Autonomous Planetary Landing via Deep Reinforcement Learning and Transfer Learning

Giulia Ciabatti[1], Shreyansh Daftry[2] and Roberto Capobianco[1,3]
giulia.ciabatti8@gmail.com *

Abstract

The aim of this work is to develop an application for autonomous landing. We exploit the properties of Deep Reinforcement Learning and Transfer Learning, in order to tackle the problem of planetary landing on unknown or barely-known extra-terrestrial environments by learning good-performing policies, which are transferable from the training environment to other, new environments, without losing optimality. To this end, we model a real-physics simulator, by means of the Bullet/PyBullet library, composed by a lander, defined through the standard ROS/URDF framework and realistic 3D terrain models, for which we adapt official NASA 3D meshes, reconstructed from the data retrieved during missions. Where such model were not available, we reconstruct the terrain from mission imagery - generally SAR imagery. In this setup, we train a Deep Reinforcement Learning model - using DDPG - to autonomous land on the lunar environment. Moreover, we perform transfer learning on the Mars and Titan environment. While still preliminary, our results show that DDPG can learn a good landing policy, which can be transferred to other environments.

1. Introduction

Autonomous landing and navigation represent an important challenge for space exploration [11, 12]. While system-embedded Artificial Intelligence (AI) has made significant progress during the last few years, space exploration generally still follows a more traditional approach in practical applications [19, 1].

Recent researches have been carried out with promising results in the field of autonomous landing, like in the case of the applications of Deep Learning and Meta-Reinforcement Learning for autonomous lunar landing, presented, respectively, by Furfaro *et al*. [9] and by Scorsoglio *et al*. [22] or Deep Reinforcement Learning for six-degree-of-freedom Mars landing, presented by Gaudet *et al*. [5] and the Deep Reinforcement learning application to learn the parameters of the Adaptive Generalized ZEM-ZEV Feedback Guidance, presented by Furfaro *et al*. [10].

Inspired by these results, we tackle the autonomous planetary landing problem (in particular the terminal landing phase). In order to do so, we utilize a Deep Reinforcement Learning algorithm that allows us to handle high-dimensional and heterogeneous input data, such as vision-based navigation imagery from an RGB camera, LiDAR detections and pose data. Given that our action space is continuous, we use the Deep Deterministic Policy Gradient algorithm - DDPG [8].

In order to evaluate the performance of DDPG on this task, we develop a simulator exploiting the Bullet/PyBullet physical engine and adapting official NASA 3D terrain models - or reconstructing them from SAR imagery retrieved from missions, when none is available, as for the Titan environment. In order to perform Reinforcement Learning experiments, we additionally wrap this simulator in the OpenAI gym interface [2].

After evaluating the performances of the DDPG algorithm in our simulator, we preliminarly test the transfer learning performances on the reconstructed Titan environment. This was, to the best of our knowledge, the first time transfer learning was attempted for planetary landing.

2. Purpose Statement

The purpose of this paper is twofold. First, a pure Python-embedded, real-physics simulator for planetary landing is developed. This first task is necessary due to the lack of open-source frameworks and simulators for planetary landing - especially for the terminal landing phase - which could also be exploited to implement and train AI algorithms. Thus, the aim of this first part of work is to provide an open-source, easily customizable simulator dedicated to the study of these problems. Both the simulator and the lander model are also fully compatible with the ROS[1] framework. Second, a Deep Reinforcement Learn-

*[1]Department of Computer, Control and Management Engineering, Sapienza University of Rome, Italy. [2]Jet Propulsion Laboratory, California Institute of Technology, USA. [3]Sony AI

[1] https://www.ros.org/

ing approach is exploited in order to perform vision-based autonomous planetary landing and domain transfer on the developed simulator. In particular, we use the Deep Deterministic Policy Gradient - DDPG [8]. The aim of this second part is (1) to tackle the difficulty in controlling a robotic system on an extra-terrestrial environment - in this case during the terminal landing phase - from remote and in absence of positioning systems - such as GPS - available on Earth and (2) to test the ability to transfer the learned knowledge from a known domain to another one. A good performance in transfer learning can grant the ability to perform efficient landing also on unknown domains, especially for those where precise modeling and simulations are impossible due to the lack of data/imagery and their difficult retrieval - e.g. asteroids, unknown or barely-known planet and satellite surfaces, etc [14, 21]. Also, efficient transfer learning applications may represent a solution to "fill the reality gap" when transferring from the simulation domain to the real domain, as the reality gap is oftentimes an obstacle to implement many robotics solutions in real world.

3. Theory Background: Deep Reinforcement Learning

We cast our problem as an episodic Deep Reinforcement Learning [24, 15] task. At each time-step t, our agent/lander interacts with a fully-observable environment $\mathcal{E}$ in the form of an Markov Decision Problem. As such, it has to choose an action $a_t \in \mathcal{A}$, based on the observation $s_t \in \mathcal{S}$, receiving back from the environment a reward r_t and a new state s_{t+1}. In our environments, all the actions are real-valued: $\mathcal{A} \subset \mathbb{R}^N$.
The policy π determines our agent's behaviour, mapping states to actions: $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The action-value function describes the expected return obtained after executing an action a_t in state s_t and then following π:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[\sum_{i=t}^{T} \gamma^{i-t} r_i | s_t, a_t] \tag{1}$$

3.1. Deep Deterministic Policy Gradient - DDPG

In order to handle a continuous action space, we use the Deep Deterministic Policy Gradient - DDPG - first introduced by Timothy P. Lillicrap *et al.* [8]. The basic architecture of the DDPG algorithm consists of an actor-critic model. The critic is learned by means of the Bellman equation:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]], \tag{2}$$

rewritten, in case of deterministic target policy, as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\mu}(s_{t+1}, \mu_{s_{t+1}})]], \tag{3}$$

making it possible to learn Q^{μ} off-policy, while the actor exploits the parameterized function $\mu(s|\theta^{\mu})$, which defines a deterministic map from state to specific action, in accordance to the policy π, and is updated by means of the chain rule to the expected return:

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s_t \sim \rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t, \theta^{\mu})}] \tag{4}$$

$$= \mathbb{E}_{s_t \sim \rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_t}] \tag{5}$$

where J is the initial distribution.
The DDPG algorithm makes use of the Ornstein-Uhlenbeck process [25] to improve exploration efficiency in physical control problems.
DDPG allows us to better handle large observation spaces, in particular the "raw" input from the RGB camera pixels, which has a dimension of 256x256, making use of a Replay Buffer and a separate target network to calculate y_t, first introduced by Mnih *et al.* [15], where:

$$y_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^{Q}) \tag{6}$$

and the Loss function to minimize in order to optimize the function approximators θ^{Q}:

$$L(\theta^{Q}) = \mathbb{E}_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E}[(Q(s_t, a_t|\theta^{Q}) - y_t)^2] \tag{7}$$

3.2. Transfer Learning

Generally, Transfer Learning tasks consist in exploiting the acquired knowledge in a task and /or in a domain to perform similar tasks in relatable domains, i.e. domain transfer.

In our experiments, we test the capacity of our lander/agent to transfer the knowledge learned in a lunar landing task to a general planetary landing, which presents different terrain morphology and different physical conditions, in particular, for this first application, a different gravitational acceleration.

Good-performing Transfer Learning applications may represent a valid solution for many robotics problems, in particular for the ones involving not-well-known domains [4], for which data and imagery lack and where robotic systems need to be fully autonomous - e.g. for interplanetary exploration.

4. Simulator Setup

The simulator for planetary landing is entirely built in Python. In particular, it is developed using the PyBullet/Bullet library [3], that allows to implement an effective physical engine. Our purpose is to develop an open-source simulator that would allow to effectively implement not only vision-based navigation and proximity operations, but also physical interactions between spacecraft/lander and environment, since, even after some detailed research, we

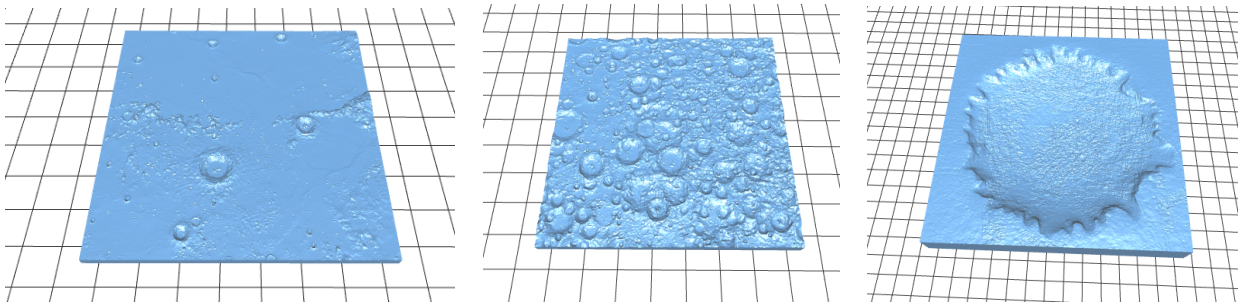

Figure 1. NASA 3D Model of (a) Moon's Near Side (b) Moon's Far Side and (c) Mars' Victoria Crater

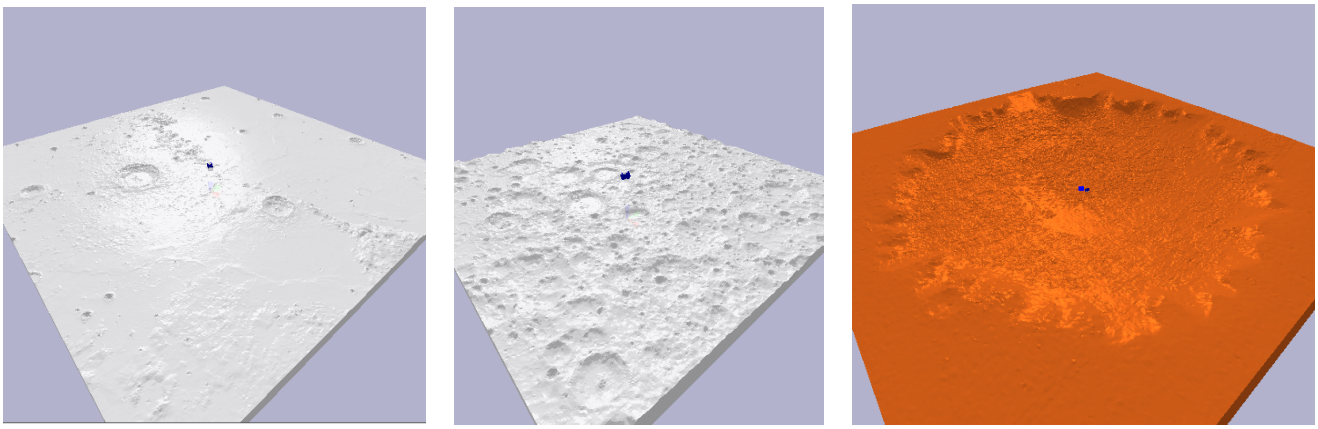

Figure 2. Terrain 3D Simulation of (a) Moon's Near Side (b) Moon's Far Side and (c) Mars' Victoria Crater

found it very difficult to find one. Our simulator includes realistic 3D terrain models and a lander model, developed using the classical pipeline to build a visual robot model. As previously stated, the simulator is compatible with ROS and is wrapped in the OpenAI gym [2] interface.

4.1. Terrain Simulation

In order to effectively implement the training algorithm, it is necessary to dispose of a real-physics environment simulator to model the force actions - such as collisions and gravitational field - and a realistic, heterogeneous terrain, that has to be wide enough to allow the change of initial conditions and three-dimensional to simulate different heights and roughness. At this purpose, we adapt the 3D mesh models provided by the official NASA GitHub repository [16] for the Moon and Mars terrains - mesh models and model descriptions are also available here [17].

The first terrain we use is a lunar landscape, in particular the "near side" of the Earth's Moon, as shown in Figure 1(a). This side is smoother since the craters were filled by the large lava flows billions of years ago. A preliminary training is performed on this model, also to test the soundness of the Deep Reinforcement Learning model-simulator interactions. A longer training phase - the "main" training - is performed on the lunar "far side" terrain. This model presents a dense distribution of craters and significant roughness, as illustrated in Figure 1(b).

Mars' Victoria Crater 3D model was first chosen and adapted to test Transfer Learning performances. Figure 1(c). This landscape simulates the 800 m wide crater on the Martian surface and the terrain presents a significant roughness, recalling the Moon's far side scenario.

The Moon and Mars 3D meshes have been appropriately scaled, textured and hence used to generate the height map in the simulator workframe and, thus, the physical interactions between environment and lander by means of the PyBullet/Bullet library. The final 3D terrain rendering in the simulator for the Moon's Near Side, Far Side and Mars' Victoria Crater respectively is represented in Figure 2(a), 2(b), and 2(c).

4.1.1 Titan Terrain

In order to test our model's Transfer Learning performance on a semi-unknown environment, we choose to try to reconstruct Saturn's moon Titan terrain. This choice is dictated by NASA's interest in future missions to explore Titan, such as the Dragonfly mission [13], in order to collect important samples. In particular, we try to model the heterogeneous Polar Lake district - Figure 3(a) - which represents an important landing site for sample gathering, because of its peculiar morphology due to the presence of small, densely dis-

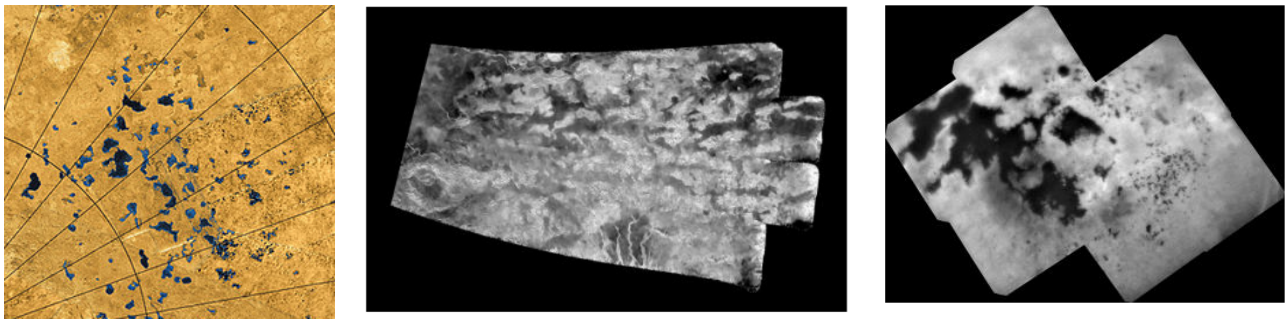

Figure 3. (a) Titan's Polar Lacus Zone, and examples of the SAR imagery retrieved by NASA's Cssini-Huygens mission from (b) Titan's Xanadu Annex region and (c) Titan lakes

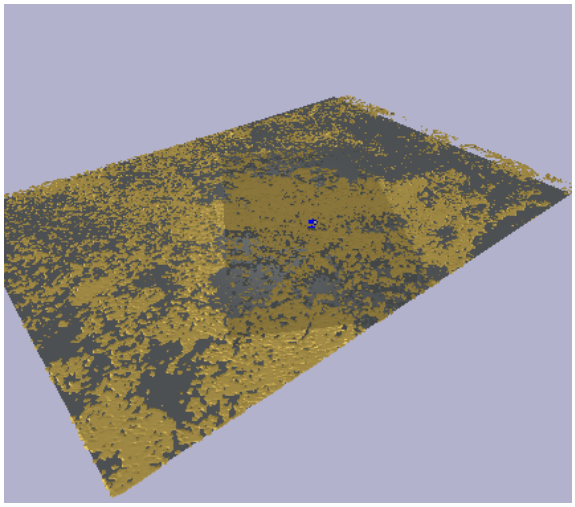

Figure 4. Titan's Terrain 3D Simulation

tributed hydrocarbon seas [23].

Since 3D accurate meshes - such as the ones we use to generate Moon and Mars terrains - are not available for Titan, we try to reconstruct a realistic landscape starting from the official Cassini-Huygens mission imagery - generally SAR imagery [20], Figure 3(b) and 3(c) - provided by NASA [18]. Starting from such imagery, a height map is reconstructed as a 3D mesh. The generated mesh is then rendered to simulate the terrain, as previously done for Moon and Mars – see Figure 4. Note: the blue "dot" at the center of the terrains represents the lander.

4.2. The Lander

The lander model is defined in the Unified Robot Description Format - URDF, following the classical ROS pipeline to build a visual robot model, i.e. subsequently attaching links to the base through joints, and thus modeling also collisions, mass and inertia. In particular, the lander model is simply defined as a central body as base, composed by a cylindrical and a spherical bodies, defined through meshes, with four links as legs, attached to the base through fixed-type joints and a sensor box. Four propulsive forces are defined as external forces, acting at the edge of the base, equidistant from the center of mass. We find that this simple vertical lifting force model can best simulate a generic propeller's resultant force.

The lander model is provided with two sensors: an RGB camera and a LiDAR, which are virtually attached to the sensor box. In Figure 5(a), it is shown an example of the simulator graphical rendering of the lander model landing on the Moon's far side environment: the up-left box shows the point of view from the lander's RGB camera, while the green "beam cone" represents the LiDAR detection (green rays mean "hit": the LiDAR is sensing another physical object, i.e. the terrain. "Miss" rays are red). In Figure 5(b), it is shown the landing on Titan's surface: the dark-grey spots representing the hydrocarbon lakes are classified as "miss" by the LiDAR, i.e. as non-landing zones. It is also possible to detect and visualize the fragmented nature of the terrain combining a Depth Camera and a Segmentation Mask - respectively second and third box on the left from above.

5. Experiment Setup

Several experiments are executed on the four terrains. In particular, we try several training sessions on the lunar scenario switching between the two lunar terrains for hyperparameter fine tuning in accordance to the task.

In particular, our DDPG implementation presents a lower learning rate with respect to the original paper *et al.* [8] and we set the "theta" parameter, the mean and the standard deviation to a higher value for our Ornstein-Uhlenbeck process [25] implementation. Transfer learning is then executed on Mars' Victoria Crater and the Titan terrain. The initial state is randomly initialized - with the exception of the altitude along the z-axis - in order to let the lander/agent explore the environment configurations as widely as possible. The agent's task is to land upright and stand on the four legs, without flipping over. During the aerial phase, the agent has to keep as steady as possible, modulating the four vertical thruster forces in order to

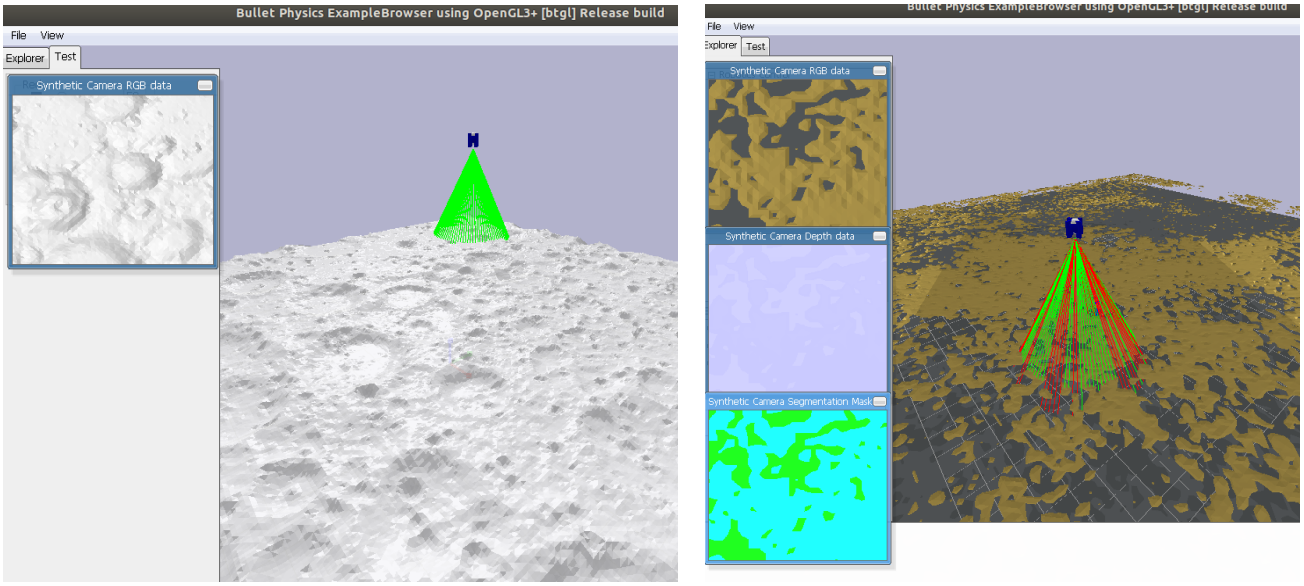

Figure 5. Examples of Simulator's graphical rendering from landing on (a) Moon's Far Side and (b) Titan

slow down without gaining altitude back again, and avoid no-landing zones if necessary by steering aside, but without rolling on itself.

The action space is continuous and four-dimensional - in fact, the lander is provided with four thrusters. The observation space is composed by the RGB camera pixels - a 256x256-dimensional image and the four RGB channels, plus the alpha channel - the LiDAR input, composed by a hundred rays, indicating the distance between the lander and the hit object (here, the terrain), a 3D vector for the position and the quaternion for the orientation - summing up to a 262251-dimensional observation space.

5.1. Training Sessions

At first, we train the DDPG model on the Moon's Near Side for 200 episodes, order to test the Deep Reinforcement Learning model-simulator interactions. Even if the training is short for this kind of task, the model seems to perform very well. In Figure 6(a) is reported a plot of the average episodic reward for this case.

We train the model on the Moon's Far Side keeping the same hyper-parameters and the same number of episodes. It is clearly visible in Figure 6(b), even if the average episodic reward visibly increases, how the agent struggles more to find the optimal action policy, especially in the initial episodes: this is due to the heterogeneity of the terrain and the consequential higher variation of the observation space - i.e. RGB camera and LiDAR input.

We then train the model in a "main training" session. We set the starting altitude at double the value of the first two experiments and train the model for a thousand episodes on the Moon's Far Side. This experiment setup is much closer to a real simulation of a terminal landing phase, in particular for what concerns physical interactions with the environment and sensor input. The average episodic reward curve presents a sharp increase up to the 250th episode and then mainly stabilizes, presenting oscillations - Figure 6(c).

5.2. Transfer Learning

Considering the encouraging results of the training sessions, we set up two domain transfer learning tasks. The initial conditions and the landing task are kept as they were during the training sessions, but the environment is changed in order to simulate at first a Martian environments, using the Victoria Crater 3D model, and then a plausible Titan environment through our terrain, reconstructed by means of SAR imagery converted to a 3D model. The gravitational acceleration is changed to represent, respectively, Mars' and Titan's real values. As a first approximation, the atmosphere is neglected for these terminal landing phase experiments.

5.2.1 Transfer Learning Results

We test our lander performances on Mars and Titan. Even in a completely different environment, both from a visual point of view and for what concerns the interactive forces of the gravitational field - on Mars the gravitational force is $3.711 m/s^2$, while the Moon's - where the agent was trained - is $1.62 m/s^2$. The lander manages to land correctly, keeping a steady pose during the aerial landing phase, keeping steadily on its legs at touchdown - without flipping over and rolling on itself - and maximizing the reward. In Figure 7(a) and 7(b) the reward on 20 episodes for Mars and Titan landing tasks, respectively, are plotted. It is interesting to notice

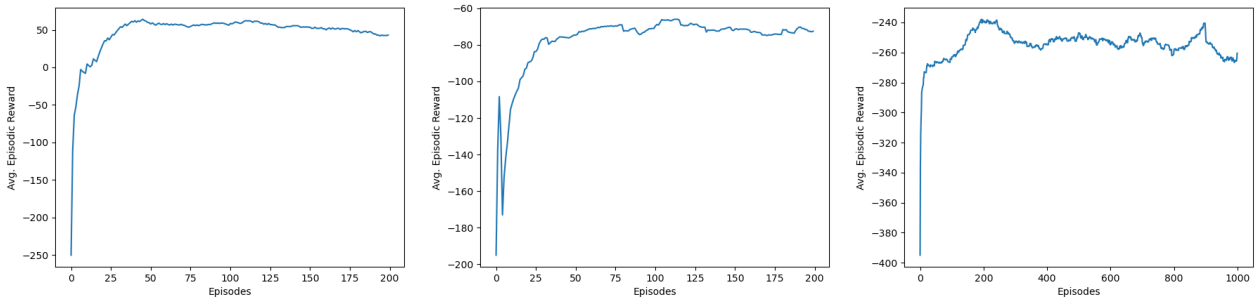

Figure 6. Preliminary training results for (a) 200 episodes on the Moon's Near Side, (b) 200 episodes on the Moon's Far Side and (c) 1000 episodes on the Moon's Far Side at double the altitude.

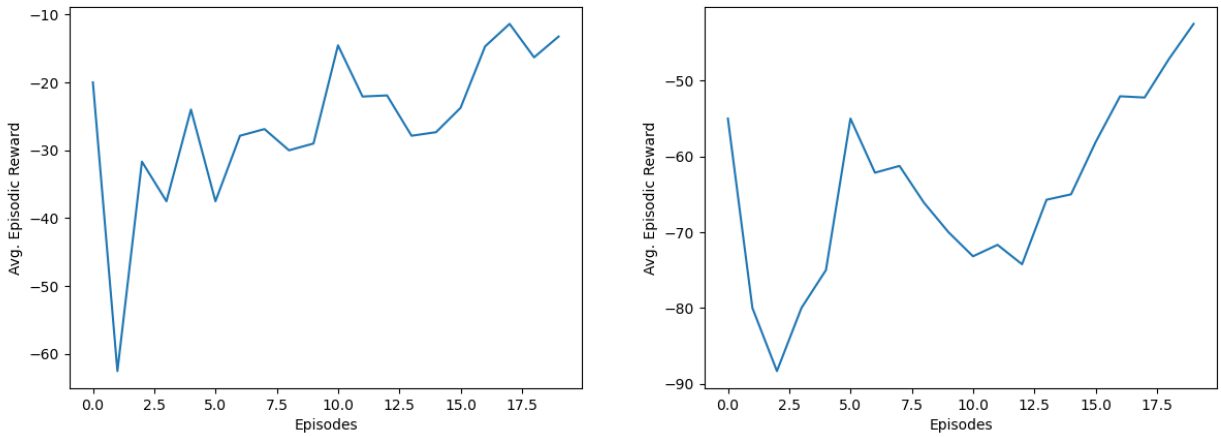

Figure 7. Transfer Learning reward for (a) Landing on Mars and (b Landing on Titan)

how the reward values for the Mars scenario tends to be closer to the case of the Moon's Near Side scenario: this is due to the fact that Mars Victoria Crater's terrain, with the exception of the boundary around the crater, is generally smoother than the Moon's far side, where the main training was executed, presenting a roughness quite similar to the Moon's Near Side terrain and leading to similar - but slightly lower - reward scores. On the other hand, Titan's terrain, resembling a Lacus zone, seems very heterogeneous, but the landing sites are generally smoother than the Moon's Far Side: Thus, the reward scores a little lower than the one of the homogeneous, relatively flat Victoria Crater, but higher that the rougher Moon terrain.

In Figure 8(a) and 8(b), the approaching and the touchdown phase are represented for Mars landing, and, analogously, for Titan landing in Figure 9(a) and 9(b). Note that the lander turns to red when detecting contact with the ground.

6. Conclusions and Future Work

In this work, we have presented a newly implemented simulator to model real-physics phenomena by means of the Bullet/PyBullet physical engine and to render realistic terrain 3D models. We adapted official NASA terrain 3D models where available or reconstructed a realistic Titan 3D terrain through imagery retrieved by NASA's Cassini-Huygens mission. We have defined a lander in ROS/URDF to interact with the simulator. We have implemented an algorithm of Deep Reinforcement Learning - the DDPG algorithm - in order to train our lander/agent to perform a terminal-phase landing task. We have exploited the knowledge learned on the lunar terrain in order to test a preliminary transfer learning task on two different environments - Mars and Titan.

Our future work includes improving the lander/agent performances, in particular for different transfer learning tasks. In order to do so, we are testing several Deep Reinforcement Learning models (such as TD3, first presented in the paper by Fujimoto *et al*. [6] and a SAC, first presented in the paper by Haarnoja *et al*. [7]. We also aim to improve our simulator to include more terrain models and environments - in particular for newly explored planetary environments - and to implement obstacle avoidance. A new simulator is also being studied in order to consider the entire EDL phase, including also the atmosphere where needed.

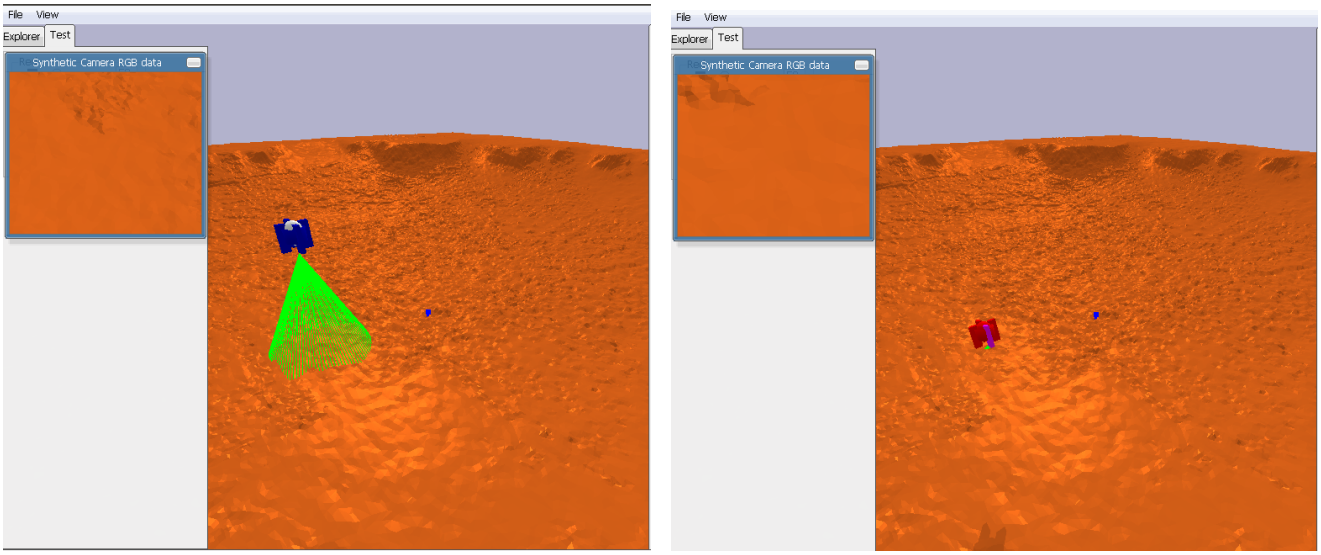

Figure 8. Testing transfer learning for landing on Mars during (a) approach phase, and (b) at touchdown.

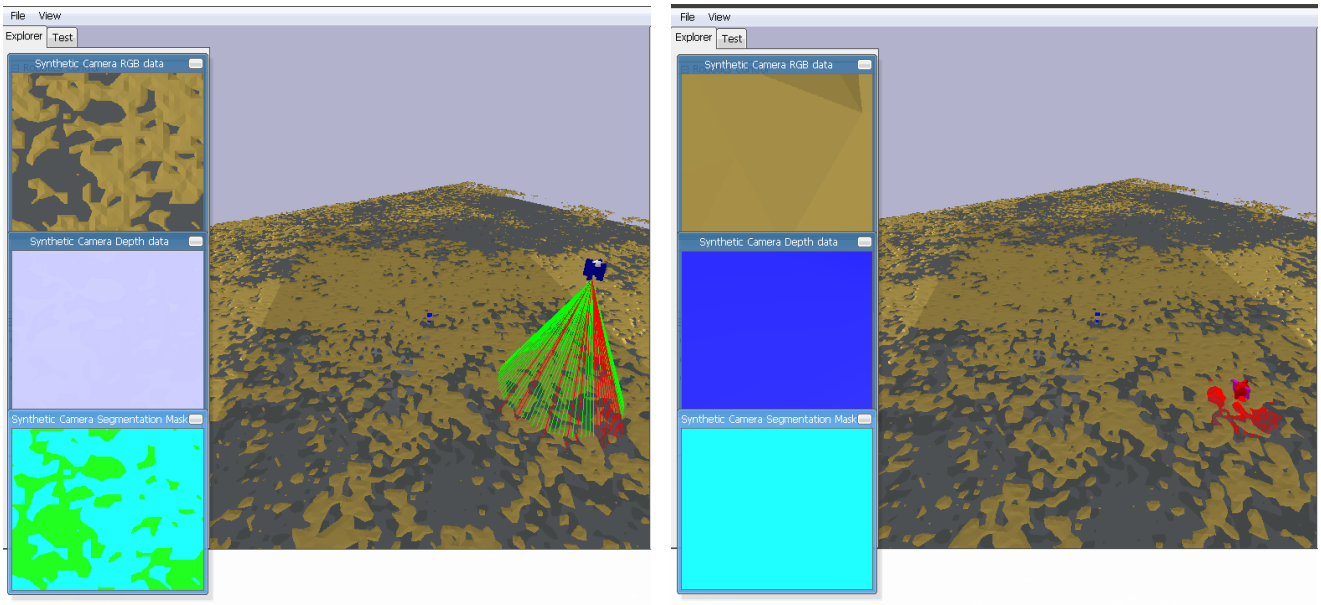

Figure 9. Testing transfer learning for landing on Mars during (a) approach phase, and (b) at touchdown.

References

[1] Neil Abcouwer, Shreyansh Daftry, Siddarth Venkatraman, Tyler del Sesto, Olivier Toupet, Ravi Lanka, Jialin Song, Yisong Yue, and Masahiro Ono. Machine learning based path planning for improved rover navigation (pre-print version). *arXiv preprint arXiv:2011.06022*, 2020.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Erwin Coumans. Bullet physics library. *Open source: bulletphysics. org*, 15:5, 2013.

[4] Shreyansh Daftry, J Andrew Bagnell, and Martial Hebert. Learning transferable policies for monocular reactive mav control. In *International Symposium on Experimental Robotics*, pages 3–11. Springer, 2016.

[5] Brian Gaudet et al. Deep reinforcement learning for six degree-of-freedom planetary powered descent and landing. *arXiv:1810.08719*, 2018.

[6] Fujimoto et al. Addressing function approximation error in actor-critic methods. *arXiv:1802.09477*, 2018.

[7] Haarnoja et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv:1801.01290*, 2018.

[8] Timothy P. Lillicrap et al. Continuous control with deep reinforcement learning, 2016.

[9] Roberto Furfaro and Richard Linares. Deep learning for autonomous lunar landing. 08 2018.

[10] Roberto Furfaro, Andrea Scorsoglio, Richard Linares, and Mauro Massari. Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach, 03 2020.

[11] Andrew Johnson, Adnan Ansar, Larry Matthies, Nikolas Trawny, Anastasios Mourikis, and Stergios Roumeliotis. A general approach to terrain relative navigation for planetary landing. In *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, page 2854, 2007.

[12] Andrew E Johnson and James F Montgomery. Overview of terrain relative navigation approaches for precise lunar landing. In *2008 IEEE Aerospace Conference*, pages 1–10. IEEE, 2008.

[13] Ralph D Lorenz, Elizabeth P Turtle, Jason W Barnes, Melissa G Trainer, Douglas S Adams, Kenneth E Hibbard, Colin Z Sheldon, Kris Zacny, Patrick N Peplowski, David J Lawrence, et al. Dragonfly: A rotorcraft lander concept for scientific exploration at titan. *Johns Hopkins APL Technical Digest*, 34(3):14, 2018.

[14] Larry Matthies, Shreyansh Daftry, Brandon Rothrock, Anthony Davis, Robert Hewitt, Evgeniy Sklyanskiy, Jeff Delaune, Aaron Schutte, Marco Quadrelli, Michael Malaska, et al. Terrain relative navigation for guided descent on titan. In *2020 IEEE Aerospace Conference*, pages 1–16. IEEE, 2020.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[16] NASA. https://github.com/nasa/nasa-3d-resources.

[17] NASA. https://nasa3d.arc.nasa.gov/models.

[18] JPL NASA. https://photojournal.jpl.nasa.gov/target/titan.

[19] Masahiro Ono, Brandon Rothrock, Kyohei Otsu, Shoya Higa, Yumi Iwashita, Annie Didier, Tanvir Islam, Christopher Laporte, Vivian Sun, Kathryn Stack, et al. Maars: Machine learning-based analytics for automated rover systems. In *2020 IEEE Aerospace Conference*, pages 1–17. IEEE, 2020.

[20] F Paganelli, Michael A Janssen, B Stiles, R West, Ralf D Lorenz, Jonathan I Lunine, Stephen D Wall, P Callahan, RM Lopes, E Stofan, et al. Titan's surface from cassini radar sar and high resolution radiometry data of the first five flybys. *Icarus*, 191(1):211–222, 2007.

[21] Aaron Schutte, Jeff Delaune, Evgeniy Skylanskiy, Robert Hewitt, Shreyansh Daftry, Marco B Quadrelli, and Larry Matthies. Integrated simulation and state estimation for precision landing on titan. In *2020 IEEE Aerospace Conference*, pages 2–13. IEEE, 2020.

[22] Andrea Scorsoglio, Roberto Furfaro, Richard Linares, and Brian Gaudet. Image-based deep reinforcement learning for autonomous lunar landing. 01 2020.

[23] Ellen R Stofan, Charles Elachi, Jonathan I Lunine, Ralf D Lorenz, B Stiles, KL Mitchell, S Ostro, L Soderblom, C Wood, H Zebker, et al. The lakes of titan. *Nature*, 445(7123):61–64, 2007.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[25] George E. Uhlenbeck and Leonard S. Ornstein. On the theory of the brownian motion. *Physical review*, 1930.