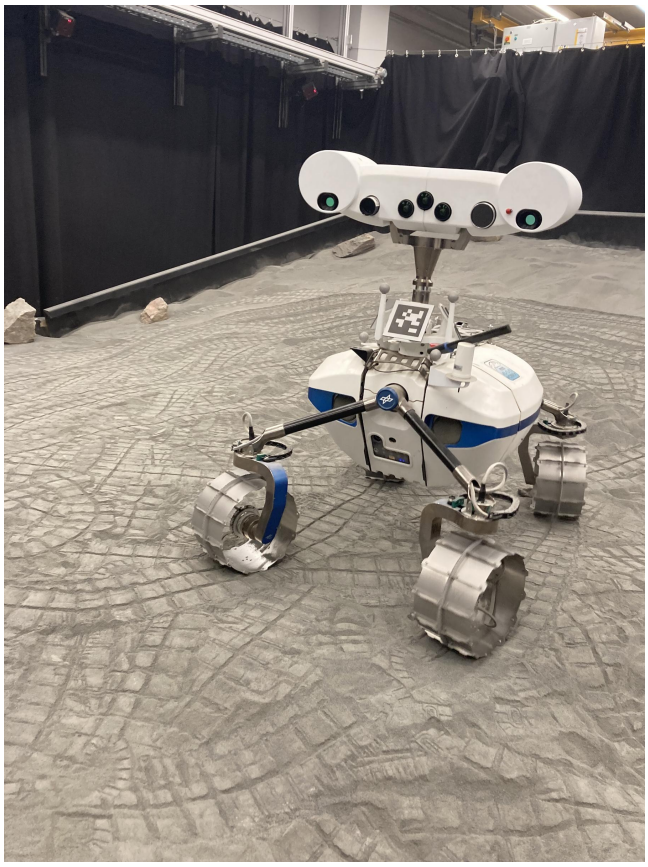


Fig. 4: LRU driving in the PEL during the test campaign. The tracking markers are mounted on the rover's neck.

Moon/Mars-analog testbed for rough-terrain rover systems. It contains an area of the dimensions $10\,\mathrm{m} \times 5\,\mathrm{m}$ which can be filled with different types of soil, depending on the test scenario. The current soil is the EAC-1A soil, a certified Moon analog. The physical properties of the sand are described in [23]. The slope of the rear part of the test plane with a size of $3\,\mathrm{m} \times 5\,\mathrm{m}$ can be set between 0 and $30\,\mathrm{deg}$. To minimize potential dust exposure during driving tests, the operators can work in a separated room containing computers and other technical infrastructure.

The PEL is equipped with an ART tracking system consisting of 10 cameras. In order to track the LRU, six marker balls have been mounted at the rover's neck (see Fig. 4). The marker balls allow the tracking of a corresponding frame with the ART tracking software *DTrack2* with $60\,\mathrm{Hz}$. The marker frame is finally calibrated w. r. t. the center of geometry of the rover, such that the tracking system can be used as ground truth position and configuration measurement for the platform.

Following the autonomy paradigm of planetary exploration missions, all relevant software components are designed to run on the rover's on-board computers. This also includes learning-based controller modules. However, no online learning is envisaged on board. The software on the rover is split into the low-level controllers running on a realtime computer with $1\,\mathrm{kHz}$, which is interfaced by the high-level controllers

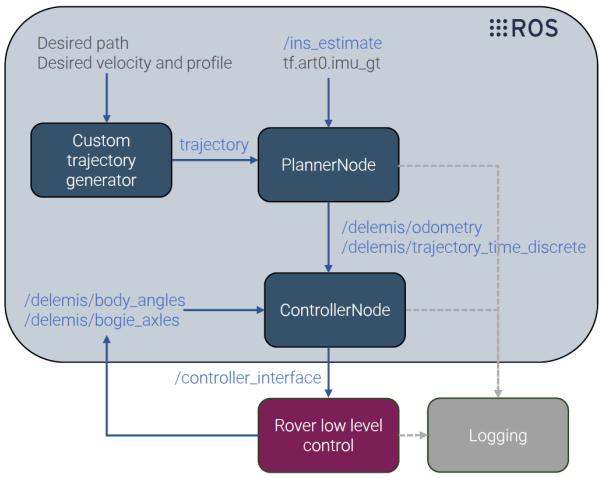


Fig. 5: Overview of the DeLeMIS nodes and their interfaces. The planner node receives a custom generated trajectory as well as input the rover's absolute position from the tracking system as ROS tf (`tf.art0.imu_gt`) and a velocity signal from the self-localization (`/ins_estimate`). From this input, a combined odometry message as well as a discretized trajectory topic is sent to the controller node. The controller node receives telemetry data from the low-level controller (`/body_angles, /bogie_axles`) and sends a desired body velocity command via the `/controller_interface` topic.

and the self-localization and mapping pipeline running on the main computer. The high-level controller software which was tested in this work has interfaces to several parts of the rover software stack, cf. Fig. 5.

On the rover, all processes, their dependencies, and real-time process communication is handled by the middleware *Links and Nodes*[1] (LN), which was developed at DLR to create and manage flexible distributed realtime systems, in particular embedded robotic systems. The higher-level software components are using ROS melodic as middleware, with custom interfaces to LN. Measurement data documenting the test runs is logged via LN topics, which guarantee logging of realtime and non-realtime topics in different sample rates together with their respective timestamps.

The test runs were performed according to the following procedure: After preparing the surface of PEL and starting the tracking system and rover software stack, a trajectory was selected and the planner node was started. Then, the selected high-level controller node was started together with the logging process. After completing the specified amount of laps, a postprocessing pipeline was triggered containing (among others) automated plotting of relevant data. Optionally, a training of the neural network was launched.

IV. ROVER CONTROL

In the following, the controllers used during the test campaign will be described.

[1] `https://gitlab.com/links_and_nodes/links_and_nodes` [Accessed 09.02.2024]

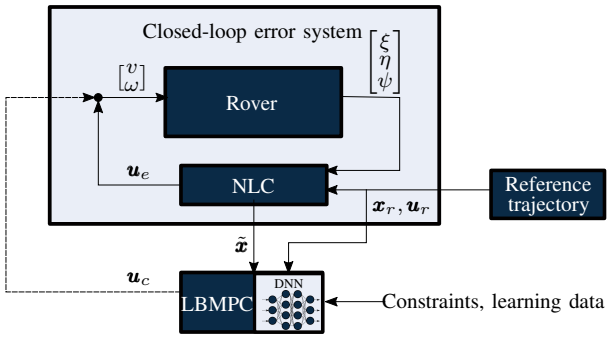


Fig. 6: Concept overview of trajectory tracking controller architecture [24]. The NLC provides the control input $\boldsymbol{u}_e$ consisting of the commanded velocity v and yaw rate ω to the rover system and receives the rover state $\boldsymbol{x} = [\xi\ \eta\ \psi]^T$ and the reference trajectory including the reference states $\boldsymbol{x}_r$ and velocities $\boldsymbol{u}_r$. The optional learning-based MPC provides the additional control input $\boldsymbol{u}_c$ and receives the error state $\tilde{\boldsymbol{x}} = [\tilde{\xi}\ \tilde{\eta}\ \tilde{\psi}]^T$, constraints and learning data if necessary.

A. Low-level control

The mobile platform features four wheels with individual steering and propulsion. This creates the possibility to change the location of the instantaneous center of rotation (ICR) arbitrarily, i. e. the rover is able to turn on the spot as well as perform car-like as well as crab-like motions according to the situation. Therefore, the steering configuration as well as the velocity of all wheels have to be controlled such that the nonholonomic rolling-without-slipping constraints of the wheels are satisfied and a unique ICR exists. In this work, a desired velocity for the geometric center of the rover is commanded in three planar directions (translational velocity in local x- and y-direction and yaw rate) from the high-level software components. This body velocity command is then used to compute the corresponding desired velocities and steering angles for the individual wheels by solving the rigid body kinematics equations. Underlying motor controllers follow the desired steering positions and wheel velocities, respecting the hardware limits of the actuators as well as the integrity of the steering configuration. Experience has shown that the performance of the underlying controllers is sufficiently fast for most trajectories, such that this simple control strategy is robust and well-suited for motions in rough terrain. However, the control strategy does not include knowledge about wheel-ground interaction. Thus, higher-level trajectory tracking algorithms building on the body-velocity command interface have to compensate for slip effects in order to improve tracking performance.

B. Trajectory tracking control

As an approach for the high-level trajectory tracking, a controller architecture with the combination of a non-linear Lyapunov controller (NLC) and a layered model predictive controller is used. Details of the overall controller architecture are described in [24] and illustrated in Fig. 6. The rover state $\boldsymbol{x} = [\xi\ \eta\ \psi]^T$ consists of the current longitudinal and lateral position ξ and η, as well as the heading angle ψ,

expressed in a fixed initial frame. The reference state from the provided trajectory is denoted by $\boldsymbol{x}_r$. The error state $\tilde{\boldsymbol{x}}$ is given as

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{\xi} \\ \tilde{\eta} \\ \tilde{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} (\boldsymbol{x} - \boldsymbol{x}_r) \ . \tag{1}$$

The NLC provides the control input $\boldsymbol{u}_e$ consisting of the commanded velocity v_e and yaw rate ω_e:

$$\begin{aligned} v_e &= -K_\xi \, \tilde{\xi} + v_r \, \cos\tilde{\psi} \\ \omega_e &= -K_\psi \, \sin(\tilde{\psi}) + \omega_r - K_\eta \, v_r \, \tilde{\eta} \ . \end{aligned} \tag{2}$$

Thereby, v_r and ω_r denote reference velocity and yaw rate of the target trajectory, and K_ξ, K_η, and K_ψ are scalar design parameters. By using the proposed non-linear controller (2) depending on the error state $\tilde{\boldsymbol{x}}$ from (1), the system can be linearized at the trajectory. From the perspective of the overlaid MPC, a linear parameter-varying system (LPV) is obtained:

$$\dot{\tilde{\boldsymbol{x}}} = \begin{bmatrix} -K_\xi & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & -v_r K_\eta & -v_r K_\psi \end{bmatrix} \tilde{\boldsymbol{x}} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}_c \ . \tag{3}$$

This design makes it possible to follow a trajectory with only the NLC, but offers the possibility to improve the motion behavior with an additional controller input $\boldsymbol{u}_c$. Specifically, MPC can be used to consider constraints, for example on the overall system inputs $\boldsymbol{u} = \boldsymbol{u}_e + \boldsymbol{u}_c$. Because of the simplicity of (3), the optimization problem remains simple and fast to solve. Details about the MPC including the formulation of the optimization problem can be found in [24].

C. Learning-based control

Due to the linearization of the system and unknowns such as slip effects, model uncertainties have to be considered. Therefore, a learning-based component using a *deep neural network* (DNN) is added to the MPC to compensate for the expected model error. In the proposed concept, state prediction over the prediction horizon is approximated using the DNN, based on various input features $\boldsymbol{\gamma}_k$. This extends the LPV system dynamic from (3) with an error prediction to

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_d \, \tilde{\boldsymbol{x}}_k + \boldsymbol{B}_d \, \boldsymbol{u}_{c,k} + NN(\boldsymbol{\gamma}_k) \ , \tag{4}$$

with the discretized system matrices $\boldsymbol{A}_d \in \mathbb{R}^{3\times3}$ and $\boldsymbol{B}_d \in \mathbb{R}^{3\times2}$. It is important to note that the MPC state prediction, including the error prediction by the DNN, is applied and optimized over the entire prediction horizon. This means that the DNN is directly integrated into the MPC problem and the associated consideration of the constraints. For this purpose, various DNN architectures were tested. A *Long Short Term Memory* LSTM network with 4 hidden layers and an average of 25 neurons per layer turned out to be particularly suitable (the exact architecture and results will be published in a further work). In this work, the focus lies on testing the improvement effects and comparing them with the classic control approaches. Thereby, the respective algorithms are first verified in simulation. Then, test results from the real rover are compared with the results from the benchmark test. In the next section, the first test results of the classical controllers are shown and compared, followed by a preview of the results of the learning-based methods.

V. TEST RESULTS AND FINDINGS

At the beginning, a test cycle is presented in detail for reference. The data includes the commanded and measured velocities, the global positions, and the deviation from the trajectory. Afterwards, individual test cases and configurations are compared with regard to their deviation from the specified trajectory. In all experiments, parameters of the NLC controller are chosen according to the stability design guidelines referenced in [24]. The obtained values are $K_\xi = 0.044731$, $K_\eta = 3.5293$, and $K_\psi = 1.2583$.

Fig. 7 shows a test ride of a deformed (drawn) eight with the presented MPC without constraints. The two upper left plots show the commanded velocities v_x and v_y of the high-level controller (NLC or MPC), the commanded velocities of the low-level controller to the wheels and the estimated velocities from the rover odometry. The plots verify the correct implementation and function of the software modules in the overall system. It can be seen that the rover is able to follow the specified trajectory. The maximum lateral deviation is $0.4\,\mathrm{m}$ at the starting position, which can be explained with the time delay between starting the trajectory generator and the controller node. In the further course of the run, the maximum tracking error is $0.11\,\mathrm{m}$. It must be highlighted that the commanded velocities cannot always be properly implemented by the low-level controller, as the commanded wheel velocities are set to zero when the steering configuration is not consistent. This can be observed in the LRU odometry measurements in the upper left plot.

In Fig. 8, two test drives under identical conditions are compared. In both cases, the same trajectory with a constant speed of $0.3\,\mathrm{m/s}$ is commanded. The slope of the rear part of the testbed is $15\,\mathrm{deg}$, a minimum of 3 laps are completed. Fig. 8 (a) depicts a ride with the presented NLC, while Fig. 8 (b) represents the ride with the MPC. The x- and y-directions and deviations refer to the global frame of the tracking system and planner. Under the assumption that the respective coordinate systems have the same origin, they are equal to ξ and η. It is evident from the comparison that both the deviation in the x- and y-directions from the reference trajectory is significantly improved by using the MPC. The improvement in the tracking performance is particularly noticeable during the incline, see Fig. 8 (a) starting from $60\,\mathrm{s}$ and from $120\,\mathrm{s}$. In the test ride with the MPC, the deviation is significant only at the beginning of the test due to the specified reference point on the trajectory, but it reduces to a maximum of $0.083\,\mathrm{m}$ in the y-direction and $0.28\,\mathrm{m}$ in the x-direction in the following laps. With the NLC, there is a maximum lateral deviation of $0.43\,\mathrm{m}$ and $1.19\,\mathrm{m}$ in the x-direction during the run. Note that it is not appropriate to compare the high deviation at the beginning of a run. The different initial errors originate in the not exact

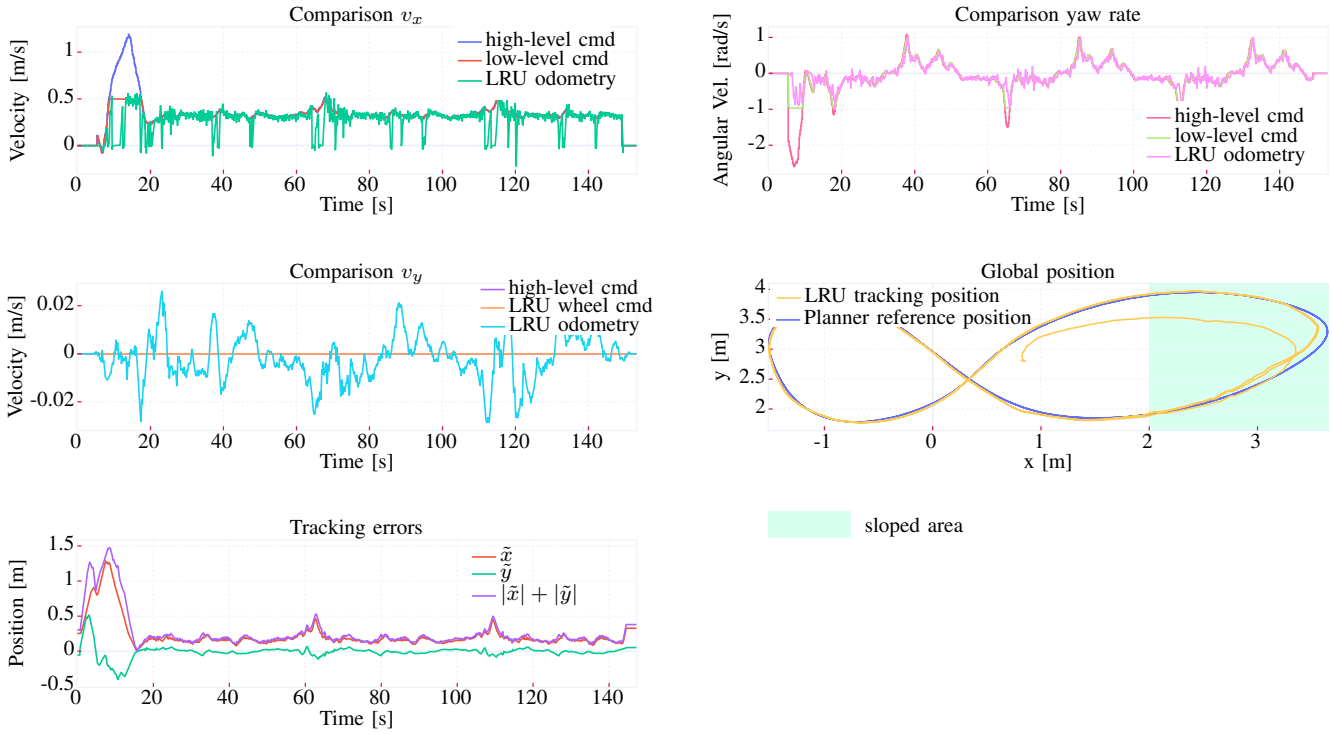


Fig. 7: Overview of the test data recorded in the PEL with an example trajectory. With the commanded high-level velocities from the MPC (blue, pink and purple), the commanded velocities of the low-level controller to the wheels (red, green and orange) and the estimate odometry from the rover (green, 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.

same start positions of the rover and the different start times w. r. t. the reference trajectory during the test runs.

The results indicate that the NLC is capable of following the trajectory well but encounters issues, particularly with higher deviations and inclines. On the other hand, the MPC performs well and reduces lateral deviation to a few centimeters. However, there is still potential for improvement in the MPC, especially in predicting the error in the rover's direction of travel. Here, the approach described in Sec. IV-C is expected to achieve an improvement.

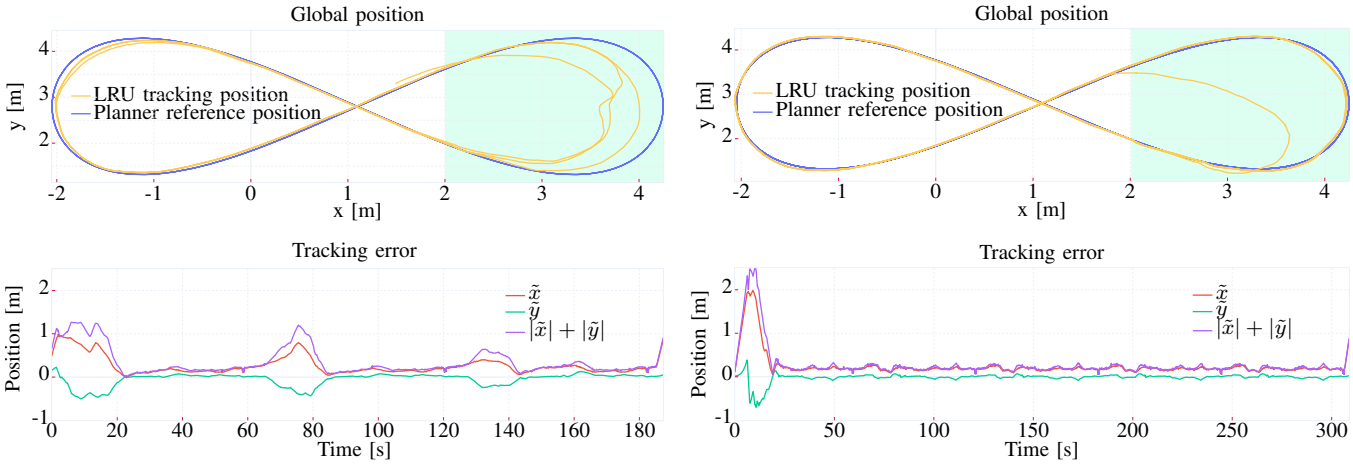
In Fig. 9, some initial results of the learning-based error prediction during an open-loop test drive with the real rover in the PEL are presented[2]. It can be observed that the DNN is partially able to predict the errors of the internal MPC states $\tilde{x}$. In addition, it can be seen that all major errors are predicted correctly in both the x- and y-directions. However, the magnitude of the error is not yet exactly accurate and should be further improved in the future, e.g. through more training and a better DNN architecture. We want to highlight that only the data obtained from approx. 3 laps with the rover in the PEL was used as training data for the model, the size of the dataset was 4 MB. The offline training of the DNN lasted only around 3 minutes on the Rover hardware. A more detailed analysis and results of the learning-based approach, especially of the closed loop results and behavior at unknown and not trained trajectories, will be provided in a subsequent work.

[2]In this context, open loop means that the neural network performs predictions during the run, but these predictions are not fed back to the control loop.

VI. DISCUSSION

The goal of this work is the implementation and examination of different control approaches for a wheeled mobile robot in the context of a planetary exploration mission. However, the experiments are conducted in a terrestrial setup, which poses the question of relevance of the results w. r. t. the goal scenario.

The first relevant aspect is the test site and the rover hardware. As mentioned before, the soil is a certified Moon-analog, while the variable slope and the possibility to shape the surface in PEL manually create the opportunity to test wheel-ground interaction effects in a realistic setting. Even more realistic results are expected to be obtained in future experiments using the new DLR Moon-Mars Test Site, which was inaugurated in Sept. 2023 [25]. The LRU is a terrestrial prototype of a planetary exploration rover, and many components are not space-qualified. However, the drive trains for wheel propulsion and steering as well as the design of the wheel geometry is strongly based on existing planetary exploration missions [26]. Concerning the software, it must be mentioned that the rover's on-board computers possess significantly more computational power



(a) NLC (b) MPC

Fig. 8: Comparison of the tracking performance of NLC and MPC at a 15 deg slope. The upper plots show the reference path provided from the planner (blue) and the measured path of the rover by the tracking system (yellow). The sloped area is highlighted in light green. The lower plots show 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.

than existing solutions in real space missions, and thus the requirements for the software are different. On the other hand, time is not a crucial factor in real space mission (whereas the dangers arising from undesired slip effects might be), such that it should be possible to run selected software components in reduced rates. We believe that the overall software architecture is generally feasible for space vehicles due to the fact that all relevant software components run on the rover directly, besides the training of the neural network. However, this is part of future work and should also be possible on-board, given the fact that the training does not have to run at a specified time or in a specified rate.

Another difference w. r. t. real space applications is the usage of a tracking system for determination of the rover's position and orientation within the planner and controller software. This choice was made for two reasons: First, it enables analyzing the results only w. r. t. the performance of the control software, without interfering with a potentially drifting self-localization. Second, in the confined space of the PEL, drift of the self-localization could easily lead to aborted test runs, as the rover hits the boundary of the sand box. Both factors are irrelevant during a space mission, where the rover navigates locally, avoiding obstacles in its surrounding. If a ground-truth measurement is needed, e. g. for initial training while filtering out drift effects of the pose estimation, it could be possible to measure the rover pose exactly by referring to the known position of the lander in a real mission. The other assumption that was made was that the desired trajectory is feasible without obstacles. We want to point out that we decided to use a static trajectory planner in the test setup. This choice together with the feasibility assumption guarantees that the obtained data for different controllers driving the same trajectory can be compared in a meaningful way. However, in a real mission, trajectories are generated piecewise by a local trajectory planner, avoiding impassable obstacles.

Finally, it is a well-known fact that wheel-ground interaction effects depend not only on terrain properties, but also on contact pressure and geometry of the wheels. Thus, model-based approaches are either inherently inaccurate or computationally expensive. We believe that approaches which rely on machine learning techniques are more likely to be applicable under previously unknown circumstances as model-based approaches. The preliminary results shown in this work justify the approach w. r. t. future space applications.

VII. CONCLUSIONS

In this work, we propose a test setup which allows the qualitative and quantitative comparison of the performance of tracking controllers in a Moon-analog setting. A test campaign is conducted which examines among others the possibility to estimate and predict effects of unmodeled wheel-ground interaction by using machine learning techniques. First results of the experiments imply that the proposed controller setup, which is extended by a deep neural network component for slip estimation and compensation, is well-suited for trajectory tacking of wheeled vehicles in previously unknown terrain. An extensive review of further results will be subject of a follow-up publication.

ACKNOWLEDGMENT

This work was funded from the ESA/ESTEC project DeLeMIS (ESA 4000136972/21/NL/AS).

REFERENCES

[1] M. Bajracharya, M. W. Maimone, and D. Helmick, "Autonomy for mars rovers: Past, present, and future," *Computer*, vol. 41, no. 12, pp. 44–50, Dec 2008.

[2] J. P. Grotzinger, J. Crisp, A. R. Vasavada *et al.*, "Mars Science Laboratory Mission and Science Investigation," *Space Science Reviews*, vol. 170, no. 1, pp. 5–56, Sep. 2012.

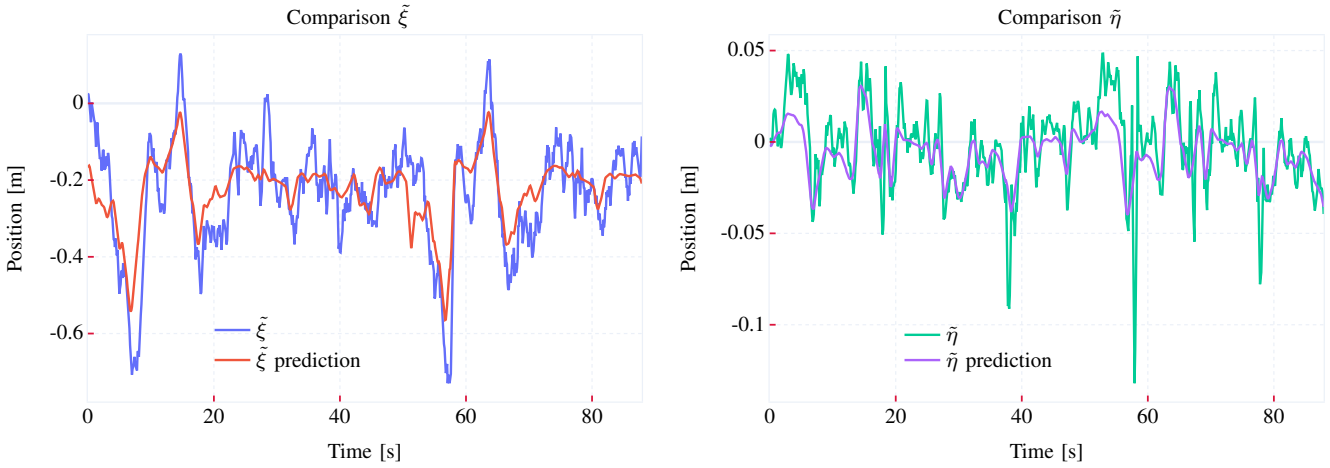


Fig. 9: Prediction of the MPC error state $\tilde{\boldsymbol{x}}$. The prediction is computed by the DNN for the next step in the prediction horizon. The prediction error of the MPC $\tilde{\xi}$ (red) and the measured error $\tilde{\xi}$ (blue) are shown on the left side. The right plot shows the prediction error of the MPC $\tilde{\eta}$ (green) and the measured error $\tilde{\eta}$ (purple).

[3] Mars Communications Team at NASA's Jet Propulsion Laboratory, 2009, https://mars.nasa.gov/mer/mission/rover-status/spirit/2009/all [Accessed: 09.02.2024].

[4] B. H. Wilcox, D. B. Gennery, and A. H. Mishkin, "Mars Rover Local Navigation And Hazard Avoidance," in *1988 Robotics Conferences*, W. J. Wolfe, Ed., Boston, MA, Mar. 1989, p. 72.

[5] A. J. Spiessbach, "Hazard Avoidance For A Mars Rover," in *Mobile Robots III*, vol. 1007. SPIE, Mar. 1989, pp. 77–84.

[6] M. Heverly, J. Matthews, J. Lin, D. Fuller, M. Maimone, J. Biesiadecki, and J. Leichty, "Traverse Performance Characterization for the Mars Science Laboratory Rover," *Journal of Field Robotics*, vol. 30, no. 6, pp. 835–846, 2013.

[7] R. Gonzalez and K. Iagnemma, "Slippage estimation and compensation for planetary exploration rovers. State of the art and future challenges," *Journal of Field Robotics*, vol. 35, no. 4, pp. 564–577, Jun. 2018.

[8] R. Haarmann, Q. Mühlbauer, L. Richter *et al.*, "Mobile Payload Element (MPE): concept study of a small, outonomous and innovative sample fetching rover," in *Proceedings of the 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2013.

[9] M. J. Schuster, C. Brand, S. G. Brunner *et al.*, "The LRU Rover for Autonomous Planetary Exploration and Its Success in the SpaceBotCamp Challenge," in *Proceedings of the International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Bragança, Portugal: IEEE, May 2016, pp. 7–14.

[10] M. J. Schuster, S. G. Brunner, K. Bussmann *et al.*, "Towards Autonomous Planetary Exploration: The Lightweight Rover Unit (LRU), its Success in the SpaceBotCamp Challenge, and Beyond," *Journal of Intelligent & Robotic Systems (JINT)*, Nov. 2017.

[11] A. Wedler, M. Vayugundla, H. Lehner *et al.*, "First Results of the ROBEX Analogue Mission Campaign: Robotic Deployment of Seismic Networks for Future Lunar Missions," in *68th International Astronautical Congress (IAC)*. Adelaide, Australia: International Astronautical Federation (IAF), Sep. 2017.

[12] A. Wedler, K. Bussmann, A. Dömel *et al.*, "From the ROBEX experiment toward the Robotic Deployment and Maintenance of Scientific Infrastructure for future Planetary Exploration Missions," in *42nd COSPAR Scientific Assembly*, vol. 42, Pasadena, California, Jul. 2018.

[13] A. Wedler, M. G. Müller, M. Schuster *et al.*, "Finally! Insights into the ARCHES Lunar Planetary Exploration Analogue Campaign on Etna in summer 2022," in *73rd International Astronautical Congress (IAC)*. Paris, France: International Astronautical Federation (IAF), Sep. 2022.

[14] L. Burkhard, R. Sakagami, K. Lakatos, H. Gmeiner, P. Lehner, J. Reill, M. G. Müller, M. Durner, and A. Wedler, "Collaborative Multi-Rover Crater Exploration: Concept and Results from the ARCHES Analog Mission," in *2024 IEEE Aerospace Conference*, Big Sky, Montana, USA, in press.

[15] L. Preston, M. Grady, and S. Barber, *TN2: The Catalogue of Planetary Analogues*, Dec. 2012, published: The Planetary and Space Sciences Research Institute, The Open University, UK.

[16] K. Bussmann, L. Meyer, F. Steidle, and A. Wedler, "Slip modeling and estimation for a planetary exploration rover: Experimental results from mt. Etna," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 2449–2456.

[17] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[18] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?" *Proceedings of the American Control Conference*, vol. 2022-June, no. 1, pp. 342–357, 2022.

[19] J. Wang, M. T. H. Fader, and J. A. Marshall, "Learning-based model predictive control for improved mobile robot path following using Gaussian processes and feedback linearization," *Journal of Field Robotics*, vol. 40, no. 5, pp. 1014–1033, 2023.

[20] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[21] U. Rosolia and F. Borrelli, "Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[22] N. Baldauf, T. Lubiniecki, A. Turnwald, K. Lakatos, and N. Panagiotopoulos, "Learning-based motion control of a rover on unknown ground," in *12th International Conference on Guidance, Navigation & Control Systems (GNC)*. ESA, Jul. 2023.

[23] V. S. Engelschiøn, S. R. Eriksson, A. Cowley, M. Fateri, A. Meurisse, U. Kueppers, and M. Sperl, "EAC-1A: A novel large-volume lunar regolith simulant," *Scientific Reports*, vol. 10, no. 1, p. 5473, Mar. 2020.

[24] N. Baldauf and A. Turnwald, "Iterative learning-based model predictive control for mobile robots in space applications," in *27th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2023, pp. 434–439.

[25] M. Görner, A. Dömel, M. Durner *et al.*, "The DLR Moon-Mars Test Site for Robotic Planetary Exploration," in *International Conference on Space Robotics (iSpaRo24)*, Jun. 2024, accepted for publication.

[26] A. Wedler, B. Rebele, J. Reill, M. Suppa, H. Hirschmüller, C. Brand, M. Schuster, B. Vodermayer, H. Gmeiner, A. Maier *et al.*, "LRU-lightweight rover unit," in *Proceedings of the 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2015.