Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning

Andrej Orsula[1] Simon Bøgh[2] Miguel Olivares-Mendez[1] Carol Martinez[1]

Abstract*— Extraterrestrial rovers with a general-purpose robotic arm have many potential applications in lunar and planetary exploration. Introducing autonomy into such systems is desirable for increasing the time that rovers can spend gathering scientific data and collecting samples. This work investigates the applicability of deep reinforcement learning for vision-based robotic grasping of objects on the Moon. A novel simulation environment with procedurally-generated datasets is created to train agents under challenging conditions in unstructured scenes with uneven terrain and harsh illumination. A model-free off-policy actor-critic algorithm is then employed for end-to-end learning of a policy that directly maps compact octree observations to continuous actions in Cartesian space. Experimental evaluation indicates that 3D data representations enable more effective learning of manipulation skills when compared to traditionally used image-based observations. Domain randomization improves the generalization of learned policies to novel scenes with previously unseen objects and different illumination conditions. To this end, we demonstrate zero-shot sim-to-real transfer by evaluating trained agents on a real robot in a Moon-analogue facility. *The source code and datasets are available at https://github.com/AndrejOrsula/drl_grasping.

I. Introduction

Planetary exploration aims to provide scientific insights to advance our knowledge about other planets, their geology and available resources. Analysis of samples plays a key role in this endeavor, where extraterrestrial robotic systems are instrumental in their acquisition for either in-situ analysis or sample return [1]. Reports of these findings are also necessary for future in-situ resource utilization [2] to enable extraction of Hydrogen and Oxygen for localized production of rocket propellant and operation of life support systems. In this way, the required payload during the initial launch from Earth would be significantly reduced while decreasing the dependency on additional resupply missions. There is an increasing effort put into sample return missions that could provide this data. Lunar material was recently returned to Earth by the Chang'e 5 mission [3], and NASA has selected companies to collect Moon rocks towards the progress of the Artemis program [4]. Mars Sample Return is another proposed mission, where an ESA rover is planned to fetch samples that are being collected by the NASA rover Perseverance [5]. Unfortunately, communication delay causes remote teleoperation to be inefficient, which reduces the amount of scientific data that rovers can gather throughout their mission. Therefore, autonomy for extraterrestrial rovers becomes essential as the complexity of missions increases.

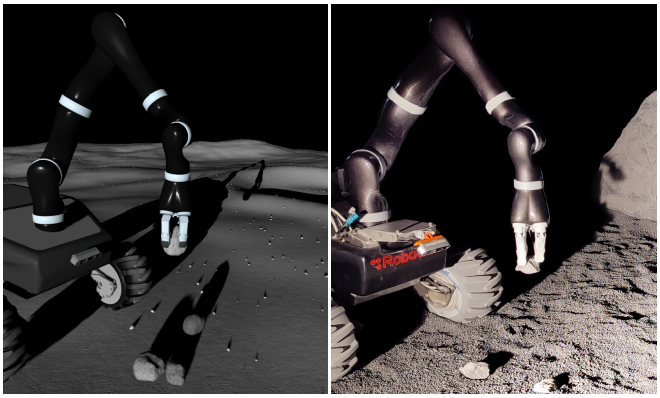

Fig. 1. Our agents learn vision-based robotic grasping under challenging conditions in a heavily randomized simulation environment. Their performance is then evaluated on a real robot inside a Moon-analogue facility.

Rovers equipped with robotic arms have many potential applications in extraterrestrial environments. Besides manipulating science instruments to closely analyze areas of interest, these rovers could also perform assembly and maintenance tasks by interacting with various tools and technical equipment. Many subroutines involved in such tasks require an object or a tool to be firmly grasped prior to performing them. Therefore, robotic grasping is a fundamental skill that is essential for flexible mobile manipulation. In order to achieve this flexibility, rovers are required to grasp a diversity of objects that can differ in their geometry, appearance and mechanical properties.

We present an approach for applying end-to-end deep reinforcement learning to the task of vision-based robotic grasping in lunar environments. The primary focus of this paper is to learn end-to-end policies for robotic grasping under challenging conditions of the Moon, i.e. unstructured scenes with uneven terrain, diverse rocks and harsh illumination. As the training of agents directly in extraterrestrial conditions is unfeasible due to the high cost and safety requirements of space robotic systems, our goal is to employ simulations with the aim of transferring learned policies to a real robot.

The main contributions of this work are as follows:

- A simulation environment of the Moon, which enables learning of mobile manipulation skills that are transferable to the real-world domain due to its realistic physics, physically-based rendering and extensive use of domain randomization with procedurally-generated datasets for simulating the wide variety of lunar conditions.

[1]Space Robotics Research Group (SpaceR), Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg. {andrej.orsula, miguel.olivaresmendez, carol.martinezluna}@uni.lu

[2]Robotics & Automation Group, Department of Materials and Production, Aalborg University, Denmark. sb@mp.aau.dk

- A novel approach for utilizing 3D octree visual observations with multi-channel features for end-to-end deep reinforcement learning. Octrees are used to efficiently represent the 3D scene, while an octree-based convolutional neural network is applied to extract abstract features that allow agents to generalize over spatial positions and orientations.
- A demonstration of learning robotic grasping inside a realistic simulation environment of the Moon and subsequent zero-shot sim-to-real transfer to a real robot inside a Moon-analogue facility.

II. Related Work

Extraterrestrial environments pose many challenges to both the hardware and software of robotic systems. Even though rovers deployed to Moon and Mars can travel long distances, their actions are controlled remotely by operators from Earth due to their limited autonomous capabilities [6]. Current manipulation for planetary exploration relies on using calibrated systems to achieve the required precision. However, their control is based on the manual selection of target waypoints that are acquired from stereo vision [7]. Due to limitations caused by communication delays, an effort has been put into increasing the autonomy of extraterrestrial rovers. One example of an autonomous rover featuring a robotic arm with a mechanical gripper was presented in [8]. With high-level commands, this rover was able to autonomously pick and assemble known objects by planning motions to predefined configurations relative to their pose estimate. More complex mobile manipulation tasks with the same rover were later conducted in a Moon-analogue environment [9], albeit the mechanical gripper was replaced with a docking interface to reduce the requirement for high precision positioning by increasing misalignment tolerance. Our work similarly investigates manipulation in lunar environments but focuses on grasping previously unknown objects with a general-purpose mechanical gripper to enable application versatility, despite increasing the task difficulty.

The mobility of rovers with robotic arms extends their reachable workspace, which in turn enhances their capabilities for interacting with the environment. However, the complexity of extraterrestrial environments poses several difficulties due to their unstructured nature and uncertainties caused by limited knowledge of surroundings with imperfect sensory perception. Analytical approaches for robotic grasping lack the required generalization even for terrestrial applications [10], despite their effectiveness for task-specific problems [8], [9]. Contrary to this, supervised learning provides a way to learn grasp synthesis empirically from labeled datasets. However, this approach requires a large volume of data in order to achieve the desired level of generalization [11]. Reinforcement learning enables acquiring policies for sequential decision-making problems by interacting with the environment in a self-supervised manner. Recent research has also applied this method in the space robotics domain for tasks such as planetary soft landing [12] and rover path planning [13]. The application of deep reinforcement learning, i.e. a combination of deep learning and reinforcement learning, has been extensively explored for terrestrial robotic grasping in the last few years [14]. Many of these contributions focus on the final performance using a single object [15] or several different objects with simple geometry [16], [17]. More recent work strives to increase this variety by training on objects with more complex geometry [18], [19], as it is considered to be one of the most important challenges for learning-based robotic grasping. In [19], a general policy capable of grasping diverse objects was achieved by training on multiple real robots over the course of several weeks. Our system strives to achieve the same goal while training agents solely inside a simulation. To bridge a possible reality gap per our first contribution, we create a simulation environment with realistic physics and physically-based rendering while relying on domain randomization [16], which is a popular technique to facilitate the sim-to-real transfer.

Manipulation tasks with high-dimensional continuous action and observation spaces pose many challenges when applying reinforcement learning due to its sample inefficiency, brittleness to hyperparameters, training instability and limited reproducibility [14]. Therefore, several different approaches have been analyzed over the years. Pixel-wise action space has been exploited for selecting action primitives based on traditional motion planning techniques in previous research such as [17], where a grasp pose synthesis was incorporated with the pushing of objects. Alternative approaches focus on end-to-end learning to directly control the motion of robots either via joint commands [15], [20] or actions that are expressed as Cartesian displacement of the gripper pose [19]. Our proposal similarly employs end-to-end learning of a policy that maps raw observations directly to continuous actions in Cartesian space.

The vast majority of research using deep reinforcement learning for robotic grasping relies on visual image observations coupled with convolutional neural networks. Here, RGB images are commonly used [16], [19], [20], where depth maps or their inclusion in RGB-D data are also common [17]. However, it is argued that 2D convolutional layers do not provide the desired level of generalization over spatial position and orientation for robotic manipulation compared to their well-established generalization over the horizontal and vertical position in the image plane [18]. Therefore, our approach employs 3D octree observations due to advancements in their utilization in deep learning [21], [22]. As opposed to [21] that employs octree-based convolutional neural networks to analyze 3D shapes, we utilize octrees for real-time control with deep reinforcement learning. Although [23] recently applied octrees to learn planning of trajectories for mobile robots due to their efficiency, there is currently a lack of methods that employ 3D observations for end-to-end learning of manipulation skills. In this way, we study the importance of generalization over the full 6 DOF workspace in which robots operate and compare them to more traditional image-based observations as a part of our second contribution.

The application of this paper is closely related to the work presented in [24] and [25], both of which focus on grasping stones and boulders. Multi-finger humanoid hand is used in [24] with computationally-efficient control that enables both grasping and pushing actions. In [25], an excavator with a two-jaw gripper is used for the autonomous assembly of a large-scale stone wall with the capability to reorient grasped boulders. Whereas these approaches perform 3D object segmentation followed by grasp synthesis and traditional motion planning, our work focuses on end-to-end reinforcement learning. This enables agents to explore the necessary interactions with objects in a self-supervised manner, without the need to manually define complex subroutines such as pushing and reorientation. Furthermore, our focus on lunar environments introduces additional challenges such as uneven terrain and demanding illumination conditions.

III. Problem Formulation

We focus on the robotic grasping of visually-perceived objects within the reachable workspace. It is assumed that any previous or subsequent motion of the rover and its arm is performed by traditional methods or other policies that are part of a larger hierarchy. We do not distinguish between isolated objects and those in cluttered scenes because rovers in extraterrestrial conditions would encounter both cases. We consider the task to be episodic, where each episode is successful once an object is grasped and lifted above a required threshold. Due to uneven terrain, this requirement is specified to be 25 cm above the base footprint of the robot. Each episode is further limited to 40 s, meaning that termination with failure occurs after a corresponding number of steps without success.

The task of robotic grasping can be formulated as a Markov decision process, where the behavior of an agent is defined by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that provides a mapping from states to actions. At each discrete time step t, the environment utilizes a reward function $r(s_t, a_t) \in \mathbb{R}$ to emit an immediate reward for executing an action a_t in a state s_t, which brings the agent to the next state s_{t+1}. The primary objective of the agent is to find the optimal policy π^* that maximizes the expected sum of all future rewards $\sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, which are discounted by $\gamma \in [0, 1]$ over a time horizon. Due to our focus on episodic grasping, T indicates the terminal state of an episode.

IV. Learning to Grasp from Octree Observations

Our approach incorporates visual octree observations with end-to-end learning of robotic grasping in a procedurally-generated simulation environment. An overview of the approach is presented in Fig. 2.

A. Truncated Quantile Critics

We use Truncated Quantile Critics (TQC) [26] in this work to learn a policy for robotic grasping. It is a model-free off-policy actor-critic algorithm that incorporates both an actor for optimizing a stochastic policy $\pi(a|s, \theta)$ parameterized by θ, and one or more critics to estimate the action-value function $Q(s, a)$ by iteratively minimizing the temporal difference error δ_t for a set of all available continuous actions that the agent is allowed to take. Critics evaluate the actor based on how rewarding the selected actions are. TQC is an extension of Soft Actor-Critic (SAC) [27] and therefore employs entropy regularization to optimize a trade-off between expected return and entropy, which represents the randomness of the policy. Unlike SAC, TQC utilizes a distributional representation for the action-value function $Q(s, a)$ of critics and truncates a number of topmost atoms from such distributions to address the problems with overestimation. We employ the specific implementation of TQC from Stable Baselines3 [28].

B. Observation Space

The observation space of the agent consists primarily of visual observations that are represented as 3D octrees. Visual data originates from a stereo camera that is considered to be a space-suitable sensor as it is already used onboard current rovers [6]. In addition to depth maps produced by the camera, we also use monochromatic images captured by one of the imaging sensors to provide agents with supplementary information about the scene, e.g. the notion of shadows, as visualized in Fig. 2. No meaningful information would be gained from multi-channel color images because lunar environments are generally homogeneously colored. Proprioceptive observations are also included to provide information about the state of the gripper in the case it is occluded or outside of the camera view. Only the gripper pose $(x, y, z, R_1, R_2, R_3, R_4, R_5, R_6)$ and its state $g_s \in \{closed: -1, open: 1\}$ are used in order to keep the observation space invariant to the utilized robot. Gripper pose is represented with respect to the robot base, and its orientation R is expressed as a continuous 6D rotation representation from [29] due to its suitability for deep learning. Furthermore, the last two observations are stacked together at each time step, similar to the image stacking method presented in [30]. This promotes the Markov property of the decision process formulation by providing the agent with temporal information about environment states and dynamics, such as the notion of motion.

Conversion of visual observations to octrees begins with the aforementioned depth map and monochromatic image, which are used to create a point cloud of the scene in the form of an unstructured list of (x, y, z, i) tuples. This intermediate representation is then transformed into the coordinate frame of the robot base. Such transformation between the camera and robot is assumed to be known or estimated via a hand-eye calibration procedure. Once transformed, the point cloud is cropped to occupy a fixed volume in space with a uniform aspect ratio. All remaining points are then used to construct an octree by performing a recursive subdivision of the occupied 3D volume until a certain depth d_{max} is reached. Contrary to the inefficiency of voxel grids, octrees create hierarchical tree-like data structures where only occupied cells are recursively decomposed into eight child octants. Observable features can then be stored as channels

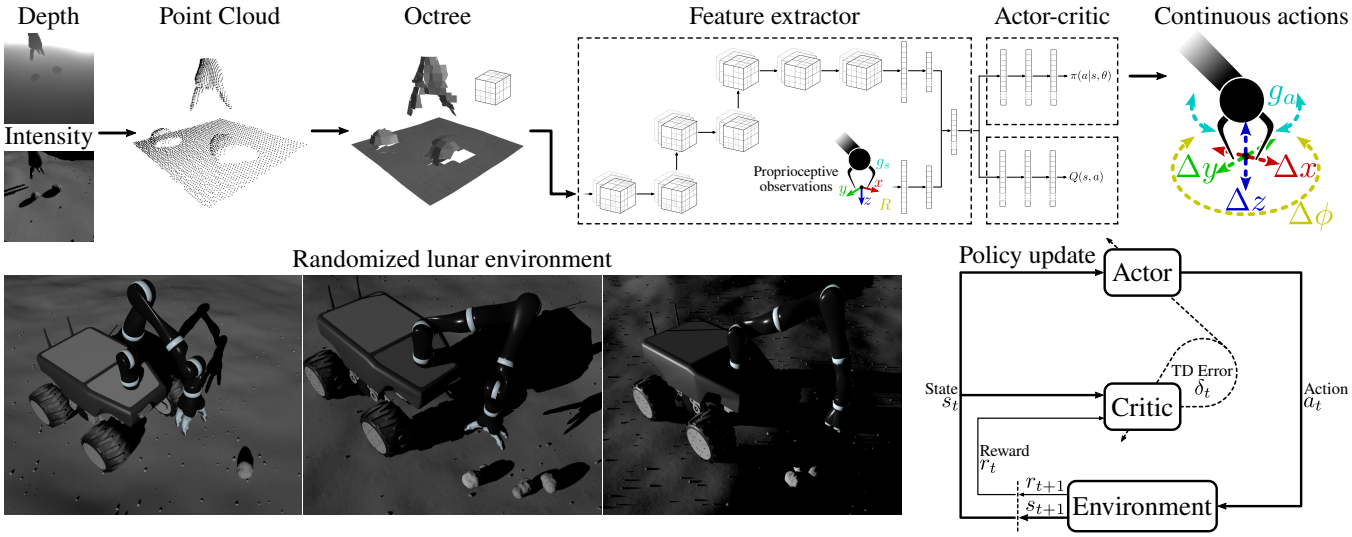

Fig. 2. Overview of our approach for training reinforcement learning agents using a model-free actor-critic algorithm in lunar environments. For visual observations, we employ octrees that are constructed from depth maps and monochromatic images via an intermediate representation in the form of point clouds. Together with proprioceptive observations, a shared feature extractor is utilized to provide abstract features that are used as input for both actor and critic networks. The actor network optimizes a stochastic policy that provides continuous actions in Cartesian space in the form of gripper displacement and action. All of these networks are optimized simultaneously via end-to-end learning while an agent is trained inside a randomized simulation environment.

at the finest leaf octants of the new octree, i.e. the smallest possible octants at depth d_{max}. For each of these octants, all points from the point cloud that occupy the same volume in space are used to extract the relevant features. We store three distinct attributes at each finest leaf octant, i.e. the average unit normal vector $(\overline{n}_x, \overline{n}_y, \overline{n}_z)$ estimated from the points, the average distance $\overline{d}$ from the center of an octant cell to all points used during its formation, and the average intensity $\overline{i}$. Both $\overline{d}$ and $\overline{i}$ are normalized to be in the range $[0, 1]$.

C. Action Space

The agent is allowed to control the motion of the robotic arm at each time step through continuous actions in Cartesian space. These were selected in favor of joint commands due to their invariance to the specific kinematic configuration of the utilized robot. As illustrated in Fig. 2, the action space for control of the gripper pose incorporates the relative translation $(\Delta x, \Delta y, \Delta z) \in [-1, 1]$ and yaw rotation $\Delta\phi \in [-1, 1]$, which are expressed with respect to the robot base frame. Prior to executing these actions, they are mapped to their respective metric and angular ranges of ± 10 cm and $\pm 45°$. Traditional collision-free motion planning and execution are accomplished by MoveIt 2 [31] while utilizing TRAC-IK [32] kinematics solver and RRTConnect [33] for path planning. Control of the gripper is similarly accomplished via a continuous action $g_a \in [-1, 1]$, where positive values open and negative values close the gripper. The control frequency of the agent is set to 2.5 Hz because it is designed to provide high-level decision-making commands, while the lower-level controllers running at 200 Hz in simulation and 500 Hz on the real robot are responsible for real-time execution. With the aforementioned limitation of 40 s, this results in a maximum of 100 discrete time steps per episode during which the agent can select its actions.

D. Composite Reward Function

To accelerate training, we use a composite reward function that combines sparse rewards from four distinct stages of the entire grasping task, i.e. reaching, touching, grasping and lifting. These follow a hierarchical flow, where each stage can give a reward upon its completion only once per episode. Their corresponding reward is set to increase exponentially as $r_b^{r_i - 1}$ based on their order $r_i \in [1, 4]$ in the hierarchy. The base of the exponential function can be tuned, where $r_b = 8$ was selected for TQC through empirical evaluation. This results in a theoretical maximum reward of 585 for each successful episode. In addition to the composite reward, a small reward of -0.1 is subtracted at each time step until the termination in order to encourage the agent to accomplish the task as fast as possible.

E. End-to-End Learning from Octrees

We employ a novel end-to-end approach for learning a policy that maps 3D octree observations directly into continuous actions via function approximation in the form of neural networks that are visualized in Fig. 3. Even though TQC requires separate networks for the actor and critics, it is beneficial to share a portion of their networks to reduce the total number of learnable parameters when dealing with high-dimensional visual observations such as octrees. Therefore, we adapt the octree-based convolutional neural network architecture from [21] into a feature extracting module that is optimized simultaneously with the actor and critic networks during the training. We utilize octrees with a maximum depth of $d_{max} = 4$ as the input, which are processed through a series of 3D convolutional and pooling layers that progressively increase the number of channels while reducing the depth of the octree. In order to enable the

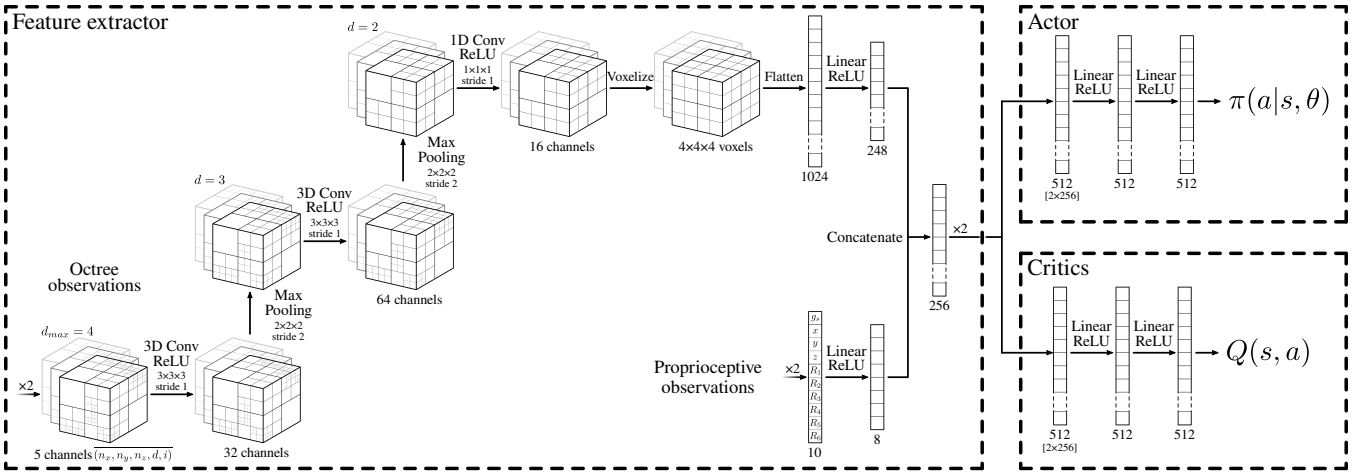

Fig. 3. The full network architecture with a shared octree-based feature extractor and separate actor-critic networks.

use of standard network layers that operate on data structures with a fixed size, each octree is voxelized after reaching the depth $d = 2$ by assigning zeros to all channels of every empty cell. Visual and proprioceptive features are then concatenated together into a feature vector. The feature extractor is applied in parallel to the entire temporally-stacked observation as a part of the same mini-batch. The output of this operation is then concatenated into a single feature vector that is used as input for the separate actor-critic networks that comprise of fully-connected layers.

F. Training Environment

We create a new simulation environment for the training of agents under lunar conditions. Our environment provides a standardized OpenAI Gym [34] interface while utilizing Gazebo [35] robotics simulator (formerly known as Ignition). Gazebo was selected due to its open-source nature that encourages reproducibility and plugin-based architecture that makes it easily extensible. We employ DART physics engine with a step size of 5 ms to simulate rigid-body dynamics. Furthermore, physically-based rendering (PBR) capabilities are facilitated by using OGRE 2. Gym-Ignition [36] is utilized for interfacing Gazebo due to its focus on reinforcement learning research. Lastly, ROS 2 [37] is used as the middleware to facilitate communication among the primary nodes while simplifying the sim-to-real transfer by providing an identical interface for both simulated and real robots.

Datasets of models for terrain and objects are necessary to simulate a realistic-looking lunar environment that encapsulates its variety. However, there is a lack of available datasets from this domain that would provide the required level of detail at the scale of the robot workspace. For this reason, we create our datasets for both lunar surfaces and rocks through procedural generation. Our synthetic mesh generation pipelines are based on the Geometry Nodes feature of Blender [38]. For each model, we apply displacement to individual vertices by a randomized 3D pattern that is formed as a combination of multiple random procedural textures at different scales. For lunar terrain, a subdivided 2D plane is used as the original geometry while imitating uneven terrain with impact craters. Lunar rocks are initialized from subdivided convex polyhedrons that are displaced to represent rocks of different sizes and shapes, including non-convex geometry. To improve training times, each model is generated at two different levels of detail, where a simpler one is used for its collision geometry. Furthermore, each model is assigned a random set of PBR material textures during its insertion into the environment. Using this simple technique, we can synthesize unique environments with nearly an unlimited number of permutations. During our experiments, we utilize a dataset with 250 surfaces, 250 rocks and 35 PBR texture sets.

Domain randomization is employed in order to increase the variety of the environment even further, which is illustrated by the initial states of three example episodes in Fig. 2. The following attributes are randomized from uniform distribution during environment resets: the pose of the rover within the environment and the initial joint configuration of its arm above the workspace; terrain model, its surface friction and material; object count (1 – 4), their models, densities, surface frictions, materials and poses relative to the rover; as well as direction and elevation of the simulated Sun. The pose of the virtual camera is also randomized relative to its mounting surface on the rover ($\pm$5 cm). Furthermore, Gaussian noise $\mathcal{N}(0, 0.001)$ is added to the captured depth maps and monochromatic images in order to increase the realism of observations.

G. Curriculum

In order to accelerate the training in the early stages, a curriculum is applied to the height requirement for the successful lifting of objects. This requirement is set to be proportional to the current success rate smoothed by a rolling average over the last 100 episodes. The requirement is initialized at 7.5 cm and increased linearly to 25 cm until a success rate of 33% is reached. With this addition, successful grasps become more common during the early stages of training while the actions of agents are nearly random.

V. EXPERIMENTS

We evaluated the real-world applicability of our approach while comparing the 2D image and 3D octree observations.

A. Experimental Setup

Throughout the experiments, we used a Summit XL-GEN mobile manipulator from Robotnik equipped with a 7 DOF Kinova Gen2 robotic arm and a three-finger mechanical gripper. The robot was controlled via MoveIt 2 both inside simulation for training and during sim-to-real evaluation in a Moon-analogue facility. To obtain visual observations inside the simulation, a virtual camera with a resolution of 128×128 px was used at a framerate of 15 Hz. On the real robot, we used Intel RealSense D435 to capture images of 424×240 px at 15 Hz, which were cropped and resized to 128×128 px before use. We attached the camera statically at the front of the rover while randomizing its simulated pose during the training. To estimate the transformation between the real camera and the base of the real robot, we performed a hand-eye calibration. We did not employ any additional artificial lights mounted on the robot and relied solely on ambient illumination that originated either from the simulated Sun or a light source inside the Moon-analogue facility that emulates the solar illumination. For octrees, we used an observable volume of 40×40×40 cm in front of the rover, which results in the size of 2.5 cm for the finest leaf octants at the selected maximum depth of $d_{max} = 4$. In order to avoid exploring areas of no interest, the position of the gripper was restricted to a 35×35×60 cm workspace centered at the origin of observable volume.

B. Hyperparameters

Because the selection of hyperparameters can significantly affect the learning curve and final performance of learned policies, we optimized hyperparameters in two consecutive steps. First, an automatic optimization was performed using the hyperparameter optimization framework Optuna [39]. This was followed by subsequent fine-tuning of targeted hyperparameters via manual optimization. During this process, we also tuned the aforementioned reward scale, the number of observation stacks and the size of neural networks. The utilized set of hyperparameters can be seen in Table I.

TABLE I
HYPERPARAMETERS USED FOR THE TRAINING OF ALL AGENTS.

Hyperparameter	TQC
Optimization algorithm	Adam [40]
Learning rate schedule	Linear, $2.0 \cdot 10^{-4} \rightarrow 0$
Mini-batch size	64
Gradient steps per update	100 (after every episode)
Size of the replay buffer	50000
Discount factor γ	0.99
Target update rate τ	$4 \cdot 10^{-5}$
Entropy coefficient α	Automatic [27]
Entropy target	$-dim(\mathcal{A}) = -5$
Number of critics	2
Number of atoms	25
Number of truncated atoms	3
Exploratory action noise	$\mathcal{N}(0, 0.025)$

C. Training Procedure

We trained agents in our simulation environment to learn policies for grasping lunar rocks by the use of TQC. In order to evaluate the proposed use of octree observations, we trained separate agents that use either image or octree observations. For image observations, we employed images with the resolution of 128×128 px and two channels that contain the depth and intensity values for each pixel. Both of these values are normalized to be in the range $[0, 1]$, where the maximum distance of 1 m is selected for the depth map. Similar to octrees, two consecutive images are stacked together for each observation to preserve temporal information. To provide a fair comparison, an analogous feature extraction network architecture is created for image observations based on the octree-based feature extractor from Fig. 3 by replacing 3D operations with their 2D variants. To achieve approximately the same number of learnable parameters for the feature extractor, i.e. 315k parameters, we increase the number of channels in 2D convolutional layers. The resulting network architecture for the image-based feature extractor is illustrated in Fig. 4.

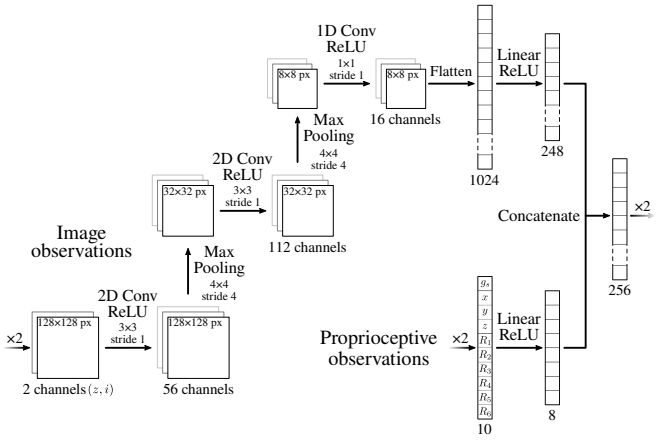

Fig. 4. The network architecture of the image-based feature extractor, which is analogous to the octree-based feature extractor presented in Fig. 3.

We also evaluated the importance of domain randomization for sim-to-real transfer in lunar applications. To do this, we trained agents on two different variants of the environment for each observation type. The first variant is the environment with complete domain randomization, whereas the second variant has a reduced level of randomization. In this reduced environment, only the pose of objects and the initial joint configuration are randomized for each episode. From the utilized datasets, one model of the lunar surface and four models of lunar rocks are randomly selected and used throughout the training. All other variables, such as the pose of the camera and the direction of illumination, are similarly selected at random but kept static throughout the training.

All agents were trained for 500k time steps and periodically evaluated every 10k steps on 20 episodes by utilizing their current policies with deterministic actions. For each analyzed variant, we trained three agents with different seeds for the pseudorandom generator that initializes the simulation

environment and all learnable parameters. Since we evaluate the agents in a Moon-analogue facility located on Earth, the gravity of the simulated environment is set to 9.807 m/s^2. Learning curves of all agents can be seen in Fig. 5.

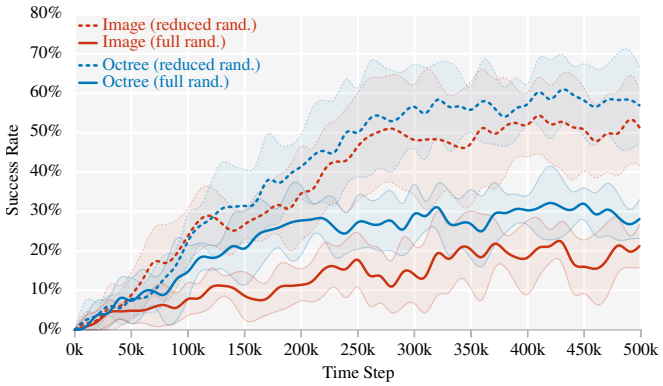

Fig. 5. Learning curves of the trained agents that are evaluated every 10k steps. Solid lines represent smoothed mean, whereas shaded areas indicate the standard deviation over three runs with different random seeds.

D. Sim-to-Real Transfer

We evaluated the feasibility of sim-to-real transfer in LunaLab [41], which is a Moon-analogue facility at the University of Luxembourg. We assessed all variants of trained agents, where the best performing agent at the end of its training inside the simulation is selected for each variant. The policies of these agents were then used for deterministic selection of actions during execution on the real robot. A total of eight different rocks from Fig. 6 were utilized during this experiment, where 1 – 4 rocks were randomly scattered within the workspace for each episode. Every agent was evaluated over the course of 25 episodes that were considered successful if the robot grasped and lifted one of the rocks.

Fig. 6. Eight rocks used during the evaluation of the sim-to-real transfer.

Quantitative results of the sim-to-real success rate appear in Table II, whereas an example of a successful grasp sequence is visualized in Fig. 7. During episodes that failed due to timeout, it was observed that agents got frequently stuck in a loop attempting to repeatedly grasp a specific rock without success due to improper positioning of the gripper. This behavior occurred for all evaluated agent variants, but it was most prominent for agents trained in environments with reduced domain randomization.

TABLE II

QUANTITATIVE RESULTS OF THE ZERO-SHOT SIM-TO-REAL TRANSFER.

Observation type	**Level of randomization**	**Success rate** (n=25)
Image	Reduced	12%
Image	Full	20%
Octree	Reduced	8%
Octree	Full	32%

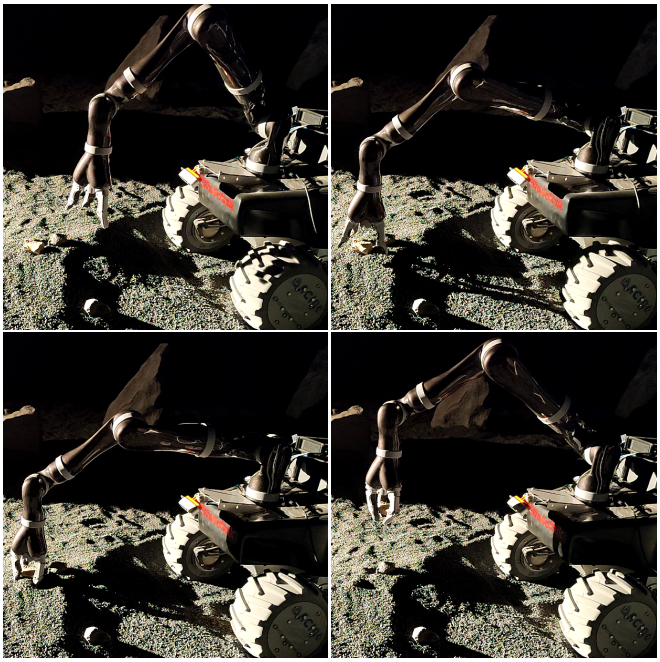

Fig. 7. An example of a successful grasp sequence using the real robot.

VI. DISCUSSION

By training agents at two different levels of domain randomization, we investigated its effect on the sim-to-real transfer. However, the level of randomization significantly impacted the overall learning of agents. With fully-enabled randomization, agents encountered training instability that restricted their achievable success rate compared to agents that experienced a reduced subset of the lunar variety. We attribute the instability to high variance in gradient estimation due to large variations in observations. Despite this, sim-to-real experiments indicate that domain randomization enables robust transferability to the real-world domain, where agents trained in fully randomized environments performed better when transferred to the real robot despite their lower success rate in simulation. Although domain randomization enables learning of policies capable of generalization under challenging conditions, training stability must be significantly improved via methods such as active domain randomization [42] in order to take full advantage of this approach.

The experimental evaluation indicates that 3D visual observations in the form of octrees provide better performance than image-based observations when applied for end-to-end learning of robotic grasping. This result is attributed to the better ability of 3D convolutions to generalize over spatial positions and orientations, unlike 2D convolutions that generalize well over planar image coordinates. Another advantage of 3D observations is their ability to provide learned policies

with invariance to the camera pose, further simplifying the transfer to a different system or application domain. Sensory fusion of data from multiple depth sensors could also be applied to obtain a single 3D data structure that provides observations with improved quality and reduced occlusions. Therefore, these and other techniques must be investigated further to reveal the full potential of 3D observations in self-supervised learning of robot manipulation skills.

VII. Conclusion

In this work, we presented an approach for learning robotic grasping on the Moon with end-to-end deep reinforcement learning. We analyzed the application of 3D octree observations and compared their performance to 2D images. We also investigated the effects of employing domain randomization in lunar environments by demonstrating zero-shot sim-to-real transfer to a real robot in a Moon-analogue facility. Overall, we believe that deep reinforcement learning is a promising method for acquiring various manipulation skills for robots in space, despite its many challenges. Improving the learning stability in diverse environments is one of the necessary steps before similar approaches can be robustly employed for the wide range of applications within space robotics.

References

[1] T. Zhang *et al.*, "The progress of extraterrestrial regolith-sampling robots," *Nature Astronomy*, vol. 3, pp. 487–497, Jun. 2019.

[2] R. W. Moses and D. M. Bushnell, *Frontier In-Situ Resource Utilization for Enabling Sustained Human Presence on Mars*, ser. NASA technical memorandum. NASA, Langley Research Center, Apr. 2016.

[3] J. Liu *et al.*, "Landing Site Selection and Overview of China's Lunar Landing Missions," *Space Science Reviews*, vol. 217, no. 6, Feb. 2021.

[4] S. Schierholz and J. Finch, "NASA Selects Companies to Collect Lunar Resources for Artemis," *NASA*, Dec. 2020.

[5] B. K. Muirhead and A. Karp, "Mars Sample Return Lander Mission Concepts," in *IEEE Aerospace Conference*, Mar. 2019, pp. 1–9.

[6] J. P. Grotzinger *et al.*, "Mars Science Laboratory Mission and Science Investigation," *Space Science Reviews*, vol. 170, pp. 5–56, Jul. 2012.

[7] K. Nickels, M. DiCicco, M. Bajracharya, and P. Backes, "Vision guided manipulation for planetary robotics — position control," *Robotics and Autonomous Systems*, vol. 58, pp. 121–129, Jan. 2010.

[8] M. J. Schuster *et al.*, "The LRU Rover for Autonomous Planetary Exploration and its Success in the SpaceBotCamp Challenge," in *International Conference on Autonomous Robot Systems and Competitions*, May 2016, pp. 7–14.

[9] P. Lehner *et al.*, "Mobile manipulation for planetary exploration," in *IEEE Aerospace Conference*, Mar. 2018, pp. 1–11.

[10] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, Mar. 2012.

[11] J. Mahler *et al.*, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," *arXiv preprint*, Mar. 2017.

[12] X. Xu, Y. Chen, and C. Bai, "Deep Reinforcement Learning-Based Accurate Control of Planetary Soft Landing," *Sensors*, vol. 21, no. 23, Dec. 2021.

[13] X. Jin, W. Lan, T. Wang, and P. Yu, "Value Iteration Networks with Double Estimator for Planetary Rover Path Planning," *Sensors*, vol. 21, no. 24, Dec. 2021.

[14] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," *Journal of Machine Learning Research*, vol. 22, no. 30, pp. 1395–1476, Jan. 2021.

[15] I. Popov *et al.*, "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation," *arXiv preprint*, Apr. 2017.

[16] J. Tobin *et al.*, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2017.

[17] A. Zeng *et al.*, "Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2018, pp. 4238–4245.

[18] M. Gualtieri, A. t. Pas, and R. Platt, "Pick and Place Without Geometric Object Models," in *IEEE International Conference on Robotics and Automation*, May 2018, pp. 7433–7440.

[19] D. Kalashnikov *et al.*, "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," *arXiv preprint*, Jul. 2018.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, Jan. 2016.

[21] P.-S. Wang, Y. Liu, Y.-X. Guo, C. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics*, vol. 36, no. 72, pp. 1–11, Aug. 2017.

[22] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6620–6629, Jul. 2017.

[23] B. Trăsnea *et al.*, "OctoPath: An OcTree-Based Self-Supervised Learning Approach to Local Trajectory Planning for Mobile Robots," *Sensors*, vol. 21, no. 11, May 2021.

[24] R. Grimm, M. Grotz, S. Ottenhaus, and T. Asfour, "Vision-Based Robotic Pushing and Grasping for Stone Sample Collection under Computing Resource Constraints," in *IEEE International Conference on Robotics and Automation*, May 2021, pp. 6498–6504.

[25] M. Wermelinger, R. L. Johns, F. Gramazio, M. Kohler, and M. Hutter, "Grasping and Object Reorientation for Autonomous Construction of Stone Structures," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5105–5112, Jul. 2021.

[26] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics," in *International Conference on Machine Learning*, vol. 119, no. 515, Jul. 2020, pp. 5556–5566.

[27] T. Haarnoja *et al.*, "Soft Actor-Critic Algorithms and Applications," *arXiv preprint*, Dec. 2018.

[28] A. Raffin *et al.*, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, Nov. 2021.

[29] Y. Zhou *et al.*, "On the Continuity of Rotation Representations in Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp. 5738–5746.

[30] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[31] I. A. Sucan and S. Chitta, *MoveIt 2*. [Online]. Available: https://moveit.ros.org

[32] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, Nov. 2015, pp. 928–935.

[33] J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, Apr. 2000, pp. 995–1001.

[34] G. Brockman *et al.*, "OpenAI Gym," *arXiv preprint*, 2016.

[35] Open Robotics, *Gazebo*. [Online]. Available: https://gazebosim.org

[36] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci, "Gym-Ignition: Reproducible Robotic Simulations for Reinforcement Learning," in *IEEE/SICE International Symposium on System Integration*, Jan. 2020, pp. 885–890.

[37] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[38] Blender Development Team, *Blender 3.0*. [Online]. Available: https://blender.org

[39] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2019, pp. 2623–2631.

[40] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, Dec. 2014.

[41] P. Ludivig, A. Calzada-Diaz, M. Olivares-Mendez, H. Voos, and J. Lamamy, "Building a Piece of the Moon: Construction of Two Indoor Lunar Analogue Environments," in *International Astronautical Congress*, Oct. 2020.

[42] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active Domain Randomization," in *Conference on Robot Learning*, vol. 100, Oct. 2020, pp. 1162–1176.