OffWorld Gym: Open-Access Physical Robotics Environment for Real-World Reinforcement Learning Benchmark and Research

Ashish Kumar
OffWorld, Pasadena, CA, USA

John B. Lanier
OffWorld, Pasadena, CA, USA

Qiaozhi Wang
OffWorld, Pasadena, CA, USA

Alicia Kavelaars
OffWorld, Pasadena, CA, USA

Ilya Kuzovkin
OffWorld, Pasadena, CA, USA
ilya.kuzovkin@offworld.ai

Abstract

Success stories of applied machine learning can be traced back to the datasets and environments that were put forward as challenges for the community. The challenge that the community sets as a benchmark is usually the challenge that the community eventually solves. The ultimate challenge of reinforcement learning research is to train *real* agents to operate in the *real* environment, but there is no common real-world benchmark to track progress of RL on physical robotic systems. To address this issue we have created OffWorld Gym – a collection of real-world environments for reinforcement learning in robotics with free public remote access. In this work we introduce the system and present experimental results that demonstrate the feasibility of learning on a real robotic system. We train a mobile robot end-to-end to solve simple navigation task relying solely on camera input and without the access to location information. Close integration into existing ecosystem allows the community to start using OffWorld Gym without any prior experience in robotics and takes away the burden of managing a physical robotics system, abstracting it under a familiar API. To start training, visit `https://gym.offworld.ai`.

1 Introduction

Reinforcement learning [1] offers a strong framework to approach machine learning problems that can be formulated in terms of *agents* operating in *environments* and receiving *rewards*. Coupled with the representational power and capacity of deep neural networks [2], this framework has enabled artificial agents to achieve superhuman performance in Atari games [3], Go [4], and real time strategy games such as Dota 2 [5] and StarCraft II [6]. Deep reinforcement learning has been successfully applied to simulated environments, demonstrating the ability to solve control problems in discrete [7, 8, 9] and continuous [10, 11] action spaces, perform long-term planning [12, 13], use memory [14], explore environments efficiently [15], and even learn to communicate with other agents [16]. These and many other capabilities proven by deep reinforcement learning (DRL) methods [17] hold an inspiring promise of the applicability of DRL to real world tasks, particularly in the field of robotics.

Despite the fact that many consider operations in real world settings to be the ultimate challenge for reinforcement learning research [18], the search for solutions to that challenge is being carried

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual

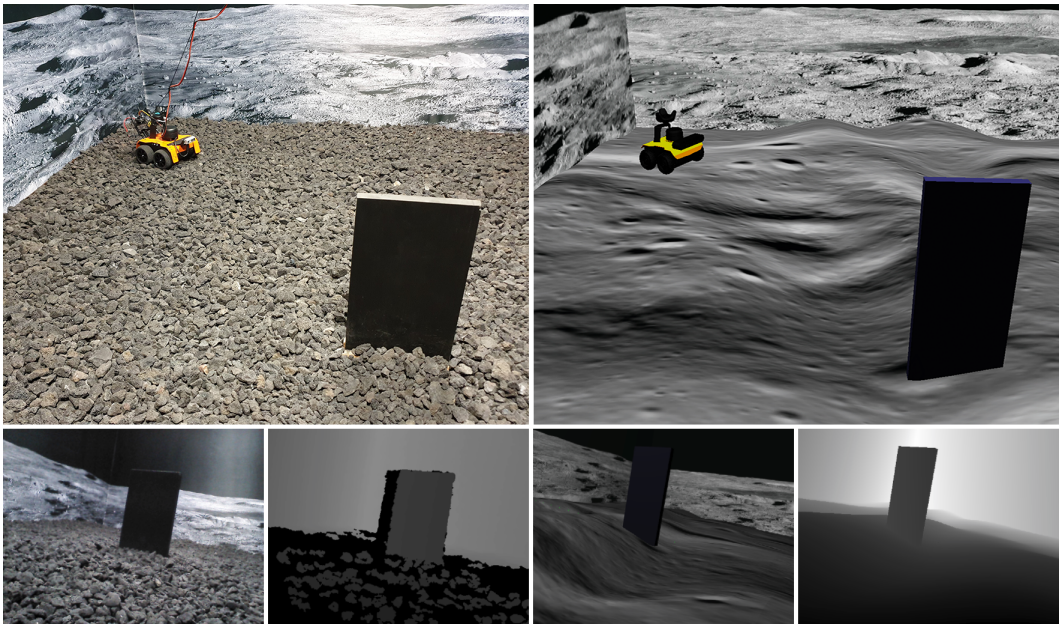

Figure 1: The top row shows the real (left) and the simulated (right) instances of the `MonolithDiscrete` environment. The users have same access to both via the same API interface, allowing for a seamless transition between a simulated and a real versions of an environment. The bottom row shows RGB and depth inputs in both instances from the robot's perspective.

out predominantly in simulated environments [19, 20, 21, 22, 11, 23, 24, 8, 25, 26]. This focus on simulated environments as opposed to physical ones can be attributed to the high difficulty of training in real world environments. High sample complexity of modern DRL methods makes collecting a sufficient amount of observations on a real robotic system both time consuming and challenging from a maintenance standpoint. As a result, the training of real world agents has been approached in a variety of ways, both directly [27, 28, 29, 30, 31, 32, 33, 34] and using simulation-to-real transfer learning to minimize experience needed in a real setting [35, 36, 37]. Recent works on imitation learning [30, 38, 39, 40, 41, 42] and reduction of sample complexity [43, 44, 45, 46, 47] also provide a path towards making training in real feasible.

From the previous major successes of machine learning, we see that the goal the community sets as a benchmark is usually the goal that the community eventually solves. Thus to solve the hard problems in RL for the real world, the RL community must add real-world environments to their set of benchmarks. Adding a common physical benchmark environment to the set of canonical reference tasks such as Atari games [48] and MuJoCo creatures [49] would enable future research to take into account, and hopefully accelerate, the applicability of RL methods to real world robotics.

2 A Physical Robotic Environment for Reinforcement Learning

In this work, we present four real-world, publicly-accessible, remote-operated robotics RL environments from the OffWorld Gym framework[1], consisting of two tasks in both discrete and continuous control formulations. These environments conform to the OpenAI `gym` API while remote-controlling a real robot maintained by the authors and address general robotics challenges such as locomotion, navigation, planning, and obstacle avoidance. In each task, the robot must reach a visual beacon while relying solely on visual input. Simulated variants of these environments are also provided. For an overview of similar systems please see the "Related work" section of supplementary materials.

The first pair of environments features a navigation task in a walled enclosure ($3 \times 4 \times 2$ meters) in which a wheeled robot has to traverse an uneven Moon-like terrain to reach an alluring visual beacon introduced by Kubrick et al. [50]. The robot receives 320×240 RGBD camera input and nothing else. The environment control server tracks the robot location using an HTC Vive™ tracker and two

[1] A video of an agent controlling the robot in a real environment: `https://youtu.be/HDViFqvB3Co`

Figure 2: The real environment with obstacles and a sparse reward for reaching the monolith. An agent has to solve visual obstacle avoidance to complete the task.

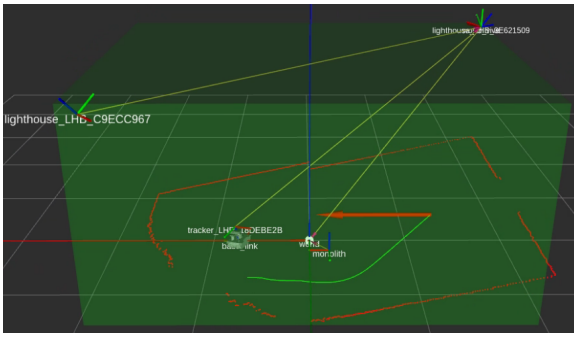

Figure 3: Object tracking for the environment control system. The two `lighthouse` components track robot's location. The monolith is installed in the middle of the world coordinate frame.

base stations (Figure 3) but this information is not available to the learning agent and is only used internally by the environment control script to calculate rewards, reset the environment and achieve new initial locations at the start of each episode.

The `MonolithDiscreteReal` environment has a discrete action space with four basic actions: left, right, forward, backward, each applying a fixed velocity to the robot with a 2-second step duration. The continuous action space variant, `MonolithContinuousReal` provides smooth controls to the linear and angular velocities of the robot. A sparse reward of $+1.0$ is assigned when the robot (Husarion Rosbot [51], dimensions $20.0 \times 23.5 \times 22.0$ cm) approaches the monolith within a radius of 40.0 cm. The environment is reset upon successful completion of the task, reaching the limit of 100 steps or approaching the boundary of the environment. After each reset, the robot is moved to a random position and orientation. Figure 1 (left) shows the environment and the input stream that the agent receives. The second pair of environments currently under development inherits all of the characteristics of the first one, but is made more challenging by adding obstacles that the robot has to avoid (see Figure 2). Developing a robust solution to this task would demonstrate the applicability of reinforcement learning to the problem of visual obstacle avoidance in the absence of a map and location information.

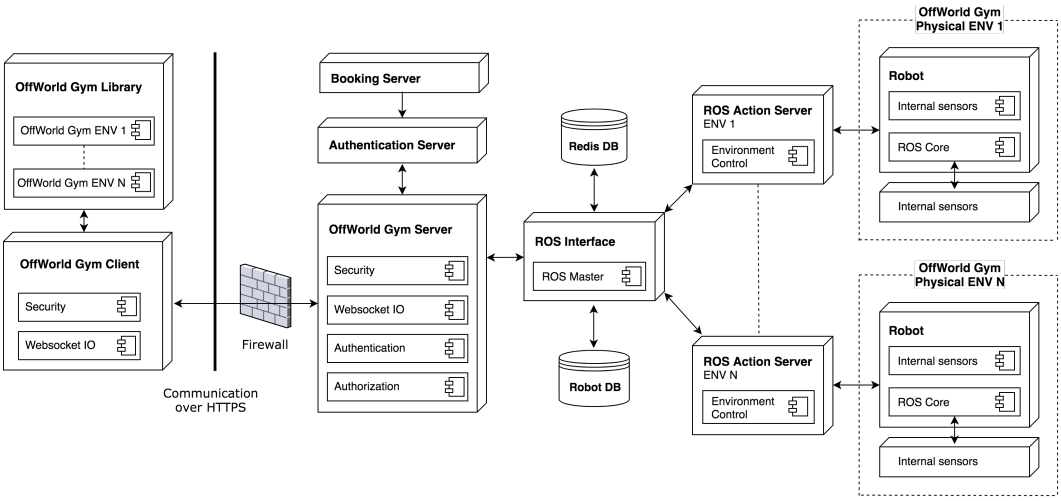

Figure 4: The diagram of the major components of the system architecture of OffWorld Gym

As we further expand the OffWorld Gym framework's collection by building additional enclosures with various robotic tasks, we will cover a wide range of challenges for robotic systems, provide stable benchmarks, and make a step toward applicability of developed solutions to real world and industrial applications. OffWorld Inc. is committed to providing long-term free support and maintenance of the physical environments.

2.1 The architecture of the system

OffWorld Gym consists of three major parts: (a) a Python library that is running on the client machine, (b) the server that handles communication, resource management and control of the environment (reward, episode reset, etc.), (c) the physical enclosure that provides power and network infrastructure, and (d) the robot itself. Figure 4 gives an overview of the architecture, its components and interactions. Please see supplementary materials for the details on hardware specification.

The OffWorld Gym library provides the API to access the environments. The client side of the library forwards RL agent's actions to the gym server. The server processes the actions and forwards them to the robot. The robot completes the requested action (movement, position reset, etc) and sends telemetry back to the action server. The server processes the telemetry and creates the state variable that is sent back to the client as an observation for the agent. The control logic and the learning process are executed on user's workstation and the user is thus free to explore any algorithmic solutions and make use of any amount of computational resources available at their disposal.

3 Experimental Validation

The purpose of our experimental work is threefold: to demonstrate the soundness of the system and feasibility of learning, provide the first set of benchmark results for modern RL algorithms, and to empirically estimate the sample complexity of learning a visual navigation task end-to-end on a real robot from camera inputs directly to actions.

We trained Double DQN [52] and Soft Actor-Critic (SAC) [53] agents in the discrete action space variant of `MonolithReal` environment and a SAC agent in the continuous variant of the same environment. Figure 5 shows the learning curves for all three experiments.

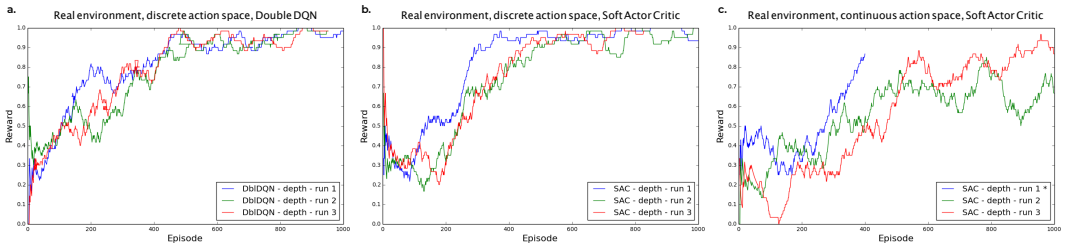

Figure 5: Learning curves in discrete and continuous variants of the environment. **a.** Double DQN trained end-to-end with discrete actions space: `https://youtu.be/HDViFqvB3Co` **b.** Soft Actor-Critic solves the discrete action space variant of the environment. **c.** SAC achieves intelligent behavior in the continuous variant of the environment: `https://youtu.be/Nbgq6c78yJg`.

The Double DQN agent's neural network architecture consisted of a 320×240 visual (depth channel only) input, three convolutional layers each with four 5×5 filters and max pooling, followed by two fully connected layers of size 16. Leaky ReLU activations were used. The network has 3381 trainable parameters. The Adam optimizer was used with a learning rate of 0.001 and a batch size of 32. The circular replay buffer was of size $25,000$, and experience was gathered in an epsilon-greedy fashion, where epsilon was linearly annealed from 0.9 to 0.1 over the first $40,000$ steps. The discount factor was 0.95.

The SAC agent's neural network architecture has 84×84 visual (depth channel only) input, followed by three convolutional layers with 16, 32, and 64 filters of sizes 8×8, 4×4, and 1×1, respectively. This was followed by two fully connected layers of size 64. ReLU activations were used. In total the network has 757,236 trainable parameters. The Adam optimizer was used with a learning rate of 0.0003 and a batch size of 1024. Updates were performed after every experience step. The circular replay buffer was of size $500,000$. α was learned to match an entropy target of $0.2 * -\log(1/|A|)$ for discrete spaces and $0.2 * -\dim(A)$ for continuous spaces. The discount factor was 0.99.

The results confirm the overall soundness of the proposed system and demonstrate feasibility of learning. We count on community involvement to evaluate other existing algorithms, explore different architectures and methods in order to identify the state of the art algorithms for the tasks presented in OffWorld Gym and supplement the experimental evaluation of newly proposed methods with a set of challenge in the real physical world.

Acknowledgments

The authors would like to thank Eric Tola, Matt Tomlinson, Matthew Schwab and Piyush Patil for their help with the mechanical and electrical design and implementation.

References

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

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[3] 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, 2015.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[5] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

[6] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, et al., and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strateg` `y-game-starcraft-ii`, 2019.

[7] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[12] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346, 2014.

[13] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sébastien Racanière, Théophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, et al. An investigation of model-free planning. *arXiv preprint arXiv:1901.03559*, 2019.

[14] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[15] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[16] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. *arXiv preprint arXiv:1810.11187*, 2018.

[17] Yuxi Li. Deep reinforcement learning. *arXiv preprint arXiv:1810.06339*, 2018.

[18] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[19] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[20] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[21] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[23] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[24] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[25] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.

[26] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. 2018.

[27] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[28] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

[29] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[30] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[31] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 79–86. IEEE, 2017.

[32] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[33] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. *arXiv preprint arXiv:1809.07731*, 2018.

[34] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[35] Andrei A Rusu, Mel Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[36] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017.

[37] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[38] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[39] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[40] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[41] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[42] Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019.

[43] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

[44] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *arXiv preprint arXiv:1809.05214*, 2018.

[45] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

[46] Himanshu Sahni, Toby Buckley, Pieter Abbeel, and Ilya Kuzovkin. Addressing sample complexity in visual tasks using hindsight experience replay and hallucinatory gans. 2019.

[47] Giulia Vezzani, Abhishek Gupta, Lorenzo Natale, and Pieter Abbeel. Learning latent state representation for speeding up exploration. *arXiv preprint arXiv:1905.12621*, 2019.

[48] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[49] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[50] Stanley Kubrick, Arthur Clarke, Keir Dullea, Gary Lockwood, Geoffrey Unsworth, and Ray Lovejoy. 2001: a space odyssey. 1968.

[51] Husarion. Rosbot pro 2.0. `https://husarion.com/manuals/rosbot-manual/`, 2019.

[52] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[53] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[54] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[55] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[56] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247, 2016.

[57] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. *arXiv preprint arXiv:1904.01201*, 2019.

[58] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[59] Oleg Klimov and J Schulman. Roboschool, 2017.

[60] OpenAI. Ingredients for robotics research. `https://openai.com/blog/ingredients-for-robotics-research/`, 2018.

[61] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.

[62] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[63] David V Gealy, Stephen McKinley, Brent Yi, Philipp Wu, Phillip R Downey, Greg Balke, Allan Zhao, Menglong Guo, Rachel Thomasson, Anthony Sinclair, et al. Quasi-direct drive for low-cost compliant robotic manipulation. *arXiv preprint arXiv:1904.03815*, 2019.

[64] Brian Yang, Jesse Zhang, Vitchyr Pong, Sergey Levine, and Dinesh Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019.

[65] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. *arXiv preprint arXiv:1811.02790*, 2018.

[66] Lilian Weng Maciek Chociej, Peter Welinder. Orrb: Openai remote rendering backend. In *eprint arXiv*, 2019.

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

[68] Stephen James, Marc Freese, and Andrew J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[69] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[70] Larry D Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The darpa lagr program: Goals, challenges, methodology, and phase i results. *Journal of Field robotics*, 23(11-12):945–973, 2006.

[71] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.

[72] Andry Tanoto, Ulf Witkowski, and Ulrich Rückert. Teleworkbench: A teleoperated platform for multi-robot experiments. In *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 49–54. Springer, 2006.

Supplementary Materials

A. Related work

Publicly available simulated environments play an important role in the development of RL methods, providing a common ground for comparing different approaches and allowing progress in the field to be explicitly tracked. However, they do not allow to bridge the gap between simulation and reality. Simulated environments address various general aspects of reinforcement learning research such as control [48], navigation [54, 55, 56, 57], physical interactions [49] and perception [58]. More domain-specific simulated environments explore such fields as robotics [59, 60, 61] and autonomous driving [62].

Following the signs of applicability of RL in real-world robotics, RL-oriented hardware kits became available in the past year to support the development of reproducible RL in robotics research [63, 64]. Mandlekar at al. [65] and Orrb et al. [66] introduce platforms for generating high fidelity robot interaction data that can be used to pre-train robotic RL agents.

OpenAI Gym [67] has provided an elegant ecosystem and an abstraction layer between the learning algorithms and the environments. Currently OpenAI gym supports classical control tasks and such environments as Atari, MuJoCo, Box2D and OpenAI robotics environments based on MuJoCo that support simulated creatures, Fetch research platform and Shadow Dexterous Hand™. OpenAI Gym was created to provide a benchmarking platform for RL research by introducing strict naming and versioning conventions, making it possible to compare the results achieved by different algorithms and track the progress in the field.

Zamora et al. [61] introduced an interface to integrate the Gazebo robotics simulator with the OpenAI Gym ecosystem, allowing to extend the set of possible RL environments to any that can be simulated in Gazebo. In their recent work, James et al. [68] introduced a toolkit for robot learning research based on V-REP simulator. Another step in this direction is the PyRobot project [69] that provides a high-level interface for control of different robots via the Robot Operating System (ROS).

Although these tools provide an easy access to a variety of environments with the focus on specific tasks, all of these publicly accessible environments are still limited to simulation, only tangentially addressing the challenge of creating intelligent agents in the real physical world. The very few projects that have provided physical systems for community-driven robotics research are the LAGR [70] project from DARPA, Georgia Tech's Robotarium [71] and TeleWorkBench [72] from Bielefeld University. While being the closest to the concept of OffWorld Gym, the LAGR program has concluded and is not active anymore. TeleWorkBench and Robotarium did not postulate a specific task and thus do not serve as a benchmark challenge. Robotarium's maximum script execution time of 600 seconds makes it unsuitable for RL research. Moreover, none of the previous systems provided close integration into modern RL research ecosystem, proposed specific and version-controlled challenges nor had the same level of public accessibility as OffWorld Gym.

B. Hardware specification

The Husarion Rosbot is equipped with an ASUS Up Board (Quad Core Intel CPU, Ubuntu 16.04) on-board computer, Orbbec Astra RGBD camera and a CORE2-ROS robot controller. The robot controller runs the firmware layer and the on-board computer runs the sensor drivers, ROS sensor packages and robot motion controller ROS package. Since all of the learning happens on the client workstation, the on-board capabilities of the robot can be kept minimal. An Intel NUC (Core i7, 32 GB RAM, Ubuntu 16.04) computer runs the OffWorld Gym server, the robot mission management software and the ROS packages that control the environment. An IBM workstation (Intel Xeon, 32 GB RAM, Nvidia Quadro, Ubuntu 16.04) interfaces with the HTC Vive lighthouse setup. It runs the HTC Vive driver and a ROS package which publishes the robot's localization data.

C. Video links

A Doulbe DQN agent controlling a physical mobile robot in a real environment with discrete control: https://youtu.be/HDViFqvB3Co

A Soft Actor-Critic achieves intelligent behavior in the continuous variant of the environment: https://youtu.be/Nbgq6c78yJg

D. Source code

The source code of the library and the examples mentioned in this work are available in the following GitHub repository: https://github.com/offworld-projects/offworld-gym