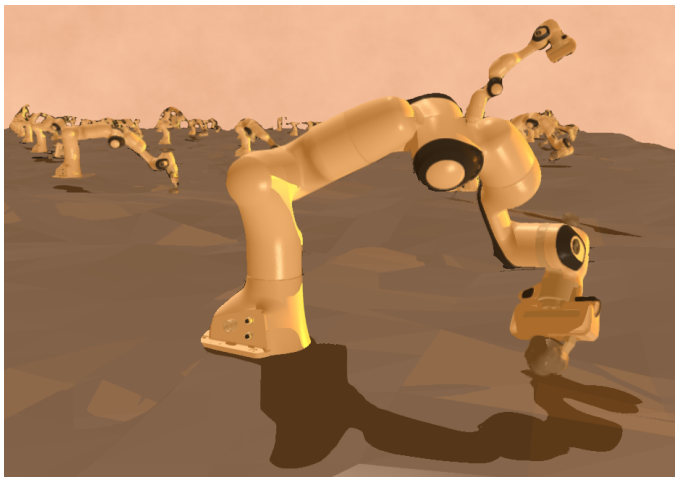
Reinforcement Learning for Robotic Rock Grasp Learning in Off-Earth Space Environments

Anton Bock Andersen
Department of Electronic Systems
Aalborg University
Aalborg, Denmark
aan17@student.aau.dk

Marius Willemoes Jørgensen
Department of Electronic Systems
Aalborg University
Aalborg, Denmark
mwja17@student.aau.dk

***Abstract*—The time delay in communications between Earth and Mars pose a challenge for the control of rovers on Mars, for tasks such as rock grasping. This paper investigates the application of Deep Reinforcement Learning (DRL) as a method for robotic grasp learning of Martian rocks. NVIDIA Isaac Gym employed to simulate a randomly generated Mars-like environment, populated with randomly generated rocks. Proximal Policy Optimisation (PPO) is used to train a Franka Emika Panda robotic manipulator in the task of picking up the randomly generated rocks. As a Mars environment can have large changes in terrain height, joint control of the robot manipulator is implemented to enable 6-DOF grasping of rock samples. The proposed method show a 91.51% success rate, grasping randomised rocks ranging from 40-400 cm^3 in size. The randomness of the environment is found to create a robust agent, while laying a solid foundation for sim-to-real in the future. This paper presents a novel approach for 6-DOF object grasping, using joint control and domain randomisation.**



I. Introduction

Robotic grasping is an essential skill to have for a robot to perform different manipulation tasks. Grasping is not a solved problem for robotic manipulators, in cases with highly varied environments or with completely novel objects [1]. Work involving robotic grasping has developed from being able to grasp simple shapes such as cubes and cylinders to bigger object sets with variety in object shape and size. Even though recent successes in robotic grasping, such as Gou et al. [2], have been able to grasp varied objects with a high success rate, there are still several unsolved aspects of robotic grasping if it has to be applicable outside the lab or factory floor.

A use case where both novel objects and an unknown environment is present is for rovers operating on the Moon or Mars. A rover should be able to pick up novel objects in the form of different rock samples or sample tubes left behind by previous rovers, such as the Perseverance Mars rover [3].

To overcome the challenge of varied environments and novel objects in robotic grasping, different strategies have been utilised [4]. A method that has seen more use in robotics recently for challenges in both grasping [5], [6] as well as navigation of mobile platforms [7] is Deep Reinforcement Learning (DRL). DRL has in recent years been made an even more effective method for teaching robot skills as computational power has increased, along with more sample efficient methods and better simulation tools [8]. A recent simulation tool such as NVIDIA Isaac Gym [9], allows for running numerous environments in parallel and utilises high GPU performance to speed up the physics simulation. This paper focuses on utilising high parallelisation with NVIDIA Isaac Gym to train DRL agents for robotic grasping, with varied objects and terrain. This is done within the context of exploration of extraterrestrial bodies, with a focus on Mars.

In a Mars environment where communication is delayed 5-20 minutes depending on Earth-Mars alignment [10], it is especially important for a system to be able to adapt and adjust its own behaviour. Even though robotic grasping solutions for space is currently limited to rovers for picking up samples, the European space agency (ESA) and other space agencies have plans for setting up extraction of local resources to enable a human long term presence on the moon before 2040 [11]. This will present many additional areas where robotic grasping can be utilised to aid in automatic resource extraction.

II. Related Work

DRL has begun to be more widely applied to robotic manipulation tasks, such as robotic grasping. Several different reinforcement learning (RL) algorithms is used in recent work to solve the task of robotic grasping. Some of these include Proximal Policy Optimisation (PPO) [12], [6], [13], Deep Q

networks (DQN) [14], SAC [15], [16] and TD3 [17]. Even though many robotic grasping solutions are based on the same RL algorithms, the specifics of a DRL setup can vary a lot based on the use of simulation, network architecture, domain randomisation, camera setup, and hyper-parameter tuning among many others.

The architecture of the network, or networks, used to create a DRL robot solution capable of grasping, have several approaches. A big distinction between approaches is if DRL is used end-to-end, or if other networks, such as an object detection network or a dedicated grasp synthesis network, are utilised. Using several networks, each responsible for a different function, can help simplify the DRL problem. An example of this is Chen et al. [6], who perform robot grasping with 4 degrees of freedom (DOF) (x, y, z, θ_z). They use an object detection network along with principal component analysis (PCA) to extract the location and orientation of the objects of the image. This approach simplifies the observations used for the DRL agent to only contain robot and object poses, instead of the entire image. An additional benefit of this approach is that the two networks can be trained separately.

Another approach is using the camera observations directly as observations for the DRL agent and letting the policy learn the full problem of interpreting the observations from the camera and moving the robot to grasp. In this approach, it is not possible to train networks separately. An example of this is in Huang et al. [13] where they train a PPO policy with image observations for 3-DOF(x, y, θ_z) grasping of rock samples with an 80%+ success rate. A feature extraction network is used to extract image features and is trained simultaneously with the policy during training.

Joshi et al. [14] propose using double deep Q learning for robotic grasping. The solution uses a multi-camera view setup, with a static RGB-D camera above the work surface and a wrist-mounted RGB camera. Using multiple viewpoints improves the grasping success rate from 65% to 80%+, compared to a single camera in their setup.

In an article from 2021, Ibarz et al. [8] identifies different current challenges in the use of DRL for training robots for real-world skills. Some of the challenges highlighted are generalisation and sim-to-real.

Introducing a large variety of objects can be a challenge for a grasping solution. A recent success is Kalashnikov et al. [5] with a success rate of 87% on a set of about 1000 different objects doing bin picking. However, due to the complexity of the task, a lot of grasps were used for training. 580 thousand real-world grasps were performed to train the QT-opt model used. The authors also notes that they see a significant performance improvements going from 320 to 520 thousand grasps, which improves the success rate from 78% to 87%. The collection of data for this work required 400+ robot hours.

Using simulation when training a policy allows for a huge amount of training to be performed, compared to doing training in the real world. Different strategies for bridging the reality gap when transferring from simulation to the real world can be employed. Domain randomisation is a method used for improving the sim-to-real transfer. Randomising various parameters of the simulation during training such as texture, objects, pose and lighting has been shown to improve the performance of a policy when transferring into the real world [18], [19], [20].

Peng et al. [21] expands the use of randomisation by randomising dynamics parameters during simulation. This allows their robotic manipulator to adapt to real-world dynamics while only training in simulation. Only a small drop in performance is observed when moving out of simulation using this strategy. Another method for achieving better sim-to-real transfer is domain adaptation. Bousmalis et al. [22] trains a network to make images from simulation resemble the real world.

To enable robots to operate on Mars it has to be able to work around obstacles and generalise well across varied terrain and differently shaped objects. This increases complexity of a DRL solution, thus requiring a large amount of training. In many cases it might not be feasible to collect enough data on a real robot, without many robots working in parallel. Using simulation presents several advantages when using DRL, as large amounts of training data can be collected in a time efficient manner. However, using policies trained in simulation can be challenging to transfer into the real world. Domain randomisation is identified as a crucial aspect of training policies to be able to transfer behaviour well from simulation to the real world.

Domain randomisation techniques applied with previous grasping methods often focuses on varying a few factors during training or implements low-DOF grasping. There is an opportunity to investigating high domain randomisation in conjunction with a 6-DOF grasping approach when training DRL agents.

The paper is structured as follows. Section III covers the system developed, the methods used and setup for experiments. Section V describes the experiments that are carried out. Section VI presents the results of the experiments. Section VII and VIII discusses and concludes on the results.

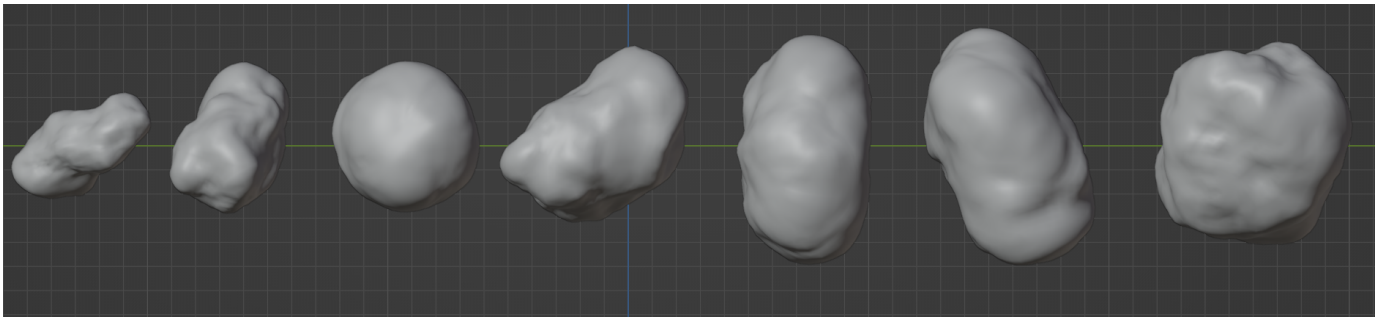
III. Experimental setup

The experimental setup is based on the hardware available for the project. The robot used in the project, is the Franka Emika Panda 7-DOF robot arm. A computer running Ubuntu 20.04 with NVIDIA GeForce RTX3070 GPU and an AMD Ryzen 5 5600X CPU is used for simulation and training.

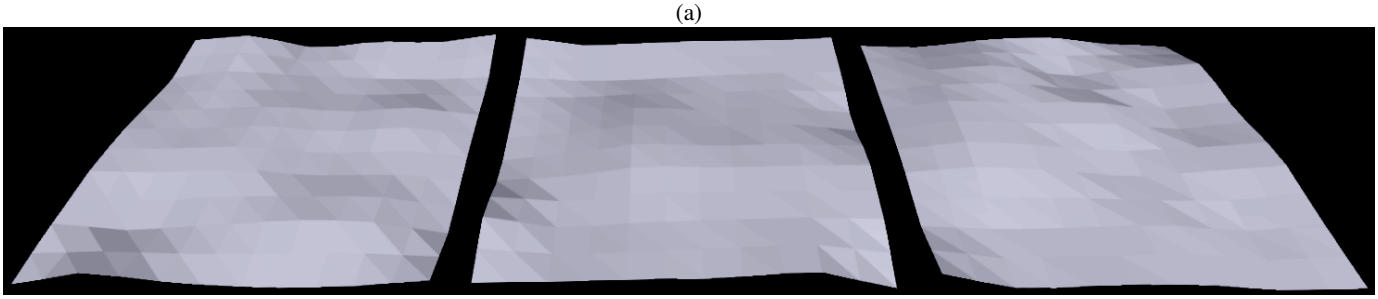
A. Simulation environment

NVIDIA Isaac Gym is chosen as the simulation tool, as it provides the ability of training on many parallel environments, with a high simulation speed [9].

The environment can be divided into three customisable parts: The robotic arm, the terrain, and the grasp object. The environment is made iteratively more complex. In this paper, only the last iterations will be shown. For more details on the simulation environment please see the appended technical report, Chapter 5. For all training iterations, please see the appended technical report, Chapter 6.



(a)



(b)

Fig. 1: The randomly generated grasp objects and terrains. **(a)** shows seven randomly generated rocks, showing the possible size range. Grid cells represents 1 cm^2. **(b)** shows three randomly generated terrains, simulating a Mars-like environment. The terrain is generated from a random uniform distribution of points between -20 cm and 0 cm in height.

- **The Franka Emika Panda robot** is incorporated into the training environment as a URDF model. The Panda robot can either be inserted with a fixed starting pose, or with randomised joint positions. Furthermore, Gym allows the actors in a simulated environment to have either fixed or randomised dynamics. For the purpose of domain randomisation, the starting joint positions are randomly set to within $\pm 7°$.
- **The grasp object** is designed to simulate objects of interests on Mars, in this case rocks. A script is used to randomly generate rocks within ranging from 50 cm^3 to 400 cm^3 in size [23]. Goossens et al. [24] finds the crust density of Mars to be 2.6 ± 0.2 $\frac{g}{cm^3}$. As such the mass of the rocks generated are chosen based on the crust density. Examples of the randomly generated rocks can be seen in Figure 1a.
- **The terrain** can be randomised in Isaac Gym, similarly to the Panda arm. The terrain is generated from a uniformly randomly distributed height-map. The height-map ranges from -20 cm to 0 cm in height, in order to simulate a bumpy terrain on Mars, that is still flat enough to be traversable by a rover. An example of the randomly generated terrain can be seen in Figure 1b.

B. DRL robotic grasping

A RL interaction cycle demonstrating how an action applied to an environment changes environment state and rewards, can be seen in Figure 2.

The agent generates an action, r, at time t, according to its policy $\pi_\theta(s_t|a_t)$, where S_t is the state of the environment.

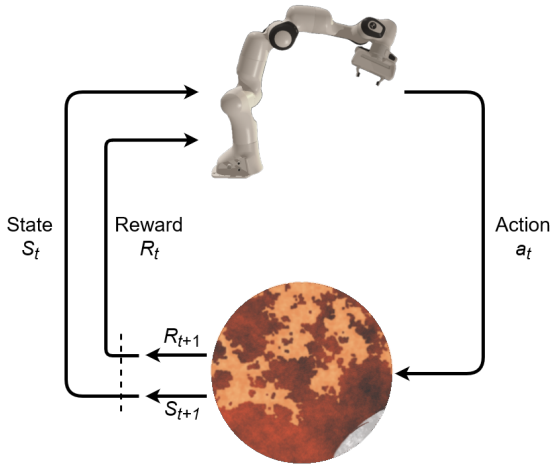


Fig. 2: The RL interaction between the agent (Franka Emika Panda) and the environment (Mars).

After the action impacts the environment, the state transitions to s_{t+1} and likewise a reward, R_{t+1}, is generated.

The problem of robotic grasping in DRL can be formulated as a Markov Decision Process (MDP), which describes the connection between the agent and the environment, through actions, states, and rewards. The goal of the RL agent is to find a policy that maximises the rewards received from the environment. These rewards are called the return(G). As the policy makes decisions that impact future rewards, a discount factor(γ) is introduced to reduce the weight of future rewards.

The policy seeks to achieve the highest discounted return possible [25]. This concept is shown in Eq. (1).

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

There are multiple DRL algorithms that could be applied for training the agent. For this project, PPO is selected as previous research have shown good results with this method. See section II for more information on this. The objective function of PPO can be seen in Eq. (2)[26].

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{2}$$

The clipped objective function, L^{CLIP}, is the main part of the PPO objective function. It is responsible for optimising the policy to follow gradient ascent. A problem that occurs when purely following gradient ascent, is making changes to the policy that are too large. This can result in an agent that will not converge, and is therefore important to avoid. The clipped objective function combats this by limiting how much the policy can be changes on each update. The state-value loss function, L_t^{VF}, allows PPO to learn the state-value function parallel to learning the policy. The entropy, S, encourages the algorithm to explore different policies, instead of only choosing actions that the current policy predicts are good. The two hyperparameters, c_1 and c_2, are simple scalars that are used to weigh the influence of the two terms on the final objective function[26].

1) System architecture: The Isaac Gym environment provides the observations used for training the policy. The PPO implementation used is the one provided by the `SKRL` library [27]. The `SKRL` library wraps the Isaac gym environment and receives observations and rewards at each time-step and provides actions to Isaac Gym for execution by the robot. The network structure used can be seen in Figure 3.

The network used is has an input layer, output layer and 4 hidden layers. Each hidden layer is fully connected and uses RELU activation. The network structure is used both for the policy and value network. Both networks get the observations from the Isaac Gym environment as input. The policy network learns to generates the actions used to control the robot in the environment, where the value network learns to evaluate if those actions takes the environment to a state with a higher value.

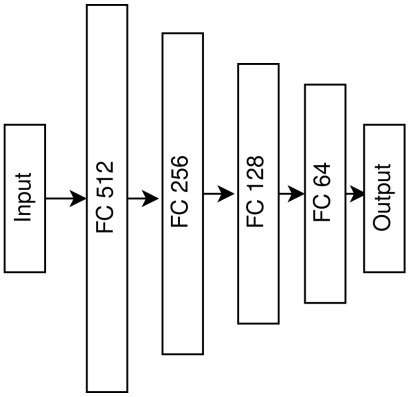


Fig. 3: Network structure used with 4 fully connected hidden layers of size: 512, 256, 128, 64.

TABLE I: Observations and actions of the system. All Observations and Actions are continuous of type, box.

	Name	No. of DOFs	Unit
Observations	Robot joint positions	7	θ
	Robot joint velocities	7	$\dot{\theta}$
	Gripper	2	m
	Distance from gripper to target	3	m
Actions	Target	7	θ
	Gripper	1	m

2) Observations and actions: Using joint control of the Franka Emika Panda robot in the simulated environment results in 19 observed states and 8 actions. These can be seen in Table I.

The robot has joint positions and joint velocities. The gripper position is represented by a value that determines how open the gripper is, along with a value for how far the gripper is above the robots base. Finally, The distance from the gripper to the target object in (x, y, z) direction is part of the observations. The continuous actions of the Franka Robot are a goal position for each of the joints, along with a single value determining how open the gripper is.

3) Reward: The rewards for the task are based on the steps involved in grasping a rock: Reaching, grasping, and lifting. The reach reward is based on the euclidean distance between the gripper of the robot, and the rock. The distance reward maxes out at 2 when there is no difference in position between the gripper and the object. The grasping reward is set to 50, and given at every timestep the gripper meets resistance when closing, without having the gripper closed entirely. The lift reward is based on the height of the rock, and starts being applied when the rock is above 5 cm. From here, the reward rises exponentially until the rock is at 10 cm. Here, the reward is 100, until the rock is lifted above 30 cm where the reward starts dropping off. This is to motivate the agent to keep the rock within 10-30 cm, which is also set as the criteria for a successful grasp. In addition to the aforementioned rewards, a success reward is given when the success criteria is met for more than 30 timesteps. Lastly a collision penalty is given if a high force is detected in any of the robot's links. The reward function can be seen in Table II.

TABLE II: The reward function with the formula for each element shown.

Reward type	Reward function
Reaching	$\begin{cases} (\frac{1}{1+d^2})^2, & \text{for } d > 6cm \\ 2(\frac{1}{1+d^2})^2, & \text{for } d \leq 6cm \end{cases}$
Grasping	50
Lifting	$\begin{cases} h^2, & \text{for } h < 10cm \\ 100h, & \text{for } h > 10cm \\ 400 - 10h, & \text{for } h > 30cm \end{cases}$
Success	200∗remaining timesteps
Collision	-1

IV. Hyperparameters

Several experiments are carried out for tuning hyperparameters of the training. This is done on a simplified environment to form a basis for choosing hyperparameters. Five runs are performed for each hyperparameter setting. The hyperparameters investigated are learning rate and entropy loss. Results are shown in Figure 4 and 5. The optimal learning rate is found to be 0.0005 and the optimal entropy value is found to be 0.001. Further details on the hyperparameter tuning can be found in Chapter 6 in the appended technical report.

V. Experiments

Two experiments are carried out to verify different aspects of the challenge presented in this paper:

- Domain randomisation
- Obstacle avoidance

In both experiments, the agent is trained using 1024 parallel environments. The trained agent is tested 10 times with random starting seeds for both experiments. All experiments feature 250 randomly generated rocks as the grasping object. The results of the experiments are presented in section VI.

For further implementation details on the two experiments, see the appended technical report Chapter 5 and 6.

A. Domain randomisation

The domain randomisation test is carried out in an simulated environment featuring a randomly generated terrain, randomly generated rocks, random position of rocks, randomised starting joint positions of the Panda arm, and randomised dynamics of the Panda arm. The simulation environment can be seen in Figure 6.

Because of the more complex environment, it is found that lowering the learning rate from 0.0005 to 0.0002 is necessary to obtain good performance. Likewise, the entropy is increased from 0.001 to 0.005 to ensure that the agent explores enough different options to converge on a good result.

B. Obstacle avoidance

The obstacle avoidance experiment is carried out in a simulated environment featuring a flat terrain, randomly generated rocks, fixed starting joint positions of the Panda arm, and fixed dynamics of the Panda arm. The purpose of the experiment is to test if the 6-DOF joint control of the Panda arm allows for rock grasping behind obstacles. A wall, acting as the obstacle, is therefore added to the environment, separating the rock and the Panda arm. The environment can be seen in Figure 7.

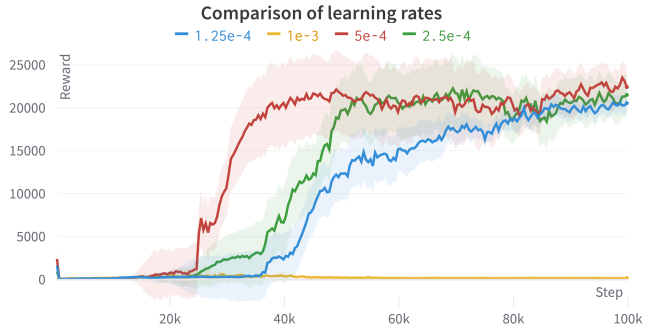


Fig. 4: Graph showing the median of the reward ±1 standard deviation for different Learning rates. No useful learning performed with the highest learning rate of 0.001 (Yellow). Lower learning rates all have stable learning and converged at a similar reward. The learning rate of 0.0005 (Red) performed the best with fast convergence.

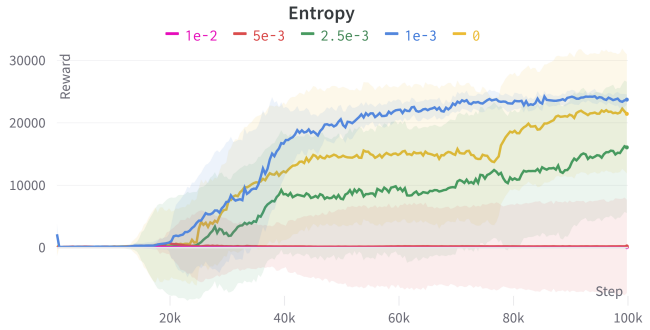


Fig. 5: Graph showing the median of the reward ±1 standard deviation for different Entropy loss scales. No entropy (Yellow) converged at a high reward with high variance between runs. Entropy of 0.001 (Blue) converged at the highest reward and have small variance between runs.

During training of the obstacle avoidance experiment the collision reward tuning is tuned to avoid behaviours that collides with the obstacle. A higher collision penalty of -5 is used to make the robot avoid the obstacle completely.

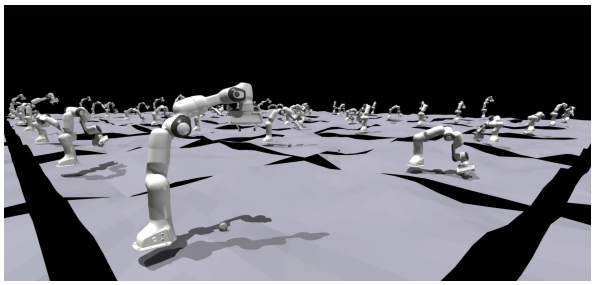


Fig. 6: The simulation environment for the domain randomisation experiment.

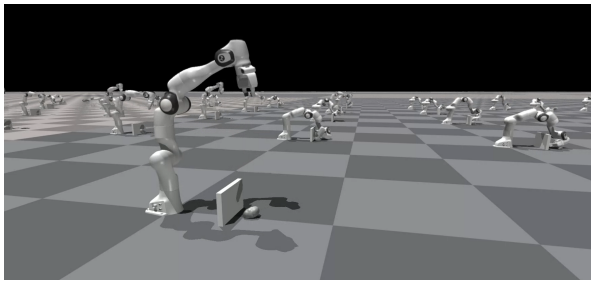


Fig. 7: The simulation environment for the obstacle avoidance experiment.

TABLE III: Success rates for each experiment, based on 100,000 grasp attempts in simulation.

	Success rate
Domain randomisation	91.51%
Obstacle avoidance	88.67 %

VI. Results

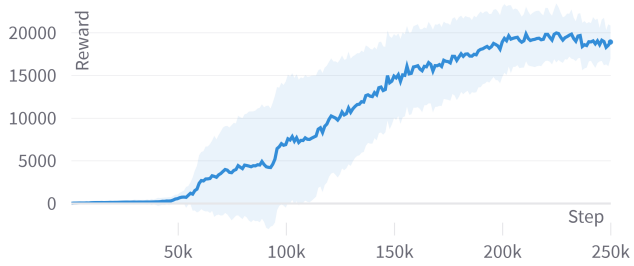
In both experiments, the agent is successful in learning to pick up the rock, with a high success rate. The success rate of each experiment is based on 100,000 grasp attempts, taken from the most successful run in each experiment. A table showing and comparing the results from the three experiments can be seen in Table III.

The training took 250,000 timesteps, taking around one hour. Training graphs showing the reward obtained during training for both experiments, can be seen in Figure 8. From the training graphs, it is evident that the Domain randomisation experiment has a more robust and consistent learning across runs, as all 10 runs converge at a similar rate, at the same total reward. The training graph for the Obstacle avoidance experiment show that the learning rate across these runs is significantly more varied, with the various runs not converging at a similar rate or at the same total reward.

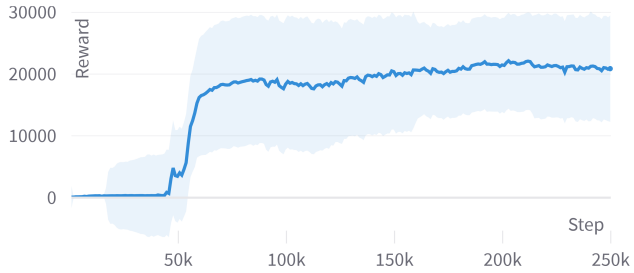
The robot is able to pick up the rocks consistently, even when the rocks are rolling. On failed grasps, the robot is able to retry pickups. A picture series and link to a video of the behaviour can be seen in Figure 9

VII. Discussion

The trained agent is able to grasp the rock with a high success rate in both experiments. The domain randomisation experiment likewise demonstrates stable training, with low variance in the rewards across runs. When observing the behaviour during multiple runs from the same experiment, this is also evident, since they all performed similarly well. While the success rate of the obstacle avoidance experiment is high, the variance of training is also quite high. This suggests that the method shown in the obstacle avoidance experiment has potential to converge to a good result, but does so inconsistently. It is believed that the method could be further improved with more hyperparameter optimisation. While the training could be more robust, the results show that the 6-DOF joint control of the Franka robot arm is very useful when



(a) Domain randomisation



(b) Obstacle avoidance

Fig. 8: Graphs showing the median of the reward ± 1 standard deviation over 10 runs with random starting seed. In the domain randomisation experiment, **a**, the agents can be seen learning at a somewhat similar rate, resulting in very consistent behaviours. The agents in the obstacle avoidance experiment, **b**, can be seen learning at a much more varied rate, resulting in more inconsistent behaviours.

operating in a space with obstacles. This can be observed, by the robot extending its arm and reaching around the wall, in order to lift the rock. This method of grasping would not be possible if 4-DOF control was used.

Videos and further detail on behaviours observed during experimentation can be found in Chapter 6, in the appended technical report.

VIII. Conclusion

This project has been focused on grasping rocks in a space environment such as Mars by using DRL. A novel approach for 6-DOF object grasping, using joint control and domain randomisation has been presented, using PPO to control a Franka Emika Panda robotic manipulator. Using this approach, a success rate of 91.51% on rock grasps is achieved in an simulated environment with high domain randomisation. Furthermore, it is shown that 6-DOF joint control allows for the Franka robot to grasp rocks behind obstacles, with a success rate of 88.67%, which allows for grasps that would not be possible with lower DOF control.

This paper therefore concludes that using DRL policy gradient methods to joint-control a 6-DOF robot arm is capable of grasping rocks in a space environment.

A GitHub repository containing the code for the project can be found at https://github.com/AntonBock/RL4RobotGrasping

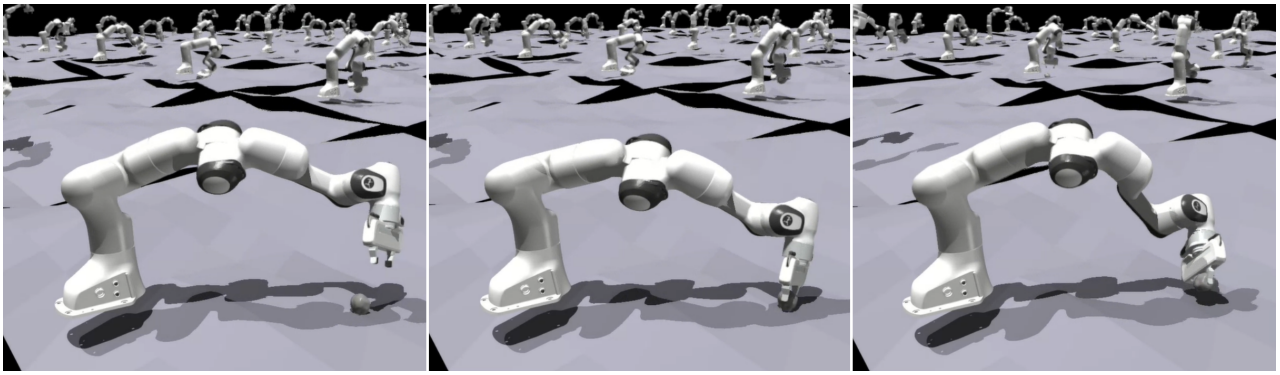


Fig. 9: The pick up from the domain randomisation experiment. A video of the behaviour can be seen at: https://youtu.be/5SpWWB_peWc. Note that the robot is able to pick up the rock while it is rolling around, showing a very robust grasping behaviour.

Acknowledgment

Thank you to Simon Bøgh, Associate Professor at the Department of Materials and Production at AAU, for assistance as supervisor, and the NVIDIA Isaac Gym community for giving technical insight into the simulation software.

References

[1] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, and J. Morimoto, "Deep learning, reinforcement learning, and world models," *Neural Networks*, 2022.

[2] M. Gou, H.-S. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu, "Rgb matters: Learning 7-dof grasp poses on monocular rgbd images," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 459–13 466.

[3] NASA, "Sample handling," (visited on 01-04-2022). [Online]. Available: https://mars.nasa.gov/mars2020/spacecraft/rover/sample-handling/

[4] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[5] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[6] Z. Chen, M. Lin, Z. Jia, and S. Jian, "Towards generalization and data efficient learning of deep robotic grasping," *arXiv preprint arXiv:2007.00982*, 2020.

[7] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[8] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[10] NASA, "Communications," (visited on 18-04-2022). [Online]. Available: https://mars.nasa.gov/mars2020/spacecraft/rover/communications/

[11] ESA, "Off-earth manufacturing: using local resources to build a new home," (visited on 20-05-2022). [Online]. Available: https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/Off-Earth_manufacturing_using_local_resources_to_build_a_new_home

[12] A. A. Shahid, L. Roveda, D. Piga, and F. Braghin, "Learning continuous control actions for robotic grasping with reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 4066–4072.

[13] H. Huang, X. Xie, L. Tang, H. Liu, N. Liu, and M. Li, "Autonomous collision avoidance sample grasping method for extraterrestrial exploration," *Acta Astronautica*, 2022.

[14] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1461–1466.

[15] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[16] P. Chen and W. Lu, "Deep reinforcement learning based moving object grasping," *Information Sciences*, vol. 565, pp. 62–76, 2021.

[17] Z. Ding, Y.-Y. Tsai, W. W. Lee, and B. Huang, "Sim-to-real transfer for robotic manipulation with tactile sensory," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6778–6785.

[18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[19] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[20] C. Scheiderer, N. Dorndorf, and T. Meisen, "Effects of domain randomization on simulation-to-reality transfer of reinforcement learning policies for industrial robots," in *Advances in Artificial Intelligence and Applied Cognitive Computing*. Springer, 2021, pp. 157–169.

[21] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[22] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[23] A. Orsula, "Deep reinforcement learning for robotic grasping from octrees," (visited on 22-05-2022). [Online]. Available: https://projekter.aau.dk/projekter/files/421582447/Deep_Reinforcement_Learning_for_Robotic_Grasping_from_Octrees.pdf

[24] S. Goossens, T. J. Sabaka, A. Genova, E. Mazarico, J. B. Nicholas, and G. A. Neumann, "Evidence for a low bulk crustal density for mars from gravity and topography," *Geophysical research letters*, vol. 44, no. 15, pp. 7686–7694, 2017.

[25] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[27] A. Serrano-Muñoz, N. Arana-Arexolaleiba, D. Chrysostomou, and S. Bøgh, "skrl: Modular and flexible library for reinforcement learning," *arXiv preprint arXiv:2202.03825*, 2022.