Towards Real Robot Learning in the Wild: A Case Study in Bipedal Locomotion

Anonymous Author(s)
Affiliation
Address
email

Abstract: Algorithms for self-learning systems have made considerable progress in recent years, yet safety concerns and the need for additional instrumentation have so far largely limited learning experiments with real robots to well controlled lab settings. In this paper, we demonstrate how a small bipedal robot can autonomously learn to walk with minimal human intervention and with minimal instrumentation of the environment. We employ data-efficient off-policy deep reinforcement learning to learn to walk end-to-end, directly on hardware, using rewards that are computed exclusively from proprioceptive sensing. To allow the robot to autonomously adapt its behaviour to its environment, we additionally provide the agent with raw RGB camera images as input. By deploying two robots in different geographic locations while sharing data in a distributed learning setup, we achieve higher throughput and greater diversity of the training data. Our learning experiments constitute a step towards the long-term vision of learning "in the wild" for legged robots, and, to our knowledge, represent the first demonstration of learning a deep neural network controller for bipedal locomotion directly on hardware.

Keywords: Legged Locomotion, Reinforcement Learning, Vision

1 Introduction

Enabling robots to learn and adapt completely autonomously only through interaction with their environment is one of the long-standing goals of AI and robotics. The idea that robots might one day learn and cohabit outside controlled laboratory environments, taking full advantage of the richness of the real world, is appealing especially for mobile robots. However, even though algorithms for self-learning systems have made considerable progress in recent years in terms of data efficiency and robustness (e.g. [1, 2]), concerns around safety and the need for instrumentation (for instance, for reward computation or resets) have so far largely limited experiments with real robots to controlled lab settings [3, 4] or to sim-to-real transfer [5, 6, 7].

While transfer of behaviours from simulation to real can sidestep the challenges around data efficiency and safety, it requires significant engineering effort in the design of high fidelity simulations that represent not only the correct dynamics of the robot but also offer increasingly complex and rich scenarios. More importantly, learning only in simulation exposes the agent to the sim-to-real gap, leaving questions around adaptation to unfamiliar situations after deployment.

Despite first successful applications (e.g. [8]), learning "in the wild" directly on hardware is often thought of as an infeasible method for real legged robots. In this paper, we show that using state-of-the-art model-free deep reinforcement learning (RL) and intrinsically defined rewards, we can learn to walk and avoid walls autonomously in uncontrolled environments. In particular, we show how small humanoid robots can learn to walk and interact with their environment in a learning scenario outside of the laboratory, relying only on on-board sensors and limited prior knowledge of the hardware.

The robot uses proprioceptive sensing consisting of an inertial measurement unit (IMU) and joint position encoders together with egocentric RGB vision to perceive its own body and the environ-

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

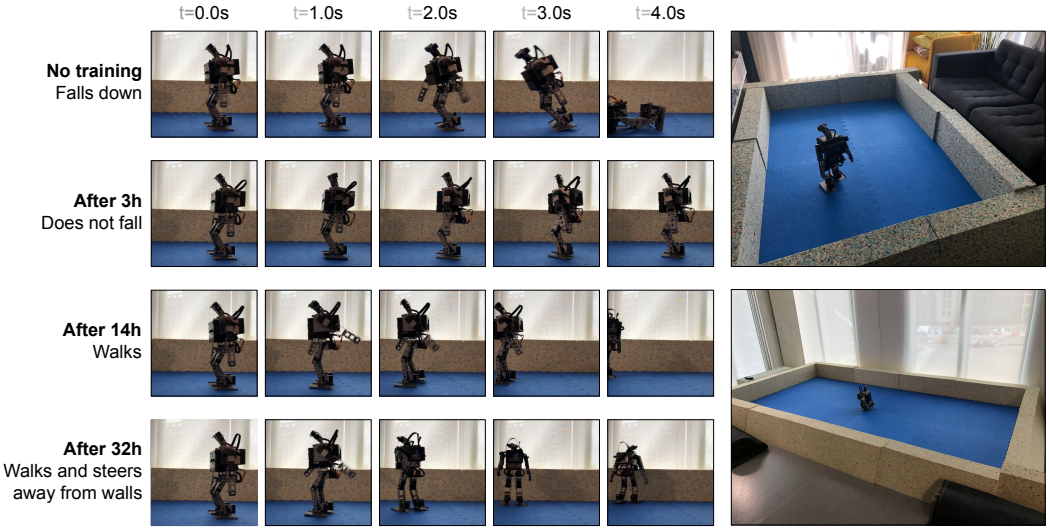

Figure 1: Two robots are trained simultaneously in two different locations depicted on the right. They are sharing the same training data. On the left we show one of the robots and the approximate experiment time it takes for the policy to reach important milestones: maintaining upright position, walking, and eventually walking and avoiding walls. We define experiment time as the total time during which at least on of the robots was on.

ment. Rewards and termination signals are derived from the proprioceptive sensors only. We employ a data-efficient off-policy algorithm [1] that allows for distributed training with a variable number of actors. The learning setup does not rely on additional external hardware for state estimation or resets. This allows us to train multiple robots simultaneously, outside of the controlled laboratory environment, in different geographic locations, exposing the robot to a range of environments with varying shape, appearance, and lighting conditions; see Figure 1 (right).

Our results show that such an intrinsic learning signal designed to promote forward movement (combined with the sensory capabilities of the robot and the partly structured environment) is sufficient to learn bipedal gaits and simple wall avoiding behavior requiring only a minimal amount of prior knowledge, see Figure 1 (left) and supplementary video. The distributed training with multiple robots improves data throughput and thus the complexity of the behavior that can be learned. Overall, the results provide a glimpse of the possibility of real-world learning for legged robots.

Our key contributions are: 1. First demonstration of learning a deep non-parametric policy for unconstrained bipedal locomotion on hardware; 2. A data-efficient autonomous learning setup for legged robots that can be easily deployed in different environments and run on multiple robots simultaneously; 3. Learned vision guided legged locomotion with simple obstacle avoidance capabilities.

2 Related Work

Conventional approaches to legged locomotion typically rely on models and optimization [9, 10] and are concerned with concepts of gait stability, such as zero moment point [11] and limit cycles [12]. While optimization based methods can yield highly effective solutions, they are laborious to implement, require rigorous domain understanding, and can struggle to generalize to diverse real-world environments and high-dimensional input modalities, such as vision.

Learning and data-driven approaches hold the promise to circumvent these limitation by avoiding the need for accurate modelling. Deep reinforcement learning for locomotion has been studied extensively in simulation [13, 14], and recent papers have shown progress in sim-to-real transfer for both quadrupeds [15, 7, 16] and bipeds [17, 6, 5]. Compared to these works, we largely lift the assumption of the availability of a simulation model.

Closest to our approach is the line of work where learning is directly applied on hardware. Unlike ours, these methods consider either a constrained, hand-designed gait parameterization [18, 19, 20, 21] or apply unconstrained deep reinforcement learning to an inherently stable [22] or a quadrupedal robot [3, 4]. None of these works consider an unconstrained humanoid robot or vision observations, and, unlike our approach, they require external instrumentation such as motion capture or other means to quantify the obtained gait.

3 Methods

3.1 Generalizable Instrumentation and Learning Autonomy

We aim to create a learning setup for a real robot to learn to walk in different *uncontrolled* environments (e.g. households). This means that we cannot rely on external sensing such as motion capture systems and we cannot use reset mechanisms such as a gantry. Moreover, the setup has to be able to deal with natural, possibly cluttered, and irregularly shaped environments, and with variable natural and artificial lighting.

In this paper, we focus on a simpler instantiation of this setting in which the robot is tasked to maintain an upright position, and to move forward as fast as possible in an area surrounded by walls. This forces the robot to use its legs for locomotion while avoiding the walls. Access to camera observations enables the robot to look ahead and adapt its behavior to its surroundings and thereby to deal with a variety of environments. For generality and robustness reasons, we do not employ any (possibly brittle) image pre-processing steps, e.g. for wall detection. Instead, we learn the mapping from the camera pixels to the control signal end-to-end.

The design of our rewards and episode termination conditions is based on [8]. The reward is generated when the feet exhibit a positive forward velocity in the robot's body coordinate frame. To assess this, the velocity of the robot body is computed by assuming that the lowest foot is in contact with the ground and the contact point is stationary (no-slip condition). While these assumptions may not always hold, Hafner et al. have shown experimentally that this reward may still be sufficient to learn forward motion when combined with a reward for keeping the torso upright. This uprightness is in turn estimated from a gravity vector obtained via the IMU's internal orientation estimate (based on a complementary filter). Given that the velocity reward encourages forward movement in the robocentric coordinate frame, there is a risk of promoting walking in small circles. To avoid this, we penalize large changes in the heading direction, again based on the IMU's orientation estimate. The three reward terms (velocity, uprightness, and no turning) are linearly combined to form the final reward. Episodes are terminated whenever the robot's body inclination exceeds a certain threshold (see the supplementary material for more details about the reward).

Finally, we make use of a simple reset mechanism to allow autonomous learning: At the end of each episode, if the robot has fallen (as inferred from the IMU orientation estimate), we use a robust pre-programmed feed-forward controller to stand up. On rare occasions multiple attempts may be required. The robot does not need to be moved and simply continues from the location where it ended the previous episode.

3.2 Robot Platform

We focus on the off-the-shelf OP3 bipedal robot platform [23], or its simulated analog modelled in MuJoCo [24]. This platform is built of 20 Dynamixel servo motors (XM430-W350-R) that we use in *position control* mode. All controllers use 20Hz control frequency. In real robot experiments, all controllers run directly on the onboard Intel NUC i3 computer using ROS [25] to send commands to the actuators. While the low weight and low cost of the robot are advantageous for real robot learning experiments, the low actuator and sensory specifications increase the difficulty of the learning task.

3.3 Distributed Data Efficient Reinforcement Learning

The input to our policy consists of proprioceptive observations listed in Table 1, and the two most recent RGB camera observations down-sampled to 64x64 resolution. The output of the policy is a distribution over 20 dimensional actions which represent position commands for the robot's actuators.

Observation	Dimension
history of joint positions	5×20
history of IMU accelerations	5×3
history of IMU angular velocities	5×3
history of egocentric gravity directions	5×3
history of past actuator commands	5×20
history of linear velocity estimates (see [8])	5×3
heading change over 500ms	2

Table 1: List of proprioceptive observations received by our policies.

Our policies are trained using an asynchronous distributed version of the actor-critic MPO algorithm [1, 26] which uses a distributional critic [27] (DMPO), and which has shown to be effective in addressing high-dimensional tasks with continuous action space. The algorithm is implemented in JAX [28] using the code[1] released as part of [29] as a reference. All hyperparameters are listed in the supplementary material.

The robots' onboard computers run *actor* processes which synchronize the policy's neural network weights with a remote *learner* process at the beginning of each episode. After an episode of executing the policy on the robot, the actor process sends all data experienced in that episode to a remote replay buffer implemented using the Reverb library [30]. The learner process continuously reads data from the replay buffer, and updates the network weights via gradient descent on the DMPO actor and critic losses. In order to prevent overfitting, the learner process is blocked when the total number of gradient steps exceeds some threshold times the number of control steps executed by all robots. In our experiments, this threshold is $1/4$. The remote learner process runs in the cloud. Communication with the robots is implemented using the Launchpad API [31].

One advantage of this setup is that we can scale from one robot to multiple robots in a practical and application-relevant way. But it also shifts the computational load away from the robot and the location of the experiment. This in principle allows to learn with robots having weaker onboard computing power, basically requiring only an internet connection to allow learning experiments.

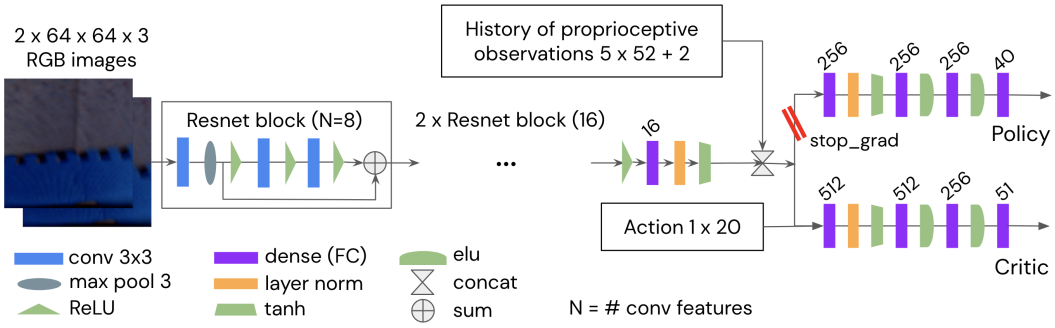

Figure 2: Network architecture. A 3-block Resnet produces image representations, which are concatenated with proprioceptive observations, and fed to separate policy and (action-conditioned) critic networks; see text for more details on tensor shapes and losses.

3.4 Network Architecture

The network architecture is depicted in Figure 2 and consists of an image encoder shared between the policy and critic networks. It processes the latest two RGB images received from the robot's camera. The outputs of the encoder are concatenated with proprioceptive observations (see Table 1) and processed by separate policy and critic networks. The shared encoder is trained using gradients from critic losses only. The output of the critic is 51-dimensional corresponding to the logits pa-

[1] https://github.com/deepmind/acme/tree/master/acme/agents/tf/dmpo

rameterizing the distribution of future returns [27]. The policy outputs the mean and the standard deviation of a diagonal Gaussian distribution from which we sample the actions for the actuators.

4 Experiments

For safety reasons and to ensure autonomous learning, the workspace in our experiments consists of a rectangular court delimited by foam walls. This prevents the robot from walking away, and, in combination with stand-up resets, represents a very effective way to minimize human intervention. In some sense, this is similar to the play pens used for babies when they start learning to walk. Except for this, all other environmental conditions are uncontrolled, e.g. we train in different daylight conditions, with various furniture items present around the workspaces, and different court sizes (see Figure 1). To avoid entanglement with the power cable, real robots are powered by a battery pack which needs to be changed roughly every 30 min.

At the beginning of training, the policy explores randomly which causes the robots to fall very often. To reduce the risk of damaging the hardware, we equip the robots with a 3D-printed body armour covering the front and back of the torso. In addition, we limit the range of the joints to prevent self-collisions that might damage individual actuators or other hardware parts. We also apply an exponential action filter on the actions sampled from the policy before sending the reference position commands to the actuators. History of these filtered actions is included in the policy's inputs (see Table 1) in order to mitigate partial observability due to delays.

4.1 Simulation Experiments

A simulation environment is used to make preliminary tests ahead of the real world experiments. Aside from using it to tune network architecture and hyperparameters, we want to know if the sample complexity is within a reasonable range and whether vision can be used as viable alternative to instrumented state estimation.

We roughly replicate the real world setup – a 4×4 m court surrounded by walls – in simulation (see Figure 3). We randomize the textures for the floor and walls in every episode to roughly mimic the diversity of visual inputs in the real world setup where the learner receives trajectories from actors running in different locations and under various lighting conditions. The position and orientation of the robot are randomised at the beginning of every episode. We do not make additional efforts to make this simulated environment more realistic as we do not aim for sim-to-real transfer (see Section 1). Given the lack of rich textures, realistic lighting conditions, and motion blur it would be unlikely for a policy to transfer (see Fig. 6 in supplementary).

Figure 4 depicts the training curves of three different agents: 1) a *vision* agent using raw images, 2) a *pose* agent having access to ground-truth robot pose (position and orientation), and 3) a *blind* agent without additional non-proprioceptive measurements. All are trained in an open square court but tested either with or without an additional disturbing wall (see Figure 3). Starting with the *blind* agent, we can observe that the attained performance is only marginally affected by the presence of an additional wall. Looking at the learned policy shows how the robot uses its arm to detect the presence of a wall which is a behavior that is fairly robust to changes of wall configuration (see supplementary material).

When having access to ground-truth pose (*pose* agent), the robot learns very quickly to walk within the court. But when adding a wall at test time the performance strongly degrades as the robot collides with the wall and is unable to recover. This is improved by the proposed *vision* agent. While performing almost on par with the *pose* agent in the simple square court, it is also able to naturally generalise when the wall configuration is altered. This shows that there is a significant benefit when including vision in real world settings were space is usually limited and more cluttered. The difference in behavior can also be appreciated when visualising the policy (see Figure 3): The policy with vision avoids the wall (which was not present during training) and turns into a different direction . Overall, the *vision* agent achieves good performance after about 1M environment steps. On a real robot, and taking into account repair and maintenance, this could be collected within 5 days.

Given the limited space of our environments all experiments have the tendency to develop circular motion patterns (see Figure 3) despite the "no turning" component of our reward. This is expected–

Figure 3: Overview of the simulation setup. Our simulated model of the OP3 robot is in a 4×4 m square court with random textures. Top: training setup. Bottom: test setup with additional wall to observe the reaction of the robot. We also visualise trajectories of the best performing *vision* agent which is also the one we train on the real robot. One can observe the circular motion that is learned and how the robot avoids walking into the walls. The white square highlights the start position.

we only use the "no turning" reward to discourage walking in very small circles which would prevent the robot from exploring the whole workspace.

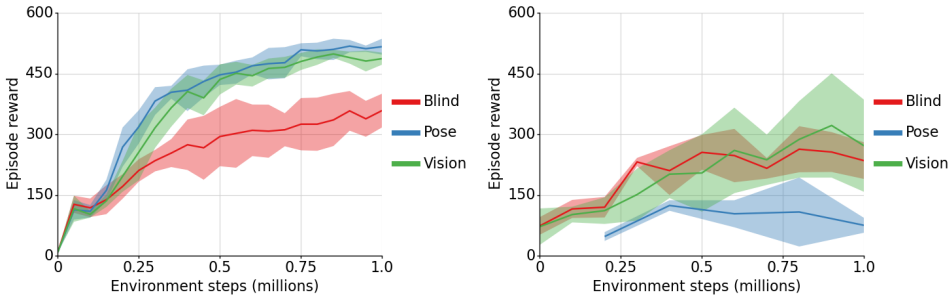

Figure 4: Simulation results comparing agents trained with vision and proprioceptive sensing (green), with groundtruth pose and proprioceptive sensing (blue), and with proprioceptive sensing only (red). The right plot shows the performance of the same agents in a court with an additional wall which was not present during training. The shaded area corresponds to 95% confidence interval across seeds.

4.2 Real robot experiments

Our robot experiments focus on sample complexity relative to simulation experiments, on assessing whether our trained policies make use of vision, and on evaluating the degree to which they can generalize to different robots and environments.

Figure 5 (left) shows the learning curve for the distributed two-robot experiment. We plot the episode results experienced by different robots in different colors to demonstrate that the robots had to be occasionally stopped for repairs (while the other robot would keep collecting data).

Interesting behaviors start to emerge after about 25k environment steps of training which corresponds to 2-3 hours of experiment time. This includes battery changes, time spent on resets and logging data in between episodes. It takes much longer to collect 25k environment steps at the beginning of the experiment than at later stages because the robot falls a lot and the experiment time is dominated by the "stand up" reset behaviors (see supplementary material). At this point the robot has learned to balance and to avoid falling which causes the episode lengths and the returns from the

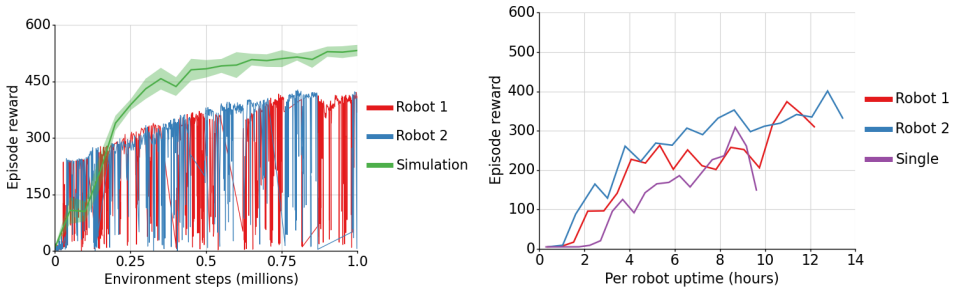

Figure 5: **Left**: Training curve for a distributed experiment with two robots called *Robot 1* and *Robot 2*. The episode rewards achieved by each robot are plotted in different colours. We do not smooth the curves to preserve information. The intervals where only one colour appears correspond to downtime of one robot. We also show a learning curve from an analogous simulation experiment. **Right**: Episode reward plotted against per robot uptime, comparing a 2-robot experiment and an independent single robot experiment.

"upright" component of the reward to increase. From this point on, the robot starts taking first steps, steadily improving its walking speed and stability. It develops a bias for turning right and converges to circular walking behavior inside the enclosed workspace. See supplementary video for recordings from the experiment.

Figure 5 (left) also shows a learning curve from an analogous simulation experiment. Once the robot learns to avoid falling, the rate of learning progress is about half that of the simulation. We believe this gap is caused by several factors which make the real environment more complex than the simulation: inaccurate model of delays in simulation; changing robot dynamics due to wear and tear of the joints causing them to become loose and eventually completely broken; changing robot dynamics due to battery voltage fluctuations; and non-stationary lighting conditions.

We also ran a shorter, single robot experiment and observed a similar sample complexity as that of the two-robot experiment (see learning curve in supplementary material). This is not obvious a priori because larger diversity of inputs (as is the case in the two-robot experiment) typically renders reinforcement learning more difficult. The single robot experiment also serves as an additional "seed" and provides evidence for the reproducibility of our setup. To demonstrate the increased throughput of the two-robot experiment, Figure 5 (right) shows the dependence of individual robots' performances on their uptime, and compares it to the same curve for the robot in the single robot experiment. We see that individual robots in the two-robot experiment, where data is shared, improve faster, pointing to a promising direction of scaling up real experiments.

Visual wall avoidance To demonstrate that the learned policy uses camera observations to avoid walls, Figure 6 depicts the reaction of the robot when encountering the wall. Because of the policy's right turning bias, we initialize the robot so that a wall is on its right (red square marks the robot's initial position). If no wall is present, then the robot turns in a large circle (top row). In comparison, when a wall is present, the robot takes a sharper turn as to only just brush the surface of the wall (bottom row).

Further robustness tests We evaluated the final policy obtained in the two-robot experiment in two additional settings: 1. on a third robot which was not used during training, and 2. on the same robot which was used in the experiment but using a black floor instead of a blue one. Table 2 shows the results of this evaluation. The policy works well on *Robot 3* achieving only a slightly lower per step reward than the robot used during training (*Robot 1*). However, it is less stable on *Robot 3* causing the robot to fall more frequently. While this difference can be due to transfer, it is likely that the state of *Robot 3* was unfavorable – as mentioned before, the dynamics of the robots vary across time and things like loose joints or screws can impact what a robot can do.

The policy performs considerably worse on a black floor demonstrating that the visual module is adapted to the blue floors encountered during training (also see supplementary video). However, as demonstrated in the simulation experiments, this could be improved by exposing the agent to black floors during training as well.

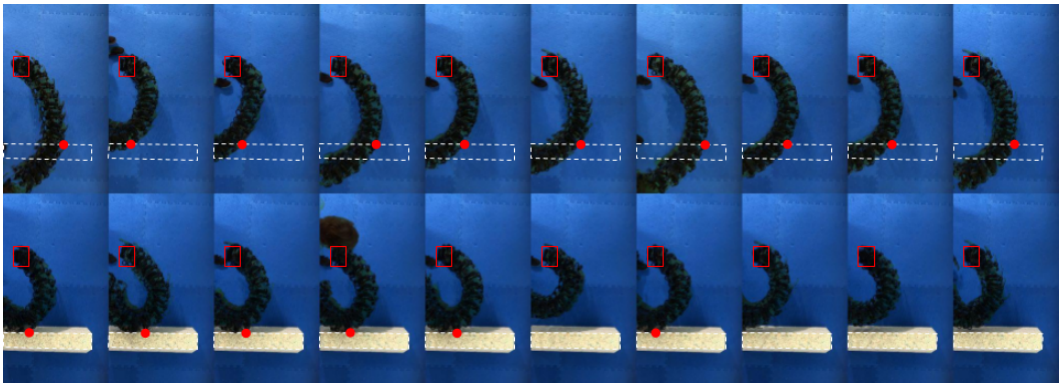

Figure 6: Qualitative comparison of robot trajectories when encountering a wall. The robot starts in the red square facing right. It then starts walking while slightly turning towards its right. The robot makes a sharper turn when there is a wall (beige foam block, bottom row) than when we remove the wall (the white dots show where it would be). The first point of contact with the (potentially virtual) wall is depicted by a red dot. In the presence of an actual wall contact can either be avoided or is shifted towards the left as to only just brush the wall.

Evaluation context	Per step reward	Episode length
Robot 1	0.408 ± 0.002	857 ± 151
Robot 3	0.389 ± 0.003	520 ± 160
Black floor	0.318 ± 0.005	136 ± 36

Table 2: Evaluation of the learned policy under various contexts.

5 Conclusions

In this paper we showed that we can learn a walking gait for a bipedal, off-the-shelf robot, directly on hardware, outside a well-controlled lab infrastructure. Using a state-of-the-art RL method (DMPO) renders learning with minimal prior knowledge feasible in terms of data efficiency.

Scalable reinforcement learning in the real world is still in its infancy, and to the best of our knowledge, this is one of the largest scale experiments for model-free learning for legged robots. Limiting ourselves to on-board sensors only and collecting data from multiple robots at various geographic locations allows us to scale up the diversity of environments which the robots encounter. While in this paper we used only two locations with different lighting conditions, arena sizes, and background visuals, in future this can include many more locations with varying terrains and obstacles. This should be contrasted to lab settings with immobile equipment for external state estimation. We further also intend to remove the requirement for a reset controller by learning to stand-up.

Specifically, we plan to explore learning more dynamic behaviours, as well as tasks specified by rewards derived from camera images such as searching for an object in a cluttered environment. Such tasks require memory, complex visual processing, and longer horizon decision making, hence it will be necessary to train for much longer than 1M steps. We believe that the presented approach is well suited for learning at such scale. We also plan to follow a "curriculum" and progressively move, or completely remove, the protective walls.

Social Impact Statement Embodied agents using perception and locomotion have the potential to enable applications with significant positive social impact. E.g. they can be deployed in search-and-rescue operations (required in force majeure events like earthquakes, landslides, nuclear disasters), infrastructure inspections that might pose risks for humans (e.g. assessing if a bridge or tunnel is in a healthy condition), and many others. However, there is also the risk that AI-driven technology in general can be used for harmful purposes, e.g. in military applications. We strongly oppose such use cases. We believe that our particular application presents a low risk of being used in such situations as it has limited potential for causing large-scale damage, but we actively support discussions around safety and fairness in Robotics.

References

[1] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation, 2018.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[3] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi:10.15607/RSS.2019.XV.011.

[4] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. *CoRR*, abs/2002.08550, 2020. URL https://arxiv.org/abs/2002.08550.

[5] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv:2105.08328*, 2021.

[6] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. *arXiv preprint arXiv:2103.14295*, 2021.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, Oct 2020. ISSN 2470-9476. doi:10.1126/scirobotics.abc5986. URL http://dx.doi.org/10.1126/scirobotics.abc5986.

[8] R. Hafner, T. Hertweck, P. Klöppner, M. Bloesch, M. Neunert, M. Wulfmeier, S. Tunyasuvunakool, N. Heess, and M. A. Riedmiller. Towards general and autonomous learning of core skills: A case study in locomotion. *CoRR*, abs/2008.12228, 2020. URL https://arxiv.org/abs/2008.12228.

[9] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[10] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson. Optimization based full body control for the atlas robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 120–127, 2014. doi:10.1109/HUMANOIDS.2014.7041347.

[11] M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[12] T. McGeer et al. Passive dynamic walking. *I. J. Robotic Res.*, 9(2):62–82, 1990.

[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[14] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[15] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[16] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[17] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. *arXiv preprint arXiv:1903.01390*, 2019.

[18] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2619–2624. IEEE, 2004.

[19] R. Tedrake, T. W. Zhang, and H. S. Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, volume 95585, pages 1939–1412. Beijing, 2005.

[20] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[21] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson. Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778. IEEE, 2018.

[22] S. Ha, J. Kim, and K. Yamane. Automated deep reinforcement learning environment for hardware of a modular legged robot. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 348–354. IEEE, 2018.

[23] Robotis op3. https://emanual.robotis.com/docs/en/platform/op3/introduction/.

[24] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi:10.1109/IROS.2012.6386109.

[25] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL https://www.ros.org.

[26] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. Riedmiller. Relative entropy regularized policy iteration, 2018.

[27] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[28] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[29] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, A. Cowie, Z. Wang, B. Piot, and N. de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020. URL https://arxiv.org/abs/2006.00979.

[30] A. Cassirer, G. Barth-Maron, E. Brevdo, S. Ramos, T. Boyd, T. Sottiaux, and M. Kroiss. Reverb: A framework for experience replay, 2021.

[31] F. Yang, G. Barth-Maron, P. Stańczyk, M. Hoffman, S. Liu, M. Kroiss, A. Pope, and A. Rrustemi. Launchpad: A programming model for distributed machine learning research, 2021.