RLROVERLAB: An Advanced Reinforcement Learning Suite for Planetary Rover Simulation and Training

Anton Bjørndahl Mortensen[1] and Simon Bøgh[2]

***Abstract*— This paper presents RLROVERLAB, an open-source reinforcement learning suite tailored for simulating planetary rovers on extraterrestrial bodies. The suite features a set of space related assets and tasks implemented using the Nvidia ORBIT framework, backed by a robust physics and graphics engine in Nvidia Omniverse. The suite aims to bridge the gap between space robotics and reinforcement learning, offering researchers a suite of assets, predefined tasks, and a versatile platform for developing, testing and benchmarking novel reinforcement learning algorithms. Consequently, the suite is designed to be flexible and modular, allowing for easy integration of new assets and tasks. Through exemplary use cases and tasks, we demonstrate RLROVERLAB's potential to advance RL applications in space exploration. Videos, documentation, and code available is at `https://github.com/abmoRobotics/isaac_rover_orbit`**

I. INTRODUCTION

The aspiration to become a multi-planetary species has led to a significant focus on exploring celestial bodies such as the Moon and Mars. Autonomous robots play a pivotal role in these endeavors, to explore and prepare for human arrival. However, controlling robots on other planetary bodies is challenging due to long communication delays, often spanning several minutes between Earth and the robot. This necessitates a high degree of autonomy in robot systems, as highlighted by [1], who present the Lightweight Rover Unit (LRU) designed for the challenges of planetary exploration, emphasizing the need for high maneuverability and autonomy in rough terrain due to communication delays.

This autonomy is crucial for ensuring safe, reliable, and efficient operations without direct human intervention. One promising approach to this challenge is leveraging reinforcement learning (RL) to enable robots to autonomously execute complex tasks. However, the impracticality of deploying robots to extraterrestrial environments for training purposes makes real-world RL training infeasible. As an alternative, high-fidelity simulations offer a viable platform for training RL agents, allowing the transfer of learned policies to real-world scenarios effectively.

In this paper, we introduce an open-source suite of RL environments and assets designed to simulate a variety of tasks for planetary robots. Building upon the modular and flexible architecture of the Nvidia ORBIT framework [2], our suite adapts and extends its capabilities to the specific needs of space rover simulations. By leveraging ORBIT's foundation, we have crafted environments that captures unique challenges of lunar and Martian terrains. These environments not only facilitate the development and testing of autonomous navigation strategies in planetary space scenarios but also significantly reduce the setup time and effort typically required to design these RL-enabled environments. In our framework we support the following RL training libraries through ORBIT: SKRL [3], RL-Games [4], RSL-RL [5], and Stable-Baselines3 [6].

Our suite addresses gaps in current autonomous space rover research, such as the absence of standardized benchmark tasks and the challenge of comparing results across different methods for autonomous robots in space exploration. By offering a diverse set of simulation environments and a benchmark tasks for evaluating various RL algorithms, we contribute to a more connected and consistent approach in the field. This initiative aims to streamline efforts in autonomous space exploration, encouraging more rapid and efficient advancements in this crucial area of robotics.

The main contributions of this paper are listed below.

1) A suite of pre-defined RL environments for planetary robots on the Moon and Mars.
2) A set of pre-trained RL policies for each task, these can be used as a baseline for future work, or as a starting point for transfer learning or sim2real.
3) Supports multiple RL techniques: record state-transitions to an offline dataset, transfer learning, teacher-student RL, Curriculum Learning.
4) A number of rover models: AAU Rover (complex), a 6-wheeled, 4-wheeled steering rover with a complex rocker-boogie suspension akin to NASA perseverance. AAU Rover (simple), a simplified variant without the rocker-boogie suspension for faster simulation. ESA ExoMy, an open-source, 6-wheeled rover that can be 3D printed and built at low cost.

The remainder of this paper is structured as follows. Section II presents related work. Section III provides an overview of the RL suite. Based on the overview, Section V presents the experiments and benchmarks performed in this paper. Lastly Section VI discusses the results and Section VII presents the future work planned for RLROVERLAB.

II. RELATED WORK

The development of simulators for space robotics has been pivotal in advancing our capabilities for extraterrestrial exploration. Simulators offer valuable platforms for testing and validation of robotic systems designed for operations

*This work was not supported by any organization

[1]Anton Bjørndahl Mortensen is with the Department of Materials and Production, Aalborg University, Denmark antonbm2008@gmail.com

[2]Simon Bøgh is with the Department of Materials and Production, Head of AAU Space Robotics, Aalborg University, Denmark sb@mp.aau.dk

on other planetary bodies. One notable example is NASA's ROAMS (Rover analysis modeling and simulation), which is a planetary rover simulation software package [7]. Despite its comprehensive modeling capabilities, ROAMS primarily focuses on Mars rover operations, with limited support for the broad range of planetary bodies and tasks that modern space missions might target.

Nvidia ORBIT provides a unified and modular framework for robot learning powered by Nvidia Isaac Sim [2]. It provides access to the advanced GPU-enabled PhysX SDK [8], and excels over prior frameworks like [9], [10] that uses CPU-based physics engines like MuJoCo [11], Bullet [12] or Gazebo [13]. RLROVERLAB utilizes the APIs provided by ORBIT, which accommodates a broad spectrum of sensor types, and adopts the many utilities from Nvidia Isaac Sim such as domain randomization, ROS support [14], and photo-realistic rendering. RLROVERLAB builds on top of Nvidia ORBIT to leverage many of the advantages of the framework for accurate and efficient reinforcement learning policy training. OmniLRS [15] focuses on improving the photorealism of simulated planetary space environments in Isaac Sim. To further support highly realistic synthetic lunar landscapes and textures our work could naturally extend OmniLRS for even more realistic training environments for RL-based learning for our crafted benchmark tasks.

Reinforcement Learning (RL) applications in space robotics have begun to showcase their potential in addressing complex, dynamic, and uncertain environments, essential for tasks such as extraterrestrial exploration and manipulation. Furthermore, planetary rovers face difficult path-planning problems. An approach to these path-planning problems based on inverse reinforcement learning (IRL) using deep convolutional networks and value iteration networks is presented in [16]. They demonstrate the effectiveness of the method in a grid world dataset as well as a realistic synthetic dataset generated from currently deployed rover mission planning tools and real Mars imagery. Other work demonstrates that a FPGA-based Q-learning architecture can significantly reduce processing time and power consumption for autonomous control systems onboard space hardware, making it suitable for planetary robotics [17]. The study by [18] explores the use of deep reinforcement learning for automating the process of robotic grasping in lunar environments. They demonstrate the effectiveness of 3D data over traditional 2D images for training robotic agents in complex tasks, such as object grasping under varied lunar conditions, achieving promising results in sim-to-real transfer applications. In [19] the authors present an approach to deep-space mission challenges through the deployment of swarm robots utilizing reinforcement learning for collaborative exploration. They demonstrate the efficiency of the multi-agent deep deterministic policy gradient (MADDPG) algorithm in training swarm robots for complex tasks on Mars.

The diversity and complexity of tasks in space robotics, ranging from autonomous navigation without maps, precise rock grasping, to swarms and building habitats, underscore the critical need for specialized simulation tools capable of rapidly training robust reinforcement learning policies, a gap RLROVERLAB is specifically designed to fill.

III. RLROVERLAB IMPLEMENTATION & FEATURES

While the ORBIT framework is designed to be modular and flexible, it remains a time consuming process to create terrain assets, robots, and create tasks to get started. Consequently, we provide a suite of environments that can be used to train RL agents to perform a number of planetary tasks. These environments are implemented in such a way that different robots, terrains and objectives can be easily exchanged and extended for future work. Furthermore, we provide a set of benchmarks for different tasks, which can be used to compare the performance of different RL algorithms. Lastly, we provide a set of pre-trained RL policies for each of the tasks, which can be used as a baseline for future work. This section presents the different terrains, robots, steering modes, tasks and benchmarks included in the RL suite.

a) Terrains In order to train and test different RL algorithms, we provide a number of terrains of varying difficulty. The terrains are seperated in three layers, where the first layer is the terrain surface with triangle collsion mesh, the second layer contains all the obstacles and are used to detect collision between the robot and the obstacles. The last layer is a merged version of the first two layers, which is used for height scanning[1]. Following this structure we provide the following premade terrain types:

1) **Flat Terrain** - The flat terrain is a simple flat surface with no obstacles, that serves as a baseline to test RL algorithms.
2) **Hill Terrain** - The hills terrain contains procedurally generated hills of varying heights and small terrain undulations. This terrain is designed to test the ability of RL algorithms to traverse uneven terrain using a height scanner and / or a camera.
3) **Obstacle Terrain** - The obstacle terrain is layered on top of the two aforementioned terrains and contains a number of obstacles that are procedurally placed on the terrain. This terrain is designed to test the ability of RL algorithms to navigate around obstacles. An example terrain is visualized in Figure 1.
4) **Maze Terrain** - This terrain type contains rocks and boulders placed in a maze-like pattern this is a challenging terrain type for POMDPs as memory or planning is required to navigate the maze.

Following the previously mentioned terrain structure we also allow other researchers to create custom terrains and integrate them into the RL suite.

b) Robots In order to train agents we provide a number of robots, that can be used to train and test RL algorithms. We provide the following robots:

1) **AAU Rover Complex** - This rover is a 6-wheeled rover with a 4-wheel steering system with a footprint of approx 1m, that features a rocker-boogie suspension

[1]ORBIT only supports one mesh for height scanning

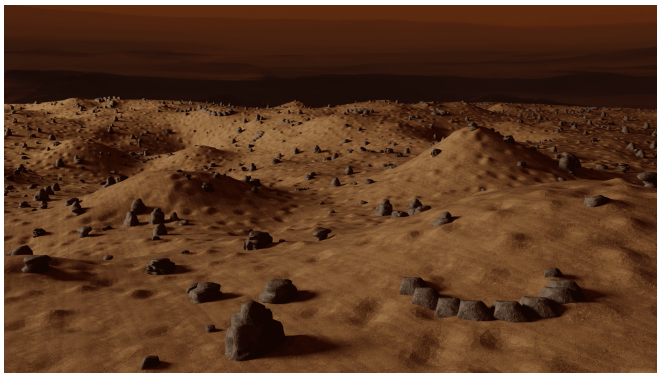

Fig. 1: Example of hilly terrain with obstacles.

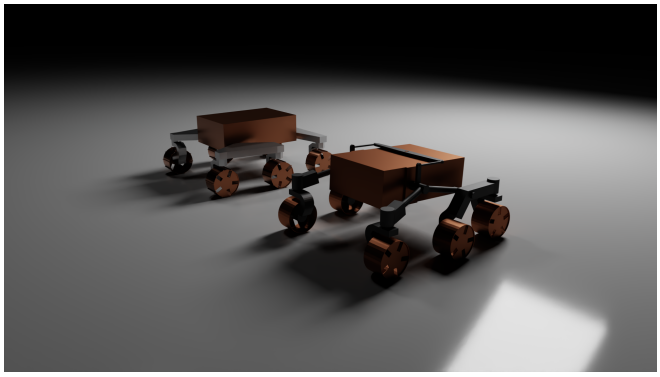

Fig. 2: Rovers: AAU Rover Complex, AAU Rover Simple

system. Due to the rocker-boogie suspension system, the computational cost of simulating this rover is high compared to non closed-chain rovers.

2) **AAU Rover Simple** - This rover is a simplified version of the AAU Rover Complex with the same dimensions and wheel configuration, but without the rocker-boogie suspensions system, thus increasing the simulation speed of this rover significantly.
3) **ExoMy** - We provide an integration of the ExoMy [20] rover in ORBIT. ExoMy is an open-source 6-wheeled rover with a 6-wheel steering system, that can be built at home using 3D printed parts.

The two AAU rovers are visualized in Figure 2. We plan to add more robots in upcoming releases. We also welcome others to integrate their robots into RLROVERLAB.

c) Steering Configurations Learning to control a robot using direct mapping of actions to the respective joints can be very challenging. Therefore, we provide a number of steering modes implemented in ORBIT, that can be used to simplifiy the learning process. These steering modes processes the actions provided by the RL agent and converts them into the respective joint positions and velocities. All the provided steering modes can be parameterized, such that integration of custom robots is possible by providing a number of parameters. We provide the following steering modes:

1) **Ackermann Steering** - The ackermann steering mode, is implemented with a maximum steering angle. When this angle is reached, the rover will turn in place. This steering mode is available for all the rovers.
2) **Skid Steering** - We provide a skid steering mode, that is available for all the rovers.
3) **Crab Steering** - This steering mode allows the rover to move in any direction, and is only available for the ExoMy rover, since this mode requires a 6-wheel steering system.
4) **Direct Mapping** - We also provide a steering mode that maps the actions of the RL agent directly to the joints of the robot.

d) Tasks/Environments Using the presented terrains, robots, and steering modes, we provide a number of basic tasks that can be used to train and test different RL algorithms. The tasks are implemented using the ORBIT framework. ORBIT uses Isaac Sim to simulate the physics environment, and implements tasks as OpenAI/Farama-Foundation Gym environments. Consequently, the tasks can be used with any RL framework that is compatible with the OpenAI Gym API. The examples provided in the RLROVERLAB uses SKRL as the RL framework, and the tasks are implemented in such a way that different terrains, robots and steering modes can be easily exchanged and extended for future work. We provide the following tasks:

1) **Autonomous Navigation** - In this task we provide a skeleton for creating navigation based task. We specifically provide an example where the robot learns to navigate to a target position using a height map, distance and angle to the target. The autonomous navigation task can be adapted to different scenarios by changing the terrain in the configuration file. Furthermore, it is straightforward to change observation space, action space, the neural network model and use different robots.
2) **Grasping** - This task implements learning-based grasping, and we provide sample tubes and rocks, that can be configured for grasping.

e) Data Recording In addition to the tasks, we also provide a data recording system that can be used to record data from the simulation. This is useful for e.g. Offline RL, which has become increasingly popular in recent years. Likewise, the recording system can collect additional sensor information such as camera images, that are not used by the RL agent for training/inference. This data can then be used for e.g. imitation learning, learning by cheating, or training a perception system. The data is stored sequentially once an episode is finished, and is stored as a HDF file. The information stored in the HDF file includes the following: 1) Observations, 2) Actions, 3) Rewards, 4) Dones, 5) Optional extra sensor information.

f) Benchmarks In order to compare the performance of different RL algorithms, hyperparameters, evaluate the performance of different robots, terrains and steering modes, we provide a number of baseline benchmarks and pre-trained policies for each of the tasks. These can be use as a baseline for future work, and to compare performance between different RL algorithms such as PPO, SAC, DDPG,

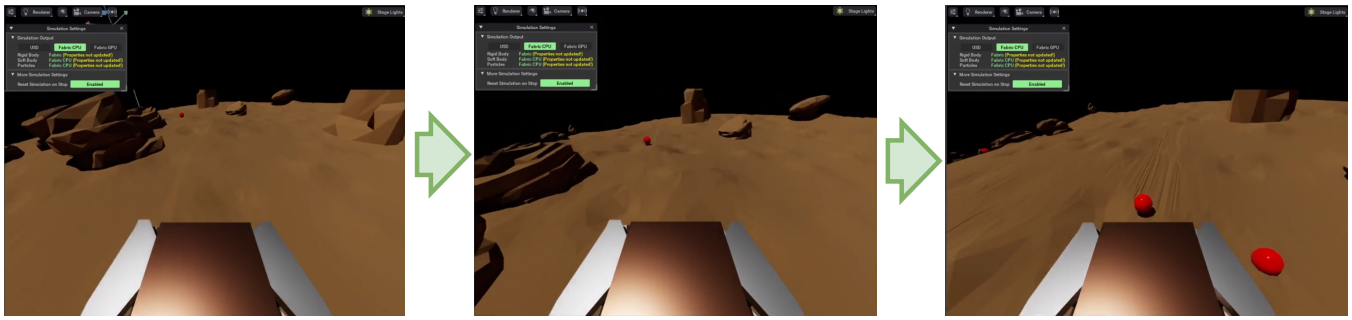

Fig. 3: Example of running inference with a policy using an RGB camera. The policy is trained using a two-stage training process, where the first stage is a pre-trained policy using a heightmap as input, and the second stage is a policy trained using the recorded camera data and imitation learning.

DQN, etc.

IV. Documentation

To ease the use of RLRoverLab, we provide detailed documentation of the suite and its components. The documentation is available as a github wiki page at `https://github.com/abmoRobotics/isaac_rover_orbit/wiki`, and includes the following sections:

- **Installation** - Provides a step-by-step guide on how to install the RL suite, and its dependencies.
- **Task Implementation** - Provides a detailed description of the implementation of the RL agents, using mathematical notations.
- **Examples** - Provides examples of use cases with their respective scripts and description of functionalities.
- **Benchmarking** - Provides benchmarking results of the different tasks, and how to use the benchmarking system.
- **Development** - Provides a guide on how to develop new components or algorithms for the RL suite.

V. Exemplar workflows with RLRoverLab

The presented RL suite is designed to be easy-to-use, modular and extensible. In this section we showcase a subset of use-cases for which the RL suite can be used.

*1) **Configuration of the environments**:* The first use-case we showcase, is the ability to configure the environments, and experiment with different configurations. In Figure 4 we show the learning curves for different observation spaces. The learning curves are generated by training a PPO agent for 50k steps in each environment. In this case the observation space consists of a heightmap of size 3x3m, 5x5m and 6x6m. The results show that using a smaller observation space leads to faster learning.

*2) **Using the pre-trained policies**:* Another application of the suite, is utilizing the pre-trained policies. One of the ways to take advantage of the pre-trained policies is to use them for transfer learning or learning-by-cheating. For instance, consider the scenario where our objective shifts towards employing a RGB camera for the observation space, instead of a heightmap. In this case, we can utilize the heightmap navigation model to drive around in an environment, and record RGB camera data. This data can then be used to train a new model using the recorded RGB camera data, and imitation learning. Three images illustrating a rover navigating using RGB camera trained using this process are shown in Figure 3.

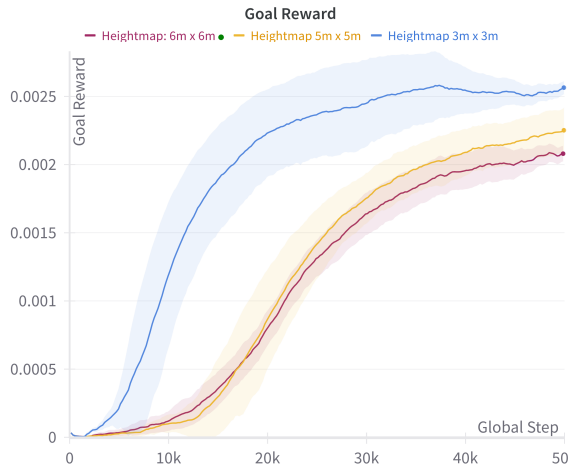

Fig. 4: This graph shows the reward over time for different observation spaces. The observation space consists of heightmaps of size 3x3m, 5x5m and 6x6m.

*3) **Benchmarking**:* Another use-case of the RL suite is to use it for benchmarking. This is relevant when comparing to different algorithms, pre-trained policies and can also be used to compare traditional methods with RL-based methods. An example of this is shown in Figure 5, where Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and Twin Delayed DDPG (TD3) are benchmarked in the context of the navigation task outlined in section III. Each benchmarked algorithm is trained five times and uses the default parameters from SKRL [2], and the training consists of 50,000 timesteps. The results show that PPO learns the task quickly, TRPO aggregates alot of negative reward in the beginning due to oscillations in the

[2]The horizon is modified in order to make sense in the context of the navigation task.

output, and TD3 in unable to learn the correct behaviour with the default parameters.

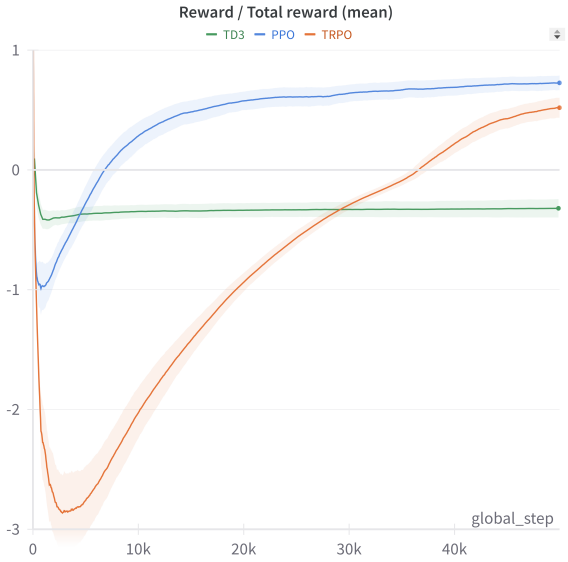

Fig. 5: This figure presents the total mean reward and benchmarks PPO, TRPO, and TD3 within the context of the navigation task outlined in section III. The implementation of the algorithms only modifies the horizon and otherwise uses the default parameters provided by SKRL.

VI. Discussion

In this paper we introduce a suite of space related assets and tasks for reinforcement learning. We showcase the capabilities of the RLRoverLab by providing a set of examples and tasks that demonstrate the potential of the suite. The suite bridges the gap between space robotics and reinforcement learning, providing a platform for researchers to develop and test their algorithms in a space environment. The core strength of RLRoverLab is that it is built on top of the ORBIT framework, utilizing Isaac Sim's high-fidelity physics engine and rendering capabilities. Another core strength is the flexibility and modularity of the suite which allows for easy integration of new space related assets and tasks. By providing a suite for space robotics, we aim to facilitate the development of reinforcement learning algorithms for space related applications. Furthermore, we hope to that the suite will be further developed and extended in the future to support the future of autonomous space exploration.

VII. Future Work

Acknowledging that RLRoverLab is in its early stages, there is a clear pathway to support the future of autonomous space exploration. Most importantly we aim to extend the catalogue of assets and tasks to include a wider range of scenarios, robots, and challenges. This includes adding more complex terrains, more advanced robots, and more challenging tasks. Additionaly, integrating dynamic environment conditions, such as realistic lightning patterns and dust dynamics will be a key feature to add in the future.

References

[1] Martin J Schuster, Sebastian G Brunner, Kristin Bussmann, Stefan Büttner, Andreas Dömel, Matthias Hellerer, Hannah Lehner, Peter Lehner, Oliver Porges, Josef Reill, et al. Towards autonomous planetary exploration: The lightweight rover unit (lru), its success in the spacebotcamp challenge, and beyond. *Journal of Intelligent & Robotic Systems*, 93:461–494, 2019.

[2] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[3] Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Simon Bøgh, and Nestor Arana-Arexolaleiba. skrl: Modular and flexible library for reinforcement learning. *Journal of Machine Learning Research*, 24(254):1–9, 2023.

[4] Denys Makoviichuk and Viktor Makoviychuk. rl-games: A high-performance framework for reinforcement learning. `https://github.com/Denys88/rl_games`, May 2021.

[5] David Hoeller and Nikita Rudin. Rsl rl: Fast and simple implementation of rl algorithms. `https://github.com/leggedrobotics/rsl_rl`.

[6] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[7] Jeng Yen, Abhinandan Jain, and J Balaram. Roams: Rover analysis modeling and simulation. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, page 249, 1999.

[8] Nvidia physx sdk.

[9] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[10] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[11] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[12] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[13] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, volume 3, pages 2149–2154. IEEE, 2004.

[14] Nvidia isaac ros, https://nvidia-isaac-ros.github.io.

[15] Antoine Richard, Junnosuke Kamohara, Kentaro Uno, Shreya Santra, Dave van der Meer, Miguel Olivares-Mendez, and Kazuya Yoshida. Omnilrs: A photorealistic simulator for lunar robotics. *arXiv preprint arXiv:2309.08997*, 2023.

[16] Max Pflueger, Ali Agha, and Gaurav S Sukhatme. Rover-irl: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robotics and Automation Letters*, 4(2):1387–1394, 2019.

[17] Pranay Reddy Gankidi and Jekan Thangavelautham. Fpga architecture for deep learning and its application to planetary robotics. In *2017 IEEE Aerospace Conference*, pages 1–9. IEEE, 2017.

[18] Andrej Orsula, Simon Bøgh, Miguel Olivares-Mendez, and Carol Martinez. Learning to grasp on the moon from 3d octree observations with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[19] Yixin Huang, Shufan Wu, Zhongcheng Mu, Xiangyu Long, Sunhao Chu, and Guohong Zhao. A multi-agent reinforcement learning method for swarm robots in space collaborative exploration. In *2020 6th international conference on control, automation and robotics (ICCAR)*, pages 139–144. IEEE, 2020.

[20] Miro Voellmy and Maximilian Ehrhardt. Exomy: A low cost 3d printed rover. 10 2020.