The Open Ant: A Robot Platform for Reinforcement Learning Research

Elena Sorina Lupu, Patrick Spieler, Khurram Javed, Kris De Asis, John D. Martin, Martha Steenstrup, Joseph Modayil

Keywords: Reinforcement Learning, Robotics, Embodied Intelligence, Research Platform

Summary

Reinforcement learning (RL) research has demonstrated success in both physical and simulated domains; however, the predominant methodology remains rooted in simulations. The predominance of simulations makes translating research to physical reality uncertain for both algorithms and researchers. We propose a physical platform that is designed to simplify the transition. In this paper, we present the Open Ant: a physical variant of the commonly used Gymnasium Ant environment, along with a simulation. We demonstrate that competent walking policies can be learned from scratch in approximately one hour directly from the physical robot's experience for two substantially different RL algorithms: SARSA(λ) and Soft Actor-Critic (SAC). Separately, we show policies that were learned in simulation transfer to reality. We also examine how well the platform supports a nimble experimental ecosystem. Specifically, we observe the speed with which new users from diverse backgrounds achieve their first success with the platform, and how easily the platform can be repaired and updated when hardware issues arise. Both the hardware design and software are available as open-source on GitHub for ease of customization. In summary, we advocate for the use of the Open Ant for RL researchers who frequently use simulated environments, so they can more easily include robot experiments in their evaluations.

Contributions

1. We developed a physical robot platform with an accompanying simulation, designed to support reinforcement learning researchers who are not accustomed to working with robots. The robot is inspired by the Gymnasium Ant used in continuous-control research, but several alterations were required to make a useful physical research platform. The hardware design and software are released to the community as an open-source platform.
   Context: The Gymnasium Ant (Schulman et al., 2015) is a popular simulated domain, with an abstract physical design. Reinforcement learning algorithm research is commonly conducted with simulated domains, though the community has used several robot platforms to validate their algorithms on embodied systems. The proposed platform is intended to support such efforts.
2. We demonstrate that this platform is compatible with multiple reinforcement learning methods. The platform supports both simple and complicated RL algorithms. The platform supports learning directly from the hardware experience and also policy transfer from the simulator to reality.
   Context: The reinforcement learning community has adopted a wide range of methods and research directions. By successfully demonstrating the use of different RL algorithms and methods with this platform, we make it easier for RL researchers to adopt the platform.
3. We show that the platform supports a nimble ecosystem for RL experimentation, with the rapid on-boarding of inexperienced users and rapid revisions when hardware failures are encountered. The platform was used for RL experimentation by multiple researchers with limited prior experience in robotics. The platform supports rapid revision to encountered problems, through the use of 3D printing and commercially available components.
   Context: RL researchers who used robots previously experienced challenges with both the initial adoption of a new platform, and with maintaining a robot as a research platform.

The Open Ant: A Robot Platform for Reinforcement Learning Research

Elena Sorina Lupu[1*], Patrick Spieler[*], Khurram Javed[2], Kris De Asis[1], John D. Martin[1, 3], Martha Steenstrup[3], Joseph Modayil[1, 2]

[1]**Openmind Research Institute, Canada**
[2]**Keen Technologies**
[3]**Department of Computing Science, University of Alberta, Canada**

[*]equal contribution

Abstract

Reinforcement learning (RL) research has demonstrated success in both physical and simulated domains; however, the predominant methodology remains rooted in simulations. The predominance of simulations makes translating research to physical reality uncertain for both algorithms and researchers. We propose a physical platform that is designed to simplify the transition. In this paper, we present the Open Ant: a physical variant of the commonly used Gymnasium Ant environment, along with a simulation. We demonstrate that competent walking policies can be learned from scratch in approximately one hour directly from the physical robot's experience for two substantially different RL algorithms: SARSA(λ) and Soft Actor-Critic (SAC). Separately, we show policies that were learned in simulation transfer to reality. We also examine how well the platform supports a nimble experimental ecosystem. Specifically, we observe the speed with which new users from diverse backgrounds achieve their first success with the platform, and how easily the platform can be repaired and updated when hardware issues arise. Both the hardware design and software are available as open-source on GitHub for ease of customization. In summary, we advocate for the use of the Open Ant for RL researchers who frequently use simulated environments, so they can more easily include robot experiments in their evaluations.

1 Introduction

Reinforcement learning (RL) has been successful in both physical and simulated domains, though many successes rely heavily on accurate simulations. Several of the notable successes include games such as Backgammon (Tesauro, 1995) and Go (Silver et al., 2016), where accurate and computationally efficient simulators can capture the exact dynamics. Success in complex online games can introduce differences between training and deployment environments (Wurman et al., 2022). Successes in complex physical domains has often relied on human expertise to select the most relevant dynamics to simulate, with policies trained in simulation prior to a deployment phase, as seen in balloon navigation (Bellemare et al., 2020), fusion plasma control (Degrave et al., 2022), or gravitational wave detectors (Buchli et al., 2025).

Reinforcement learning research might be a victim of its own success with simulations, as relatively few studies focus on learning directly in physical reality. The ability to learn directly from physical experience has ample evidence in animal learning experiments, which served as motivation for early computational RL algorithms (Barto et al., 1983), with a modified version demonstrated on hardware (Benbrahim et al., 1992). This early motivation was followed by evidence that RL

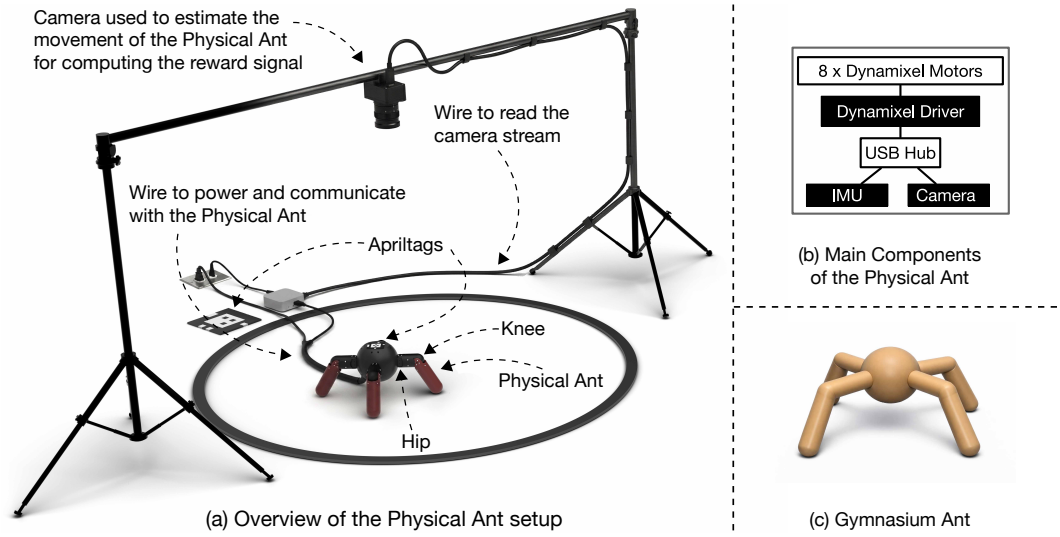

Figure 1: **Physical Ant platform, learning arena, and system overview.** (a) An overhead webcam tracks the fiducial markers to compute reward signals and the heading vector of the ant. The robot is connected by cables to AC power and to an external computer where the agent is running. (b) The main components of the Physical Ant. (c) The Gymnasium Ant (Schulman et al., 2015; Towers et al., 2025), which was the inspiration for the Physical Ant.

algorithms provide computational models for the activity of dopamine neurons (Montague et al., 1996; Schultz et al., 1997). In addition to animal studies, many engineered systems have demonstrated the use of RL algorithms to learn behaviors directly from physical experience without the need for a simulator, with early examples on locomotion (Kohl & Stone, 2004; Tedrake et al., 2004) and later work demonstrating the use of a single RL algorithm across multiple robots and problem formulations (Mahmood et al., 2018b; Wu et al., 2023).

Despite repeated success with RL algorithms on physical robots, researchers with primarily simulation-based experience encounter significant challenges when experimenting with physical robots. One challenge is that the overhead (e.g., cost and time) for experiments with physical robotics is substantially higher than using simulations alone.

Even in situations where adequate lab resources and expertise are available, the long delay caused by the assembly of the robot and troubleshooting before the first successful experiment can exhaust a considerable portion of a researcher's typical residency. Consequently, experts may graduate and depart research labs shortly after successful experiments, resulting in slower experiment iteration and difficulties with future re-use.

Contributions. We present the Open Ant, an open-source research robot platform available on GitHub. The robot comes with a MuJoCo simulation, which can be used to learn competent policies. The robot body is designed to look like the widely used Gymnasium Ant environment (Towers et al., 2025). The Open Ant makes physical robot experiments an easier addition to a standard RL research pipeline. The robot is designed to be built and maintained by AI researchers without backgrounds in robotics engineering. The robot is designed to withstand the substantial wear incurred during RL exploration, and to be easy to repair when components are damaged.

The learning experiments presented here are designed to yield positive results within an hour; furthermore, we demonstrate that the robot platform provides reliable, repeatable results across different RL algorithms. In these ways, we envision this platform lowering the barrier to physical robotics research for the broader RL community.

2 Platform Motivation and Background

Our overall goal is to create a research platform that simplifies the process for researchers to extend their simulation results in RL research. Based on our team's experience on previous robot platforms, we identified several core criteria: the robot should closely mirror an existing simulated domain (see Figure 1), remain affordable, and integrate with standard RL software tools (see Section 5). Furthermore, running experiments should require minimal intervention, support diverse RL methods, and yield results on hardware within timescales comparable to simulation (see Section 5). Finally the overall platform should be accessible to newcomers and relatively easy to modify and maintain (Section 6). Although many prior works have demonstrated some of these capabilities, we are not aware of a platform that provides all of them. As such, our goal is not to produce a static benchmark, but to introduce a platform that the research community can flexibly revise over time—similar to the Arcade Learning Environment (Bellemare et al., 2013).

In addition to making robot experiments easier, we seek a platform that supports RL researchers to better study fundamental questions that arise when algorithms learn from physical experience. Invited talks at the 2025 RL Conference by Kaelbling (2025) and Sutton (2025) highlighted the complementary considerations for RL algorithms that happen at *design-time* (before regular deployment) and *run-time* (during regular deployment). Design-time knowledge of the problem structure already informs many facets of a robot system using RL, including the selection of the observation and action spaces (Mahmood et al., 2018a), the creation of the initial behavior policies (Silver et al., 2018), the physical design of the robot's body, and the selection of appropriate simulators. In contrast, run-time knowledge can only be acquired in deployment. During run-time, the agent may encounter phenomena that are either unknown or imperfectly modeled at design-time.

Traditional adaptive control (Slotine & Li, 1991) addresses run-time learning by continuously updating a linear controller to compensate for changes in the environment parameters while the robot remains in operation. Examples of online learning using adaptive control on hardware include remote-controlled airplanes (Shi et al., 2020), quadcopters (O'Connell et al., 2022), and ground vehicles (Lupu et al., 2025). Reinforcement learning shares this same online learning/adaptation property but extends it beyond tracking or stabilization, enabling complex decision-making through interaction with the environment (Sutton & Barto, 2018). Physical robots provide a natural setting for studying algorithms that learn continually during deployment, where adaptation must occur from ongoing experience rather than repeated offline retraining.

In practice, most RL experiments are conducted in a lab without deployment (sim-to-real); however, we aim to study algorithms that can adapt behavior in deployment, outside of simulators and with limited researcher intervention. A primary constraint of the run-time setting is learning while behaving, without pausing the environment. Moreover, the robot's physical experience differs from a simulation: a ground-truth simulator state is unavailable, and physical effects like overheating are difficult to model. Additional considerations include ensuring learning algorithms are compatible with local compute, communication, and timing constraints, as well as limiting the need for manual episodic resets. Our aim is to make these run-time conditions accessible and practical for RL researchers through a single, integrated robotic platform for learning from physical experience.

Although several papers have demonstrated run-time reinforcement learning on physical robots, it remains unclear whether these platforms are easily adopted by current RL researchers. One fruitful approach is to design custom robots that mirror existing simulated domains. For instance, NoodleBot (Berrueta et al., 2024) replicates the Gymnasium Swimmer benchmark, while RealAnt (Boney et al., 2020) is a quadrupedal platform inspired by the Gymnasium Ant. In contrast to ours, the RealAnt requires soldering and overhead camera calibration, and lacks clear guidance on setup time and reproducibility. In addition, its morphology also differs from the standard MuJoCo Ant model. Run-time reinforcement learning has also been demonstrated on commercial platforms. For example, Smith et al. (2022) show run-time policy learning on the Unitree A1 quadruped, and Kohl & Stone (2004) demonstrate a policy gradient RL algorithm that learns the parameters of a walking gait for the Sony Aibo quadruped. More recently, Preiss et al. (2025) demonstrate non-episodic, online

Table 1: Comparison of the Gymnasium Ant and the Physical Ant platform.

Ant Type	Torso Diameter [m]	Upper Leg [m]	Lower Leg [m]	Total Mass [kg]	Hip Motor	Knee Motor
Physical Ant	0.15	0.08	0.16	1.20	XC430-W240	XM430-W210
Gymnasium Ant	0.50	0.28	0.56	0.91	N/A	N/A

policy optimization on a Crazyflie open-source quadcopter (Preiss et al., 2017). Even more closely aligned with our objectives, Mahmood et al. (2018a) previously demonstrated the instantiation of a Gymnasium Reacher environment with a commercial UR5 robot. We chose to pursue a different platform from the Reacher to access more domain complexity (higher dimensional observations and actions, mobility with contact dynamics) at a fraction of the cost. Commercial platforms can offer a relatively low barrier of entry for RL researchers familiar with simulations; however, they carry risks of product obsolescence, high purchase costs, long and costly repairs, and unfamiliar or rigid software interfaces which may hinder certain lines of research.

3 The Open Ant Platform

In this section, we introduce the Open Ant (Figure 1): an *open-source* quadruped platform for RL and robotics research. Our platform includes both physical and simulated robot bodies inspired by the Gymnasium Ant (Schulman et al., 2015), which serves as a popular benchmark for continuous-control research (Towers et al., 2025). Our physical robot body is called Physical Ant and the simulation is called Simulated Ant. Our platform is designed for researchers to easily move between simulation and reality when experimenting.

3.1 Design Overview

Design Principles. Our design adheres to three fabrication principles. First, the design *prioritizes simplicity and ease of assembly* by avoiding any need for soldering and relying exclusively on commercial off-the-shelf (COTS) components that are widely available. Second, the robot must *support long-term learning* without interruption; thus it is powered directly from an AC wall outlet, bypassing the maintenance overhead of rechargeable batteries. Third, the robot requires *no special equipment*; it directly connects to a personal computer via USB, eliminating the need for specialized embedded systems code or complex network configurations.

Physical Attributes. The Physical Ant is one third the scale of the Gymnasium Ant (Schulman et al., 2015). By default, the Gymnasium Ant stands approximately 75 cm tall with a total mass of 0.9 kg, its torso is 0.50 m in diameter; the upper leg segment measures 0.28 m and the lower leg segment 0.56 m. These dimensions imply an extremely low mass-to-size ratio that would be difficult to realize physically with conventional materials. The Physical Ant's reduced size results in a relatively higher mass-to-length ratio, improving both its robustness and ease of manufacturing. See Table 1 for the design details.

Construction. The robot consists of a spherical body (torso) housing the onboard electronics and four articulated legs. The torso and legs are 3D printed, enabling rapid revision and localized repairs. Each leg is equipped with two metal-geared Dynamixel actuators: one at the hip (XC430-W240) and one at the knee (XM430-W210). The spherical body shell houses the onboard electronics, including an IMU, a USB communication converter, and a USB hub that interfaces these components with an external computer (Figure 1, (b)). Additionally, the Physical Ant features an on-board camera, to support future vision-based learning tasks. The complete list of components is available on our GitHub repository. The robot communicates via USB with an external computing platform, a design choice that allows users to flexibly select their preferred compute hardware. As a result, researchers

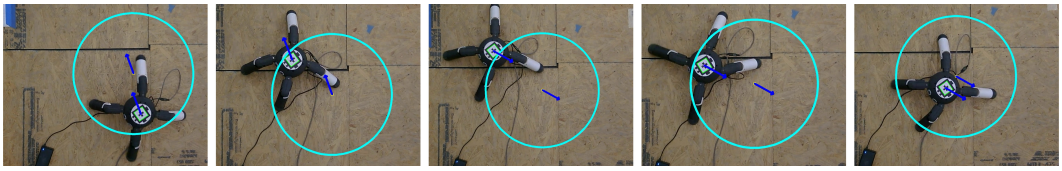

Figure 2: **Overhead snapshots during run-time learning from physical experience**. The cyan circle represents the boundary where the reward direction (the blue arrow) changes direction if the Physical Ant exceeds it. In the first figure, the robot travels towards 11 o'clock. In the second figure, it approaches the boundary. In the third figure, the reward direction flips and in the last two figures, we see the ant traveling in the direction of the reward direction.

can develop and run their algorithms without needing to adapt them to the computational constraints of an onboard platform. Power is provided via a wall-plug power supply, and the feet are equipped with 3D-printed thermoplastic polyurethane (TPU) "socks" to increase ground friction. The total cost for all the components is approximately USD 2200, excluding 3D-printing filament.

There exists a possibility to replace the more expensive motors (XM430-W210) with plastic-geared motors (XL430-W250) on both joints, which we call the Physical Ant Lite. For this lower-cost variant, the overall price is approximately USD 500 (excluding 3D printed parts). While more affordable, these actuators are more susceptible to wear and overheating. We therefore recommend that interested users carefully review the thermal analysis (Section 10.1), as well as the discussed electrical and mechanical improvements (Section 10.2 and Section 6) before deciding on the best motors for their needs.

3.2 Differences and Similarities between the Gymnasium and the Open Ant Environments

We highlight several aspects of the Gymnasium Ant environment (Schulman et al., 2015) that motivated our design. The Physical Ant adopts the same software interface as the Gymnasium Ant, while varying in the specific choices for its sensing, actuation, and reward.

Observations. Compared to its Gymnasium counterpart, the Physical Ant has a compact observation space. Its observations have twenty-four dimensions, consisting of joint angles (8), joint angular velocities (8), the torso's angular velocity (3), and linear acceleration (3) measured by an onboard IMU. In addition, we augment the observation with a two-dimensional unit vector encoding the angle between the heading vector and the goal direction (Equation 2) for the back-and-forth task presented below. Lastly, the platform provides interfaces for augmenting the observation space with other sensor modalities, such as motor loads and motor temperatures. In contrast, the observation space of Gymnasium Ant is 105 dimensions, composed of body-part positions, velocities, and external forces acting on each segment. Observations exclude inertial planar position. Many of these observations in the Gymnasium Ant, such as the external forces on the body parts, are difficult to obtain on hardware.

Actions. The Physical Ant's action space has eight continuously-valued, bounded dimensions. Actions represent position commands for the hip and knee joints of all four legs. Each command specifies a desired joint position to a Dynamixel actuator. For the Simulated Ant in MuJoCo, desired positions are tracked using proportional–derivative control whose gains and force limits were chosen to match the physical hardware (see Section 8 for system identification). In comparison, the Gymnasium Ant adopts a torque control strategy. Some prior real-world RL work uses position control (Miki et al., 2022). The software control interface can be changed to use position, velocity, or torque commands for other experiments.

Rewards and Tasks. The Gymnasium Ant's default task is to move forward as quickly as possible. The reward function is a sum of multiple terms: a reward for staying upright, called a *healthy*

reward, a *forward reward* for making forward progress, a *control cost* penalizing large actions, and an optional *contact cost* penalizing large external forces. The Gymnasium Ant also has episodic resets back to a starting configuration, which truncates episodes after long trajectories or on arrival to an unhealthy state. In contrast, we aim for learning in a non-episodic setting on the Physical Ant with a *simple reward formulation* based on progress alone.

Towards this end, we design a non-episodic task of moving back-and-forth. We define the robot's planar position at time $t \in \mathbb{N}$ to be $\mathbf{p}_t = [x_t, y_t] \in \mathbb{R}^2$, measured in the inertial (world) frame and a reward direction unit vector $\mathbf{u}_t \in \mathbb{R}^2$ defined in Equation 2. The instantaneous reward $r_t \in \mathbb{R}$ is given by the projection of the position displacement onto the current reward direction $\mathbf{u}_t$

$$r_t = (\mathbf{p}_t - \mathbf{p}_{t-1})^\top \mathbf{u}_t. \tag{1}$$

We further define a circle of radius $R > 0$ centered at $\mathbf{o} \in \mathbb{R}^2$. When the robot's position is outside the circle and it has made it to the half plane opposite from the previous reward direction update, a change in reward direction is triggered that compels the robot 'bounce back' from the circle, as follows:

$$\mathbf{u}_{t+1} = \begin{cases} \dfrac{\mathbf{o} - \mathbf{p}_t}{\|\mathbf{o} - \mathbf{p}_t\|_2}, & \text{if } \|\mathbf{p}_t - \mathbf{o}\|_2 > R \text{ and } (\mathbf{p}_t - \mathbf{o})^\top \mathbf{u}_t > 0, \\ \mathbf{u}_t, & \text{otherwise.} \end{cases} \tag{2}$$

An illustration of the switching behavior in Equation 2 is shown in Figure 2. The advantage of the back-and-forth task, in contrast to the forward task specified in the Gymnasium Ant, is that it does not require the user to bring a physical robot back to the origin when it reaches the edge of the learning arena. In this way, the robot learns forever, with minimal user intervention.

Learning Arena. The robot pursues the back-and-forth walking task in an arena similar to the one rendered in Figure 1(a). An overhead webcam supported by a tripod system tracks two fiducial markers (Olson, 2011): one placed on the ant's torso and another one placed on the ground. The system provides the position $\mathbf{p}$ used to compute the reward in Equation 1 and the heading.

3.3 Measuring Performance.

The performance of the learned policy is measured with the *average reward per second* defined as

$$\bar{r}_t = \frac{1}{N} \sum_{k=t-N+1}^{t} \frac{r_k}{\Delta t}, \tag{3}$$

where $r_k \in \mathbb{R}$ is the instantaneous reward, $N \in \mathbb{N}$ is the number of steps contained in the averaging window (corresponding to a user-defined window length), $\Delta t \in \mathbb{R}_{>0}$ is the time duration of one environment step, and $t \in \mathbb{N}$ is the current time step.

4 Reinforcement Learning Methods

We provide demonstrations of learning from physical experience for two representative RL algorithms, under the back-and-forth walking task described in the previous section. The first algorithm is an on-policy action-value method: SARSA(λ) with linear approximation of the value function (Sutton & Barto, 2018). SARSA(λ) has a small number of parameters and is easy to implement. The second algorithm is Soft Actor-Critic (SAC) (Haarnoja et al., 2018), a deep RL off-policy entropy-regularized method for continuous control problems. For each algorithm, we describe the implementation details in this section and report the corresponding experimental results in Section 5. These two demonstrations show that learning from physical experience is feasible. They also provide insight into the behavior and limitations of run-time learning with the Physical Ant platform.

4.1 Discrete Action RL: SARSA(λ)

SARSA(λ) is an on-policy RL algorithm where an agent interacts with the environment, learning an action-value function that defines the policy (for example, using ε-greedy action selection). The algorithm is defined with a discrete set of actions, so a transformation is needed from the robot's continuous action space.

Before applying SARSA(λ) as a learning algorithm for the Open Ant, we first defined a slower and more abstract agent-environment interface. The abstract learning agent interface operated at a slower rate (0.5 s) than what was used for communication to the robot (0.05 s), where each agent timestep lasted for $\tau = 10$ robot interactions. These timing values were a design choice. On each agent timestep, the agent received the latest observation from the robot. The intermediate rewards from the robot were accumulated before being sent to the learning agent, where the agent's reward between the robot timesteps t and $t + \tau$ is the discounted sum $r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{\tau-1} r_{t+\tau}$, where γ is the discount factor. For simplicity, we use the discounted formulation of SARSA(λ) rather than an average-reward formulation. The robot's continuous command space was abstracted into a discrete set of long duration motion primitives, which formed the learning agent's actions. We defined each agent action $\bar{\mathbf{a}}$ with an open-loop motion primitive $\langle \mathbf{a}_1, ...\mathbf{a}_\tau \rangle$, where τ was the fixed duration and $\mathbf{a}_i$, with $i \in [1, \tau]$, was the joint-space command sent to the robot at each robot timestep. The design of these motion primitives was inspired by observing the behavior of standard quadruped robots and animals during locomotion. Specifically, our motion primitives were defined as follows. For a hip joint, we used a linear ramp. For a knee joint, we use a half-sine with amplitude 1 or -1, corresponding to "lift leg" versus "push leg into the ground" primitives. A diagram of these motion primitives is shown in Section 15.

We approximated the true action-value function $Q(\mathbf{s}, \bar{\mathbf{a}})$, where $\mathbf{s} \in \mathcal{S}$ was the state, using linear function approximation with a tile coder (Sutton & Barto, 2018). Tile coding is based on the Cerebellar Model Articulator Controller (CMAC) (Albus, 1975) and was applied in RL before in Watkins (1989) and Sutton (1995). The robot's observation vector served as the agent's state. Each state $\mathbf{s}$ was mapped by the tile coder to a sparse binary feature vector $\phi(\mathbf{s})$, and each agent action $\bar{\mathbf{a}}$ had a separate weight vector $\mathbf{w}_{\bar{\mathbf{a}}}$. The estimated action value was defined by $\hat{Q}(\mathbf{s}, \bar{\mathbf{a}}) = \mathbf{w}_{\bar{\mathbf{a}}}^\top \phi(\mathbf{s})$. Each agent action was selected using an ε-greedy policy with respect to the current action-value estimate.

Using the abstract agent interface, we applied the standard SARSA(λ) algorithm with eligibility traces as described in Sutton & Barto (2018). Because our tasks were non-episodic, learning proceeds without episodic resets, and eligibility traces decay without being cleared at episode boundaries. With this formulation, SARSA(λ) operated as a fully online, continuing-control algorithm directly on the physical platform. The results of this implementation are presented in Section 5.

4.2 Continuous Action RL: Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy RL algorithm designed for continuous action spaces. SAC combines actor–critic methods with entropy regularization. The original algorithm (Haarnoja et al., 2018) learns a stochastic policy network, two action-value functions, and one state-value function network, all using a replay buffer. Newer implementations of the SAC algorithm omit learning the state-value function and achieve similar performance (Achiam, 2018). We have modified the CleanRL implementation (Huang et al., 2022) and applied it to the Physical Ant, with the observations and actions presented in Section 3.2. The action bounds were adjusted to constrain the knee to a 20° range with a 50° offset, and the hip to a 45° range with 0° offset.

5 Reinforcement Learning Results

We begin by introducing the common methodology employed by both SARSA(λ) and SAC, where learning is performed directly on board the Physical Ants. We then present the hardware results of these two reinforcement learning algorithms introduced in Section 4.

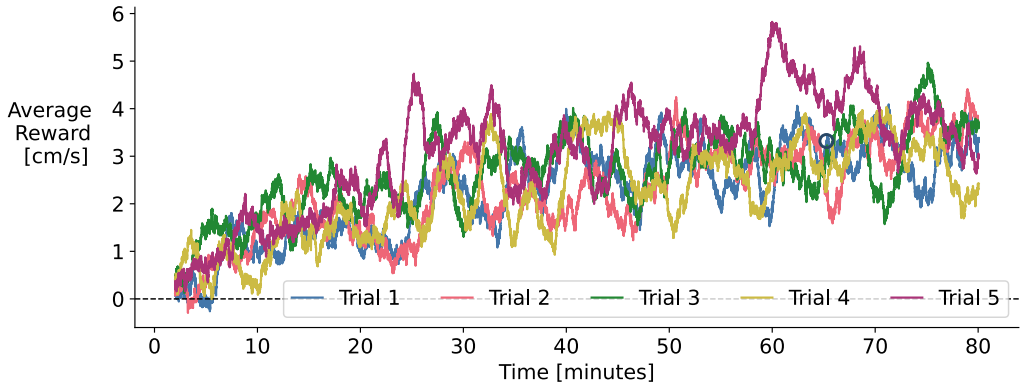

Figure 3: **Run-time learning performance of SARSA(λ) executed onboard the Physical Ant across 5 trials.** Average reward is computed using Equation 3 on a window of 120 seconds. These 5 trials had a total of one interruption (indicated with circles) caused by the leg failure in Figure 6(g). The leg was repaired within 10 minutes and the experiment was continued. The experiments were run on a Desktop computer with the Intel Core i9-13900K CPU.

Common Methodology. The task was the back-and-forth objective defined in Equation 1, in which the robot must walk within a specified circle. For all experiments, the radius of this circle was set to $R = 0.3$ m. This value was determined by the dimensions of the operational arena (Figure 1(a)). At the start of each trial, the power and communication cables were positioned outside the circular region. The Physical Ant was then placed inside the circle and the cables (power and communication) were arranged so that they were not entangled with each other and with the robot's body. We refer to this configuration as the start configuration. During learning, the robot occasionally became entangled in the cables. When this occurred, the experimenter manually paused the experiment, disentangled the robot, then reset the robot by returning it to the start configuration within the circle, and resumed the experiment. We have also ensured that the interaction was performed within the same fixed-duration control loop Δt. Lastly, for both tasks, have not designed a complex reward for the experiments presented, but used a simple progress reward, as seen in Equation 1.

Each RL algorithm was run on the robot for 80 minutes per trial, and we conducted five independent trials. SARSA(λ) was evaluated only on the Physical Ant, and SAC was evaluated on both the Physical Ant and the Physical Ant Lite. The evaluations used the performance metric from Section 3.3.

Evaluation of SARSA(λ). Learning with SARSA(λ) on hardware included several design decisions. First, to select good algorithm parameters at design time, we used Optuna (Akiba et al., 2019) for automated parameter optimization with the simulation. We conducted a sweep over the parameters listed in Section 12, evaluating each of 100 configurations across ten random seeds. The configuration that achieved the highest average reward was selected for the final experiments and is shown in Table 2.

Table 2: Parameters for SARSA(λ) with tile coding used for the experiments in Section 5.

Δt	Elig. Tr. λ	γ	ϵ	Step Size	Tiles per Dim.	Tilings	Tile Coder Table Size	Actions
0.5 s	0.964	0.998	0.255	0.0001	4	192	2^{25}	8

Next, the specification of observation bounds for the tile coding was important. Because we used a tile coding library Tiles3 that depends on the scale of the inputs, we precomputed realistic observation ranges by driving the robot through diverse motions in multiple directions. These measured

Table 3: Parameters for SAC used for the run-time learning on the Physical Ants.

γ	τ	Batch Size	Buffer Size	Learn Start	Reward Scale
0.99	0.005	256	10^6	2000	100.0
Δt	Actor Step Size	Critic Step Size	Entropy Coeff. Step Size	Policy Update Frequency	Target Update Frequency
0.12	0.0003	0.001	0.001	2	1

bounds were then used to normalize the observations prior to the generation of tile-features by the software library.

Third, the design of the motion primitives and ensuring they are realizable on hardware substantially influenced learning. In particular, we found that primitives coordinating two legs simultaneously, rather than actuating a single leg in isolation, produced more dynamically meaningful behaviors.

Results from five independent trials are shown in Figure 3. In every trial, the agent achieved slow yet competent walking behaviors (average reward of 2-4 cm/s) within approximately one hour of on-hardware learning. The key result is that learning is reliably observed across trials, even with a very simple RL algorithm.

Evaluation of SAC. We implement Soft Actor–Critic (SAC) on the two Ant platforms, the Physical Ant and the Physical Ant Lite. Both platforms share identical observation spaces, action spaces, and reward functions. Except for the modifications stated below, the algorithm parameters (Table 3) and network architectures for the policy and action-value functions follow the default configuration in the implementation of CleanRL (Huang et al., 2022).

First, we reduced the number of random interaction steps before learning begins from 5000 to 2000 (approximately 4 minutes of on-hardware random interaction). Empirically, this earlier start of gradient updates produced average reward performance comparable to the default setting, while reducing random actions taken on the robot, which could potentially damage it, and enabling faster learning (see Section 13.1 in the supplementary material).

Second, we incorporated Layer Normalization (LayerNorm) (Ba et al., 2016) into both the policy and critic networks. Prior work suggests that LayerNorm can stabilize learning (Elsayed et al., 2024); we verified this effect using our Simulated Ant, as shown in the supplementary material Section 13.1, across 30 simulation seeds. LayerNorm both accelerates learning and reduces variance. This faster convergence is particularly important in hardware settings, where interaction time is costly.

Third, we found reward scaling to be important for stable SAC training. The SAC algorithm has an entropy component that is sensitive to the reward scale. An initial reward scaling factor of 1.0 led to poor learning performance on hardware, whereas scaling rewards by a larger factor substantially improved performance and enabled consistent learning across seeds. We validated the effect of the scaling reward parameter in simulation across 30 seeds (supplementary material Section 13.1) and demonstrated that a larger value produces faster learning.

Lastly, we apply the modifications recommended in De Asis & Sutton (2024) to the return definition. This alleviates a idiosyncratic dependence on time-discretization for the algorithm. It decouples the optimization objective from Δt, making it a solution parameter instead of a problem parameter.

Figure 4 shows the average back-and-forth reward [cm/s] over time for five trials, on both physical platforms. By the end of training, all trials on both platforms exhibit stable, visually plausible walking behaviors, demonstrating that SAC can reliably learn locomotion directly on hardware in this configuration. We note that the SAC behavior had more interventions to clear cable entanglement than encountered with SARSA(λ), but the robot was also moving more with the SAC policies.

Translating policies from the simulator to physical reality using SAC. A popular approach to obtain competent robot behavior is to simulate an approximation of the robot and its environment,

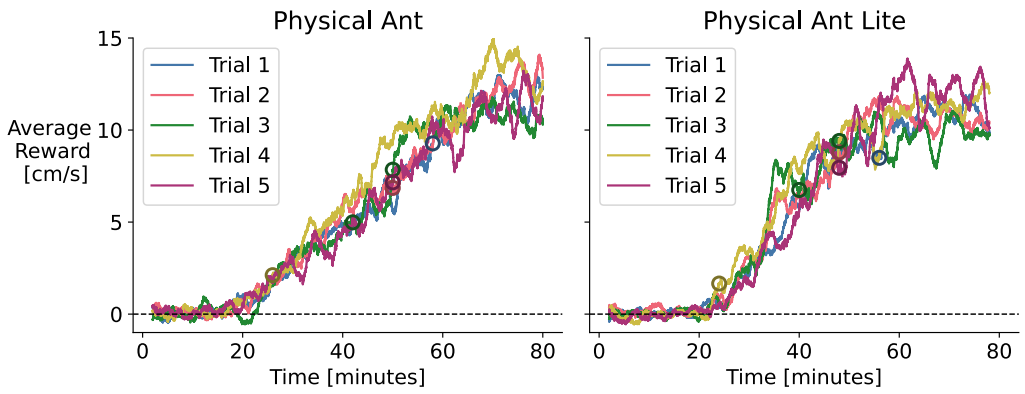

Figure 4: **Run-time learning performance using SAC on hardware for 5 trials.** Average reward is computed using Equation 3 for both the Physical Ant and the Physical Ant Lite on a window of 120 seconds. Circles indicate interventions, which are manual stops due to cable entanglement with the robot's legs. The experiment was run on a Macbook M1 with 16 GB of memory. The performance for one of the runs can be seen in Movie 2.

and learn a policy in simulation. The policy can then be deployed on the real robot as is, or adapted on the robot with additional experience.

We tested how policies learned on the Simulated Ant performed on the Physical Ant without further learning. We learned 2,304 policies in simulation using SAC with several parameter configurations (supplementary material Section 11) for a duration of 400,000 timesteps (the equivalent of about 13 hours). We have not done any domain randomization (Tobin et al., 2017). We then sampled ten policies, such that the average reward of the policies in simulation were uniformly distributed between 13 cm/s to 23 cm/s (which was the fastest policy observed). We label these policies 1 to 11 in decreasing order of their performance in simulation, with policy 1 being the best and policy 11 being the worst. Finally, we evaluated these policies on the Physical Ant for 5,000 steps (10 minutes), with no interventions. We report the mean performance in Figure 5. We noticed two key results.

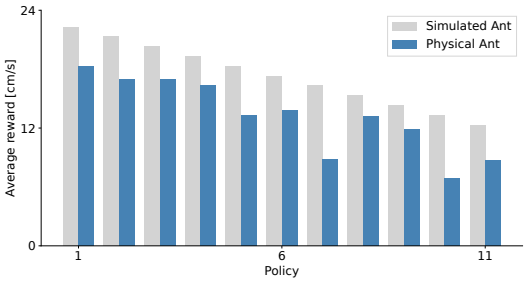

Figure 5: Evaluating policies learned using SAC on the Simulated Ant with the Physical Ant on the back-and-forth task. We chose ten policies of the Simulated Ant with reward rates between 23 cm/s to 13 cm/s and measured their performance on the Physical Ant.

First, all policies transferred with some success and enabled the robot to walk in the correct direction. This shows that the Simulated Ant captures the key aspects of the Physical Ant well. We also noticed that the less proficient policies make the ant entangled with the cable a lot more, for example policy 10 and policy 7.

Second, the ranking of the policies changed when they were transferred to the Physical Ant. For example, the policy that performed the best on the Simulated Ant was the sixth best on the Physical Ant. This change in ranking is an important result because policy improvement requires the ability to compare two policies. If the comparison is not accurate, then finding the best policy from a large number of policies learned in simulation poses challenges for hardware deployment.

Note that the policies from our sim-to-real analysis are not directly comparable because they are chosen from a set of parameters that do not always overlap with the parameters used for learning from physical experience (Table 3).

6 Examining the Platform Suitability for RL researchers

Most RL researchers have limited familiarity with robotics. Indeed, the authors have frequently heard RL researchers say that they avoid robotics experiments, due to difficulties they experienced in the past. One common problem is the long delay to the first successful experiment for a researcher using a novel robot with RL. Another common problem is the difficulty in repairing or adapting the physical system when failures arise. We share our experiences on both these common concerns. We have observed that multiple researchers could use the platform within a few days to learn policies, and we have found that many failures with the hardware can be rapidly corrected, due to the 3D printed design, and the use of commercially available (COTS) components.

6.1 Easy for Many to Build and Use

An important goal for the Open Ant platform is to enable researchers to build, deploy, and iterate on the platform with minimal effort. To date, the robot has been independently assembled and tested across multiple locations, including Canada, USA, and Malaysia. Once all the components were purchased, it took between 2 to 5 hours to assemble a full robot. The Open Ant was used for experimentation with RL at two meetings of academic researchers, with initial use at a workshop and a more thorough testing at the Openmind Research Institute Winter School in Malaysia.

The winter school provided evidence that the platform can be successfully used by researchers who were inexperienced in RL and robotics. At the school, five independent teams (total of 20 participants) successfully used and modified SAC and SARSA(λ) implementations. The participants tested sim-to-real transfer on the physical robot in an episodic setting (a walking forward task). The teams consisted of researchers who had limited experience with robotics and RL, and no prior experience with this particular platform. Nevertheless, all teams were able to demonstrate successful learning and transfer with three days of access to the physical platform. The physical platform was assembled by a local team of university robotics students, with a combination of locally available 3D printers, and externally sourced commercial components. The assembly process highlighted the platform's low barrier to entry; as noted in a testimony from the local assembly team: "once we understand how all the parts fit together, it takes less than two hours to assemble everything. When we tried building it (the Open Ant) without a guide, it took us around four to six hours working on it on and off. Most of that time was spent assembling, realizing mistakes, and then taking things apart to fix them". The participants used their own laptops to control the systems with a variety of operating systems (Linux, MacOS, Windows). An illustration of the experiments performed at the Winter School can be seen in Movie 6. These deployments demonstrate that the system is accessible to a broad range of researchers, and that the platform provides a rapid path to successful RL experiments with a robot.

We speculate that some design choices make this a good introductory physical platform for RL researchers. The combination of the simplicity of the platform, AC-powered long-duration operation, a standard Gymnasium interface, and compatibility with external compute platforms created a streamlined and engaging researcher experience.

6.2 Platform Reliability

From earlier experiments performed during the development of this platform, we observed several mechanical failure modes on the Physical Ant. As shown in Figure 6, these issues were primarily caused by ground impacts, joint loadings, and accumulated stress in 3D-printed components. In response, we iteratively improved the hardware design by reinforcing interfaces, redesigned vulnerable parts, and added protective elements such as 3D printed Thermoplastic Polyurethane (TPU)

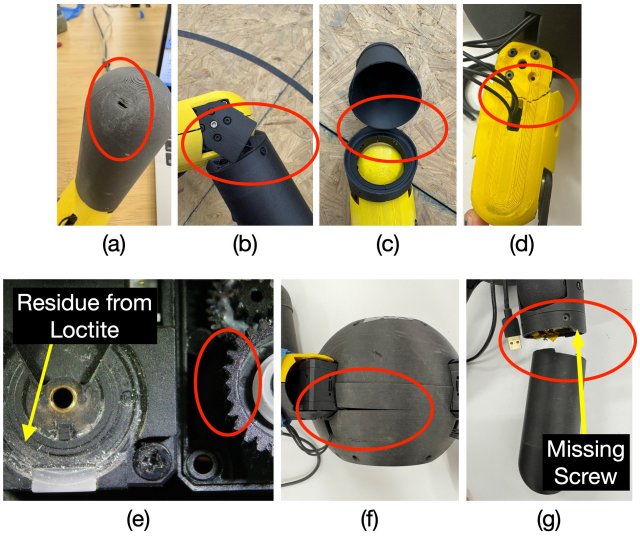

Figure 6: **Different failures observed during the development of the Physical Ant** (a) Foot-tip abrasion from repeated impacts, mitigated with 3D-printed TPU socks. (b) Hip cover fracture due to locomotion stress, redesigned with a stronger interface. (c) Leg shell damage after manual overloading; reinforced with internal ribs. (d) Leg housing cracks from accumulated stress, likely related to (e), interface strengthened. (e) Motor contamination from excess Loctite and plastic gear wear (XL430-W250-T). (f) Outer shell crack, mitigated by tightening screws to maintain structural integrity. (g) Leg broken during one of the SAC experiments. We notice a missing screw which could have affected the load distribution.

foot socks. We have also performed thermal improvements based on a detailed thermal analysis (Section 10.1), and power and communication fixes (Section 10.2). These modifications were important for enabling long learning runs. By releasing the hardware design as open-source, we aim to encourage further community-driven improvements to the platform's structural robustness and long-term reliability.

7 Discussion and Future Directions

We have presented the Open Ant as a platform for RL researchers who are familiar with simulation domains to include robot experiments in their evaluations. For our primary purpose of easing adoption, we have made the platform intentionally close to existing simulation environment and tasks, so that researchers can succeed with their initial attempts.

An RL researcher who is familiar with algorithm development in simulation may question the value of including robots as part of their research methodology. One argument in favor is that if our ultimate goal is to build learning agents that operate in the physical world, then experiments with the algorithms in the physical world are essential. Even for research programs that target non-physical domains, like agents for the PlayStation game GranTurismo (Wurman et al., 2022), researchers may not have direct access to the deployment environment with human participants. Those researchers may also find it valuable to evaluate their algorithms with a physical robot, where they can access a non-simulated deployment environment. A physical robot platform provides a valuable testbed for validating algorithms and exposes discrepancies between simulation and reality that would otherwise remain hidden.

7.1 Practical Concerns for Future Research

We consider some factors that can make the direct application of reinforcement learning algorithms to hardware more practical. These considerations are taken with respect to the current state-of-the-art.

Random Exploration. To safely deploy *current* learning algorithms on physical robots, their typical random exploration must not compromise the integrity of the robot. For example, motors, gearboxes, and structural components must tolerate the wear induced by this exploratory behavior. The Open Ant is designed to tolerate such exploration; it uses relatively large motors for its size, and the joint limits and kinematics prevent any self-collisions. The safety requirement imposes constraints on the robot design or the algorithm. The Open Ant addresses this issue in a straightforward way by over-sizing its actuators. More complex robots could require substantially more effort.

Unrecoverable States and Resets. Second, unrecoverable states should be rare. For example, our the robot can continue walking even if it rolls onto its back, provided the full knee range of motion ($\pm 70^\circ$) is used. The back-and-forth locomotion task was designed to require minimal human intervention by keeping the robot within the camera field of view. This allows the robot to learn without frequent interventions. The cable entangling with the robot did occasionally require intervention and in the future we will explore mitigating this through the use of a cable management system utilizing elements such as retractable cables, slip rings, or pulley systems. An adaptation of the Gymnasium humanoid walking task, an alternative we considered early in the design phase would require interventions which are trivial to implement in simulation (by resetting the environment) but burdensome in physical experiments. In summary, the design of the physical experiment should minimize physical human interventions and resets.

Performance vs. Compute We also observed that learning performance can vary depending on the specific computer instance on which the agent is executing (Section 14). Although we adjust the duration of the environment interaction step to maintain a fixed interaction frequency, the timing between issuing a motor command and reading sensor values for the next observation will depend on the time the agent takes to decide on its action. The impact of this timing on the Open Ant platform, as well as other factors that depend on compute speed, like camera processing latency, deserve further analysis. Furthermore, conventional agent environment interfaces, such as Gymnasium, lack specification of timing aspects that are unavoidable when operating in the real world. We acknowledge that timing remains a concern despite some existing works addressing it, for example Farrahi & Mahmood (2026); Yuan & Mahmood (2022); Rupam Mahmood et al. (2018).

Environment Changes. The environment changes over time as the agent learns from physical experience. For example, we observed that the robot gradually created holes in its feet during learning, altering their structural integrity, as shown in Figure 6(a). We have mitigated this by building 'socks' out of thermoplastic polyurethane material. In addition, repeated walking on a Medium-density fiberboard (MDF) wooden floor generated fine dust, which accumulated over time and changed the friction between the feet and the surface. Such effects are difficult to capture accurately in simulation and are often difficult for the designer to anticipate in advance, which motivates continual learning.

Reward Drop. We observed occasional runs in which the performance in simulation drops to nearly zero. Similar behavior has also been observed on the physical robot, as illustrated in Movie 4 for SARSA(λ). In both cases, the agent eventually recovers and resumes learning, as shown in Movie 5 for the simulation. It is unclear what precisely causes this performance drop, but it requires further investigation. This phenomenon is illustrated in Figure 13 in Section 13.2.

Simulator Exploits. A practical limitation of sim-to-real is that policies can exploit inaccuracies in the simulator rather than learn behaviors that transfer to the real world. Workarounds include intensive domain randomization (Tobin et al., 2017) and reward engineering. Although this concern does not apply to our hardware learning experiments, it was present in our early sim-to-real experiments (Section 5). As discussed in Section 8 and shown in Movie 3, we observed policies that achieved

high rewards in simulation by rapidly jittering the robot's feet, resulting from a bang-bang control strategy. While highly rewarded in simulation, this behavior could not be reproduced on the physical robot. This example illustrates how policies may exploit imperfect dynamics or contact models that are absent in the real world. Learning directly on hardware avoids this failure mode because optimization occurs under the true system dynamics rather than a simulated approximation.

Acknowledgments

The authors would like to thank Richard S. Sutton, Jun Luo, and Arsalan Shariffnassab for engaging discussions, Mohamed Elsayed for the suggestions to use CleanRL and LayerNorm for the SAC implementation, and Jose Uribe for help with some of the experimental setup building. We would also like to thank students Kevin Chew Ken Yi, Sim Sheng Wei and Lim Zi Quan at Universiti Tunku Abdul Rahman (UTAR), Malaysia, for implementing five instantiations of the Physical Ant for the Openmind Research Institute Winter School. We thank the NASA Jet Propulsion Laboratory for access to lab space and the Openmind Research Institute for funding.

References

Joshua Achiam. Spinning Up in Deep Reinforcement Learning, 2018.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019. DOI: 10.1145/3292500.3330701.

James Albus. New approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems*, 1975-09-30 1975. URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=820151.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. DOI: 10.1109/TSMC.1983.6313077.

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.

Marc G. Bellemare et al. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

H. Benbrahim, J.S. Doleac, J.A. Franklin, and O.G. Selfridge. Real-time learning: a ball on a beam. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pp. 98–103 vol.1, 1992. DOI: 10.1109/IJCNN.1992.287219.

Thomas A. Berrueta, Alex Pinosky, and Timothy D. Murphey. Maximum diffusion reinforcement learning. *Nature Machine Intelligence*, 6:558–567, 2024.

Rinu Boney, Jussi Sainio, Mikko Kaivola, Arno Solin, and Juho Kannala. RealAnt: an open-source low-cost quadruped for research in real-world reinforcement learning, 2020. URL https://arxiv.org/abs/2011.03085.

Jonas Buchli et al. Improving cosmological reach of a gravitational wave observatory using deep loop shaping. *Science*, 389(6764):1012–1015, 2025. DOI: 10.1126/science.adw1291. URL https://www.science.org/doi/abs/10.1126/science.adw1291.

Kris De Asis and Richard S. Sutton. An idiosyncrasy of time-discretization in reinforcement learning, 2024. URL https://arxiv.org/abs/2406.14951.

Jonas Degrave et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

General Dynamics. Brush-type DC motors handbook. PDF (online), 2020. URL https://www.gd-ots.com/wp-content/uploads/2020/07/Brush-Type-DC-Motors-Handbook_NAV.pdf. Accessed: 2025-11-26.

Mohamed Elsayed, Gautham Vasan, and A. Rupam Mahmood. Streaming deep reinforcement learning finally works, 2024. URL https://arxiv.org/abs/2410.14606.

Homayoon Farrahi and A. Rupam Mahmood. Learning without time-based embodiment resets in soft-actor critic. In *Proceedings of The 4th Conference on Lifelong Learning Agents*, volume 330 of *Proceedings of Machine Learning Research*, pp. 112–135. PMLR, 11–14 Aug 2026. URL https://proceedings.mlr.press/v330/farrahi26a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Shengyi Huang et al. CleanRL: high-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Leslie Kaelbling. RL: Rational Learning. Keynote talk presented at the Reinforcement Learning Conference (RLC), 2025, 2025. URL https://rl-conference.cc/2025/keynotes.html.

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pp. 2619–2624 Vol.3, 2004. DOI: 10.1109/ROBOT.2004.1307456.

Elena Sorina Lupu, Fengze Xie, James Alan Preiss, Jedidiah Alindogan, Matthew Anderson, and Soon-Jo Chung. MAGIC$^{\mathrm{VFM}}$-Meta-learning adaptation for ground interaction control with visual foundation models. *IEEE Transactions on Robotics*, 41:180–199, January 2025. ISSN 1552-3098. DOI: 10.1109/TRO.2024.3475212. URL https://doi.org/10.1109/TRO.2024.3475212.

A. Rupam Mahmood, Dmytro Korenkevych, Brent J. Komer, and James Bergstra. Setting up a reinforcement learning task with a real-world robot, 2018a. URL https://arxiv.org/abs/1803.07067.

A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. *CoRR*, abs/1809.07731, 2018b. URL http://arxiv.org/abs/1809.07731.

Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. DOI: 10.1126/scirobotics.abk2822. URL https://www.science.org/doi/abs/10.1126/scirobotics.abk2822.

P. Read Montague, Peter Dayan, and Terrence J. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16(5):1936–1947, 1996.

Michael O'Connell et al. Neural-Fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66), May 2022. DOI: 10.1126/scirobotics.abm6597. URL https://doi.org/10.1126%2Fscirobotics.abm6597.

Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pp. 3400–3407, 2011. DOI: 10.1109/ICRA.2011.5979561.

James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE International Conference on Robotics and Automation*, pp. 3299–3304, 2017. DOI: 10.1109/ICRA.2017.7989376.

James A. Preiss, Fengze Xie, Yiheng Lin, Adam Wierman, and Yisong Yue. Fast non-episodic adaptive tuning of robot controllers with online policy optimization, 2025. URL https://arxiv.org/abs/2507.10914.

A. Rupam Mahmood, Dmytro Korenkevych, Brent J. Komer, and James Bergstra. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4635–4640. IEEE Press, 2018. DOI: 10.1109/IROS.2018.8593894. URL https://doi.org/10.1109/IROS.2018.8593894.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using Generalized Advantage Estimation. *CoRR*, abs/1506.02438, 2015. URL https://api.semanticscholar.org/CorpusID:3075448.

Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

Xichen Shi, Patrick Spieler, Ellande Tang, Elena-Sorina Lupu, Phillip Tokumaru, and Soon-Jo Chung. Adaptive nonlinear control of fixed-wing VTOL with airflow vector sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5321–5327, 2020. DOI: 10.1109/ICRA40945.2020.9197344.

David Silver et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Tom Silver, Kelsey R. Allen, Josh Tenenbaum, and Leslie Pack Kaelbling. Residual policy learning. *CoRR*, abs/1812.06298, 2018. URL http://arxiv.org/abs/1812.06298.

Jean-Jacques E. Slotine and Weiping Li. *Applied nonlinear control*. Prentice Hall, Englewood Cliffs, N.J, 1991. ISBN 978-0-13-040890-7.

Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning, 2022. URL https://arxiv.org/abs/2208.07860.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995. URL https://proceedings.neurips.cc/paper_files/paper/1995/file/8f1d43620bc6bb580df6e80b0dc05c48-Paper.pdf.

Richard S. Sutton. The OaK architecture: A vision of SuperIntelligence from experience. Keynote talk presented at the Reinforcement Learning Conference (RLC), 2025, 2025. URL https://rl-conference.cc/2025/keynotes.html.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT Press, second edition, 2018.

Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pp. 2849–2854 vol.3, 2004. DOI: 10.1109/IROS.2004.1389841.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38 (3):58–68, 1995.

Joshua 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. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

Mark Towers et al. Gymnasium: A standard interface for reinforcement learning environments, 2025. URL https://arxiv.org/abs/2407.17032.

Christopher Watkins. Learning from delayed rewards, 1989. URL https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pp. 2226–2240, 2023.

Peter R. Wurman et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Yufeng Yuan and A. Rupam Mahmood. Asynchronous reinforcement learning for real-time control of physical robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.