Streamlined Sim-to-Real Transfer for Deep-Reinforcement Learning in Robotics Locomotion

Luigi Campanaro

Somerville College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2023

Abstract

Legged robots possess superior mobility compared to other machines, yet designing controllers for them can be challenging. Classic control methods require engineers to distill their knowledge into controllers, which is time-consuming and limiting when approaching dynamic tasks in unknown environments. Conversely, learning-based methods that gather knowledge from data can potentially unlock the versatility of legged systems.

In this thesis, we propose a novel approach called CPG-Actor, which incorporates feedback into a fully differentiable Central Pattern Generator (CPG) formulation using neural networks and Deep-Reinforcement Learning (RL). This approach achieves approximately twenty times better training performance compared to previous methods and provides insights into the impact of training on the distribution of parameters in both the CPGs and MLP feedback network.

Adopting Deep-RL to design controllers comes at the expense of gathering extensive data, typically done in simulation to reduce time. However, controllers trained with data collected in simulation often lose performance when deployed in the real world, referred to as the sim-to-real gap. To address this, we propose a new method called Extended Random Force Injection (ERFI), which randomizes only two parameters to allow for sim-to-real transfer of locomotion controllers. ERFI demonstrated high robustness when varying masses of the base, or attaching a manipulator arm to the robot during testing, and achieved competitive performance comparable to standard randomization techniques.

Furthermore, we propose a new method called Roll-Drop to enhance the robustness of Deep-RL policies to observation noise. Roll-Drop introduces dropout during rollout, achieving an 80% success rate when tested with up to 25% noise injected in the observations.

Finally, we adopted model-free controllers to enable omni-directional bipedal locomotion on point feet with a quadruped robot without any hardware modification or external support. Despite the limitations posed by the quadruped's hardware, the study considers this a perfect benchmark task to assess the shortcomings of sim-to-real techniques and unlock future avenues for the legged robotics community.

Overall, this thesis demonstrates the potential of learning-based methods to design dynamic and robust controllers for legged robots while limiting the effort needed for sim-to-real transfer.

Streamlined Sim-to-Real Transfer for Deep-Reinforcement Learning in Robotics Locomotion

Luigi Campanaro
Somerville College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2023

Acknowledgements

I would like to express my sincere gratitude to my supervisor Ioannis, for his guidance throughout my PhD journey.

I would also like to express my heartfelt appreciation to Daniele, Sid, and Wolf, my post-docs and friends, for their invaluable contributions, stimulating discussions, and constant encouragement. Their expertise and enthusiasm have been critical in helping me overcome challenges and push my research further.

I owe a debt of gratitude to my family for their unyielding love, support, and inspiration. My mother, Angela, my father, Vincenzo, and my brother, Domenico, have been my pillars of strength, always believing in me and encouraging me to pursue my dreams.

I would like to acknowledge the unwavering support of my girlfriend, Mariana, whose love, understanding, and encouragement have sustained me throughout my PhD journey. Her unwavering presence has been a constant source of motivation and joy.

My friends from Somerville college, Laura and Ondine, have been my confidants and companions, offering their unwavering support, kindness, and patience. I am deeply grateful for their friendship and the memories we've shared.

I am also grateful to my friends from GOAL, Charlie, Lara, Mark, and Mohamed, and my friends from the SRL, for their stimulating discussions, and unwavering support.

Finally, I would like to express my appreciation to all those who have supported me in countless ways throughout my PhD journey, whether through conversations, collaborations, feedback, or simply their presence. Thank you all for helping me achieve my goals and reach new heights.

Abstract

Legged robots possess superior mobility compared to other machines, yet designing controllers for them can be challenging. Classic control methods require engineers to distill their knowledge into controllers, which is time-consuming and limiting when approaching dynamic tasks in unknown environments. Conversely, learning-based methods that gather knowledge from data can potentially unlock the versatility of legged systems.

In this thesis, we propose a novel approach called CPG-Actor, which incorporates feedback into a fully differentiable CPG formulation using neural networks and Deep-RL. This approach achieves approximately twenty times better training performance compared to previous methods and provides insights into the impact of training on the distribution of parameters in both the CPGs and MLP feedback network.

Adopting Deep-RL to design controllers comes at the expense of gathering extensive data, typically done in simulation to reduce time. However, controllers trained with data collected in simulation often lose performance when deployed in the real world, referred to as the sim-to-real gap. To address this, we propose a new method called ERFI, which randomizes only two parameters to allow for sim-to-real transfer of locomotion controllers. ERFI demonstrated high robustness when varying masses of the base, or attaching a manipulator arm to the robot during testing, and achieved competitive performance comparable to standard randomization techniques.

Furthermore, we propose a new method called Roll-Drop to enhance the robustness of Deep-RL policies to observation noise. Roll-Drop introduces dropout during rollout, achieving an 80% success rate when tested with up to 25% noise injected in the observations.

Finally, we adopted model-free controllers to enable omni-directional bipedal locomotion on point feet with a quadruped robot without any hardware modification or external support. Despite the limitations posed by the quadruped's hardware, the study considers this a perfect benchmark task to assess the shortcomings of sim-to-real techniques and unlock future avenues for the legged robotics community.

Overall, this thesis demonstrates the potential of learning-based methods to design dynamic and robust controllers for legged robots while limiting the effort needed for sim-to-real transfer.

Contents

List of Figures

Qui se ultro morti offerant facilius reperiuntur quam qui dolorem patienter ferant.

It is easier to find men who will volunteer to die than to find those who are willing to endure pain with patience.

— Gaius Iulius Caesar's *Commentarii de Bello Gallico*

1 Introduction

Contents

1.1 Motivation

Advancements in automation technology have the potential to transform numerous industries by eliminating the need for human labor in physically repetitive and hazardous tasks. This transition towards greater reliance on autonomous robots is set to have a significant impact on sectors such as heavy industry, agriculture, construction, and forestry. By freeing human workers from the burden of these strenuous occupations, this new wave of automation holds the promise of ushering in a new era of productivity and prosperity for society as a whole. However, for robots to fully realize their potential in these domains, they require advanced cognitive skills and agile control of their bodies, equivalent to those of humans. Moreover, the emergence of versatile robotics solutions that can perform multiple physical tasks with the same hardware, without requiring specific hand effectors or bodies, would enable the focus to shift towards developing intelligent machines instead of solely engineering problems and solutions.

In this context, mobility is a critical skill that would allow robots to undertake a variety of tasks in diverse locations, with legs providing superior ability to navigate various terrains, ranging from artificial environments to natural ones.

Thus, the aim of this thesis is to investigate locomotion to enable robots to serve as reliable and effective mobility solutions in the near future.

Designing locomotion controllers involves navigating several pitfalls, including the complicated kinematic structure of a robot, multiple unspecified contacts with the environment, uncertain and partially known dynamics, and numerous physical constraints. While traditional approaches based on dynamic models have limitations in dealing with such complexities, learning-based methods have shown competitive performances.

To facilitate the collection of training data in a practical amount of time, Deep-RL methods often rely on simulators. In these simulations, complex objects are approximated as rigid bodies connected by joints and powered by actuators. This provides a potentially infinite data source and alleviates safety concerns with real robots, while allowing for the efficient training of agents to accomplish a variety of tasks, including locomotion control. Nonetheless, the gap between the simulated and real worlds can degrade the performance of policies once they are deployed to real robots. Consequently, multiple research efforts are being directed towards closing this sim-to-real gap and achieving more efficient policy transfer.

This thesis leverages Deep-RL to create robust locomotion controllers introducing innovative approaches to minimize hyper-parameter tuning and to address the sim-to-real gap. Moreover, the efficacy of learning-based methods applied to quadrupeds was pushed to the extreme by teaching quadruped robots bipedal locomotion, this experiment demonstrated that the sim-to-real gap remains a challenge in highly dynamic scenarios.

1.2 Contribution

The primary objective of this study was to develop more convenient methods for designing, training, and deploying robust locomotion controllers for legged robots using learning-based techniques.

We began by exploring the potential of CPGs for facilitating the process of designing locomotion controllers, leveraging their inherent characteristics, such as oscillatory behavior and robustness to perturbations. We developed a hopping controller in simulation combining our custom differentiable CPGs formulation with Deep-RL, this framework allowed us to easily include/train a Multilayer Perceptron (MLP) to provide non-linear feedback to the oscillators.

Building on these simulation experiments, we focused on deploying locomotion controllers on real machines, this time adopting purely model-free Deep-RL techniques. In particular, we aimed to reduce the burden of hyper-parameters selection and tuning required to address the sim-to-real gap, thereby obtaining very robust policies at the expenses of tweaking only a couple of parameters.

Having observed the impressive robustness of Deep-RL on quadruped robots, we sought to push the limits of these model-free controllers by adopting them to train a policy for the bipedal locomotion of a quadruped robot. The resulting policy was extremely performing in simulation, but the shortcomings of current sim-to-real techniques were evident when transferred to the real robot.

The main scientific contributions of the work undertaken as part of this Doctor of Philosophy study are:

- Development of a fully differentiable formulation of CPGs, allowing for direct training of parameters using gradient-based optimization techniques, and incorporation of an MLP network for feedback processing.

  Chapter 3, *Luigi Campanaro et al. "CPG-Actor: Reinforcement Learning For Central Pattern Generators". In:* Towards Autonomous Robotic Systems: 22nd Annual Conference, TAROS 2021, Lincoln, UK, September 8–10, 2021, Proceedings*. Lincoln, United Kingdom: Springer-Verlag, 2021, pp. 25–35.* URL: *`https://doi.org/10.1007/978-3-030-89177-0_3`*

- Introduction of a novel method, ERFI, for transferring locomotion controllers trained in simulation to hardware by randomizing only two parameters, eliminating the need for exhaustive system and distribution identification.

  Chapter 4*, Luigi Campanaro et al.* Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization*. 2022.* URL: *`https://arxiv.org/abs/2209.12878`*

- Development of Roll-Drop, a method to improve the robustness of Deep-RL based locomotion controllers to observation noise without exhaustive system identification, achieving an 80% success rate with up to 25% noise injected.

  Chapter 5*, Luigi Campanaro et al. "Roll-Drop: accounting for observation noise with a single parameter". In:* Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. Ed. by Nikolai Matni, Manfred Morari, and George J. Pappas. Vol. 211. Proceedings of Machine Learning Research. PMLR, 15–16 Jun 2023, pp. 718–730.* URL: *`https://proceedings.mlr.press/v211/campanaro23a.html`*

- Development of a the first bipedal omni-directional controller for quadruped robots with point-feet, trained in simulation and validated against hardware execution data.

  Chapter 6

- Development of the first bipedal controller for quadruped robots with point feet that can take 11 steps in the real world without hardware modifications.

  Chapter 6

To summarise, legged robots are versatile machines with superior mobility, but their complex controllers are difficult to design. Classic methods require engineers to distill their knowledge into the controllers, which can be time-consuming and limiting when approaching dynamic tasks in unknown environments. Conversely, learning-based methods gather knowledge from data, and have the potential to unlock the versatility of legged systems. This thesis followed the latter path to address several challenges in locomotion with the following contributions: a CPG-Actor architecture combining differentiable formulations, neural networks, and Deep-RL for joint CPG and MLP feedback learning; the ERFI method for streamlined sim-to-real transfer of locomotion controllers; Roll-Drop, a technique addressing observation noise in sim-to-real; and a model-free controller enabling omni-directional bipedal locomotion of quadruped robots in simulation, and for eleven steps on the real machine.

Ut est rerum omnium magister usus.

Experience is the teacher of all things.

— Gaius Iulius Caesar's *De Bello Civili*

2 Preliminaries

Contents

2.1 Introduction

The field of legged robotics has experienced a significant surge in interest in recent years, resulting from the availability of affordable hardware, particularly quadrupeds, on the market.

One of the key challenges in legged robotics is orchestrating the relative motions of the limbs, footholds, and reflex generation in the presence of external perturbations. This challenge remains a major area of focus, and researchers have proposed various techniques to address locomotion, with the two major approaches, model-based and learning-based, discussed in detail in sections 2.4 and 2.5.

2.2 Modern legged robots

In the 1980s, Marc Raibert's Leg Lab at MIT developed some of the first legged machines that exhibited dynamic behaviors closer to animals. These included a 3-D one leg hopper that could dynamically balance [4], a 3-D biped robot that could perform back-flips [5], and a quadruped robot controlled with a single virtual leg [6].

Some years later, Honda was working on ASIMO [7], a 52 kg humanoid robot equipped with various sensors, including cameras and microphones. ASIMO was capable of walking at a speed of 1.6 km/h.

Following the success of the Leg Lab at MIT, Raibert left academia and founded Boston Dynamics, which has produced some of the most capable and recognizable robots, including BigDog, Petman [8], Atlas, LS3, Spot, Wildcat, SpotMini, Handle, and Stretch. Initially, many of these robots were equipped with combustion engines and hydraulic actuators, which made them noisy.

Later, researchers at the Italian Institute of Technology (IIT) began working on HyQ, a 90 kg hydraulically actuated quadruped robot with an external power supply [9]. The experience gained from HyQ led IIT to develop a more compact version called HyQ2Max [10]. In 2019, they introduced HyQReal, which featured an on-board power supply.

As electric motors and batteries improved, researchers began to focus on this technology. One of the first examples was StarlETH, which featured twelve Series-Elastic Actuators (SEAs) that made it robust, energy-efficient, and capable of accurately measuring torque at the joints [11]. It demonstrated trotting at 0.7 m/s [12] and performed multiple dynamic maneuvers [13]. The development of

StarlETH led to the ANYmal robot [14], which demonstrated 1.5 m/s locomotion and versatility in industrial scenarios [15] and international robotic competitions [16].

In 2013, a game-changing design for electric actuators was proposed [17], which incorporated a low-ratio planetary gear set and a large-gap-radius motor for superior power density. Based on this design, Cheetah 2 demonstrated highly dynamic maneuvers [18, 19] that previous robots could not achieve with other electrical actuators. Its successor, Cheetah Mini, demonstrated even more dynamic maneuvers, including back-flips [20] and abrupt changes in direction [21]. The new planetary gear motors developed at MIT have also led to the production of cheaper machines by several companies, including Unitree Robotics and Deep Robotics.

In addition to Raibert's lab, other research groups were interested in bipedal locomotion. For example, ATRIAS [22] demonstrated dynamically stable stepping in 3D without external support using a spring-mass model. Its successor, Cassie, was built by the spin-off company Agility Robotics from Oregon State University and demonstrated impressive skills, such as walking blindly over stairs [23] and transitioning through several bipedal gaits [24]. The current iteration, Digit, is equipped with arms for manipulation [25].

While this thesis omits many other robots that have contributed to the progress of legged robotics, the emphasis on high-power machines is motivated by their potential to carry out useful tasks. High-power legged machines are characterized by motors that can exert high torque (∼30 N m to 80 N m, relatively to their weights ∼12 kg to 50 kg) and high velocities (∼10 $\mathrm{rad\,s^{-1}}$ to 20 $\mathrm{rad\,s^{-1}}$), as depicted in Figure 2.1.

In order to achieve navigation autonomy in real-world applications, legged robots must be equipped with advanced batteries, sensors, and robot arms, as shown in Figure 2.2. This additional equipment adds weight to the robot and places constraints on its shape and structure. Moreover, human-scale mobility is often required to navigate places built for human utilization or to explore disaster sites. As such, the focus on high-power machines in this thesis is driven by the need for legged robots that are capable of performing useful tasks in challenging environments.

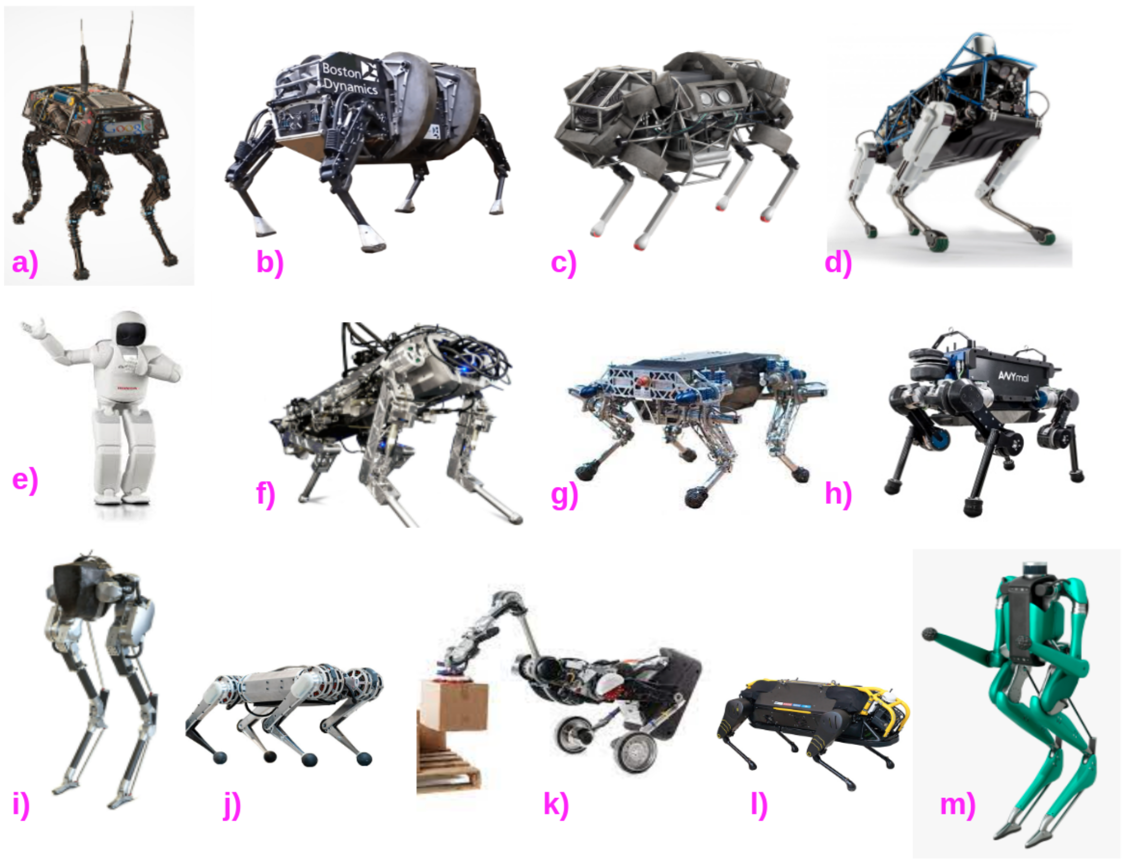


Figure 2.1: a) BigDog, b) LS3, c) WildCat, d) Spot Classic, e) Asimo, f) HyQ, g) StarlETH, h) ANYmal B, i) Cassie, j) Cheetah Mini, k) Handle, l) HyQ Real, m) Digit

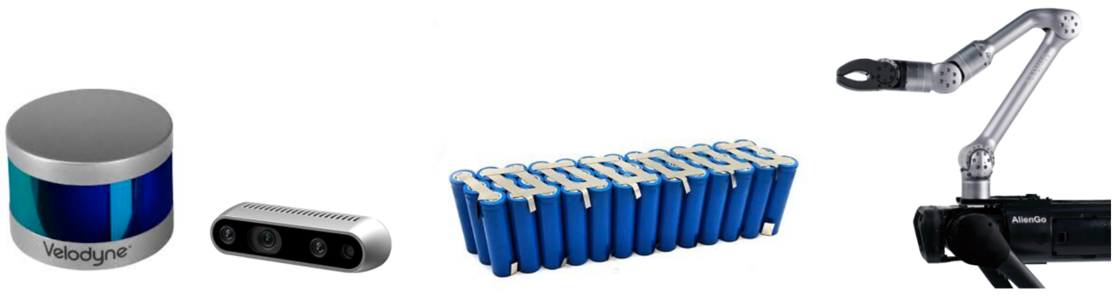


Figure 2.2: Laser scanners, depth cameras, battery packs, and robotic arms are some of the most common payloads legged robots mount to be autonomous and carry out useful tasks.

2.3 Platform systems overview

The platforms used in this study consists of three different robots: the ANYbotics ANYmal B, the ANYbotics ANYmal C, and the Unitree A1. These are all quadruped robots, each with its own set of capabilities and limitations (e.g. SEAs), which were carefully considered when designing the policies to ensure transferability from the simulation environments to the real world.

2.3.1 ANYbotics ANYmal B

The ANYmal B is a quadruped robot that uses SEAs for its motors. SEAs are a type of actuator that combines a motor with a spring, which provides mechanical compliance to the system and this compliance allows the ANYmal B to absorb impact loads and improve its dynamic performance. The SEAs on the ANYmal B have a maximum speed of 12 rad/s and a maximum torque of 40 Nm, this gives it a running speed of 0.8 m/s and a payload capacity of 5 kg. The robot is powered by a 48 V battery that provides an endurance of 1.5-2 hours. The ANYmal B is also equipped with a variety of sensors, including joint encoders, joint-torque sensors, IMU, Lidar, cameras, and a depth sensor, these sensors allow it to accurately control its motion and avoid obstacles.

2.3.2 ANYbotics ANYmal C

ANYmal C is the next generation of ANYbotics' autonomous legged robot, designed specifically for industrial inspection tasks. It features more powerful motors (80 Nm) and a higher payload (10 kg) than its predecessor, the ANYmal B. Additionally, ANYmal C is equipped with a variety of sensors, including LIDAR, depth cameras, and a pan-tilt inspection module with a visual and thermal camera (optional). This allows it to accurately map its environment and detect defects in equipment. ANYmal C is also more rugged and reliable than the ANYmal B, making it ideal for use in harsh industrial environments. It is IP67 water and dustproof, and can operate for up to 2 hours on a single battery charge.

2.3.3 Unitree A1

The Unitree A1 is a 12kg quadruped robot with a maximum running speed of 3.3m/s (11.88km/h) and a maximum payload of 5kg. The robot's kinematics are based on a four-bar linkage system with actuated joints at the hip, knee, and ankle. It has 12 independent brushless DC motors with a maximum joint torque of 33.5N.m and a maximum joint speed of 21rad/s. The motors are powered by a 24V battery and have an endurance of 1-2.5 hours. It is equipped with a depth camera for obstacle avoidance and a variety of sensors, including foot-end pressure sensors (not used for the applications discussed in this thesis), IMUs, and encoders.

2.4 Model-based approaches

Generating feasible motions for legged robots is a challenging task, as the movements of the base are not directly generated but instead result from the interaction between the limbs and the environment. To achieve the desired behavior, the contact forces must be carefully generated, while also accounting for constraints such as the fact that a force can only be generated if the feet are touching the ground, and that feet can only push, not pull, into it.

Crafting valid trajectories for the body, feet, and forces can be a laborious task due to the complexity of respecting these constraints. A wide range of control models exist for legged robots, ranging from simplified templates to full rigid body models.

The rigid body dynamics model of a quadrupedal system can be expressed in the form of generalized equations of motion:

$$\mathrm{M}\dot{\mathrm{u}} + \mathrm{h} = \mathbf{S}^T \tau_j + \mathrm{J}^T \lambda, \tag{2.1}$$

where $\mathrm{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is the mass matrix relative to the joints, $\mathrm{h} \in \mathbb{R}^{6+n_j}$ comprises Coriolis, centrifugal and gravity terms, $\mathbf{S}^T = [\mathbf{0}_{n_j \times 6} \;\; \mathbf{I}_{n_j \times n_j}]^T$, and J is the Jacobian which maps the contact forces $\lambda \in \mathbb{R}^{n_f}$ at $n_f = 4$ feet to generalized forces.

The representation of a dynamical system with Equation (2.1) is based on the assumption that the bodies in the system do not deform when subjected to external forces. As more assumptions are introduced, the resulting models become simpler and faster to compute, but their accuracy decreases. Continuing on this line of simplification, neglecting the momentum produced by joint velocities and assuming that the full-body inertia remains similar to the one in nominal joint position results in a less accurate model known as the Single Rigid Body Dynamics (SRBD). Adding more assumptions to the SRBD, as: the height of the Center of Mass (CoM) is constant, the angular velocity and acceleration of the base are zero, and that footholds are at a constant height, lead to the Linear Inverted Pendulum (LIP) model.

In the following sections, I categorized model-based control approaches for legged robots into two groups based on the complexity of the model employed. However, there is a gray area where trajectories are optimized adopting simplified models [26, 27].

The first group is a modular controller design, in which each module utilizes a template or heuristic. The second group is optimal control, which aims to compute a trajectory that minimizes a single high-level cost function. While this approach can result in highly optimized movements, it typically requires more computational resources than the modular approach. Overall, choosing the appropriate control model for a particular task involves balancing the complexity of the model with the available computational resources.

An additional variant involves the combination of trajectory optimization approaches with data-driven methods. In their work [28], the authors employed a Neural Network (NN) to expedite the convergence of a non-linear programming solver aimed at generating trajectories that anticipate obstacles for a quadruped robot. This procedure entailed generating diverse data sets for various tasks using offline trajectory optimization techniques. Following this step, a NN was trained in a supervised manner to provide a meaningful initialization for the Model Predictive Controller (MPC).

2.4.1 Hierarchical Controllers

Orchestrating the coordinated and effective motion of the limbs is a challenging task. To address this challenge, methods have been adopted that solve smaller, more tractable sub-problems. Controllers are assigned to each sub-problem, and new modules can be easily swapped in to generate different behaviors. Hierarchical architectures, which are interpretable by humans, scalable, and easier to maintain, are often used for this purpose.

One well-known example of modular controllers is the LIP model, which was initially developed by [29] to approximate the locomotion of single-leg hoppers. However, this method is not limited to controlling single-leg machines, and has been extended to biped and quadruped robots by incorporating virtual feet. These models simplify complex systems by assuming symmetries or low acceleration/velocity of the links, which reduces the complexity of the control problem.

In the hierarchical framework, the models represent only a single aspect of control. For instance, the inverted pendulum model is used to compute only the foot placement position or the virtual foot position (for biped and quadruped robots), and multiple small controllers operate in concert to maintain balance.

Following Raibert's findings, the introduction of the Zero-Moment Point (ZMP) [30] better formalized the adoption of fake contact points to control the motion of the CoM. This fast and effective approach was applied to real hardware locomotion, and additional evidence of its efficacy is shown in [31], which was the state-of-the-art controller for rough terrain locomotion for over ten years.

In this work, the authors proposed a cascade of different components: the footstep planner calculates optimal foothold choices for the next four steps, the pose finder optimizes the 6-D pose of the robot body - given the 3-D locations of the stance feet - to maximize kinematic reachability and to avoid configurations that collide with the terrain. The body trajectory generator uses the next four planned footholds to create smooth and ZMP-stable body trajectories. The foot trajectory planner, given a stable body trajectory and the desired footstep locations, generates a trajectory for the swing leg that avoids collisions with the terrain. Finally, the controller executes the plan with accuracy and robustness to perturbation, resorting to PID Control, Inverse Dynamics, and Force Control.

Later works such as [26] proposed a hierarchical controller that can execute dynamic gaits on quadruped robots, including trot, pace, dynamic lateral walk, as well as gaits with full flight phases as jumping, pronking, and running trot with smooth transitions among the gaits. The controller is based on an online zero-moment point motion planner that continuously updates the reference trajectory as a function of the contact schedule and the state of the robot. The desired footholds are obtained by solving a separate optimization problem, and the resulting motion plans are tracked by a hierarchical whole-body controller.

In [32], the authors developed a controller for the quadruped MIT Cheetah capable of running at a maximum speed of 6 m/s. The motion generation and tracking are organized into three modules: a programmable virtual leg provides instantaneous reflexes to external disturbances and facilitates self-stabilizing, a tunable stance-trajectory design adjusts impulse at each foot-end in the stance phase according to the equilibrium-point hypothesis, and a gait-pattern modulation imposes a desired pattern based on proprioceptive feedback.

Another proposed framework in [33] consists of two modules: the generation of elliptic trajectories for the feet based on task space CPGs and controlling the stability of the whole robot via a null space-based attitude control for the trunk and a push recovery algorithm (based on the capture point).

Despite the advantages of hierarchical controllers, their complexity requires a lengthy and tedious development process. They must be modified drastically for every new task, and crawling or other maneuvers that require different contact points may necessitate significant controller modifications. Additionally, hierarchical controllers often consider the legs massless, neglecting their dynamic effects and performing poorly in dynamic maneuvers. Lastly, modular controllers must optimize contact scheduling, which is a combinatorial problem beyond the capabilities of current algorithms, resulting in controllers based on heuristics or hand-coded contact scheduling.

2.4.2 Optimal Control

In response to the limitations of hierarchical controllers, which necessitate designing specific control architectures for the given task and tuning each submodule, researchers have increasingly turned to optimization algorithms to elicit emerging behaviors.

Typically, these algorithms take the form of an optimal control problem aimed at achieving long-term optimization of a scalar cost function. This involves considering a controllable, discrete-time, dynamical system represented by $x_{n+1} = f(x_n, u_n)$, where x_n denotes the system state at time n and u_n represents the control inputs. The main objective of optimal control is to specify a metric that evaluates the controlled system's performance and to determine the control inputs u_n at each time that optimize this performance.

However, the high dimensionality of locomotion renders designing trajectories with solvers from scratch impractical, necessitating simplifications or initial solutions to attain convergence within a reasonable time frame. For example, as is the case with hierarchical controllers, pre-specifying the contact points (such as the feet) allows to ignore selecting the body and limbs for such tasks, speeding up optimization.

Dynamic Programming (DP) is one of the numerical methods adopted to solve the optimal control problem, in this case the original problem is decomposed into a collection of simpler sub-problems, which are solved individually, and their solutions are stored. This iterative process estimates the value function through consecutive Bellman updates, DP and RL share a common approach that involves leveraging value functions to organize and structure the search for good policies [34]. Interest in DP-based approaches has recently increased, with studies applying them to controlling humanoids [35] and quadruped robots [36–39].

2.5 Learning-based control approaches

Learning-based control methods offer a promising alternative to classic control approaches by enabling the learning of effective controllers directly from experience, without relying on approximations of the robot's dynamics or engineering behaviors. Notably, learning-based techniques retrieve information about a controller's performance in simulation to optimize one or more task-representing functions, leading to highly dynamic and effective controllers. These methods typically employ a parameterized control policy representation, with learning achieved through optimization of these parameters via maximization of a loss function, such as the reward. The subsequent sections will explore two prominent learning-based control techniques: CPGs and Deep-RL.

2.5.1 Central Pattern Generators

CPGs are neural circuits that can produce rhythmic patterns in the absence of sensory inputs [40, 41]. They have been observed in both invertebrates and vertebrates. In the latter, the locomotor system is organized such that the spinal CPGs are responsible for producing basic rhythmic patterns, while higher-level centers such as the motor cortex, cerebellum, and basal ganglia modulate these patterns according to environmental conditions. This distributed organization presents several interesting features, such as reducing time delays in control loops by using short feedback from the spinal cord and reducing the dimensionality of the descending control signals by allowing a few inputs from higher-level centers to control the whole spine, as shown in Figure 2.3.

In the field of control engineering, artificial CPGs are implemented in the form of cyclic graphs that generate rhythmic patterns for control, Equation (2.2).

$$\begin{aligned}
\dot{\theta}_i^t &= 2\pi\nu_i(d_i^t) + \zeta_i^t \\
\zeta_i^t &= \textstyle\sum_j r_j^{t-1} w_{ij} \sin(\theta_j^{t-1} - \theta_i^{t-1} - \phi_{ij}) \\
\ddot{r}_i^t &= a_i(\tfrac{a_i}{4}(\rho_i(d_i^t) - r_i^{t-1}) - \dot{r}_i^{t-1}) \\
x_i^t &= r_i^t \cos(\theta_i^t)
\end{aligned} \tag{2.2}$$

where $\cdot^t$ describes the value at the t-th time-step, θ_i and r_i are the scalar state variables representing the phase and the amplitude of oscillator i respectively, ν_i and ρ_i determine its intrinsic frequency and amplitude as function of the input command signals d_i, and a_i is a positive constant governing the amplitude dynamics. The effects of the couplings between oscillators are accounted in ζ_i and the specific coupling between i and j are defined by the weights w_{ij} and phase ϕ_{ij}. The signal x_i represents the burst produced by the oscillatory centre used as position reference by the motors.

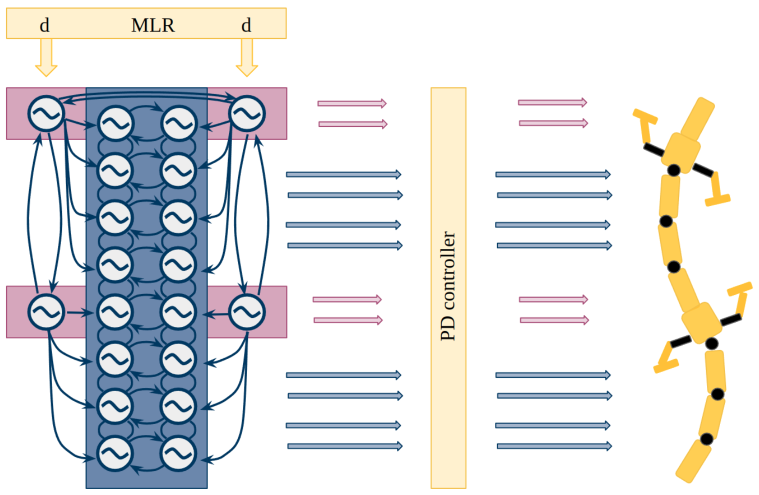


Figure 2.3: In the work presented in [42] the robot has ten motors that control the six hinge joints of the spine and four rotational joints for the limbs. The CPGs consists of two parts: body CPGs and limb CPGs. The body CPGs have 16 oscillators that control the spine motors, while the limb CPGs have four oscillators that control the limb motors. The CPGs receive signals from the brain stem that determine the setpoints for the motor controllers, which in turn control the motor torques to achieve the desired angles. The gait of the robot can be modified by adjusting the signals sent to the CPGs.

Unlike the deep learning community, which aims for a general representation for multiple purposes, the CPG community looks for a configuration that sufficiently describes the behavior of a specific animal or robot. As a result, CPGs require more experience and craftsmanship to be designed and tuned.

Models of CPGs have been used to control a variety of different types of robots and modes of locomotion, including swimming robots [42], quadruped walking robots [43, 44], hexapods, octapods, biped robots [40], and modular robots [45, 46]. The optimization of CPG-based controllers usually occurs in simulation through techniques such as Genetic Algorithms (GA) [40], Particle Swarm optimization (PSO) [46, 47], or expert hand-tuning [43, 44, 48, 49].

Prior work has evaluated the performance of CPGs for blind locomotion over flat ground [47]. However, navigating on rough terrain requires sensory feedback (e.g. to handle early or late contact), as shown in [44], where a hierarchical controller based on CPGs relied on a state machine to activate feedback. In particular, stumbling correction and leg extension reflexes were impulses triggered by the state machine, while the attitude control relied on information such as the contact status of each leg, the joint angles read by encoders, and the rotation matrix indicating the orientation of the robot's trunk. All of this data was processed in a virtual model control fashion and then linearly combined with the CPG equations. The angle of attack between the leg and terrain, which is useful for accelerating/decelerating the body or locomoting on slopes, was controlled by the sagittal hip joints and linearly combined with the CPG equations to provide feedback.

Similarly to [44], [49] also adopted feedback, this time based on gyroscope velocities and optical flow from a camera, to modify the CPGs output to maintain balance. The authors first tuned CPGs in an open-loop setting and then trained a NN with PSO to provide feedback while keeping the CPGs parameters fixed. Their method relied on a simple NN with seven inputs, four from the camera/optical flow and three from the gyroscope, and a single hidden layer.

In [50], the authors demonstrated how leg coordination can self-organize through a CPG model, with spontaneous gait transitions [51] between the most energy-efficient patterns exhibited only by changing the intrinsic angular velocity of oscillators in the CPG model without any pre-programmed gait patterns.

In robotics, CPGs offer several advantages, such as the ability to generate smooth output, and the ease of introducing sensory feedback without causing discontinuities in the output. Moreover, CPGs can encode arbitrary limit cycles, which can be useful for controlling various types of locomotion in robots.

However, despite their benefits, designing CPGs for a particular locomotor problem remains a challenge, and there is currently no standardized methodology for doing so. Additionally, proving the stability of the entire CPG-robot system is a difficult task, and designing effective feedback signals is an ongoing challenge [40].

Overall, while CPGs have shown promise for controlling a variety of different types of robots and modes of locomotion, there is still much work to be done in optimizing their performance and refining the design process. Future research in this area will likely focus on developing more effective design methodologies, and exploring new feedback injection techniques to enhance the capabilities of CPGs in robotics applications.

2.5.2 Reinforcement Learning

The objective of reinforcement learning is to determine the optimal policy for a Markov Decision Process (MDP). The MDP is characterized by a tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$, where $S \in \mathbb{R}^n$ and $A \in \mathbb{R}^m$ represent the state space and action space, respectively. The state transition probability $P : S \times A \times S \to [0, 1]$, with $P(s_{t+1} \mid s_t, a_t)$ denoting the probability of s_{t+1} given that at state s_t, the system takes the action a_t. The reward function $r : S \times A \to \mathbb{R}$ assigns a scalar reward for each system transition, and $\gamma \in [0, 1]$ is the discount factor. The goal of reinforcement learning is to learn a policy π that is parameterized by θ, where $\pi_\theta\left(a|s\right)$ denotes the probability of taking a_t given s_t, that solves the following optimization problem:

$$\begin{aligned} \max_\theta J_{rl}(\theta) =& \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \\ \text{subject to } & s_{t+1} \sim P(. \mid s_t, a_t) \\ & a_t \sim \pi_\theta(. \mid s_t) \end{aligned}$$

Policy gradient algorithms [52] are commonly used to tackle this problem. These algorithms estimate $\nabla_\theta J_{rl}$ using on-policy samples, i.e., data collected from the current stochastic policy.

Prior to pursuing the search for an optimal policy, it can be advantageous to contemplate techniques for evaluating the effectiveness of a specified policy. In the domain of RL, value functions serve as a key concept, offering an estimation of the expected future returns of a policy given a particular state [34].

The (state) value function, Equation (2.3), of a policy π, typically denoted by $V^\pi(s)$, provides an estimate of agent's expected return from following π starting at a given state s.

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \tag{2.3}$$

The $V^{\pi}(s_t)$ can be estimated, through consecutive interactions with the environment, in different ways: using Monte Carlo methods waiting until the actual return G_t is known, or by bootstrapping adopting Temporal Difference (TD) methods, which instead waits only until the next time-step to update the $V^{\pi}(s_t)$.
In practice, for Monte Carlo methods $V^{\pi}(s_t)$ is updated as: $V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha[G_t - V^{\pi}(s_t)]$, while for TD methods $V^{\pi}(s_t)$ is updated as: $V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha[R_{t+1} - \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)]$.
A way to interrelate Monte Carlo and TD methods consists in averaging the n-steps updates, this average would contain all the n-steps updates, each weighted proportionally to λ^{n-1} where $\lambda \in [0, 1]$; this algorithm is called $TD(\lambda)$,

Deep-Reinforcement Learning

Classic RL algorithms rely on a tabular representation of the value function, which limits their applicability to tasks with small numbers of states and actions. The main issue is not only the memory required to store large tables, but also the time and data needed to accurately fill them. When dealing with continuous variables or images, most encountered states will not be present in the table, making generalization a crucial aspect of the learning process [34].

A vivid illustration of this concept emerges when examining a 7 Degree of Freedom (DOF) manipulator. Upon discretizing its action space, considering values of $a_i = \{-k, 0., k\}$ for each joint, the resulting action space expands to $3^7 = 2187$ permutations. This challenge is widely recognized as the curse of dimensionality, illustrating the exponential growth in complexity.

To overcome these limitations, modern RL methods rely on function approximators, which is the essential ingredient for extending the capabilities of classic RL algorithms. Modern Deep-RL uses NNs as the primary method for implementing these parametrizations [34].

Physical control tasks, especially those involving legged robots, provide a tangible example of the complexities arising from the continuous and high-dimensional nature of action and observation spaces. These intricacies extend to other domains such as robotics manipulation, self-driving cars, and more. The pioneering work of [53] tackled these challenges by synthesizing insights from [54] and [55], resulting in policies capable of effectively handling continuous action spaces across diverse domains, even when learning from raw pixel data.

In recent years, Deep-RL has demonstrated outstanding results in several domains ([56–59]), including robotics. Specifically, in the area of legged locomotion, this technique has become increasingly popular. For the sake of brevity and relevance, we will focus only on RL applied to this domain.

In [60], the authors proposed an MLP-based controller, in Figure 2.4, which takes the robot's state history as input and produces joint position targets as output. The use of joint space control eliminates numerical issues at singular configurations, and the learning-based approach achieved a new level of locomotion, based purely on training in simulation, without tedious tuning on the physical robot. Additionally, modelling the SEA actuators with a NN allowed deploying the trained controller directly on the physical system, leading to more accurate and energy-efficient base velocity tracking and a 25% faster walking speed than model-based controllers.

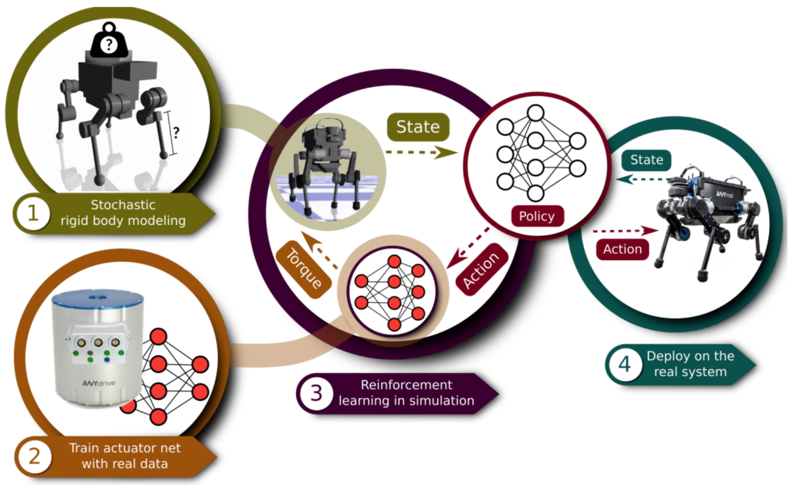

1 Stochastic rigid body modeling
2 Train actuator net with real data
3 Reinforcement learning in simulation
4 Deploy on the real system
State
Policy
Torque
Action
State
Action

Figure 2.4: In [60] the authors first identify the physical parameters of the robot and estimate uncertainties in the identification. Next, they train an actuator net to model complex actuator/software dynamics. Using the models produced in the first two steps, the authors then train a control policy. Finally, the trained policy is deployed directly on the physical system.

In addition to walking, [60] also presented a trained policy for recovering from a fall, which involves multiple unspecified internal and external contacts, as in Figure 2.5. Compared to classic approaches, learning-based methods shift all the computational costs to the training phase, resulting in faster and more reactive behaviors, and this is achieved without requiring the model of the system. Conversely, conventional optimization-based approaches require exact dynamic models and fixed contacts, struggling with multiple uncertain contact points. The same applies to controllers utilizing simplified templates and heuristics, relying on handcrafted sequences and models. Predictable behaviors in these methods hinder robustness during corner cases (e.g., legs trapped under the base) common in falls Figure 2.5, and developing such motions via traditional methods demands significant engineering endeavors [60]

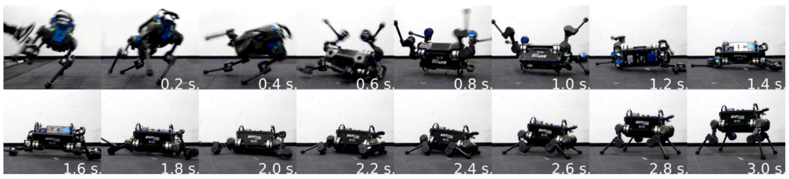


Figure 2.5: ANYmal recovering from a kick [61].

Legged locomotion can significantly expand the operational domains of robotics, particularly in environments inaccessible to conventional autonomous machines using wheels or tracks. However, conventional controllers based on elaborate state machines that explicitly trigger motion primitives and reflexes have become increasingly complex, without achieving the generality and robustness of animal locomotion. Conversely, in [62] the authors proposed a radically robust controller based on a NN that acts on proprioceptive signals and demonstrated remarkable zero-shot generalization from simulation to natural environments. Despite being trained with Deep-RL in simulation, the blind controller retains its robustness under challenging conditions that were never encountered during training, including deformable terrain, dynamic footholds such as rubble, and overground impediments such as thick vegetation and gushing water. This work opens up new frontiers for robotics, showing that radical robustness in natural environments can be achieved by training in much simpler domains.

A completely different approach, called MELA, was proposed in [63] to achieve versatile robot locomotion by learning to generate adaptive motor skills from a group of representative expert skills. MELA utilizes a Gating Neural Network (GNN) to combine pre-trained experts and dynamically synthesize a new deep neural networks (DNNs) to produce adaptive behaviors in response to changing situations, resulting in successful multi-skill locomotion on a real quadruped robot that performed coherent trotting, steering, and fall recovery autonomously.

In [24], the authors tackled the problem of achieving the full spectrum of bipedal locomotion on a real robot with sim-to-real Deep-RL. Designing intuitive reward functions to describe different gaits is a challenge in learning legged locomotion. Reference motions, such as trajectories of joint positions, are commonly used to guide learning. However, finding high-quality reference motions can be difficult, and the trajectories themselves narrowly constrain the space of learned motions. The authors proposed a reward specification framework based on simple probabilistic periodic costs on basic forces and velocities, with intuitive settings for all common bipedal gaits, including standing, walking, hopping, running, and skipping. Using this function, they successfully demonstrated sim-to-real transfer of the learned gaits to the bipedal robot Cassie, as well as a generic policy that can transition between all bipedal gaits.

Furthermore, in [23], the authors introduced a blind controller for the biped robot Cassie to traverse stairs. The controller's core is a recurrent policy trained with Deep-RL, without adopting any stair-specific reward.

In [64], the authors introduce a training setup that utilizes massive parallelism on a single (workstation) GPU to achieve fast policy generation for real-world robotic tasks. Their approach involves a game-inspired curriculum suitable for training thousands of simulated robots in parallel, and they investigate the impact of different training algorithm components in the massively parallel regime. The authors demonstrate the effectiveness of their approach by training the quadrupedal robot ANYmal C to walk on challenging terrain. Notably, their parallel approach enabled the training of policies for flat terrain in under four minutes, in twenty minutes for uneven terrain, and the policies were successfully transferred to the real robot, representing a significant speedup compared to previous work.

In [65] the authors proposed a controller that is solely trained in simulation using Deep-RL. The controller leverages exteroceptive feedback to inform the policy about the environment, enabling the robot to plan its steps in advance. While previous works [64] have demonstrated this functionality by combining model-based and learning-based methods [66], [65] increased the versatility of perceptive controllers even further, enabling them to work in cluttered environments where visual feedback can be unreliable. Specifically, the authors included an attention-based recurrent encoder, referred to as the belief encoder, in the policy that selectively attends to the input data and decides whether to trust the visual feedback or not. This new method allows the robot to work in environments with tall vegetation, where the grass can produce sudden spikes in the elevation map, or in other environments where the height map can become corrupt, leading the robot to perceive non-existent obstacles and potentially fall over.

2.5.3 Sim-to-real Transfer

Training policies directly on real robots using Deep-RL algorithms presents a challenge due to the substantial amount of interactions with the environment required, as well as the potential for damages that the robots may incur. As a result, policies are typically trained in a simulation environment before being transferred to physical robots. However, the reality-gap, which refers to the differences between the simulation and the physical robot, can cause naive transfer to be ineffective. As a result, finding a reliable method for transferring policies from simulation to physical robots is currently a significant area of research.

2.5.4 Domain Randomization

To address the reality gap, a viable solution is to train policies that are robust to potential modeling errors. By introducing randomness into the properties of the robot and into the environment it navigates during training, robust policies that successfully navigate the reality gap can be achieved. This approach is referred to as Domain Randomization and has proven successful in a variety of sim-to-real tasks, including manipulation [67, 68], quadruped [60, 61, 65, 69], and biped tasks [23, 24]. However, there are several limitations associated with the application of domain randomization. The technique often involves a broad range of randomization, which require extensive system identification to select the parameters to randomize and their respective ranges, and it can result in policies that are overly cautious and restrict the system's capabilities [70].

2.5.5 Domain Adaptation

Rather than aiming to create a policy robust to potential real-world variations, an alternative is resorting to policies that are adaptable to changes. To achieve this, a policy can be conditioned on several parameters, such as the robot's mass, the friction coefficient of the terrain, the height of the terrain [71, 72]. Often some of these parameters are available in simulation environments but not on the real robot, in this scenario it is possible to use a short history of the state as input [62] or neural networks with memory [68, 73] to allow the policy to perform adaptation implicitly.

2.5.6 Model-based Deep-RL

Model-based Deep-RL centers on the construction of environment models that allows agents to anticipate the outcomes of their actions. When provided with a state and an action, the model learns to predict the next state and reward. Learning these models offers several benefits: 1) more efficient utilization of interaction experiences, which can be used to improve the value function and policy, and to refine the model itself, and 2) the generation of simulated interaction with the environment that are used to update the policy [34].

In the context of model-based RL, our primary focus is on the contemporary application of Deep-NN as approximators for value function, policy, and environmental models, with a deliberate omission of tabular representations for conciseness. An influential milestone in this domain was the seminal work presented in [74]. This approach incorporated a Variational Autoencoder (VAE) to reduce the dimensionality of the state (2D images). The resulting low-dimensional latent representation was combined with previous actions, and a Recurrent Neural Network (RNN) was used to build the world model capturing long-term dependencies between states and actions. To demonstrate the effectiveness of their approach, the authors adopted a controller made up of a single linear layer that mapped the combination of the lower-dimensional perception representation (the 2D image compressed by the VAE) and the world model's hidden state into actions. This methodology illustrates the potential of training agents entirely within latent space worlds.

Model-based Deep-RL is not only confined to video-games [74], and found recent application on board of quadruped robots as well. In [75] the authors proposed a model-based approach for more efficient training of a locomotion policy on real robots, bypassing the need for simulation environments. This method substantially reduces real-world interactions and is rooted in algorithms introduced [76, 77], where the encoder fuses all the sensory modalities in discrete codes (made interpretable by the decoder), which are then used by a recurrent state space model to accurately predict future states. The resulting policy was trained on a quadruped robot in an hour, all without necessitating engagement with simulation environments or the employment of sim-to-real techniques.

2.5.7 Reward shaping, imitation learning, and AMP

Reward shaping is a delicate aspect of constructing the RL environment, playing a pivotal role in determining the final behavior of the agent. In more complex applications, such as legged locomotion involving quadruped robots, the specification of multiple rewards becomes necessary to precisely define the desired policy behavior [2, 60–62, 64, 66, 69]. These diverse reward terms are then aggregated through weighted sum to yield a single scalar utilized in the optimization process. The combination of reward terms and appropriate weights selection is known to as reward shaping.

Emphasized by [78], inaccurately specifying rewards lead RL agents to prioritize certain aspects of the observation and action spaces, often misaligned with human behaviors in the given context. This misalignment contradicts fundamental principles of common sense and engineering. In response, [78] presents several remedies: first, imitation learning circumvents the necessity of explicitly crafting rewards by enabling learning from demonstrations, second, human feedback can be integrated to evaluate episode quality, and third, learning across similar scenarios can produce a more intuitive reward function.

In the pursuit of developing viable locomotion policies for legged robots, we will explore streamlined approaches to reward specification: employing imitation learning with animal data, and adopting Adversarial Motion Priors (AMP) to emulate a specific style (gait) derived from a dataset.

In addressing the challenge of replicating the versatile locomotion abilities found in animals, [79] turned to imitation learning. In contrast to classic approaches, involving manual design of controllers and substantial expertise, Deep-RL offers an automated alternative to these manual efforts. Nonetheless, formulating effective learning objectives remains intricate. This study introduces an imitation learning system that empowers legged robots to acquire agile locomotion skills through emulating animal movements. Leveraging reference motion data, this approach synthesizes controllers to encompass a diverse range of behaviors, that combined with domain adaptation techniques facilitates the transfer of learned policies from simulation to an 18-DOFs quadruped robot.

With the objective of overcoming the challenge of intricate reward function tuning, the work from [80] introduces Multi-AMP, a Deep-RL approach that extends adversarial motion priors to enable the learning of multiple complex locomotion styles on legged robots. This method concurrently acquires diverse skills, even from trajectories optimization techniques. Experimental validation on a wheeled-legged quadruped robot demonstrates a wide range of abilities, including transitions between quadrupedal and humanoid modes. Multi-AMP and its precursor, AMP, offer a promising pathway to reduce reward function tuning in Deep-RL.

Techniques like imitation learning and AMP contribute to a more accurate, simpler, and more effective specification of the training objective, crucial to approach multiple and intricate tasks.

3
Training CPGs with Deep-RL

Contents

3.1 Overview of contribution

The advanced maneuverability of legged robots compared to wheeled or crawling robots demands sophisticated planning and control solutions. In the past, conventional control methods have dominated the field of legged robot control, but recently, data-driven methods have shown remarkable results surpassing classical approaches in terms of robustness and dynamic behaviors [62]. Specifically, controllers trained using deep reinforcement learning (RL) utilize a neural network (NN) policy to translate sensory information into low-level actuation commands, resulting in behaviors that cannot be engineered and are resilient to environmental events [2, 60, 61, 64, 65]. However, commonly used NN architectures, such as multi-layer perceptrons, do not inherently generate the oscillatory behavior observed in natural locomotion gaits, and require extensive training and reward-shaping to achieve smooth oscillations. Another family of controllers, CPGs, a biologically-inspired neural network capable of producing rhythmic patterns, have shown promising results for robot locomotion. However, despite these intriguing attributes, CPGs face the challenge of lacking a firmly established methodology for designing the desired limit cycle [40, 43, 81]. Also, unlike Deep-NNs, which offer versatile architectures for various applications, CPGs require a representation that adequately captures the behaviors of a specific animal/robot. Within this context, formulating the feedback –given the intricate interdependencies among the aforementioned aspects– poses notable challenges [40].

Hence, we argue that the full potential of CPGs has so far been limited by insufficient sensory-feedback integration and absence of state of the art techniques to shape the desired limit cycle. Thus, combining Deep-RL with CPGs could enhance the latter perception of the environment. However, optimizing Deep-NN in combination with CPGs requires methods to backpropagate the gradient from the loss to the parameters. To address it, this work introduces a novel approach for incorporating feedback into a fully differentiable CPG formulation adopting Deep-NNs and applying Deep-RL to jointly learn the CPGs parameters and MLP feedback. In contrast to prior work, the CPG is directly embedded as the actor in an Actor-Critic framework, allowing for direct encoding of task-specific characteristics (e.g., periodicity) without resorting to recurrent methods. The outcome is the CPG-Actor architecture, which enables end-to-end training of coupled CPGs and an MLP for sensory feedback using Deep-RL.

The main contributions of this paper are:

- For the first time, to the best of our knowledge, the parameters of the CPGs can be directly trained using gradient-based optimization techniques such as Proximal Policy Optimisation (PPO). This is made possible by proposing a fully differentiable CPGs formulation and a novel method for capturing their recurrent state without unrolling them over time.

- The fully differentiable approach also enables the incorporation and joint tuning of an MLP network responsible for processing feedback in the same pipeline.
- The results show roughly twenty times better training performance compared to previous state-of-the-art methods.

3.2 Integrated manuscript

CPG-Actor: Reinforcement Learning for Central Pattern Generators

Luigi Campanaro, Siddhant Gangapurwala, Daniele De Martini, Wolfgang Merkt, and Ioannis Havoutis

Oxford Robotics Institute
{luigi, siddhant, daniele, wolfgang, ioannis}@robots.ox.ac.uk

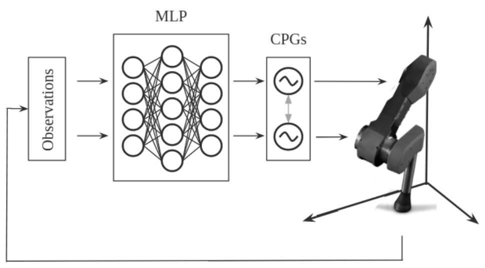
Abstract Central Pattern Generators (CPGs) have several properties desirable for locomotion: they generate smooth trajectories, are robust to perturbations and are simple to implement. However, they are notoriously difficult to tune and commonly operate in an open-loop manner. This paper proposes a new methodology that allows tuning CPG controllers through gradient-based optimisation in a Reinforcement Learning (RL) setting. In particular, we show how CPGs can directly be integrated as the Actor in an Actor-Critic formulation. Additionally, we demonstrate how this change permits us to integrate highly non-linear feedback directly from sensory perception to reshape the oscillators' dynamics. Our results on a locomotion task using a single-leg hopper demonstrate that explicitly using the CPG as the Actor rather than as part of the environment results in a significant increase in the reward gained over time (20x more) compared with previous approaches. Finally, we demonstrate how our closed-loop CPG progressively improves the hopping behaviour for longer training epochs relying only on basic reward functions.

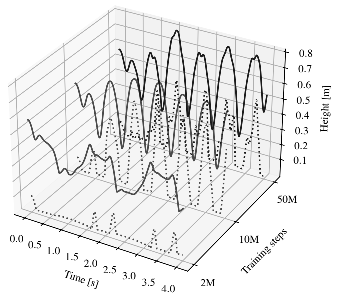
Keywords: Central Pattern Generators, Reinforcement Learning, Feedback Control, Legged Robots

1 Introduction

The increased manoeuvrability associated with legged robots in comparison to wheeled or crawling robots necessitates complex planning and control solutions. The current state-of-the-art for high-performance locomotion are modular, model-based controllers which break down the control problem in different submodules [1].This rigorous approach is rooted in the knowledge of every portion of the motion, but it is also limited by heuristics handcrafted by engineers at each of the stages.

While the field of legged robot control has been dominated over the last decades by conventional control approaches, recently, data-driven methods demonstrated unprecedented results that outpaced most of the classical approaches in terms of robustness and dynamic behaviours [2]. In particular, controllers trained using deep-RL utilise a Neural Network (NN) policy to map sensory information to low-level actuation commands. As a result, controllers trained with





(a) (b)

Figure 1: The experiments are carried out on a classic Reinforcement Learning (RL) benchmark – the single-leg hopper based on the ANYmal quadruped robot [3]. It hops along the vertical axis and is controlled by Central Pattern Generators (CPGs). Closed-loop feedback is incorporated using a jointly trained Multilayer Perceptron (MLP) network (Figure 1a). To demonstrate that the CPG-Actor progressively learns to jump higher peaks of both the hip (solid line) and foot (dotted line) heights (Figure 1b) are shown.

RL exhibit behaviours that cannot be hand-crafted by engineers and are further robust to events encountered during the interaction with the environment. However, widely-used NN architectures, such as MLP, do not naturally produce the oscillatory behaviour exhibited in natural locomotion gaits and as such require long training procedures to learn to perform smooth oscillations.

A third family of controllers have been used with promising results for robot locomotion: CPGs, a biologically-inspired neural network able to produce rhythmic patterns. However, very few design principles are available, especially for the integration of sensor feedback in such systems [4] and, although conceptually promising, we argue that the full potential of CPGs has so far been limited by insufficient sensory-feedback integration.

The ability of Deep-NNs to discover and model highly non-linear relationships among the observation – the inputs – and control signals – the outputs – makes such approaches appealing for control. In particular, based on Deep-NNs, Deep-RL demonstrated very convincing results in solving complex locomotion tasks [2, 5] and it does not require direct supervision (but rather learns through interaction with the task). Hence, we argue that combining Deep-RL with CPGs could improve the latter's comprehension of the surrounding environment. However, optimising Deep-NN architectures in conjunction with CPGs requires adequate methods capable of propagating the gradient from the loss to the parameters, also known as backpropagation.

To address this, this paper introduces a novel way of using Deep-NNs to incorporate feedback into a fully differentiable CPG formulation, and apply Deep-RL to jointly learn the CPG parameters and MLP feedback.

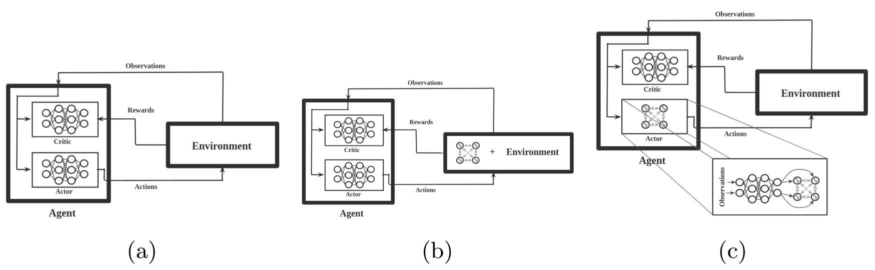


Figure 2: (a) represents the basic actor-critic Deep-RL method adopted for continuous action space control. (b) illustrates the approach proposed in [10–13], which consists in a classic actor-critic with CPGs embedded in the environment. (c), instead, is the approach proposed in the present work, which includes the CPGs alongside the MLP network in the actor critic architecture.

1.1 Related Work

Our work is related to both the fields of CPG design and RL, in particular to the application of the latter for the optimisation of the former's parameters.

CPGs are very versatile and have been used for different applications including non-contact tasks such as swimmers [6], modular robots [7] and locomotion on small quadrupeds [8]. The trajectories CPGs hereby generate are used as references for each of the actuators during locomotion and a tuning procedure is required to reach coordination. The optimisation of CPG-based controllers usually occurs in simulation through Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) or expert hand-tuning [6, 8].

To navigate on rough terrain sensory feedback is crucial (e.g. in order to handle early or late contact), as shown in [9]: here, a hierarchical controller has been designed, where CPGs relied on a state machine which controlled the activation of the feedback.

Similarly to [9], [8] also uses feedback, this time based on gyroscope velocities and optical flow from a camera to modify the CPGs output in order to maintain balance. However, in [8] the authors first tune CPGs in an open-loop setting and then train a NN with PSO to provide feedback (at this stage the parameters of the CPGs are kept fixed). We follow the same design philosophy in the sense that we preprocess the sensory feedback through a NN; yet, we propose to tune its parameters in conjunction with the CPG.

Actor-critic methods [14] rely on an explicit representation of the policy independent from the value function Figure 2a.

Researchers applied RL to optimise CPGs in different scenarios [10]. The common factor among them is the formulation of the actor-critic method; yet, they include the CPG controller in the environment – as depicted in Figure 2b. In other words, the CPG is part of the (black-box) environment dynamics. According to the authors [13], the motivations for including CPGs in the environment are their intrinsic recurrent nature and the amount of time necessary to train them, since CPGs have been considered Recurrent Neural Networks (RNNs) (which are computationally expensive and slow to train). In [10] during training and inference, the policy outputs a new set of parameters for the CPGs in response to observations from the environment at every time-step. Conversely, in [13] the parameters are fixed and, similarly to [8], CPGs receive inputs from the policy. However, whether the CPGs parameters were new or fixed every time-step, they all considered CPGs as part of the environment rather than making use of their recurrent nature as stateful networks. We exploit this observation in this paper.

1.2 Contributions

In this work, we combine the benefits of CPGs and RL and present a new methodology for designing CPG-based controllers. In particular, and in contrast to prior work, we embed the CPG directly as the actor of an Actor-Critic framework instead of it being part of the environment. The advantage of directly embedding a dynamical system is to directly encode knowledge about the characteristics of the task (e.g., periodicity) without resorting to recurrent approaches. The outcome is CPG-Actor, a new architecture that allows end-to-end training of coupled CPGs and a MLP for sensory feedback by means of Deep-RL. In particular, our contributions are:

1. For the first time – to the best of our knowledge – the parameters of the CPGs can be directly trained through state-of-the-art gradient-based optimisation techniques such as Proximal Policy Optimisation (PPO) [15], a powerful RL algorithm). To make this possible, we propose a fully differentiable CPG formulation (Section 2.1) along with a novel way for capturing the state of the CPG without unrolling its recurrent state (Section 2.1).
2. Exploiting the fully differentiable approach further enables us to incorporate and jointly tune a MLP network in charge of processing feedback in the same pipeline.
3. We demonstrate a roughly twenty times better training performance compared with previous state-of-the-art approaches (Section 4).

2 Methodology

As underlying oscillatory equation for our CPG network, we choose to utilise the Hopf oscillator [16] in a tensorial formulation, Equation (2).

Differently to previous approaches presented in Section 1.1, we embed CPGs directly as part of the actor in an actor-critic framework as shown in Figure 2c. Indeed, the policy NN has been replaced by a combination of an MLP network for sensory pre-processing and CPGs for action computation, while the value function is still approximated by an MLP network.

In practice, in our approach the outputs of the actor are the position commands for the motors. In [10], instead, the actor (MLP-network) outputs the parameters of the CPGs, that are then used by the environment (that includes the CPGs) to compute the motor commands. In this sense, there is a substantial difference in the architectures: in CPG-Actor, both the CPGs' and MLP's parameters are trained, while in [10] only the MLP's parameters are trained and the CPGs' ones are derived at runtime, being the output of the network.

However, a naïve integration of CPGs into the Actor-Critic formulation is error-prone and special care needs to be taken i) to attain differentiability through the CPG actor in order to exploit gradient-based optimisation techniques; ii) not to neglect the hidden state as CPGs are stateful networks.

We are going to analyse these aspects separately in the following sections.

2.1 Differentiable Central Pattern Generators

Since equations in [16] describe a system in continuous time, we need to discretise them for use as a discrete-time robot controller, as in Equation (1):

$$\begin{aligned}
\dot{\theta}_i^t &= 2\pi\nu_i(d_i^t) + \zeta_i^t + \xi_i^t \\
\zeta_i^t &= \textstyle\sum_j r_j^{t-1} w_{ij} \sin(\theta_j^{t-1} - \theta_i^{t-1} - \phi_{ij}) \\
\ddot{r}_i^t &= a_i(\tfrac{a_i}{4}(\rho_i(d_i^t) - r_i^{t-1}) - \dot{r}_i^{t-1}) + \kappa_i^t \\
x_i^t &= r_i^t \cos(\theta_i^t)
\end{aligned} \tag{1}$$

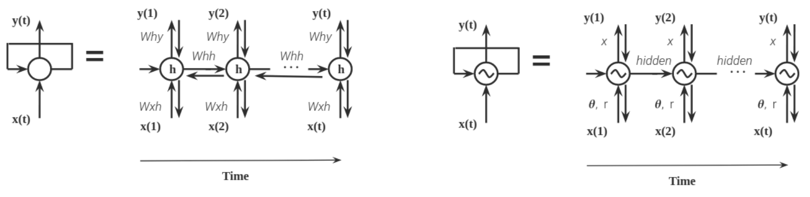
where $\cdot^t$ describes the value at the t-th time-step, θ_i and r_i are the scalar state variables representing the phase and the amplitude of oscillator i respectively, ν_i and ρ_i determine its intrinsic frequency and amplitude as function of the input command signals d_i, and a_i is a positive constant governing the amplitude dynamics. The effects of the couplings between oscillators are accounted in ζ_i and the specific coupling between i and j are defined by the weights w_{ij} and phase ϕ_{ij}. The signal x_i represents the burst produced by the oscillatory centre used as position reference by the motors. Finally, ξ_i and κ_i are the feedback components provided by the MLP network.

In order to take advantage of modern technology for parallel computation, e.g. GPUs, there is a strong need to translate the equations in [16] into a tensorial formulation (2) which describes the system in a whole enabling batch computations. Let N be the number of CPGs in the network, then:

$$\begin{aligned}
\dot{\Theta}^t &= 2\pi C_\nu(V, D^t) + Z^t\mathbf{1} + \Xi^t \\
Z^t &= (WV) * (\Lambda R^{t-1}) * \sin(\Lambda\Theta^{t-1} - \Lambda^\intercal\Theta^{t-1} - \Phi V) \\
\ddot{R}^t &= (AV) * (\tfrac{AV}{4}(P(V, D^t) - R^{t-1}) - \dot{R}^{t-1}) + K^t \\
X^t &= R^t \cos(\Theta^t)
\end{aligned} \tag{2}$$

Here, $\Theta \in \mathbb{R}^N$ and $R \in \mathbb{R}^N$ are the vectors containing θ_i and r_i, while $\Xi \in \mathbb{R}^N$ and $K \in \mathbb{R}^N$ contain ξ_i and κ_i respectively. $V \in \mathbb{R}^M$ contains the M, constant parameters to be optimised of the network composed by the N CPGs.

This said, $C_\nu : \mathbb{R}^M, \mathbb{R}^d \to \mathbb{R}^N$, $P : \mathbb{R}^M, \mathbb{R}^d \to \mathbb{R}^N$ and $A \in \mathbb{R}^{N\times M}$ are mappings from the set V and the command $D^t \in \mathbb{R}^d$ to the parameters that lead ν_i, ρ_i and a_i respectively. $Z \in \mathbb{R}^{N\times N}$ instead takes into consideration the effects of the couplings of each CPG to each CPG; all the effect to i-th CPG will be then the sum of the i-th row of Z as in $Z\,\mathbf{1}$, where $\mathbf{1}$ is a vector of N elements with



(a) (b)

Figure 3: The images above show the difference between back-propagation for classic RNNs (Figure 3a) and CPGs (Figure 3b). In particular to train RNNs, the matrices W_{xh}, W_{hy}, W_{hh} have to be tuned, where W_{hh} regulates the evolution between two *hidden states*. Instead, for CPGs only the parameters in $\dot{\theta}_i$ and $\ddot{r}_i$ (Equation (2)) need tuning, while the evolution of the *hidden state* is determined by an integration operation.

value 1. Within Z, $W \in \mathbb{R}^{N\times N\times M}$ and $\varPhi \in \mathbb{R}^{N\times N\times M}$ extrapolate the coupling weights and phases from V, while $\varLambda \in \mathbb{R}^{N\times N\times N}$ encodes the connections among the nodes of the CPG network.

The reader can notice how in (2) only already-differentiable operations have been utilised and that the MLP's output, i.e. the CPGs' feedback, is injected as a sum operation, enabling the gradient to backpropagate through the MLP network as well. This further enables us to compute the gradient of each of the parameters in (2) (CPGs and MLP) with respect to the RL policy's loss using the auto differentiation tools provided by PyTorch.

Recurrent state in CPGs In order to efficiently train CPGs in a RL setting, we need to overcome the limitations highlighted in [13]: In fact, CPGs are considered similar to RNNs (due to their internal state) and consequently they would have taken a significant time to train. In this section, we show how we can reframe CPGs as stateless networks and fully determine the state from our observation without the requirement to unroll the RNN.

RNNs are stateful networks, i.e. the state of the previous time-step is needed to compute the following step output. As a consequence, they are computationally more expensive and require a specific procedure to be trained. RNNs rely on Backpropagation Through Time (BPTT),Figure 3a , which is a gradient-based technique specifically designed to train stateful networks. BPTT unfolds the RNN in time: the unfolded network contains t inputs and outputs, one for each time-step. Undeniably, CPGs have a recurrent nature and as such require storing the previous hidden state. However, differently from RNNs, the transition between consecutive hidden states, represented by the matrix W_{hh}, in CPGs is determined a priori through simple integration operations without the need of tuning W_{hh}. This observation has two significant consequences: Firstly, CPGs do not have to be unrolled to be trained as the output is fully determined given the previous state and the new input. Secondly, eliminating W_{hh} has the additional benefit of preventing gradient explosion or vanishing during training, Figure 3b. As a result, CPGs can be framed as a stateless network on condition that the previous state is passed as an input of the system.

3 Evaluation

We evaluate our method on a classic RL benchmark: the hopping leg [17], which due its periodic task is a great fit for the application of CPGs. In fact, a single leg Figure 1a needs only two joints to hop and this is the minimal configuration required by coupled Hopf-oscillators to express the complete form; less than two would cancel out the coupling terms [16].

We based the environment on a single leg of the ANYmal quadruped robot, which was fixed to a vertical slider. Its mass is 3.42 kg, it is actuated by two series-elastic actuators capable of 40 N m torque, a maximum joint velocity of $15\,\mathrm{rad\,s^{-1}}$ and controlled at 400 Hz. We use PyBullet [18] to simulate the system and use a data-driven method to capture the real system's actuator dynamics.

At every time-step the following observations are captured: the joints' measured positions p_j^m and velocities v_j^m, desired positions p_j^d, the position p_h and the velocity v_h of the hip attached to the rail. While the torques t_j^d and the planar velocity of the foot $v_f^{x,y}$ are instead used in computing the rewards, as described in the following. To train CPG-Actor, we formulate a reward function as the sum of five distinct terms, each of which focusing on different aspects of the desired system:

$$\begin{aligned} r_1 &= (1.2 \cdot \max(v_h, 0))^2 & r_4 &= \sum_J -1.e^{-4} \cdot (t_j^d)^2 \\ r_2 &= \sum_J -0.5e^{-2} \cdot (p_j^d - p_j^m)^2 & r_5 &= -1.e^{-2} \cdot \left\| v_f^{x,y} \right\| \\ r_3 &= \sum_J -1.e^{-3} \cdot (v_j^m)^2 \end{aligned} \tag{3}$$

where $_J$ stands for joints.

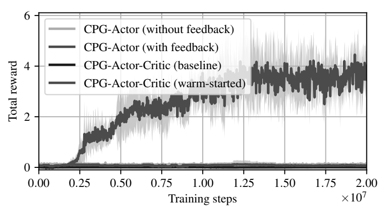
In particular, r_1 promotes vertical jumping, r_2 encourage the reduction of the error between the *desired position* and the *measured position*, r_3 and r_4 reduce respectively the *measured velocity* and the *desired torque* of the motors and finally, r_5 discourage the foot from slipping.

3.1 Experimental setup

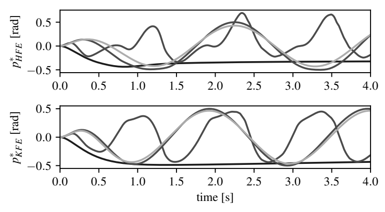
CPG-Actor is compared against [10] using the same environment. Both approaches resort to an actor-critic formulation, precisely running the same critic network with two hidden layers of 64 units each. Indeed, the main difference is the actor, which is described in detail in Section 2 for the CPG-Actor case, while [10] relies on a network with two hidden layers of 64 units each.

We trained the approaches for 20M time steps using an Nvidia Quadro M2200 GPU and an Intel(R) Xeon(R) E3-1505M v6 @ 3.00GHz CPU (8 cores) CPU; the process lasted roughly 2 hours.

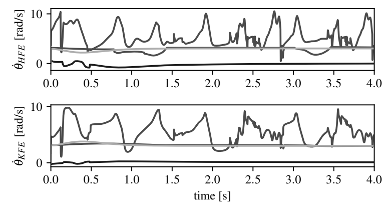
As Section 4 illustrates, an appropriate comparison between CPG-Actor and [10] required the latter to be warm-started to generate *desired positions* resulting in visible motions of the leg. Differently from the salamander [16], already tuned parameters are not available for the hopping task, hence a meaningful set from [9] was used as reference. The warm-starting consisted in training the actor network for 100 epochs in a supervised fashion using as target the aforementioned parameters.

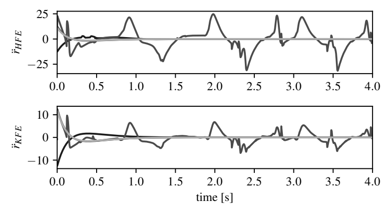


(a) Episode reward over 20M time steps horizon.



(b) Desired positions generated by CPG-Actor-Critic [10] and CPG-Actor.





(c) Comparison between $\dot{\theta}$, eq. (2), generated by CPG-Actor-Critic [10] and CPG-Actor.

(d) Comparison between $\ddot{r}$, eq. (2), generated by CPG-Actor-Critic [10] and CPG-Actor.

Figure 4: (Figure 4a) represents how the reward evolves during training, each approache run five times and averaging the rewards. (Figure 4b) trajectories generated by the different approaches: [10] warm-start produces an output similar to CPG-Actor without feedback. While CPG-Actor with feedback presents a heavily reshaped signal. The different contribution of the feedback in the two aforementioned approaches is explained by (Figure 4c) and (Figure 4d). The feedback – in CPG-Actor case – is interacting with the controller, resulting into visibly reshaped $\dot{\theta}$ and $\ddot{r}$ (green lines).

4 Results

4.1 CPG-Actor and previous baselines, comparison

The results of the comparison between CPG-Actor ans [10] can be seen in Figure 1a. Although the warm-starting procedure results in a performance improvement for [10] (red line vs blue line), CPG-Actor (green line) achieves roughly a twenty times higher reward after 20 million training time-steps.

We investigated the reason of such different performances and we argue it lies in the way the feedback affects the CPG controller. Figures 4c and 4d represent the evolution over time of the CPGs. Observing $\dot{\theta}$ and $\ddot{r}$ in experiments with [10] it is evident they do not show responsiveness to the environment, since the blue and the red lines remain almost flat during the whole episode. On the other hand, $\dot{\theta}$ and $\ddot{r}$ in CPG-Actor experiments (green line) demonstrate substantial and roughly periodic modifications over time. Although [10] relies on feedback information to infer the CPGs dynamics, in practise the effects of the feedback signals on the shape of the output variables are rather weak when compared to CPG-Actor, as visible in Figure 4b: in the case of CPG-Actor the original CPG's cosine output is heavily reshaped by the feedback, while [10] presents an almost-sinusoidal behaviour. Hence, to achieve successful hopping strong feedback information is crucial.

To further assess our intuition, we show CPG-Actor's open-loop (i.e. without feedback) behaviour (orange line), which shows performances on par with [10] after warm-start. Indeed, albeit explicitly penalised by Equation (3), both led to policies with the foot sliding on the floor and, as such, with low vertical velocity (yet slightly oscillating as if hopping); this behaviour results in low final rewards even after a large number of training episodes (20 M). It is then evident that the direct propagation of the gradient through a differentiable CPGs allows CPG-Actor to learn an effective correction to the open-loop behaviour through the sensor feedback.

4.2 Evaluation of progressive task achievement

The last set of experiments presented assess how CPGs' outputs and the overall behaviour evolve over the course of the learning. The plots in Figure 1 present the system at 1, 20 and 50 million time-steps of training. Figure 1b, shows the progress of the hopper in learning to jump; indeed, the continuous and dotted lines – respectively indicating the hip and the foot position – start quite low at the beginning of the training, to almost double the height after 50 millions time-steps.

5 Discussion and Future work

We propose CPG-Actor, an effective and novel method to tune CPG controllers through gradient-based optimisation in a RL setting.

In this context, we showed how CPGs can directly be integrated as the Actor in an Actor-Critic formulation and additionally, we demonstrated how this method permits us to include highly non-linear feedback to reshape the oscillators' dynamics.

Our results on a locomotion task using a single-leg hopper demonstrated that explicitly using the CPG as an Actor rather than as part of the environment results in a significant increase in the reward gained over time compared with previous approaches.

Finally, we demonstrated how our closed-loop CPG progressively improves the hopping behaviour relying only on basic reward functions.

In the future, we plan to extend the present approach to the full locomotion task by utilising the same architecture shown in Figure 1a with a CPG-network made of 12 neurons in order to be able to control a quadruped robot with 12 DOFs.

Acknowledgements

This work was supported by the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1], the UKRI/EPSRC RAIN [EP/R026084/1] and ORCA [EP/R026173/1] Hubs and the EU H2020 Project MEMMO (780684). The Titan V used for this research was donated by the NVIDIA Corporation. This work was part of the Human-Machine Collaboration Programme, supported by a gift from Amazon Web Services.

References

1. C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2261–2268, July 2018.
2. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986
3. M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2016, pp. 38–44.
4. L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, May 2008, pp. 819–824.
5. 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*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872
6. A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008.
7. S. Bonardi, M. Vespignani, R. Möckel, J. Van den Kieboom, S. Pouya, A. Spröwitz, and A. Ijspeert, "Automatic generation of reduced cpg control networks for locomotion of arbitrary modular robot structures," *Proc. Robotics: Science and Systems (RSS)*, 2014.
8. S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2013, pp. 194–201.

9. M. Ajallooeian, S. Gay, A. Tuleu, A. Spröwitz, and A. J. Ijspeert, "Modular control of limit cycle locomotion over unperceived rough terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, Tokyo, 2013, pp. 3390–3397.
10. Y. Cho, S. Manzoor, and Y. Choi, "Adaptation to environmental change using reinforcement learning for robotic salamander," *Intell. Serv. Robot.*, vol. 12, no. 3, p. 209–218, Jul. 2019. [Online]. Available: https://doi.org/10.1007/s11370-019-00279-6
11. A. L. Ciancio, L. Zollo, E. Guglielmelli, D. Caligiore, and G. Baldassarre, "Hierarchical reinforcement learning and central pattern generators for modeling the development of rhythmic manipulation skills," in *2011 IEEE International Conference on Development and Learning (ICDL)*, vol. 2, 2011, pp. 1–8.
12. Y. Nakamura, T. Mori, M. aki Sato, and S. Ishii, "Reinforcement learning for a biped robot based on a cpg-actor-critic method," *Neural Networks*, vol. 20, no. 6, pp. 723 – 735, 2007.
13. S. Fukunaga, Y. Nakamura, K. Aso, and S. Ishii, "Reinforcement learning for a snake-like robot controlled by a central pattern generator," in *Proc. IEEE Conf. Rob., Aut. Mech. (ICMA)*, vol. 2, 2004, pp. 909–914 vol.2.
14. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html
15. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347
16. A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007.
17. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016. [Online]. Available: http://arxiv.org/abs/1606.01540
18. E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2020.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	CPG-Actor: Reinforcement Learning for Central Pattern Generators
Publication Status	Published
Publication Details	**Campanaro, L**., Gangapurwala, S., De Martini, D., Merkt, W., Havoutis, I. (2021). CPG-ACTOR: Reinforcement Learning for Central Pattern Generators. In: Fox, C., Gao, J., Ghalamzan Esfahani, A., Saaj, M., Hanheide, M., Parsons, S. (eds) Towards Autonomous Robotic Systems. TAROS 2021. Lecture Notes in Computer Science(), vol 13054. Springer, Cham. https://doi.org/10.1007/978-3-030-89177-0_3

Student Confirmation

Student Name:	Luigi Campanaro		
Contribution to the Paper	Conceptualized the idea, implemented it, trained it, evaluated the results, contributed to final manuscript, and presented the work at the conference.		
Signature	Luigi Campanaro	Date	11/04/2023

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Dr. Ioannis Havoutis			
Supervisor comments: The contributions are as above			
Signature	Ioannis Havoutis	Date	19/04/2023

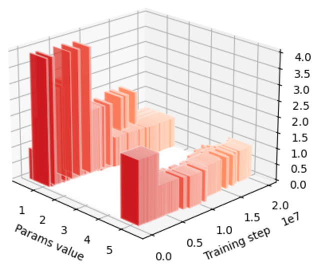
This completed form should be included in the thesis, at the end of the relevant chapter.

3.3 Further insights

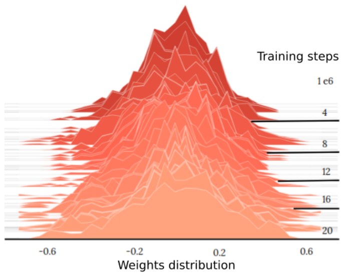
This section focuses on additional considerations that were not included in the manuscript for publication. Specifically, the impact of training on the distribution of parameters in both the CPGs and MLP feedback network, as well as qualitative results and insights into the baseline and proposed methods. Additionally, the related work will be updated based on developments that have occurred since the publication of the paper.

3.3.1 End-to-end training

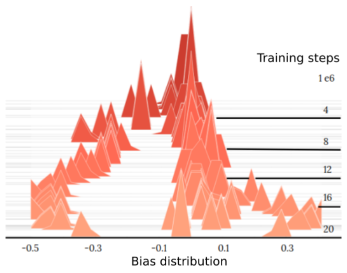
Figure 3.1 shows how the parameters belonging to both the CPG controller (Figure 3.1a) and the network that processes the feedback (Figures 3.1b and 3.1c) evolve in conjunction. This is signified by their distributions changing over the course of the learning process, from darker to lighter shades as the training process proceeds. Particular attention goes to CPGs' parameters (Figure 3.1a), they present two separate clusters due to the different magnitudes between coupling weights and the rest of the parameters, the almost discrete distribution emphasizes the low number of parameters that CPGs need compared to standard NNs.



(a) CPG-parameters distribution over time

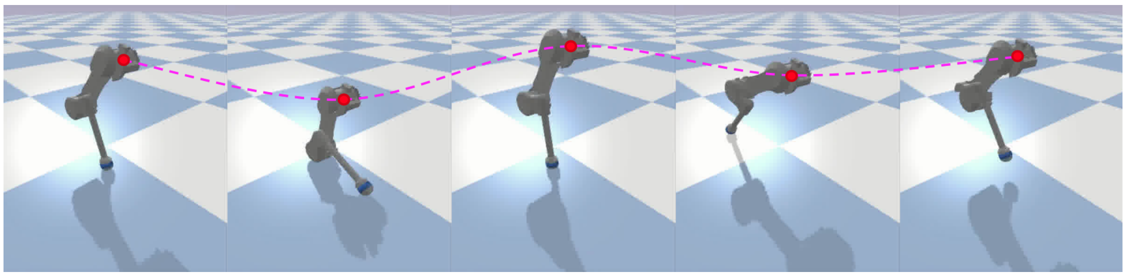


(b) MLP-feedback weights distribution over time

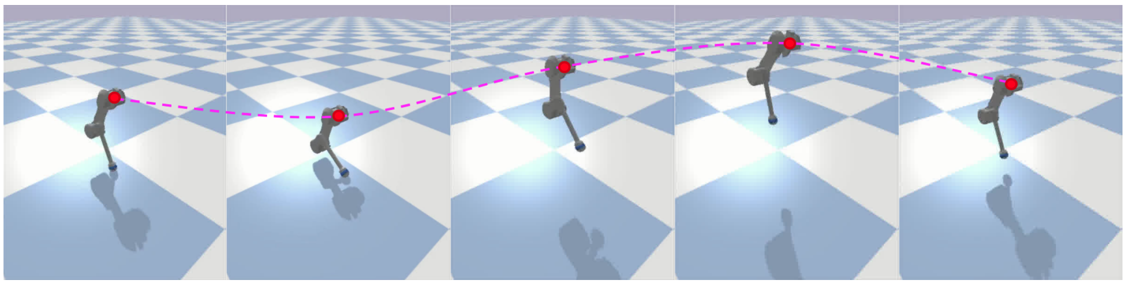


(c) MLP-feedback biases distribution over time

Figure 3.1: The set of images above show the evolution – from darker to lighter colours – of the distributions of CPGs parameters (Figure 3.1a), weights (Figure 3.1b) and biases (Figure 3.1c) of the output layer of MLP-feedback network. This demonstrates the simultaneous gradient propagation through the CPG and MLP parameters.



(a)



(b)

Figure 3.2: In Figure 3.2a we show the behavior of the baseline, while in Figure 3.2b our approach –CPG-Actor– in action.

3.3.2 Qualitative comparison

The performance differences between the baseline and our approach –CPG-Actor– are shown qualitatively in Figure 3.2. The baseline approach [82] trained on the hopping task fails to jump even after warm-starting, as depicted in Figure 3.2a. The foot drags on the ground and the base of the leg moves minimally along the vertical axis. On the other hand, CPG-Actor successfully performs tall jumps without slipping with the foot on the ground. The performance difference between the two methods can be attributed to the rewards punishing foot dragging, while at the same time encouraging the positive vertical velocity of the base of the leg.

3.4 Recurrent networks and back-propagation

Designing effective controllers for dynamical systems requires a profound understanding of the causal impact of each action on future states. When addressing this challenge using NNs, the initial impulse often involves adopting architectures with memory layers capable of capturing these relationships, such as RNNs.

However, despite the appealing attributes of RNNs, significant limitations constrain

their practical utility. Specifically, RNNs are confined to short-term horizons due to the emergence of vanishing and exploding gradient problems, which escalate exponentially with the number of time-steps considered [83, 84].
The groundbreaking work by [84] effectively addressed the issue of gradient vanishing/explosion by introducing the Long-Short Term Memory (LSTM) architecture. Through a gating mechanism that regulates access to the "memory" flow, LSTM can maintain long-term memory.
Similar to RNNs and LSTMs, CPGs are state-full networks, meaning they rely on information from the preceding time-step to compute subsequent outputs, potentially encountering similar limitations as RNNs.
However, a fundamental distinction arises: in contrast to RNNs, the transition between consecutive time-steps in CPGs is predetermined through simple integration operations. By circumventing the need to adjust hidden state weights –the underlying cause of gradient explosion/vanishing– two advantages emerge.
Firstly, CPGs do not require unrolling for training, as the output is entirely determined by the prior hidden state and new input. Secondly, the elimination of hidden state adjustments eradicates the risk of gradient-related issues during training.

3.5 Updated related work

Recently the authors of [85] proposed an approach that combines CPGs and Deep-RL, developing a framework called CPG-RL for learning quadruped locomotion control through the modulation of intrinsic oscillator amplitudes and frequencies. The method simplifies the controller structure proposed in [42]: it does so by retaining only four oscillators, one for each leg, instead of one per joint, by eliminating the coupling among them, and using inverse kinematics to control joint positions based on end-effector positions, enabling the user to set desired robot body height and swing foot ground clearance. The adoption of Deep-RL, according to the authors, opens up the possibility of exploring questions related to the roles and interactions of descending pathways, interoscillator couplings within CPG networks, and sensory feedback in gait generation in the field of neuroscience. According to the authors, the results suggest that stable locomotion can be achieved through the modulation of CPG circuits rather than direct action on muscles, that stable locomotion can be obtained with weak interoscillator couplings, and that sensing limb contact or loading is one of the most important sensory information for quadruped locomotion control.

3.6 Contribution review

The novel aspect of the CPG-RL framework, as discussed in Section 3.5, lies in its simplification of the CPG network, achieved by reducing the number of CPGs and eliminating the couplings, as well as its hardware experiments. The CPG-RL framework shares similarities with previous approaches [82, 86, 87], in that the CPG network is part of the environment and its parameters are generated by the output of the actor network instead of being optimized through gradient descent. This results in oscillator configurations that change at each time step, which differs from the fixed configurations in previous CPG studies [40, 42, 43, 47, 49, 88, 89].

On the other hand, our work [2] proposed a different approach by directly tuning the CPG parameters through gradient descent and enhancing their capabilities with feedback from an MLP network. This approach demonstrated better performance compared to previous methods that change the oscillators' configuration at every time step. At the time of submitting this document, CPG-Actor has not yet been compared to CPG-RL.

3.7 Transitioning to real-world implementation

This chapter introduced the CPG-Actor method and showcased its effectiveness through simulations involving an ANYmal B hopping leg.

Having gained valuable insights into controller design through simulation, the research trajectory naturally shifted towards practical implementation on real robotic platforms.

The adoption of data-inefficient techniques like Deep-RL, which necessitates reliance on simulators for efficient controller training, in combination with real robots, shifted the attention towards bridging the sim-to-real gap.

In a methodical approach to address this challenge, the decision was made to postpone the integration of CPG-Actor with real robots. This strategy prioritized the acquisition of the necessary skills to effectively tackle the hurdle of sim-to-real transfer first.

However, as the complexities of sim-to-real transfer were engaged, this aspect gained prominence in following studies, marking a clear shift in focus evident in the upcoming chapters.

4
Filling the sim-to-real gap with two parameters

Contents

4.1 Overview of contribution

The field of legged robotic control has been revolutionized by Deep-RL, which enables highly dynamic and sophisticated locomotion capabilities [60, 61, 64, 65]. Due to the sample complexity associated with high-dimensional problems –like locomotion– physics simulators are typically adopted for training Deep-RL control policies. However, there is often a non-negligible discrepancy between the simulated training domain and the physical target domain, known as the reality gap. Strategies to address this reality gap include the identification of sensory noise, accurate system dynamics identification, and training a NN to model actuation dynamics.

Indeed the actuators are a crucial part of legged systems, and their dynamics are difficult to model involving nonlinear/nonsmooth dissipation, feedback loops, and several internal states that are not directly observable, as in SEAs.

To address all the issues, in this work we propose a new method called ERFI that randomizes only two parameters to address the sim-to-real transfer of locomotion controllers: an actuation offset, and random torque perturbations at each step.

We demonstrated the effectiveness of ERFI on legged systems including comparisons with its predecessor, Random Force Injection (RFI) [90], variations of the same method, and standard domain randomization. The simulation experiments revealed that ERFI yielded a significant improvement in success rates for varying masses of the base and for attaching a manipulator arm to the robot during testing. Moreover, ERFI achieved competitive performance when compared to standard randomization techniques, while requiring tuning only a fraction of the parameters.

To demonstrate the efficacy of the method we successfully deployed perceptive and blind policies trained in simulation with ERFI on to the physical ANYmal C and Unitree A1 quadrupeds, showing that training of actuator networks and performing significant dynamics randomization can be substituted by this simple strategy.

4.2 Integrated manuscript

Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization

Luigi Campanaro, Siddhant Gangapurwala, Wolfgang Merkt, and Ioannis Havoutis
Dynamic Robot Systems Group (DRS), University of Oxford
{luigi,siddhant,wolfgang,ioannis}@robots.ox.ac.uk

***Abstract*— Training deep reinforcement learning (DRL) locomotion policies often require massive amounts of data to converge to the desired behavior. In this regard, simulators provide a cheap and abundant source. For successful sim-to-real transfer, exhaustively engineered approaches such as system identification, dynamics randomization, and domain adaptation are generally employed. As an alternative, we investigate a simple strategy of *random force injection* (RFI) to perturb system dynamics during training. We show that the application of random forces enables us to emulate dynamics randomization. This allows us to obtain locomotion policies that are robust to variations in system dynamics. We further extend RFI, referred to as extended random force injection (ERFI), by introducing an episodic actuation offset. We demonstrate that ERFI provides additional robustness for variations in system mass offering on average a 53% improved performance over RFI. We also show that ERFI is sufficient to perform a successful sim-to-real transfer on two different quadrupedal platforms, ANYmal C and Unitree A1, even for perceptive locomotion over uneven terrain in outdoor environments.**

Additional resources at: `https://sites.google.com/view/erfi-video`

I. Introduction

Deep reinforcement learning (DRL) has emerged as a promising approach for legged robotic control enabling highly dynamic and sophisticated locomotion capabilities [1], [2], [3]. The sample complexity associated with high-dimensional problems such as locomotion makes the use of physics simulators [4], [5] appealing for training DRL control policies. This convenience, however, often requires addressing the *reality gap* between the simulated training domain and the physical target domain.

Strategies to address this reality gap often include identification of sensory noise which is then modeled and introduced in simulation during training [6], [7]; accurate parameter identification (of properties such as Center of Mass (CoM), mass and inertia of robot links, impedance gains, system communication delays, and friction) for system modeling in addition to identification of relevant distributions suitable for domain randomization [8], [1]; and training a Neural Network (NN) to model the actuation dynamics of specific actuators, e.g. Series Elastic Actuators SEAs [7], [9].

As an alternative to exhaustive system identification and distribution identification for dynamics randomization, [10] demonstrated captivating performance in sim-to-real for manipulation tasks using an extremely simple RFI strategy. RFI enables emulation of dynamics randomization through

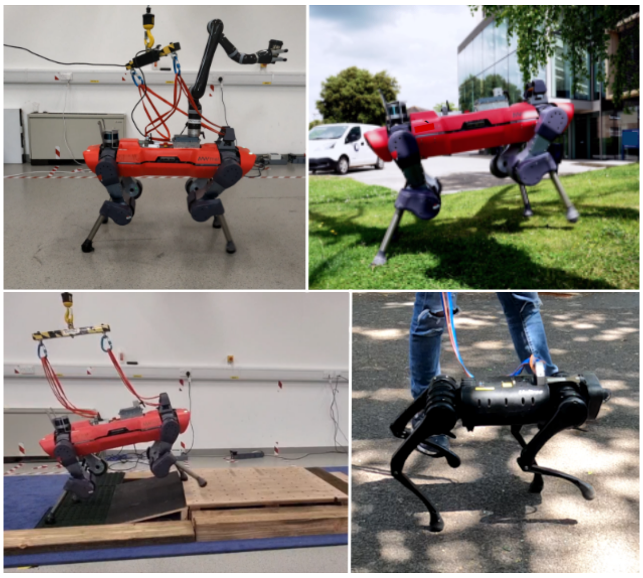


Fig. 1: Deployment of the perceptive and blind locomotion policies on the ANYmal C and Unitree A1 quadrupedal platforms trained using our proposed ERFI-50 strategy without requiring actuation modeling or explicit randomization of dynamics or actuation properties.

perturbation of system dynamics with randomized forces. However, as presented in Section VIII, locomotion policies trained using RFI exhibit subpar robustness to policies trained with explicit dynamics randomization. To address this loss of performance, we present ERFI: ERFI allows to transfer locomotion controllers trained in simulation to the hardware by randomizing only two parameters: a random episodic actuation offset and random perturbations at each step. First, we show the efficacy of the approach proposed on legged systems, not covered in previous studies, second, we compare it to its predecessor RFI [10], to variations of the same method detailed in the following chapters and to standard domain randomization. Furthermore, we demonstrate with simulation experiments a significant performance improvement over RFI (mass variations' success rate +53%) especially in unseen scenarios, which involves adding a manipulator arm on top of the robot at test time (mass variations' success rate +61%). Finally, we successfully deploy perceptive and blind policies trained in simulation with ERFI on to the physical ANYmal C and Unitree A1 quadrupeds. We show that training of actuator networks (mainly adopted for robots containing SEAs) and performing significant dynamics randomization [11], currently accepted as a standard for sim-to-real transfer can be substituted by a simple ERFI strategy. We test the controller's locomotion performance over flat and uneven terrain and further evaluate its robustness to additional mass and variation in CoM by mounting a Kinova arm on the robot's base.

II. Related Works

Actuators are an essential part of legged systems: they can be hydraulic [12], electric [13] and contain compliant elements [14]. Their dynamics is difficult to model involving nonlinear/nonsmooth dissipation, feedback loops and several internal states which are not directly observable. To accurately approximate SEAs, the authors of [7] trained an actuator network able to output an estimated torque at the joints given as inputs a history of joint position errors and joint velocities recorded from the hardware. Modeling the actuation dynamics with NNs, for robots adopting SEAs, is now considered a standard and other works employed derivations of the same approach [15], [16], [17]. The limitations of learning the actuators' dynamics can be summarized in the need of recording motors' torques (not directly measurable for direct drives), training and testing the NN. In this context it is important to underline that direct drives motors are simpler to model compared to SEAs and adopting an actuator network is not necessary, since classic system identification is enough. However, ERFI also removes the need for system identification by randomizing motors' torques.

Alongside actuator networks, the rise of highly dynamic controllers is driven by domain randomization, particularly dynamics randomization. Initially introduced in [18], [11], the approach consists in the randomization of some of the parameters of the robot's dynamics or of the environment. The additional robustness achieved can then compensate for discrepancies between simulation and the real world. In [7] the domain randomization involves adding noise to the center of mass positions, the masses of links, and joint positions. In [17] the randomization covers: mass, center of mass position, joint position, joint damping, joint friction, joint position tracking gains K_p, torque limits; while regarding the observations' perturbations: delays, joint position noise, angular velocity noise, linear acceleration noise and base orientation noise. Alongside with dynamics randomization, observations were also perturbed during training [17] by adding delay, injecting noise into the joint positions, angular velocity, linear acceleration and base orientation. A significant randomization was also adopted in [15]: gravity, actuation torque scaling, robot link mass scaling, robot link length scaling, random external forces at the base, gravity, actuation torque scaling, link mass scaling, link length scaling, actuation damping gain.

Considering the long list of parameters affected by randomization, the additional robustness it offers requires substantial efforts in system identification: especially in selecting the factors responsible of the reality gap [10] and in defining their randomization range; which if done incorrectly can severely affect the real world performances of the controller, leading to overly conservative policies [19].

An alternative technique – Random Force Injection – was proposed in [10], it aims to transfer policies trained in simulation to real systems without further tuning, with a limited number of parameters and it consists of injecting random forces into the simulator's dynamics. This method was tested on manipulation tasks, where it performed comparably to domain randomization. However, its potential was not evaluated for floating-base systems, especially when the overall stability is compromised by external perturbations.

III. Preliminaries

A. System Model

We model a quadrupedal system as a floating base B. The robot state is represented w.r.t. a reference frame W. We assume the z-axis of W, $\mathbf{e}_z^W$, aligns with the gravity axis. The base position is then expressed as $r_B \in \mathbb{R}^3$, and the orientation, $\mathrm{q}_B \in SO(3)$, is represented by a unit quaternion. The corresponding rotation matrix is expressed as $\mathbf{R}_B \in SO(3)$. The angular positions of the rotational joints in each of the limbs are described by the vector $\mathrm{q}_j \in \mathbb{R}^{n_j}$. For the quadrupeds considered in this work, $n_j = 12$. The linear and angular velocities of the base w.r.t. the global frame are written as $\mathrm{v}_B \in \mathbb{R}^3$ and $\omega_B \in \mathbb{R}^3$ respectively. The generalized coordinates and velocities are thus expressed as q and u where

$$\mathrm{q} = \begin{bmatrix} r_B \\ \mathrm{q}_B \\ \mathrm{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \mathrm{u} = \begin{bmatrix} \mathrm{v}_B \\ \omega_B \\ \dot{\mathrm{q}}_j \end{bmatrix} \in \mathbb{R}^{6+n_j}. \quad (1)$$

B. Impedance Control

In the context of this work, we consider a quadrupedal system is actuated using the joint control torques $\tau_j \in \mathbb{R}^{n_j}$. These torques are computed using the impedance control model given by

$$\tau_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) + K_d(\dot{\mathrm{q}}_j^* - \dot{\mathrm{q}}_j) + \tau_{j_{FF}}, \quad (2)$$

where K_p and K_d refer to the position and velocity tracking gains respectively, q_j^* is the vector representing desired joint positions, $\dot{\mathrm{q}}_j^*$, the desired joint velocities, and $\tau_{j_{FF}}$ refers to the feed-forward joint torques.

For locomotion, we train DRL control policies that modulate the joint actuation torques by generating q_j^*. Additionally, we set $\dot{\mathrm{q}}_j^* = 0$ and $\tau_{j_{FF}} = 0$. [20] presented that such an approach offers more stable training and better performance than a torque controller. Equation 2 can thus be simplified to

$$\tau_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j. \quad (3)$$

C. Rigid Body Dynamics Model

The rigid body dynamics model of a quadrupedal system can be expressed in the form of generalized equations of motion expressed as

$$\mathrm{M}\dot{\mathrm{u}} + \mathrm{h} = \mathbf{S}^T\tau_j + \mathrm{J}^T\lambda, \quad (4)$$

where $\mathrm{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is the mass matrix relative to the joints, $\mathrm{h} \in \mathbb{R}^{6+n_j}$ comprises Coriolis, centrifugal and gravity terms, $\mathbf{S}^T = [\mathbf{0}_{n_j\times 6} \ \ \mathbf{I}_{n_j\times n_j}]^T$, and J is the Jacobian which maps the contact forces $\lambda \in \mathbb{R}^{n_f}$ at $n_f = 4$ feet to generalized forces.

IV. Extended Random Force Injection

[10] investigated the effects of introducing random perturbations to a manipulation system. These *random force injections* aimed to diversify the visited states during training of DRL policies. In this regard, their implementation augmented the generalized equations of motion, similar to Equation 4, by random forces $f_r \sim \mathcal{U}(-f_r^{lim}, f_r^{lim})$ sampled from a uniform distribution $\mathcal{U}$ with limits $-f_r^{lim}$ and f_r^{lim}. These forces are sampled and applied at each time step to perturb the state transition P. In this work, we adapt this approach for quadrupedal systems and write Equation 4 with RFI as

$$\mathrm{M}\dot{\mathrm{u}} + \mathrm{h} = \mathbf{S}^T\tau_j + \mathrm{J}^T\lambda + f_r. \tag{5}$$

It is important to note that [10] used this approach for a fixed-base system. In our preliminary experiments, for a mobile-base quadrupedal system, we observed that perturbing the robot's base with even small forces and torques resulted in convergence to undesired locomotion behavior. For the ANYmal C quadruped, forces and torques on the base sampled from distributions with $f_{r_b}^{lim} > 5\,\mathrm{N}$ and $\tau_{r_b}^{lim} > 3\,\mathrm{N\,m}$ respectively resulted in pronking behavior. Although this behavior was robust to external disturbances on the base, the pronking gait is energy inefficient and unsuitable for transfer to the physical system. Therefore, to better handle uncertainty in the system, we only introduce perturbations to the rotary joints of the quadruped and randomize forces on DoFs that we directly control.

The impedance controller described by Equation 2 is often executed at the actuation level at a higher frequency compared to the locomotion controller which is described by the DRL control policy mapping robot state information to desired joint positions. In this article, we refer to these frequencies as impedance control frequency and locomotion control frequency. We introduce perturbations at the impedance control frequency. We then split Equation 5, describing **RFI**, into the generalized equations of motion given by Equation 4 and an augmented impedance controller given by

$$\tau_j^r = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j + \tau_{r_j}, \tag{6}$$

where τ_{r_j} refers to the random joint torque injections sampled from $\mathcal{U}(-\tau_{r_j}^{lim}, \tau_{r_j}^{lim})$ at each impedance control update step. Note that Equation 6 is only utilized during training. For deployment, we consider the actuation is governed by Equation 3.

In this work, we also investigate the effects of introduction of episodic actuation offsets during training. As opposed to randomizing τ_{r_j} at each impedance control step, we sample joint torque offsets τ_{o_j} from $\mathcal{U}(-\tau_{o_j}^{lim}, \tau_{o_j}^{lim})$, at the beginning of each training episode and apply them at each impedance control step. We refer to this as random actuation offset (**RAO**). This can be represented similarly as our implementation of RFI and is written as

$$\tau_j^o = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j + \tau_{o_j}. \tag{7}$$

This constant offset enables us to emulate a shift in the robot's mass, inertia, impedance gains and contact Jacobian. However, unlike RFI, wherein the dynamics vary at each impedance control step resulting in a more reactive control behavior robust to temporally local perturbations, with RAO, the policy learns an implicit adaptive behavior for temporally global variations in system dynamics.

We also introduce an extended variant of RFI by combining RFI and RAO to learn control policies which can be robust to temporally local and global variations in system dynamics. We refer to this as **ERFI-C**. In this case, we inject both a randomized force sampled at each impedance control step and an episodic actuation offset. The impedance controller with ERFI-C can be then written as

$$\tau_j^c = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j + \tau_{r_j} + \tau_{o_j}. \tag{8}$$

We further explore another strategy with the same motivation as for ERFI-C. In this case, we only utilize RFI with 50% of the parallelized DRL training environments. The remaining environments employ RAO. We refer to this approach as **ERFI-50**. In comparison to ERFI-C, which can be considered as RFI with randomized distribution mean thereby resulting in a possibility of a learning bias for robustness to temporally local perturbations, ERFI-50 promotes unbiased learning of both local and global variations in system dynamics.

V. Why does ERFI work?

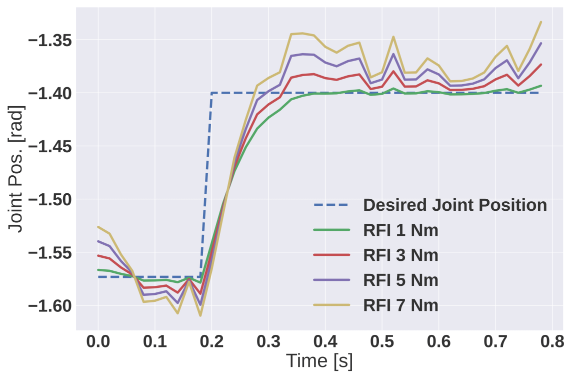
In Figure 2a and Figure 2b, respectively, we show the effects of adding RFI and RAO as a feed-forward term of the PD controller ($K_p = 15$, $K_d = 1$) when commanding a step position change of $0.17\,\mathrm{rad}$ ($\approx$$10\,\mathrm{deg}$) to the hind right knee.

A. How does RFI model delays?

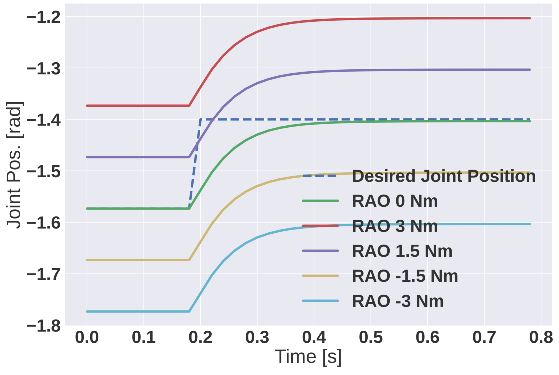
As can bee seen from Figure 2a, the yellow line reaches the desired position faster than the green line, although the green line settles earlier. This implies that RFI adds stochasticity to the rise and settling times, i.e. it either increases or reduces the rise and settling times. The increase or decrease depends on the direction of the perturbation. This allows us to implicitly randomise actuation dynamics, especially parameters that relate to delays, friction and inertia (Section "ERFI robustness to delays" of the accompanying website).

B. How does RAO model mass and kinematic variations?

In Figure 2b, the additional torque shifts the desired position of the joint and implicitly models offsets in the joint position (kinematics variations) or in the payload supported by the robot. Evidences of these effects can be found in Figure 5a and Figure 5d and video 3, 4, and 10 (on the accompanying website), demonstrating the robustness of the



(a)



(b)

Fig. 2: The magnitudes of $\tau_{r_j}^{lim}$ and $\tau_{o_j}^{lim}$ affect the dynamics of the system.

controllers even when the unmodelled payload reaches 42% of the total weight of the robot.

VI. Problem Definition

The complex SEAs present on the ANYmal C quadruped exhibit a highly nonlinear behavior [21]. To address their complex dynamics, networks modeling the actuation became common practice in the community (Section II). Evaluating the effectiveness of ERFI on such a platform thus provides a measure of the robustness of the method and of its generalization abilities.

Conversely, Unitree's A1 adopts quasi-direct drive actuators, which are affected by high levels of delay, signal noise and inaccurate tracking. Given the different technology adopted compared to SEAs and the canonical role that Unitree A1 has played in recent research works [22], [23], we also investigated the effects of ERFI for obtaining locomotion policies for A1.

A. *Perceptive Quadrupedal Locomotion*

The ANYmal C robot is used to track a velocity command $[v_x, v_y, \dot{\gamma}]_{\mathcal{B}}$ on uneven ground using proprioceptive and exteroceptive information. The state is represented as $s := \langle s_r, s_v, s_{j_p}, s_{j_v}, s_a, s_m, s_c \rangle$, where $s \in \mathbb{R}^{259}$, $s_r^{\mathcal{B}} \in \mathbb{R}^3$ is the second row of the rotation matrix, $s_v \in \mathbb{R}^6$ is the base linear and angular velocities, $s_{j_p}^{\mathcal{B}} \in \mathbb{R}^{24}$ is the sparse history of joint position errors and $s_{j_v}^{\mathcal{B}} \in \mathbb{R}^{24}$ is the sparse history of joint velocities, $s_a \in \mathbb{R}^{12}$ is the previous action, $s_m \in \mathbb{R}^{187}$ are measurements from the height-map around the robot's base and $s_c^{\mathcal{B}} \in \mathbb{R}^3$ is the velocity command. The actions $a \in \mathbb{R}^{12}$ are interpreted as the reference joint positions q_j^*. The state s is fed to an MLP network made by three layers respectively of size $[512, 256, 128]$ and the action a is subsequently tracked by the low level PD controller ($K_p = 80., K_d = 2.$).

B. *Blind Quadrupedal Locomotion*

The A1 quadruped robot is required to follow a velocity command $[v_x, v_y, \dot{\gamma}]_{\mathcal{B}}$ on flat ground using proprioceptive information. The state is represented as $s := \langle s_r, s_v, s_{j_p}, s_{j_v}, s_a, s_c \rangle$, where $s \in \mathbb{R}^{192}$, $s_r^{\mathcal{B}} \in \mathbb{R}^3$ is the second row of the rotation matrix, $s_v \in \mathbb{R}^6$ is the base linear and angular velocities, $s_{j_p}^{\mathcal{B}} \in \mathbb{R}^{84}$ is the history of joint position errors and $s_{j_v}^{\mathcal{B}} \in \mathbb{R}^{84}$ is the history of joint velocities, $s_a \in \mathbb{R}^{12}$ is the previous action, velocity and action and $s_c^{\mathcal{B}} \in \mathbb{R}^3$ is the velocity command. The actions $a \in \mathbb{R}^{12}$ are interpreted as the reference joint positions q_j^*. The state s is fed to an MLP network formed by two layers respectively of size $[512, 512]$ and the action a is tracked by the low level PD controller ($K_p = 15., K_d = 1.$). The base linear velocity in s_v is not provided by the onboard state estimator and it was estimated similarly to [24] through an MLP network of size $[128, 128]$.

VII. Experimental Setup

To evaluate our method, we employed ANYmal C as a reference platform and we trained different policies for 10,000 iterations using IsaacGym [25] each adopting one among RFI, RAO, ERFI-50, ERFI-C, and ActNetRand, where ActNetRand represents the present state-of-the-art approach implementing both actuation network and extensive domain randomization, as in [26]. The environment settings used are described in Section VI-A.

The performances of the policies trained with the different methods were assessed by addressing perceptive locomotion over stairs and rocky terrain as relevant case study (Figure 3). Moreover, to obtain more realistic results the experiments were conducted in a different simulator (RaiSim [27]) and we included an actuator network to reproduce the dynamics of SEAs (which was not used during the training of RFI/RAO/EFRI policies). The robot is always deployed at the same position, the velocity command is fixed to $0.5\,\mathrm{m\,s^{-1}}$ and it has $8\,\mathrm{s}$ to go up the stairs; the attempt is considered a failure when the robot falls on the ground or when it is not able to move forward for at least $2.5\,\mathrm{m}$. We generated 50 random stairs as in [15], which are placed just in front of the robot, and walking for $2.5\,\mathrm{m}$ from the spawning point requires tackling at least one step.

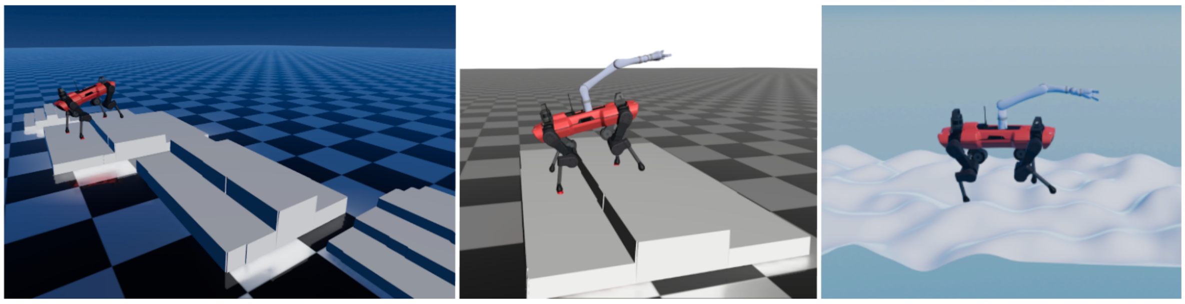


Fig. 3: (Left) Examples of stairs with varying step-height and step-depth used for evaluation. (Center) ANYmal C walking on stairs with an unmodeled Kinova manipulator. (Right) ANYmal C walking on rocky terrain during tests.

To assess the robustness of the policies to unseen conditions we introduced perturbations to the simulation environment: the application of external forces $[0\ ;\ 150]\,\mathrm{N}$ to the base (fixed value during training: 0.) for a duration of $3\,\mathrm{s}$, the application time of an external force of $50\,\mathrm{N}$ varies between $[0\ ;\ 3]\,\mathrm{s}$ (fixed value during training: 0.), the application of external torque $[0\ ;\ 75]\,\mathrm{N\,m}$ to the base (fixed value during training: 0.) for a duration of $1\,\mathrm{s}$, the friction coefficient between ground and feet in the range $[0.2, 0.8]$ (fixed value during training: 0.5), the gravitational acceleration was modified between $[-18\ ;\ -2]\,\mathrm{m/s^2}$ (fixed value during training: $-9.81\,\mathrm{m/s^2}$), the position of the knees' motors was shifted by $[-0.15\ ;\ -0.15]\,\mathrm{m}$ (fixed value during training: 0.) and the mass of the base changed between $[22\ ;\ 65]\,\mathrm{kg}$ (fixed value during training: $27\,\mathrm{kg}$). Throughout the evaluation we alter only one parameter at the time and for each of them we run 50 experiments with different terrains. Furthermore, we replicate the set of experiments above with a robotic arm mounted on top of ANYmal C, this introduces significant variations in the mass matrix M which the robot never explicitly experienced during training. Following the thorough validation presented in simulation, the best performing controller (resulted to be trained with ERFI-50) was deployed on the hardware, tested on rough and uneven terrain, both in the laboratory and outdoor environments to validate the feasibility of the method.

In addition, we demonstrate the effectiveness of ERFI-50 with hardware experiments also on Unitree A1, this time performing blind locomotion in challenging conditions. We present results of extensive hardware evaluation in Figure 4 and on our accompanying website `https://sites.google.com/view/erfi-video`.

VIII. Results

We compared ERFI-50, ERFI-C, and RAO against two baselines, RFI and ActNetRand (policy trained using dynamics randomization and actuator network). The metric adopted to assess their performances is the success rate described in Section VII. The first row of Figure 5 (Figures 5a to 5c) shows the robustness of the different approaches to changes in the base mass, in the application of external forces, or to different friction coefficients between feet and ground; in this first batch of experiments, the arm was not included. From these plots, it is evident that the standard RFI is the least performing method, while still providing decent robustness especially close to the training domain. Conversely, RAO and ERFI-50 are the better-performing ones (providing on average 53% better success rate than RFI on mass variations, Figure 5a), they are often very close and sometimes better than ActNetRand, which is currently the standard approach to deploy controllers on the hardware. Regarding ERFI-C, it does better than standard RFI (on average 41% better success rate on mass variations, Figure 5a), but still not as well as ERFI-50 and RAO (on average 12% worse success rate on mass variations, Figure 5a). The analysis presented above was repeated after mounting a fixed Kinova manipulator arm on top of the robot; the same policies, perturbation, and set of stairs were considered during the experiments. The objective of this last study is to test the robustness of the controller in real-world scenarios never encountered during training. The resulting performances are depicted in Figures 5d to 5f, where we observe the gap between RFI and RAO/ERFI-50 enlarging with a performance loss for RFI -even in the training domain- of roughly 50%, while RAO and ERFI-50 achieved roughly 62% higher success rate than RFI on this task, Figure 5d.

Furthermore, we investigated the effects of $\tau_{o_j}^{lim}$ and $\tau_{r_j}^{lim}$ on the overall performances of ERFI-50, we show the outcomes of different limits on the success rate when the base mass is increased, Figure 5i. The curves in Figure 5i show that high $\tau_{o_j}^{lim}$ provides greater robustness in combination with high $\tau_{r_j}^{lim}$ (ERFI50-40+40, red line), when compared to $\tau_{o_j}^{lim} = 20[\mathrm{N\,m}]$ and $\tau_{r_j}^{lim} = 20[\mathrm{N\,m}]$. However, for values of $\tau_{o_j}^{lim} = 30[\mathrm{N\,m}]$ and $\tau_{r_j}^{lim} = 30[\mathrm{N\,m}]$ the performance improves in one portion of the domain and it remains comparable to $\tau_{o_j}^{lim} = 20[\mathrm{N\,m}]$ and $\tau_{r_j}^{lim} = 20[\mathrm{N\,m}]$ in the remaining one.

To provide a comprehensive assessment, we present the performance on the rocky terrain in Figure 3, which complements the results obtained in the staircase environment. Notably, the survival rates in high perturbation regimes are partially reduced due to the absence of rocky terrain in the training environment. Nonetheless, the relative performances of the methods remain comparable to those observed in the staircase evaluation, Figures 5g and 5h.

The robustness to further perturbations –as varying the

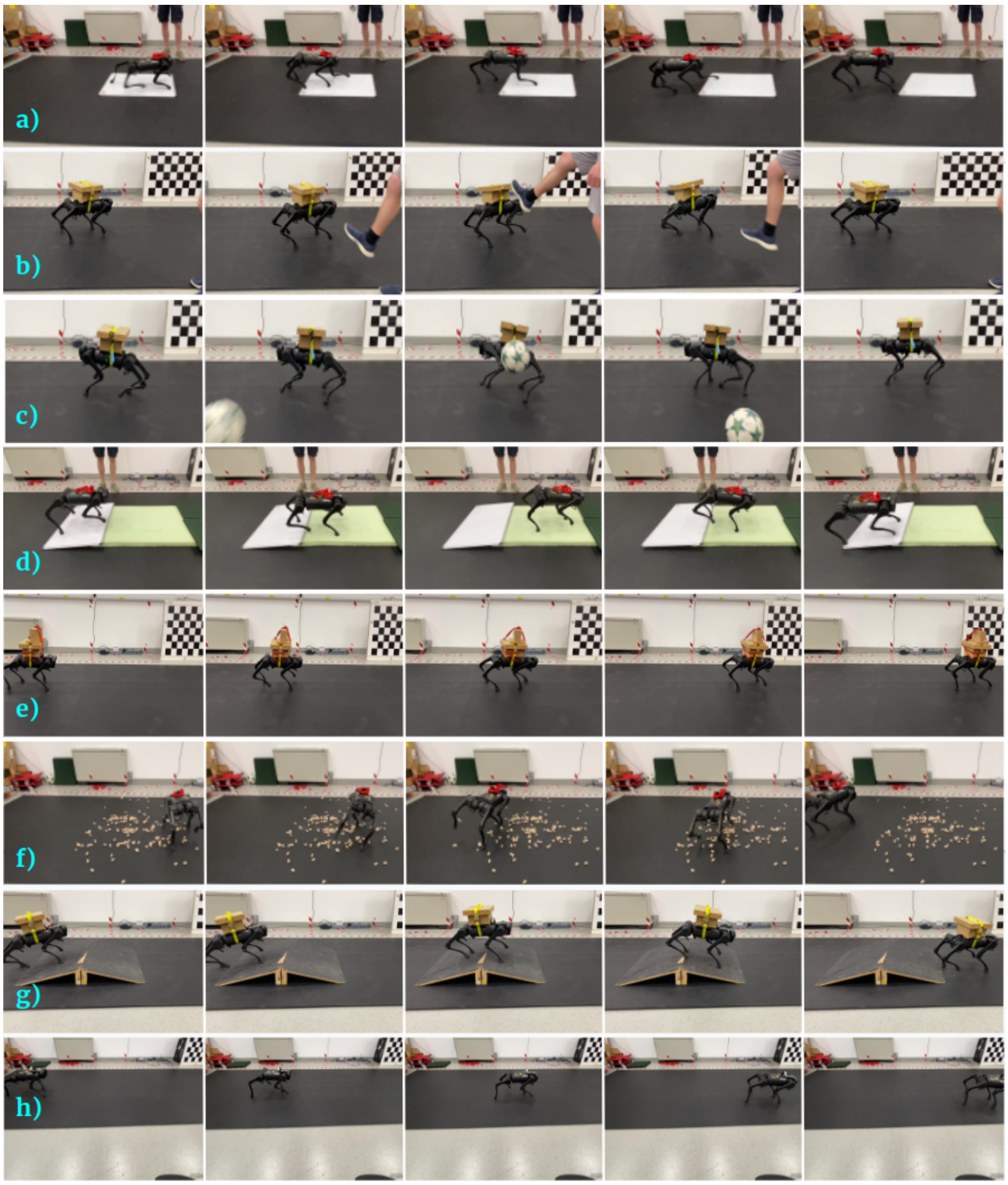


Fig. 4: This figure shows some of the experiments on Unitree A1 adopting ERFI, also part of our accompanying website `https://sites.google.com/view/erfi-video`. a) Walking on wet terrain and recovering from slipping, b) resisting to external forces, c) withstanding impulsive forces, d) walking on soft terrain, e) walking with an unknown 5 Kg payload, f) walking on wooden cylinders, g) traversing a ramp, and h) adapting to a K_p^{RHKFE} equal to a third of the original value.

duration of the external force, applying an external torque, varying the gravitational acceleration and shifting the knee motors' positions– were investigated and the results are consistent with what is already shown in Figure 5.

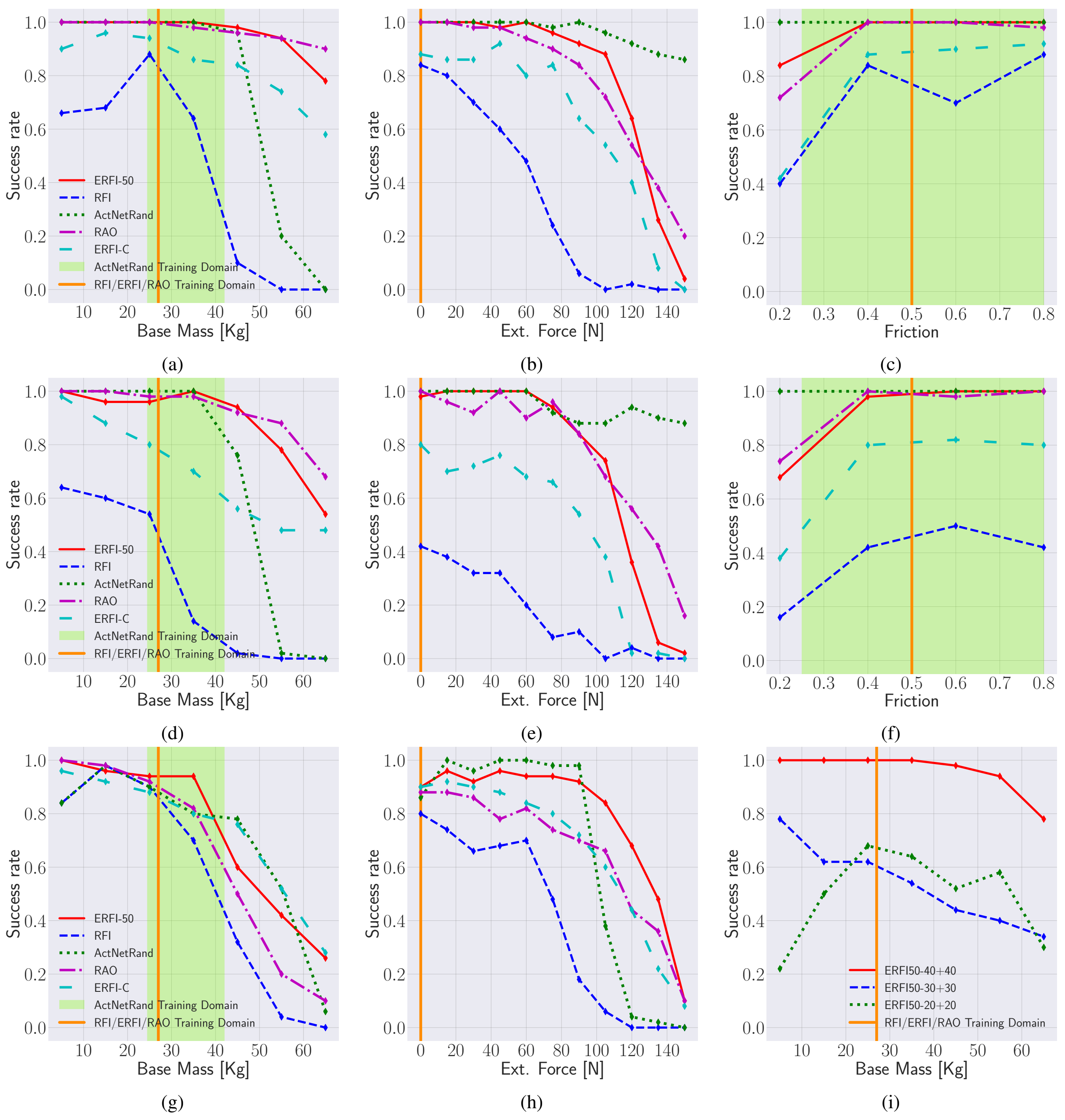


Fig. 5: Figures 5a to 5c show how RFI, ERFI-50, ERFI-C, RAO and ActNetRand resist to variations of the base mass, to external forces, or to different frictions. In Figures 5d to 5f the same experiments are replicated with a Kinova manipular on top of the robot. In Figures 5g and 5h we investigated the effects of the perturbations also on the rocky terrain in Figure 3. While, in Figure 5i we studied how different $\tau_{o_j}^{lim}$ and $\tau_{r_j}^{lim}$ affected the robustness of the controller.

IX. Conclusion

In this work we showed that transferring policies trained in simulation to real systems is possible without defining the domain randomization's parameters and their ranges, without further system identification to measure the noise to inject in the observations and without recording any of hardware data to train an additional NN to model the motors' dynamics. Instead we proposed to use a blend of episodic and continuously changing random force perturbations (ERFI-50), which has competitive performance compared to state of the art extensive domain randomization (ActNetRand) and which only requires tuning two parameters ($\tau_{o_j}^{lim}$ and $\tau_{r_j}^{lim}$); hence reducing the sim-to-real transfer efforts compared to previous approaches by a large margin. We further demonstrated the validity of our approach by transferring the controllers to the hardware and showing stable locomotion with Unitree A1, uneven terrain locomotion and mounting an unmodelled manipulator on top of the robot with ANYmal C.

References

[1] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *CoRR*, vol. abs/1901.07517, 2019. [Online]. Available: http://arxiv.org/abs/1901.07517

[2] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, p. eabb2174, 2020.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.

[4] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com

[5] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[6] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in Artificial Life*, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[7] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *CoRR*, vol. abs/1901.08652, 2019. [Online]. Available: http://arxiv.org/abs/1901.08652

[8] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *CoRR*, vol. abs/1804.10332, 2018. [Online]. Available: http://arxiv.org/abs/1804.10332

[9] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *CoRR*, vol. abs/2010.11251, 2020. [Online]. Available: https://arxiv.org/abs/2010.11251

[10] E. Valassakis, Z. Ding, and E. Johns, "Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics," *CoRR*, vol. abs/2008.06686, 2020. [Online]. Available: https://arxiv.org/abs/2008.06686

[11] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2018. [Online]. Available: https://doi.org/10.1109%2Ficra.2018.8460528

[12] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of hyq – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011. [Online]. Available: https://doi.org/10.1177/0959651811402275

[13] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3307–3312.

[14] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[15] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-Aware Legged Locomotion using Reinforcement Learning and Optimal Control," *arXiv e-prints*, p. arXiv:2012.03094, Dec. 2020.

[16] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *CoRR*, vol. abs/2201.08117, 2022. [Online]. Available: https://arxiv.org/abs/2201.08117

[17] S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, M. Wulfmeier, M. Neunert, B. Moran, N. Siegel, A. Huber, F. Romano, N. Batchelor, F. Casarini, J. Merel, R. Hadsell, and N. Heess, "Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors," 2022. [Online]. Available: https://arxiv.org/abs/2203.17138

[18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *CoRR*, vol. abs/1703.06907, 2017. [Online]. Available: http://arxiv.org/abs/1703.06907

[19] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited:a case study for quadrupedal locomotion," 2020. [Online]. Available: https://arxiv.org/abs/2011.02404

[20] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[21] C. Gehring, S. Coros, M. Hutter, C. Dario Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[22] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, "Learning free gait transition for quadruped robots via phase-guided controller," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1230–1237, 2022.

[23] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 773–783. [Online]. Available: https://proceedings.mlr.press/v164/yang22d.html

[24] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021. [Online]. Available: https://arxiv.org/abs/2108.10470

[26] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *CoRR*, vol. abs/2109.11978, 2021. [Online]. Available: https://arxiv.org/abs/2109.11978

[27] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization
Publication Status	Submitted for Publication
Publication Details	Campanaro L., Gangapurwala S., Merkt W., Havoutis I., Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization, 2023

Student Confirmation

Student Name:	Luigi Campanaro		
Contribution to the Paper	Conceptualized the idea, implemented it, trained it, evaluated the results, deployed on the robots, contributed to final manuscript.		
Signature Luigi Campanaro		Date	11/04/2023

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Dr. Ioannis Havoutis		
Supervisor comments The contributions are as above		
Signature Ioannis Havoutis	Date	19/04/2023

This completed form should be included in the thesis, at the end of the relevant chapter.

4.3 Further insights

This section delves into several unexplored aspects that were not covered in the manuscript, including an analysis of the performance differences between ERFI-C and ERFI-50, an examination of the set of experiments presented and their underlying motivations, insights into the emergence of adaptive behaviors, and a qualitative evaluation of the impact of different levels of randomization on the policy. We also explore the effects of introducing artificial delay during simulation and how the policy responds to it. Furthermore, we update the related work section to reflect recent developments since the submission of the paper.

4.3.1 ERFI-C vs ERFI-50

The comparison of Extended Random Force Injection 50% (ERFI-50) and Extended Random Force Injection Cumulative (ERFI-C) reveals that the latter displays a more erratic behavior, as evidenced by sudden variations in the velocity $\dot{q}$ of the front left knee in Figure 4.1. In contrast, ERFI-50 exhibits a smoother velocity, and a clear pattern in the relationship between high values of gradients in the saliency map and high torques, as illustrated in the same figure. Although both models are trained with the same $\tau^{lim}o_j$ and $\tau^{lim}r_j$, this pattern is less apparent for ERFI-C.

High torques can be associated with foot-ground contact, and correctly detecting it is crucial for locomotion. The discrepancy between ERFI-C and ERFI-50, and the clear pattern in ERFI-50 suggests that detecting the contact is the reason why the latter outperforms ERFI-C.

In conclusion, the discontinuous behavior of ERFI-C may affect the network's ability to interpret input observations, ultimately leading to lower performance.

4.3.2 Experiments motivation and analysis

Robustness on slippery terrains is a crucial skill for legged robots, and in our simulation setup the ground friction was constant during the whole training. To support the generality of scenarios that ERFI-50 can handle, we tested the robot's ability to maintain stability on slippery surfaces it had not encountered before.

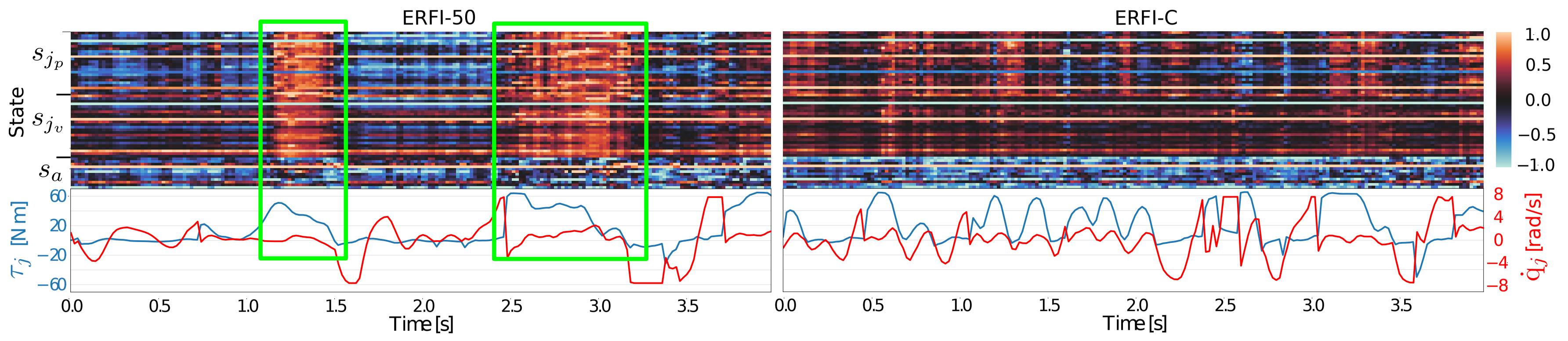


Figure 4.1: Looking at the saliency map of ERFI-50 (knee motor) we can recognize the direct correlation between high torques and high gradients (framed in green). Conversely, when inspecting ERFI-C, it shows a higher joint velocity, a less evident correspondence between torque and gradient and this explains the poorer performance of ERFI-C compared with ERFI-50, highlighting that splitting training environments in those randomizing for system model error (RAO) and those randomizing for tracking error (RFI) is vital to learning locomotion policies.

Walking on soft materials is also a standard test for non-learning-based controllers, as it makes detecting contact during the gait cycle more challenging. We therefore tested the robot's behavior on foam and observed its successful performance.

Maintaining balance in the presence of external forces is another key indicator of a locomotion controller's robustness. We tested the robot's ability to recover from impulsive forces, such as being hit by a football, and pushes of varying duration. In both cases, the robot demonstrated its ability to quickly regain balance even after strong perturbations, despite never having experienced these disturbances during training.

The ability to resist variations in the position of the CoM, weight, and inertia of the base is critical for successful locomotion control. To assess this, we added a 5 kg payload to the robot, which is roughly 42% of its weight before the additional load. Unlike other approaches [71], which use higher gains ($K_p = 100.0\,\mathrm{N\,m\,rad^{-1}}$), we adopted lower ones ($K_p = 20.0\,\mathrm{N\,m\,rad^{-1}}$) to keep the ground striking soft and the motion smooth.

Our controller for A1 is blind, so we tested its robustness in the presence of tall obstacles by conducting experiments on a ramp with a height of approximately 35% of the robot's legs.

Finally, we tested the robot's ability to traverse over wooden cylinders with a diameter of approximately 1.5 cm. This experiment approximates a rough, slippery, and non-static terrain, similar to pebbles.

In conclusion, the locomotion controller based on ERFI has shown remarkable robustness and adaptability in handling various challenging terrains and situations. These results indicate that the proposed method has great potential for real-world applications, where legged robots must navigate unpredictable and varied environments. The video results of the experiments are available through the link provided in the paper.

4.3.3 Adaptation

Adaptation to unseen scenarios is a key requirement for legged robotics controllers that are intended for deployment in the real world. To assess the adaptability of our controller trained with ERFI-50, we reduced the K_p of the right hind knee to 33% of its original value. Despite this modification, our policy tracked the desired base velocity commands, albeit with a limp in the rear leg.

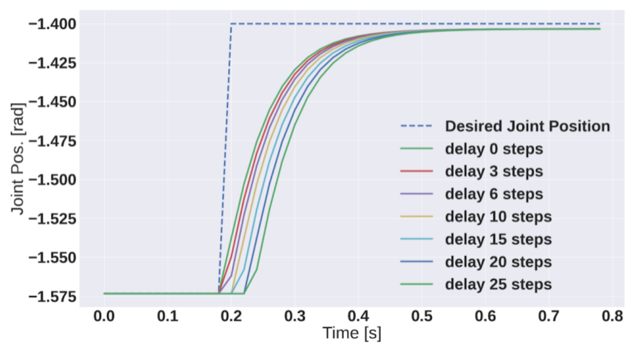
4.3.4 Different degrees of randomization

We compared the results obtained with ERFI-50 with domain randomization by training additional policies for A1 using this method. To have a reference, we initially adopted the randomization distributions and joint impedance settings described in [71], but found that the resulting locomotion was very conservative and prone to failures when the Adaptation Module was not used. To achieve more dynamic and stable locomotion, we instead used a "softer" randomization approach, which reduced the range of the parameters under randomization, in agreement with [70].

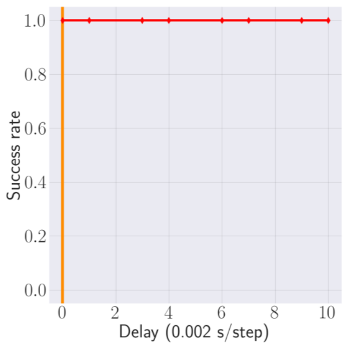
However, we observed that the absence of any randomization (not even ERFI-50) led to failures, which we attribute to delays in the actuation that were not fully modeled; we discuss delays in the following section.

4.3.5 Artificial delay injection

In simulation, actuation dynamics is ideal, and delays due to motor inertia or communication latency do not exist. However, this results in the over-fitting of the policy to the behavior of a perfect impedance controller, which is not desirable for real-world deployment. By using ERFI-50, the policy can exhibit correct locomotion and ensure robustness in unexpected scenarios. This suggests that ERFI-50 also accounts for imperfections in actuation dynamics, including delays. To demonstrate this, we injected artificial delays in the actuation, as in Figure 4.2a, and observed the policy's robustness in tracking the commanded velocity without falling (100% success rate), as shown in Figure 4.2b. We limited the delay injection to 10 steps since the delay measured on the robot is approximately 10 [ms].



(a)



(b)

Figure 4.2: In Figure 4.2a we show the effects of delays on the PD controller tracking ($K_p = 15$, $K_d = 1$), when commanding a step of +0.17 [rad] (10 [deg]) to the hind right knee. While in Figure 4.2b we injected delays in the actuation dynamics of each motor during forward locomotion at 0.5 m/s and the policy demonstrated a 100% success rate. We didn't consider injecting more than 10 steps of delay because the delay measured on the robot is 10 [ms]. In simulation the PD control is executed at 500 [Hz]: 10 [steps] * 0.002 [s/step] = 0.02 [s], and this is already double the delay measured on the robot.

4.4 Limitations

The parameters used by Random Actuation Offset (RAO) and ERFI-50 to increase the robustness of the controllers ($\tau_{o_j}^{lim}$ and $\tau_{r_j}^{lim}$) can only be empirically tuned, requiring training different policies and testing their abilities to variations of the simulation environment. And differently from domain randomization, explicitly defining the desired robustness range is not possible apriori but emerges from training and needs to be evaluated after obtaining a policy. As part of future work, we plan to investigate the correlation between robustness ranges and $\tau_{o_j}^{lim}/\tau_{r_j}^{lim}$ to eliminate the need of training and testing several policies, and adopting different limits for different joints, even if this will increase the number of parameters to tune which is something that our work here aims to avoid. Moreover, for tasks requiring specific control behaviors – for example handling significant external forces – ERFI-50 may not be sufficient, since the policy needs to experience during training such situations to learn adequate skills.

4.5 Updated related work

The direction taken recently by the legged robotics community points towards vision-based locomotion, and in this context the sim-to-real transfer is a non-trivial problem, especially when raw depth images are directly consumed by the policy during locomotion. The authors of [91] aim to develop a walking policy that can efficiently map proprioception and raw depth inputs to target joint angles at a frequency of 50Hz. However, training the system directly using Deep-RL becomes intractable due to the rendering of depth that slows down the simulation. To tackle this, a two-phase training scheme is proposed. In the first phase, low-resolution scandots are used as a proxy for depth images, and in the second phase, depth and proprioception are used as inputs to an RNN to predict target joint angles directly. This second phase is supervised using actions from the phase 1 policy. The proposed pipeline enables the whole system to be trained on a single GPU in a few days, and the deployment policy can directly predict joint angles from depth and proprioception without constructing externally the elevation maps. The second phase of the training presents two variations of the architectures: the monolithic and the RMA-based. Both are trained using PPO with backpropagation through time truncated at 24 timesteps. The reward functions proposed in previous work are extended to penalize energy consumption.

To ensure a successful simulation-to-real transfer, the authors employed a rigorous randomization during phase 1. The randomization included parameters such as the height map update frequency and latency, added mass, change in position of CoM, random pushes, friction coefficient, height of fractal terrain, motor strength, and PD controller stiffness and damping. Quantitative information about the randomization was provided, such as height map update frequency ranging from 80ms to 120ms, and motor strength ranging from 90% to 110%.

Additionally, the authors utilized a curriculum-based environment, constructing a large elevation map with 100 sub-terrains arranged in a 20x10 grid. The terrains were arranged in increasing difficulty, with each row having the same type of terrain while different rows having different terrains. Each terrain had a length and width of 8m, and high and medium fractals were added to the flat and other terrains, respectively.

4.6 Review of the Contributions

The main contribution of [91] is a walking policy that maps proprioception and raw depth inputs to joint angles. However, the paper's impact would have been reduced if the method had only been presented in simulation. The sim-to-real transfer is essential, and the paper provides little information beyond the list of randomized quantities and associated intervals. As a result, reproducibility and extension of the work depends on the authors' experience. In contrast, in our work [2] we minimized the parameters needed for successful sim-to-real transfer and provided extensive ablation studies to support the effect of their method on robustness to external perturbations. This allows less experienced practitioners to approach the sim-to-real problem, and experienced ones can take advantage of the limited number of parameters and not having to justify their selection with ablation studies. Indeed, several of the parameters randomized in [91] could have been easily addressed by ERFI-50, as demonstrated in [2], which would reduce overall complexity.

5 Mitigating noise in observations using a single parameter

Contents

5.1 Overview of contribution

In recent years, Deep-RL has emerged as a promising approach for controlling legged robots, enabling them to perform highly dynamic and sophisticated locomotion tasks [60, 61, 64, 65]. However, training Deep-RL policies for high-dimensional problems –like locomotion– in the real world can be challenging due to sample complexity, the risk of breaking machines during training, and the difficulty of manually resetting the robots after termination is reached. To address these challenges, physics simulators have become increasingly popular for training Deep-RL control policies. However, the sim-to-real gap is a significant challenge that must be addressed.

To address the reality gap, various strategies have been proposed, including accurately identifying properties such as CoM and inertia of robot links, impedance gains, system communication delays, friction, and actuation dynamics. Relevant distributions suitable for domain randomization must also be selected, and sensory noise needs to be modeled and introduced into the simulation during training. Recently, we proposed a new approach called ERFI to handle system and actuation uncertainty, which demonstrates state-of-the-art sim-to-real performance by only randomizing two parameters. However, ERFI does not explicitly encompass modelling noise in observations.

To improve the robustness of Deep-RL-based locomotion controllers to observation noise, we are proposing in the following paper a new method called Roll-Drop. This method introduces dropout during rollout, improving the robustness of Deep-RL policies to observation noise by only tuning a single parameter. In comparison with other techniques, Roll-Drop demonstrates an 80% success rate when up to 25% noise is injected in the observations. The policies were trained in simulation on flat ground and deployed on a Unitree A1 quadruped robot, which was required to follow a velocity command using proprioceptive information.

In summary, the sim-to-real gap is a significant challenge that must be addressed in training Deep-RL control policies. Current research has focused on tackling the mismatch between simulated and real sensors by directly modeling the noise from real systems and injecting it into the network state during training. Roll-Drop. instead, addresses the noise injection by only requiring and tuning one parameter, without further system identification.

5.2 Integrated manuscript

Roll-Drop: accounting for observation noise with a single parameter

Luigi Campanaro LUIGI@ROBOTS.OX.AC.UK
Daniele De Martini DANIELE@ROBOTS.OX.AC.UK
Siddhant Gangapurwala SIDDHANT@ROBOTS.OX.AC.UK
Wolfgang Merkt WOLFGANG@ROBOTS.OX.AC.UK
Ioannis Havoutis IOANNIS@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

This paper proposes a simple strategy for sim-to-real in Deep-Reinforcement Learning (DRL) – called Roll-Drop – that uses dropout during simulation to account for observation noise during deployment without explicitly modelling its distribution for each state. DRL is a promising approach to control robots for highly dynamic and feedback-based manoeuvres, and accurate simulators are crucial to providing cheap and abundant data to learn the desired behaviour. Nevertheless, the simulated data are noiseless and generally show a distributional shift that challenges the deployment on real machines where sensor readings are affected by noise. The standard solution is modelling the latter and injecting it during training; while this requires a thorough system identification, Roll-Drop enhances the robustness to sensor noise by tuning only a single parameter. We demonstrate an 80% success rate when up to 25% noise is injected in the observations, with twice higher robustness than the baselines. We deploy the controller trained in simulation on a Unitree A1 platform and assess this improved robustness on the physical system. Additional resources at: https://sites.google.com/oxfordrobotics.institute/roll-drop

Keywords: Sim-to-real; Legged Locomotion; Reinforcement Learning.

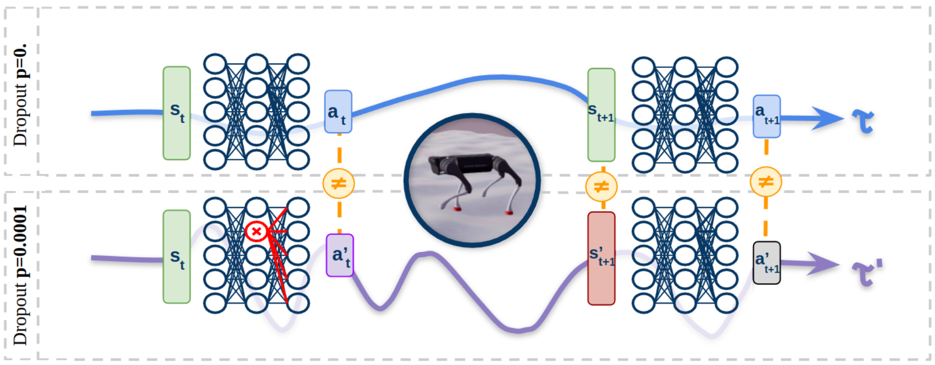


Figure 1: We present two policies, $\hat{\pi}$ trained with dropout during rollout (bottom) and π without it (top). As the time-step t precedes the occurrence of the first dropout and the training adopts the same random seed, both policies visited the same states and actions. After the first dropout is triggered, the policies will follow different trajectories, $\mathcal{T}_\pi$ and $\mathcal{T}'_{\hat{\pi}}$: $[a_t, .., a_T]_\pi \neq [a'_t, .., a'_T]_{\hat{\pi}}$, $[r_t, .., r_T]_\pi \neq [r'_t, .., r'_T]_{\hat{\pi}}$, and $[s_{t+1}, .., s_T]_\pi \neq [s'_{t+1}, .., s'_T]_{\hat{\pi}}$. This alters the visited states and prevents high sensitivity of policies to noiseless observations.

1. Introduction

Deep-Reinforcement Learning (DRL) gained traction in the legged robotics community as a promising approach to the control problem, enabling highly dynamic and sophisticated locomotion capabilities (Lee et al., 2019; Yang et al., 2020; Kumar et al., 2021). The sample complexity associated with high-dimensional problems such as locomotion, the risk of breaking the machines at the beginning of the training and the difficulty of resetting the robots make the use of physics simulators (Hwangbo et al., 2018; Makoviychuk et al., 2021) appealing for training DRL control policies. However, this convenience often requires addressing the *reality gap* between the simulated training and physical deployment domains.

Strategies to address such a reality gap include accurately identifying properties such as Center of Mass (CoM), mass and inertia of robot links, impedance gains, system communication delays, friction, and actuation dynamics (Hwangbo et al., 2019; Lee et al., 2020). In addition, relevant distributions suitable for domain randomisation need to be selected (Tan et al., 2018; Lee et al., 2019); as part of such randomisation of the environment, sensory noise needs modelling and it is introduced in simulation during training (Jakobi et al., 1995; Hwangbo et al., 2019).

We recently proposed Extended Random Force Injection (ERFI) (Campanaro et al., 2022) to handle system and actuation uncertainty as an alternative to a complete system and distribution identification for dynamics randomisation. We demonstrate state-of-the-art sim-to-real performances by only randomising (and tuning) two parameters. However, the robustness showed by ERFI in challenging conditions did not explicitly encompass modelling noise in observations.

In this work, we propose Roll-Drop, a method that improves the robustness of DRL-based locomotion controllers to observation noise by introducing dropout during rollout. In continuation with ERFI's simplicity, Roll-Drop only needs tuning a single parameter.

In the following sections, we present the method, analyse the results, and compare the robustness of alternatives to the injection of noise in the state space of the policy. Roll-Drop demonstrates an 80% success rate when up to 25% noise is injected in the observations, whereas in the same conditions other techniques experienced less than 40% success rate. The policies were trained in simulation on flat ground and deployed on a Unitree A1.

2. Related Work

Modern robots are equipped with diverse sensors to ensure acceptable levels of autonomy by estimating either the robot's state or the surrounding environment. Such sensors include Inertial Measurement Units (IMUs), joint encoders (Hubicki et al., 2016), torque and contact sensors (Hutter et al., 2016), RGBD cameras (Rudin et al., 2022; Gangapurwala et al., 2022; Miki et al., 2022), and lidar scanners (Mattamala et al., 2022). DRL approaches applied to locomotion controllers conveniently train policies that can take advantage of such rich sensory information.

Simulators are paramount here to reducing costs and training time while ensuring safety during the delicate training procedure. Moreover, simulators provide the repeatability necessary to investigate eventual undesired behaviour. However, in contrast to real sensors, simulators provide perfect and noiseless information far from what the policy would experience when deployed on a real robot, causing an additional sim-to-real gap to be addressed.

Research has focused on tackling the mismatch between simulated and real sensors by directly modelling the noise from real systems and injecting it into the network state during training. Hwangbo et al. (2019) sample the joint velocity noise from uniform distributions, similarly to Lee et al. (2020) for linear and angular velocity noise; these were then added to the simulator's observations to improve robustness. Bohez et al. (2022) use instead normal distributions to model observation noise for joint positions, angular velocity, linear acceleration, and base orientation, while Siekmann et al. (2021); Yu et al. (2022) also include the joint-encoder offsets, which were sampled from a uniform distribution. Gangapurwala et al. (2022); Miki et al. (2022), instead, focus on exteroceptive sensors and inject noise into the height maps to foster the controller robustness to artefacts and sudden spikes.

Additionally, determining the noise characteristics is a delicate and costly process. Often little detail on the process is provided, and ablation studies supporting the necessity of such randomisation are absent. Roll-Drop addresses this lack of information as one parameter is enough to characterise the implementation.

3. Problem Definition: Blind Quadrupedal Locomotion

We model a quadrupedal system as a floating base B described by the reference frame $\mathcal{B}$, represented w.r.t. a world reference frame $\mathcal{W}$, whose z-axis aligns with the gravity axis. $\mathcal{B}$'s x-axis $x_{\mathcal{B}}$ points in the forward direction of motion of B, the y-axis $y_{\mathcal{B}}$ to the left and the z-axis $z_{\mathcal{B}}$ upwards. The base position is then expressed as $r_B \in \mathbb{R}^3$, and the orientation, $\mathrm{q}_B \in SO(3)$, is represented by a unit quaternion, whose corresponding rotation matrix is denoted as $\mathbf{R}_B \in SO(3)$.

In this work, we will employ a Unitree A1 quadruped, whose four legs are composed of three joints each. We will refer to the front-right leg as `FR`, to the front-left leg as `FL`, to the hind-right as `HR`, and to the hind-left leg as `HL`. Each leg has a hip adduction/abduction `HAA`, hip flexion/extension `HFE`, and knee flexion/extension `KFE` joint. For example, we refer to the front-right hip flexion/extension as `FR_HFE`. The vector $\mathrm{q}_j \in \mathbb{R}^{n_j}$ – in our system, $n_j = 12$ – contains the angular positions of the rotational joints of all limbs, which are actuated through an impedance control, simplified as described by Peng and van de Panne (2017):

$$\Gamma_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j. \tag{1}$$

where Γ_j are the actuation torques on the joints, q_j^* is the vector representing desired joint positions, and K_p and K_d refer to the position and velocity tracking gains, respectively, which in our system are $K_p = 15.0\,\mathrm{N\,m\,rad^{-1}}$ and $K_d = 1\,\mathrm{N\,m\,s\,rad^{-1}}$.

3.1. Reinforcement Learning

The Reinforcement Learning (RL) problem is modelled as an Markov Decision Process (MDP) including a state space $\mathbf{S}$, an action space $\mathbf{A}$, an initial state distribution $p_1(s_1)$, a transition dynamics $p(s_{t+1}|s_t, a_t)$ compliant with the Markov property $p(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$ for any trajectory $\mathcal{T}_{1:t} = [(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots, (s_t, a_t, r_t)]$, where $r_i = R(s_i, a_i)$ is the reward obtained from a reward function $R : \mathbf{S} \times \mathbf{A} \to \mathbb{R}$. In all the previous, $s_i \in \mathbf{S}$ and $a_i \in \mathbf{A}$.

A policy – in our case, the controller – selects actions in the MDP given a specific state. The policy – denoted by π_θ, where $\theta \in \mathbb{R}^n$ is a vector of n parameters – is stochastic, and $\pi_\theta(a_t|s_t)$ is the conditional probability density of a_t associated with the policy. The agent uses its policy to interact with the MDP, realising the trajectory of states, actions, and rewards $\mathcal{T}_{1:T} = (s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)$.

Table 1: (a) PPO hyper-parameters used for training the Unitree A1 policy; (b) the rewards adopted during training, and their weights.

(*a*)

Hyperparameter	Value
Control dt	0.02 [s]
Sim dt	0.002 [s]
Batch size	25600
Mini-batch size	6400
Number of epochs	8
Clip range	0.2
Entropy coefficient	0.
Discount factor	0.996
GAE discount factor	0.95
Learning rate	$1e^{-4}$

(*b*)

	Definition	Weight
Base orientation	$k_c \cdot \|\|\mathbf{R}_B^z - [0,0,1]\|\|^2$	-30
Base linear velocity	$\phi(v^*_{b_{x,y}}, v_{b_{x,y}}, 5)$	15
Base angular velocity	$\phi(v^*_{b_z}, \omega_{b_z}, 5)$	15
Action smoothness	$k_c \cdot \|\|\mathrm{q}^*_{j_t} - \mathrm{q}^*_{j_{t-1}}\|\|^2$	-7
Feet clearance	$k_c \cdot \sum_{n=0}^{n<4}(0.1 - f_{z_n})^2$	-400
Feet sleep	$k_c \cdot \|\|\dot{f}_{x,y}\|\|^2$	-8
Joint position	$k_c \cdot \|\|q_j - \mathrm{q}_j^N\|\|^2$	-4
Joint velocity	$k_c \cdot \|\|\dot{q}_j\|\|^2$	-0.01
Joint torque	$k_c \cdot \|\|\tau_j\|\|^2$	-0.4
Feet swing duration	$\sum_{n=0}^{3}(\mathbf{t}_{air,n} - 0.5)$	8
Pronking gait	$k_c \cdot \sum_{n=0}^{3}(f_{c_n} \cdot 1)$	-35

The policy π is trained through an optimisation problem to maximise the cumulative discounted reward it obtained from the starting state, expressed as $\pi^* = \text{argmax}\, \mathbb{E}[r_1^\gamma|\pi]$, where r_t^γ is the total discounted reward from time-step t onward, as $r_t^\gamma = \sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k)$, where $0 < \gamma < 1$.

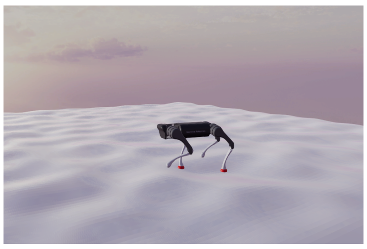
3.2. Implementation

The quadruped robot is required to follow a velocity command $s_c = [v_x, v_y, \dot{\psi}]_{\mathcal{B}}$ on flat ground using proprioceptive information. Here v_x and v_y are the linear velocities along $x_{\mathcal{B}}$ and $y_{\mathcal{B}}$ respectively, while $\dot{\psi}$ is the angular velocity around $z_{\mathcal{B}}$.

The state is represented as $s := \langle s_r, s_v, s_{j_p}, s_{j_v}, s_a, s_f, s_c \rangle \in \mathbb{R}^{196}$, where $s_r^{\mathcal{B}} \in \mathbb{R}^3$ is the last row of the rotation matrix $\mathbf{R}_{\mathcal{B}}$, $s_v \in \mathbb{R}^6$ is the base linear and angular velocities, $s_{j_p}^{\mathcal{B}} \in \mathbb{R}^{84}$ is the history of joint position errors and $s_{j_v}^{\mathcal{B}} \in \mathbb{R}^{84}$ is the history of joint velocities, $s_a \in \mathbb{R}^{12}$ is the previous action, $s_f \in \mathbb{R}^4$ is the contact state of the feet, and $s_c^{\mathcal{B}} \in \mathbb{R}^3$ is the velocity command. The actions $a \in \mathbb{R}^{12}$ are retrieved from the policy π – implemented as a Multi-Layer Perceptron (MLP) formed by three layers of size $[512, 256, 256]$ – and interpreted as the reference joint positions q_j^*, tracked by the impedance controller in Equation (1). The onboard state estimator does not provide the base linear velocity in s_v; hence, we estimate it and s_f similarly to Ji et al. (2022) through an MLP of size $[128, 128, 128]$. We train π on flat ground using Proximal Policy Optimization (PPO) (Schulman et al., 2017) until convergence (Figure 7(*b*)), adopting the rewards and hyper-parameters in Table 1(*a*).

4. Roll-Drop

The proposed method, Roll-Drop, exploits the concept of using dropout to mimic an observation noise to improve the network's robustness in a sim-to-real deployment scenario. In particular, Roll-Drop adds a customised dropout layer (Hinton et al., 2012), active only during rollouts and turned off during training. The resulting random perturbations (as shown in Figure 1) cause the policy π to explore regions of the state space **s** and action space **A** different from the standard training, as in Figure 5.



(*a*)

Simulator parameter	Value
K_p	$15.0\,\mathrm{N\,m\,rad^{-1}}$
K_d	$1\,\mathrm{N\,m\,s\,rad^{-1}}$
Torque Limit	20 [N/m]
Motor Act. Delay	0.012 [s]
Motor Static Friction	0.2
Motor Dyn. Friction	0.01
Ground Friction	0.4
Gravity	-9.81 $[m/s^2]$

(*b*)

Figure 2: (a) Evaluation environment: the non-flat terrain is more realistic and it brings stochasticity to the evaluation of the robustness to observation noise. (b) Default settings for the testing environment.

When the dropout is not present, the actions are sampled with a policy $\pi_\theta(a|s)$: $\pi_\theta(a|s) = \mu_\theta(s) + \mathcal{N}(0,\sigma)$, where $\mu_\theta(s)$ is the output of the network. When, instead, dropout is included the parameters θ become $\hat{\theta} = \theta + \delta\theta$, and consequently $\pi_{\theta+\delta\theta}(a|s) = \mu_{\theta+\delta\theta}(s) + \mathcal{N}(0,\sigma)$. Based on this, in a state s: $\mu_\theta(s) \mapsto \alpha$, while $\mu_{\theta+\delta\theta}(s) \mapsto \hat{\alpha}$ with $\hat{\alpha} = \alpha + \delta\alpha$ and δa function of $\hat{\theta}$.

Assuming deterministic dynamics and same initialisation, the transition probability can be reformulated as the transition function $\mathcal{P}(s,\alpha) \rightarrow \zeta$, when dropout is inactive, and $\mathcal{P}(s,\hat{\alpha}) \rightarrow \hat{\zeta}$ otherwise, where ζ and $\hat{\zeta}$ are the next states. Similarly to $\hat{\alpha}$, $\hat{\zeta}$ can be expressed as $\hat{\zeta} = \zeta + \delta\zeta$, where $\delta\zeta = f(\alpha + \delta\alpha)$.

At the next time-step $(t+1)$, when dropout is inactive we can expect $\mathcal{P}(s', a') \rightarrow s''$, whereas when dropout is active $\mathcal{P}(s' + \delta s', a' + \delta a') \rightarrow s'' + \delta s''$. Here $\delta s'$ and $\delta s''$ represent the discrepancy between the transitions happening adopting $\pi_\theta(a|s)$ and $\pi_{\theta+\delta\theta}(a|s)$.

In this work, we added a single layer of Roll-Drop after the second layer of the MLP network. Notably, since the dropout-injected noise happens only during rollout, π develops reflexes to recover from dangerous states and becomes more robust to perturbations; conversely, adding dropouts during training does not allow the policy to develop reactions to perturbations.

4.1. Tuning Roll-Drop probability

Similarly to other randomisation techniques (Tobin et al., 2017; Valassakis et al., 2020; Campanaro et al., 2022), the tuning of the Roll-Drop probability is carried out empirically: At first, the environment (defined in Tables 1(*a*) and 1(*b*)) is tuned for tracking a velocity command s_c on flat ground **without** any randomisation and using a fixed random seed. After the policy converges to the desired behaviour, the Roll-Drop layer is included in the network, and the dropout probability is increased (starting from $p = 0.$) until the training is stable again. This can be seen in Figure 3(*a*), where we tested different dropout probabilities and how they affected the training convergence.

5. Experimental Setup

To assess the performance of the method proposed we run several experiments with different levels of noise affecting the observations. The environment's settings are fixed as in Figure 2(*b*), the robot is commanded a constant velocity $s_c = [v_x, v_y, v_z]$, where $v_x = 0.5[m/s]$ is the only non zero component.

Alongside these settings we included a mild rough terrain to better represent realistic conditions, as in Figure 2(*a*). In the environment defined as above we varied the amount of noise (n) from 0% to 60% as in Equation (2), with a step of 5%, and 100 experiments were run for each noise configuration (randomising the spawning point of the robot on the rough terrain).

$$s_t \leftarrow s_t + n \cdot \mathcal{U}(-1,1) \cdot s_t, \text{ where n} \in [0, 0.6] \tag{2}$$

The success rate in Figure 7(*a*) is measured across the 100 experiments carried out for each percentage of injected noise. To successfully complete the evaluation the robot does not have to fall on the ground and it has to walk for at least 1 [m] in the direction of the velocity commanded, if one of the two conditions is not respected the experiment is considered a failure. The ratio between the successful runs and the total number runs gives the success rate.

6. Results and Discussion

We compared Roll-Drop ($p = 0.0001$) against *No Randomisation* –which is based on the original environment used for Roll-Drop but without dropout, in Section 4.1–, against ERFI (Campanaro et al., 2022) that demonstrated state-of-the-art robustness to external perturbations, against dropout during training ($p = 0.001$), and finally a mixture of dropout during training ($p = 0.001$) plus dropout during rollout ($p = 0.0001$). From the results in Figure 7(*a*), the most robust method to the injection of noise in the observations is Roll-Drop, which retained 80% success rate when more than 25% of noise was injected. The performance of the policies trained with other techniques degrades quickly as soon as noise is injected, suggesting strong sensitivity to observation distribution encountered during training. Note that all the controllers were trained and tested adopting the same random seed.

6.1. Dropout during training

We motivate the adoption of dropouts during rollouts (Roll-Drop) in Section 4, nonetheless we investigated the performances resulting from adopting dropouts during training, and during both training and rollouts. This is depicted in Figure 3(*b*), where we show the effects of different dropout probabilities when it is applied during training and not during rollout. A cluster of lines can be identified with dropout probability $\in [0.01, 0.1]$, their maximum reward oscillates around 0.3, which corresponds to the robot standing still. Based on our experience, the randomness injected by dropout does not allow the network to correlate inputs and outputs well, and by standing still the policy avoids the termination reward (when the robot falls on the ground), while still receiving some positive points from the rewards in Table 1(*b*). Indeed, as soon as the dropout probability is lowered to $p = 0.001$ the total reward increases, and the robot starts walking again. We compare the performance of the policies trained 1) with only dropout during training ($p = 0.001$), 2) with the dropout during training ($p = 0.001$) plus dropout during rollout ($p = 0.0001$), 3) with the policy trained with Roll-Drop only. In Figure 7(*a*), we can observe that the introduction of dropout during training, even in conjunction with dropout during rollouts is detrimental.

6.2. Dropout probability and convergence

The classical usage of dropout in supervised learning is to regularise the learning of the employed networks Srivastava et al. (2014). Randomly dropping units from the Neural Network (NN) during

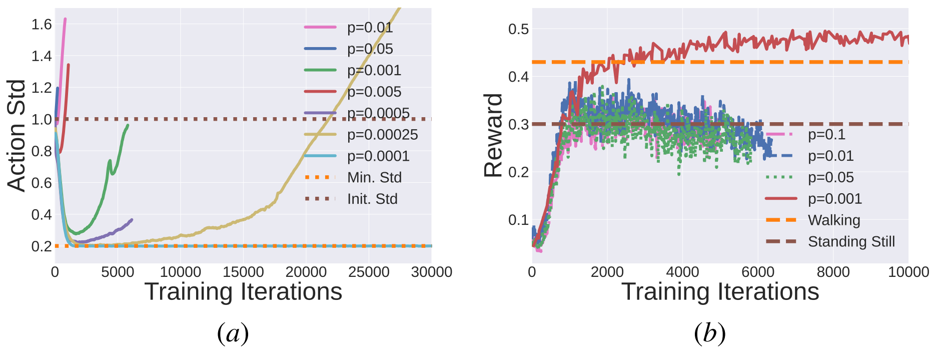


Figure 3: In Figure 3(a) the introduction of dropout during rollouts heavily affects the convergence, in this figure we show the effect of different dropout probabilities on the standard deviation used to sample actions in policy gradient algorithms. In Figure 5(c) alongside dropout during rollouts, we investigate the effects of dropout during training. When the dropout probability is too high the policy displays a standing-still behaviour, which suggests its inability to correlate inputs to outputs.

training prevents them from co-adapting, thus significantly reducing overfitting; at test time, then, the dropout is removed to approximate averaging the predictions of all these partial networks by using a complete network with smaller weights. A second dropout application is to approximate a Bayesian network Gal and Ghahramani (2016): applying dropout at inference time can generate multiple predictions from the same input through multiple inferences. This gives us a probability distribution of the outputs which we can then analyse. For such applications dropout probability typically varies between 20% and 50%, but can reach values up to 80% Srivastava et al. (2014).

As can be seen from Figure 3(a), in the case of Roll-Drop the probability is much lower: 0.01%. In fact, unlike supervised and semi-supervised learning, RL does not provide any target and the policy loss depends entirely on the actions taken by the policy itself. The better states the policy explores, the higher the reward it will receive; conversely exploring bad states can result in the policy exploring a wider actions space and eventually catastrophically diverging to even worse states. Moreover, when some neurons are dropped during rollouts the noise introduced affects the following state of the episode. Considering the latter in conjunction with having a moving target, it is clear that RL is more sensitive to dropout probabilities and that lower dropout probabilities are expected.

Figure 3(a) shows how the training diverges for dropout probabilities $\in [0.00025, 0.01]$, while for $p = 0.0001$ it converges to a stable behaviour. In fact, policy gradient algorithms like PPO (Schulman et al., 2017) explore the action space sampling from a distribution $\mathcal{N}(\mu, \sigma)$, where μ is the action a output of π, while σ is a learnt parameter. Figure 3(a) shows the σ for policies trained with different levels of dropout during rollouts (same random seed) – the initial $\sigma_0 = 1.$, and it is capped to $\sigma_{min} = 0.2$; high probabilities of dropout are responsible for the divergence.

6.3. How is Roll-Drop affecting the training?

Figure 1 depicts the effects of Roll-Drop on the training: the policies with and without dropout observe the same initial state (same random seed), and produce the same action until the first dropout is triggered. After this event, the two trainings take different trajectories $\mathcal{T}$ and $\mathcal{T}'$, as the policies output different actions and the robots experience different states. We investigated

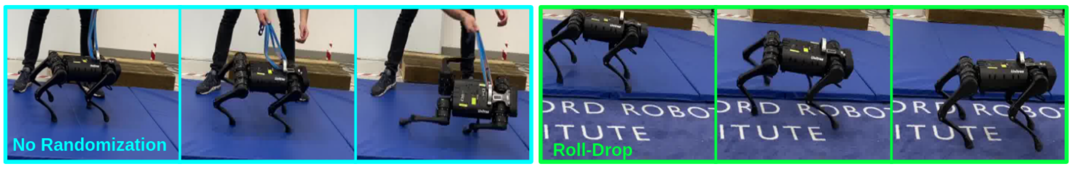


Figure 4: Policies trained with *No randomisation* (left) and Roll-Drop (right), both trained on flat ground in simulation with $K_p = 20.0\,\mathrm{N\,m\,rad^{-1}}$, and tested on the hardware with $K_p = 15\,\mathrm{N\,m\,s\,rad^{-1}}$. This gain change highlights a case of system uncertainty, demonstrating that Roll-Drop can address sim-to-real gaps.

this further in Figure 5 by recording states and actions for the first 3000 training iterations (adopting 128 parallel environments and episodes of 4 [s]) for Roll-Drop with associated probabilities $\in [0., 0.002, 0.0001]$. We show these distribution shifts for some states and actions: the joint position of `HR_KFE` in Figure 5(*a*), the joint velocity of `HL_KFE` in Figure 5(*b*), and the action of `HR_HAA` in Figure 5(*c*). These histograms demonstrate how such a tiny dropout probability – when compared to supervised/semi-supervised/non-supervised learning– affects the training: For $p = 0.0001$ the distributions of states and actions are different from $p = 0.$, while for $p = 0.002$ the training is clearly diverging (Figure 3(*a*)) with most of the states and actions distributed close to the joint position and velocity limits. Further evidences of the divergence are provided in Figures 5(*d*) to 5(*f*), where we show the mean across all environments and time-steps for each of the first 3000 iterations.

6.4. Different random seeds

We investigated how consistent the training is when different random seeds are used: We trained five policies without any randomisation and five policies with Roll-Drop across five different seeds, and compared the total reward of both groups in terms of mean and standard deviation. We considered the *No Randomisation* setting as the perfect candidate for this comparison since it is massively overfitting to the simulation environment. From Figure 7(*b*), we discovered that Roll-Drop (blue line) is on average performing better, because of its higher robustness across different seeds, and this is also supported by the smaller standard deviation, when compared to *No Randomisation*. However, as expected, *No Randomisation* is in absolute value performing better than Roll-Drop, but only for the seed the environment was originally tuned on; while for other seeds it gained lower rewards, and it has a more spread standard deviation.

6.5. Training and deployment mismatch

Apart from increasing the robustness to observation noise, Roll-Drop is also providing the policies with additional flexibility to external perturbations. Indeed, we trained two more policies -with and without Roll-Drop- with $K_p = 20.0\,\mathrm{N\,m\,rad^{-1}}$ and we deployed them on the hardware using $K_p = 15.0\,\mathrm{N\,m\,rad^{-1}}$, the target velocity command is $s_c = [0., 0., 0.]$. As can be seen from Figure 4, the policy trained without any sort of randomisation is not able to stand, while on the other hand Roll-Drop allows the policy to find equilibrium and to better follow the velocity command. Quantitative advantages of adopting Roll-Drop for the experiments above are provided in Figure 6,

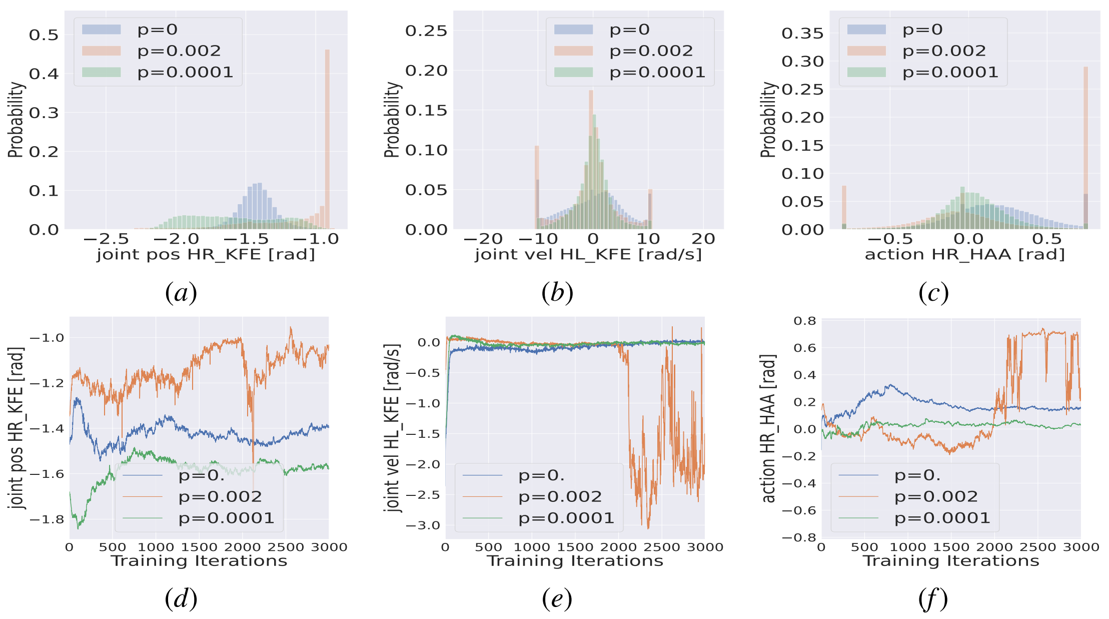


Figure 5: (a) to (c) show some state/action distributions for Roll-Drop with three different probabilities: 0 (no Roll-Drop), 0.002, and 0.0001. When $p = 0$ the policy is over-fitting to a fixed simulation environment, when $p = 0.0001$ the policy converges to the desired behaviour, but exploring a different state/action space compared to $p = 0$, and finally, when $p = 0.002$ it diverges catastrophically and it explores bad portions of the state/action spaces (often the joint position/velocity limits) as can be seen from the orange histograms. Evidences of the divergence are provided in (d) to (f), which represent the mean for each state/action along the initial 3000 iterations considered.

where we show better velocity command tracking (both linear and angular), and lower joint velocity usage.

7. Conclusion

In this work we show how to account for observation noise without tuning randomisation distributions for each of the states/sensors as is commonly used in DRL. This can be simply done by including dropout during rollouts (Roll-Drop) in the network architecture and by tuning a single parameter: the dropout probability. In fact, by turning on and off neurons during the rollouts we show a considerable improvements in noise-injection robustness (200%), and a success rate of 80% when 25% noise in injected. Alongside the results we present a thorough analysis to explain the effects of different dropout implementations and associated probabilities on performances, convergence, and state/action distributions. The approach was also validated on the hardware and tested on board of the Unitree A1 quadruped robot. Although we conducted a thorough ablation study, future research will explore the impact of changing the position of the dropout layer in the network on the controller's overall performance.

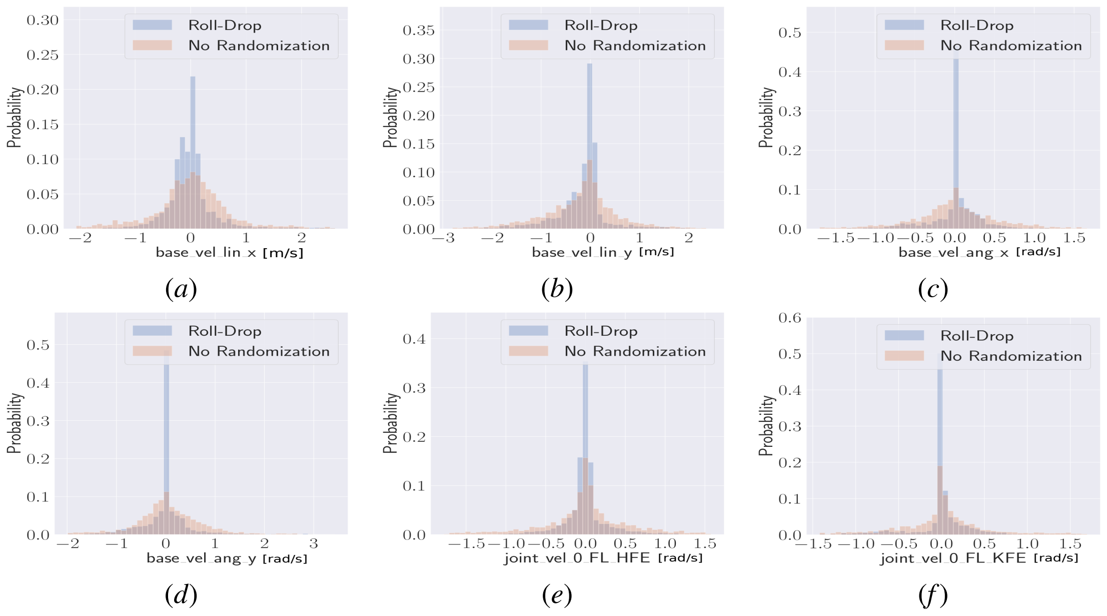


Figure 6: These policies are trained in simulation with $K_p = 20.0\,\mathrm{N\,m\,rad^{-1}}$ and deployed on the hardware (Unitree A1) using $K_p = 15.0\,\mathrm{N\,m\,rad^{-1}}$. In this context Roll-Drop demonstrated better performances in tracking the velocity command and in expending less velocity at the joints.

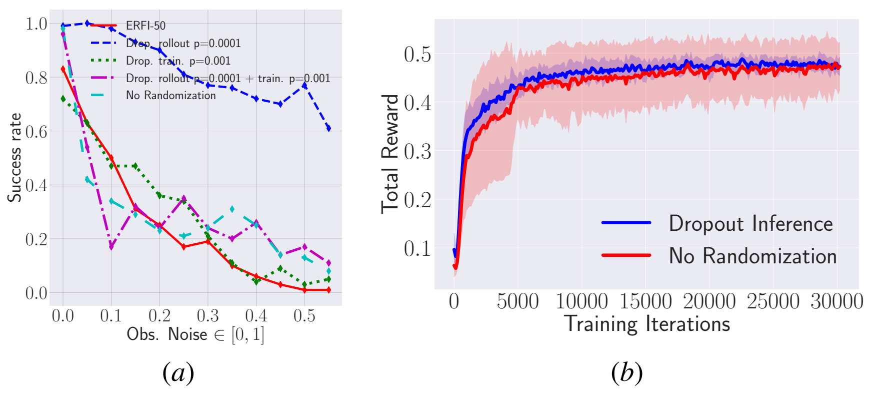


Figure 7: (a) Comparing the success rate of different methods in the presence of noise in the observations. Roll-Drop performs more than twice as good as other methods, retaining 80% success rate with more than 25% injected noise. (b) we trained the Roll-Drop and the *No Randomisation* policies using 5 different random seeds, and here we compare their total rewards. Roll-Drop is more consistent, demonstrating on average higher total reward and a smaller standard deviation. Conversely, *No Randomisation* is more sensitive to the random seed used: highest total reward on the seed on which the environment was tuned on, but the performance degraded when other seeds were adopted.

References

Steven Bohez, Saran Tunyasuvunakool, Philemon Brakel, Fereshteh Sadeghi, Leonard Hasenclever, Yuval Tassa, Emilio Parisotto, Jan Humplik, Tuomas Haarnoja, Roland Hafner, Markus Wulfmeier, Michael Neunert, Ben Moran, Noah Siegel, Andrea Huber, Francesco Romano, Nathan Batchelor, Federico Casarini, Josh Merel, Raia Hadsell, and Nicolas Heess. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors, 2022. URL https://arxiv.org/abs/2203.17138.

Luigi Campanaro, Siddhant Gangapurwala, Wolfgang Merkt, and Ioannis Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization, 2022. URL https://arxiv.org/abs/2209.12878.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon, and Ioannis Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 38(5):2908–2927, 2022. doi: 10.1109/TRO.2022.3172469.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Christian Hubicki, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research*, 35(12):1497–1521, 2016. doi: 10.1177/0278364916648388. URL https://doi.org/10.1177/0278364916648388.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C. Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. doi: 10.1109/IROS.2016.7758092.

Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018. URL www.raisim.com.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. doi: 10.1126/scirobotics.aau5872. URL https://www.science.org/doi/abs/10.1126/scirobotics.aau5872.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49286-3.

Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022. doi: 10.1109/LRA.2022.3151396.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

Joonho Lee, Jemin Hwangbo, and Marco Hutter. Robust recovery controller for a quadrupedal robot using deep reinforcement learning. *CoRR*, abs/1901.07517, 2019. URL `http://arxiv.org/abs/1901.07517`.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abc5986`.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL `https://openreview.net/forum?id=fgFBtYgJQX_`.

Matias Mattamala, Nived Chebrolu, and Maurice Fallon. An efficient locally reactive controller for safe navigation in visual teach and repeat missions. *IEEE Robotics and Automation Letters*, 7(2): 2353–2360, 2022. doi: 10.1109/LRA.2022.3143196.

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`.

Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 08–11 Nov 2022. URL `https://proceedings.mlr.press/v164/rudin22a.html`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315, 2021. doi: 10.1109/ICRA48506.2021.9561814.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, jan 2014. ISSN 1532-4435.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.010.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

Eugene Valassakis, Zihan Ding, and Edward Johns. Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5372–5379, 2020. doi: 10.1109/IROS45743.2020.9341617.

Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 5(49):eabb2174, 2020.

Fangzhou Yu, Ryan Batke, Jeremy Dao, Jonathan Hurst, Kevin Green, and Alan Fern. Dynamic bipedal maneuvers through sim-to-real reinforcement learning, 2022. URL https://arxiv.org/abs/2207.07835.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Roll-Drop: accounting for observation noise with a single parameter
Publication Status	Published
Publication Details	Campanaro L., De Martini D., Gangapurwala S., Merkt W., Havoutis I., Roll-Drop: accounting for observation noise with a single parameter, 2023

Student Confirmation

Student Name:			
Contribution to the Paper	Conceptualized the idea, implemented it, trained it, evaluated the results, deployed on the robots, contributed to final manuscript.		
Signature Luigi Campanaro		Date	11/04/2023

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Dr. Ioannis Havoutis		
Supervisor comments The contributions are as above		
Signature Ioannis Havoutis	Date	19/04/2023

This completed form should be included in the thesis, at the end of the relevant chapter.

5.3 Dropout in supervised learning vs Roll-Drop

Dropout was originally introduced to mitigate overfitting on held-out test data in supervised learning scenarios [92]. This approach involves a straightforward principle: during training, each hidden unit is stochastically deactivated with a probability of $(1 - p)$, where p signifies the likelihood of maintaining the neuron's functionality. In supervised learning contexts, p is commonly situated within the range of 0.5 to 0.8 [92, 93], resulting in a corresponding neuron deactivation probability of $(1 - p) \in [0.2, 0.5]$.

Overfitting can be understood as a common occurrence when an algorithm's performance diminishes between evaluation and target environments [94]. Nevertheless, the underlying causes of this decline in supervised learning and RL differ. In the former scenario, it is referred to as data overfitting, while in the latter, it is called environment overfitting.

5.3.1 Data overfitting

Data overfitting arises when a learning procedure returns a function tailored excessively to a specific dataset, diminishing its ability to generalize to independently sampled data from the same environment. This issue predominantly affects supervised learning, where evaluations are conducted on a static dataset, and it is mitigated through partitioning data into training and test sets. In contrast, this concern does not apply to RL, specifically in on-policy algorithms like PPO, as the dataset remains dynamic since the actions taken influence the data sampled in the following time-steps.

5.3.2 Environment overfitting

Environment overfitting occurs when a learning algorithm is tailored to the evaluation environment, resulting in poor performance within the target environment. In our case, the evaluation environment is the simulator, while the target environment is the real world. This challenge has been tackled in the literature through techniques like domain randomization or ERFI, proposed in this thesis. Nonetheless, as highlighted in [95], even in stochastic environments, memorization can still occur, and agents have demonstrated the capacity to adapt to random noises. It is worth noting, as emphasized by [95], that memorization and overfitting may not always be detrimental. Just as humans inherently overfit their muscle memories, enabling efficient actions, legged robots overfitting to a vast range of different environments produce robust controllers.

5.3.3 Observation and action shift in Roll-Drop

In contrast to the conventional dropout technique [92, 93], Roll-Drop is applied during inference, imparting stochastic perturbations to the robot's trajectories within rollouts. Despite employing dropout probabilities several orders of magnitude lower than those commonly utilized in supervised learning, this is enough to induce a notable shift in both state and action distributions. Such perturbations can trigger a significant divergence in training outcomes, as evidenced in [3]. However, a correct calibration of the Roll-Drop probability in training results in improved robustness to state noise during deployment, due to the diverse trajectories encountered by the policy during the rollouts in simulation.

5.4 Discussion and conclusion

In this work we proposed a new approach to account for observation noise in Deep-RL without requiring the tuning of randomisation distributions for each state or sensor. The proposed approach is called Roll-Drop and involves including a dropout layer during rollouts and tuning a single parameter, the dropout probability.

By turning on and off neurons during rollouts, we demonstrated significant improvements in observation-noise robustness (200%) and a success rate of 80% when 25% noise is injected.

To support the findings we presented a detailed analysis of the effects of different dropout implementations and associated probabilities on performances, convergence, and state/action distributions. Although we conducted a thorough ablation study, future research could explore the impact of changing the position of the dropout layer in the network on the controller's overall performance.

6

Bipedal locomotion with quadruped robots

Contents

6.1 Introduction

In recent years, quadruped robots have gained significant attention from both industry and academia due to their higher mobility compared to wheeled vehicles. They are probably the best-known and studied class of legged systems, and their capabilities have been demonstrated in various tasks, including jumping over obstacles [96], recovering from falls [61], trotting blind on rough terrain [62], and navigating over hurdles using vision [97].

Despite these recent advances, quadruped robots still lag behind their natural counterparts in terms of agility and dynamic maneuvers. Four-legged animals, for instance, can adopt two-legged locomotion to interact with the environment, traverse tight spaces, and even use their forelimbs for manipulation. However, such maneuvers require a fine control of the body and limbs' inertia to maintain balance, and achieving bipedal locomotion on point feet with quadruped robots is a challenging scenario for current control architectures. This challenge is compounded by the absence of adduction/abduction joints in the bipedal configuration shown in Figure 6.14, sub-optimal joint limits for standing on two legs, resulting in bent knees and higher torques, and motors not designed for bipedal locomotion.

To tackle the challenging task of omni-directional bipedal locomotion on point feet with a quadruped robot without any hardware modification or external support, we adopt model-free controllers that promise to exploit the full dynamics of the system. This work aims to develop a controller that complies with the quadruped's hardware requirements and can navigate challenging environments. The task serves as a perfect benchmark to explore the capabilities and limitations of modern Deep-RL methods while investigating new skills and avenues for the legged robotics community.

Additionally, this work examines the challenges posed by the sim-to-real gap in extremely dynamic scenarios -as the one considered here- and the limitations we encountered while adopting state-of-the-art techniques that shown their effectiveness in less dynamic locomotion applications.

6.1.1 Contributions

Prior research on bipedal locomotion of quadruped robots is limited to jumping gaits[98], requires mechanical modifications to the robot for stability [99], or adopts robots with wheels [80], which do not handle the discontinuous dynamics of stepping.

In contrast, our method discourages jumping movements through reward shaping and learns omni-directional bipedal locomotion on point feet without utilizing wheels.

Our main contributions are:

- The development of an omni-directional controller for quadruped robots capable of walking on two legs with point-feet. This approach was initially trained in simulation and subsequently validated against hardware execution data.
- The development of the first bipedal controller for quadruped robots (with point feet) capable of walking in simulation and taking 11 steps in the real world without requiring hardware modifications.
- Highlighting the shortcomings of currently adopted sim-to-real techniques in highly dynamic applications.

6.2 Related works

Planning feasible motions for legged robots is a challenging task, as base movements cannot be directly generated but result from the discontinuous interaction among limbs and the environment. To address this issue, various methods have been developed in the field of legged robotics.

One approach is to use Trajectory Optimisation (TO), as discussed in Section 2.1, to automate the generation of desired motions. TO takes high-level tasks as inputs and determines the motions and forces that comply with all restrictions using an optimizer [27]. However, these methods require selecting, and eventually approximating, a suitable dynamics model for the system.

Another approach is to use RL to design dynamic controllers. These controllers are trained in simulation using proprioceptive or/and exteroceptive information and then transferred to the real hardware. Applications of RL in legged robotics range from quadruped robots [60, 62, 66, 97] to biped robots [23, 24, 73], and the robotics community is increasingly interested in pushing the boundaries of legged locomotion by mimicking their natural counterparts.

Recent research has investigated the transition among different gaits [100, 101] as well as bipedal locomotion [98, 99]. For example, in [98] the authors presented a hopping controller that allows a legged robot to hop on its two hind legs using as template a spring-loaded inverted pendulum (SLIP) model to generate the trajectory of the CoM; Mini Cheetah was adopted as reference platform [102]. Then, the control-Lyapunov function based quadratic programming (CLF-QP) controller modulates the nominal ground reaction forces (GRFs) to balance the torso. The legs dynamics is ignored, and the fore legs remain close to the body in a fixed position during the motion; with this method the authors demonstrated a hopping gait on the two rear legs with Mini Cheetah in simulation.

Similarly, in [99] the authors proposed a multi-modal legged robot that can walk in both quadrupedal and bipedal fashion using a combination of RL and Inverse Kinematics (IK); this was also based on Mini Cheetah. Their approach required modifying the robot with a new mechanical part to improve stability (i.e., to achieve a two-point / line contact per leg) and used a scripted motion based on IK for the gait transition.

Furthermore, in [80], the authors proposed an RL-based method that combines several motion styles in a single policy without the need for excessive reward tuning. These styles include ducking, walking, and switching between quadrupedal and bipedal locomotion. However, in all the configurations, the robot mounts wheels, and this makes the contact dynamics continuous since the robot tracks the desired base-velocity command without stepping but by controlling the velocities of the motors that drive the wheels.

6.3 Preliminaries

We model the general quadrupedal robot system as a floating base B with four actuated limbs: front right (FR), front left (FL), hind right (HR) and hind left (HL). Each leg is made up of three joints: hip adduction/abduction (HAA), hip flexion/extension (HFE) and knee flexion/extension (KFE); the legs and the joint names can be combined, for example hind left leg knee flexion/extension joint is abbreviated as *HL KFE*. The robot state is described w.r.t. a *world* inertial reference frame W. The base position is expressed in this frame as ${}_W r_{WB} \in \mathbb{R}^3$, and the orientation, $\mathrm{q}_{WB} \in \mathbb{SO}(3)$, is represented using a unit quaternion. The corresponding rotation matrix is given by $\mathbf{R}_{WB} \in \mathbb{SO}(3)$. The positions of the joints are represented by $\mathrm{q}_j \in \mathbb{R}^{n_j}$. For the A1 robot used in this work, $n_j = 12$. The linear and angular velocities of the base are written as ${}_W\mathrm{v}_{WB} \in \mathbb{R}^3$ and ${}_W\omega_{WB} \in \mathbb{R}^3$, respectively. The generalized coordinates and velocities are stacked as vectors q and u where

$$\mathrm{q} = \begin{bmatrix} {}_W r_{WB} \\ \mathrm{q}_{WB} \\ \mathrm{q}_j \end{bmatrix} \in \mathbb{SE}(3) \times \mathbb{R}^{n_j}, \quad \mathrm{u} = \begin{bmatrix} {}_W\mathrm{v}_{WB} \\ {}_W\omega_{WB} \\ \dot{\mathrm{q}}_j \end{bmatrix} \in \mathbb{R}^{6+n_j}. \tag{6.1}$$

The A1 quadrupedal system is actuated using the joint control torques $\tau_j \in \mathbb{R}^{n_j}$ which are computed at the actuator level using an impedance control model written as

$$\tau_j = K_p(\mathrm{q}_j^*[-d] - \mathrm{q}_j) + K_d(\dot{\mathrm{q}}_j^* - \dot{\mathrm{q}}_j) - f_s - f_d \cdot \dot{\mathrm{q}}_j \tag{6.2}$$

where K_p and K_d refer to the position and velocity tracking gains, q_j^* is the list of vectors representing the desired joint positions, $d \in [0, 7] (d \in \mathbb{N})$ is the injected delay used to randomly select previous desired positions, $\dot{\mathrm{q}}_j^*$, the desired joint velocities, f_s and f_d refer to the static and dynamic friction, respectively. For concise presentation, we will assume that all of the translations and rotations are measured and expressed with respect to the reference frame W. In this regard, the notation ${}_W r_{WB}$ will be shortened to r_B, $\mathbf{R}_{WB}$ to $\mathbf{R}_B$, and similarly for all the other introduced notations.

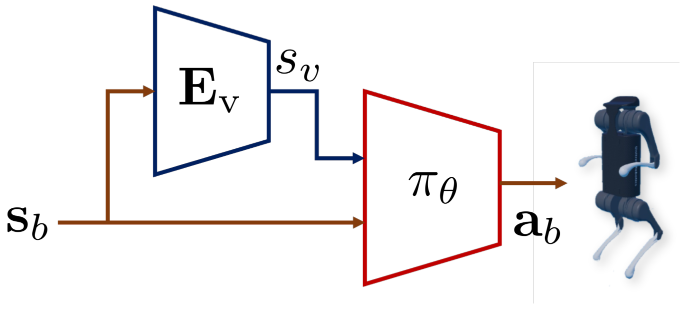


Figure 6.1: Overview of the control architecture utilized for bipedal locomotion. The proposed framework also employs a velocity estimator network $\mathbf{E}_{\text{v}}$ to predict the robot's linear velocity s_v. The locomotion policy, π_θ, then maps the robot state information into control action, $\mathbf{a}_b$, representing the desired joint positions q_j^*. We concurrently learn both $\mathbf{E}_{\text{v}}$ and π_θ.

6.4 Methodology

Data-driven deep RL approaches allow for natural emergence of sophisticated control intelligence [60]. Parameterizing the control policy as a neural network further enables complex non-linear mapping of robot state information to desired control action. In this regard, we employ a control architecture illustrated in Figure 6.1 to generate desired joint positions q_j^* at a control frequency of 50 Hz. These joint positions are then tracked at the actuator level using the impedance controller described in Equation (6.2).

This section details upon the Markov decision process (MDP) formulation employed for obtaining a bipedal locomotion policy, about the training setup, and about the significant obstacles encountered in attempting to transfer the policy to the hardware.

6.4.1 State and Action

We define the MDP state as a tuple $\mathbf{s}_e := \langle \mathbf{s}_b, s_v \rangle$ where $\mathbf{s}_b := \langle s_R, s_\omega, s_j, s_a, s_c \rangle$. The state $\mathbf{s}_e \in \mathbb{R}^{192}$ comprises terms that represent the base orientation $s_R \in \mathbb{R}^3$, base linear velocity $s_v \in \mathbb{R}^3$, base angular velocity $s_\omega \in \mathbb{R}^3$, joint state history $s_j \in \mathbb{R}^{168}$, previously generated action $s_a \in \mathbb{R}^{12}$, and desired base velocity command term $s_c \in \mathbb{R}^3$. We introduce a horizontal reference frame H such that ${}_W r_{WH} = {}_W r_{WB}$ and $\mathbf{R}_{WH} = \mathbf{R}_{WB_z}$, where $\mathbf{R}_{WB_z}$ is the z decomposition of the base rotation matrix which can be expressed as $\mathbf{R}_{WB} = \mathbf{R}_{WB_z}\mathbf{R}_{WB_y}\mathbf{R}_{WB_x}$. The velocity command $\mathbf{c}^*$ is then written as

$$\mathbf{c}^* = [\mathrm{v}_x^*\mathbf{e}_x^H \quad \mathrm{v}_y^*\mathbf{e}_y^H \quad \omega_z^*\mathbf{e}_z^H]^T \tag{6.3}$$

where $\mathrm{v}_x^*\mathbf{e}_x^H$ is the desired base linear velocity along the x-axis described in the horizontal frame. This allows for commanding the robot using a user-input (such as with a joypad) or a high-level goal planner.

Table 6.1: State term definitions for the bipedal MDP. Here, t refers to the current measurement and the duration between $t-1$ and t corresponds to 20 ms. Also, $\Delta \mathrm{q}_{j_t} = \mathrm{q}_{j_t} - \mathrm{q}^*_{j\,t-1}$ represents the joint position tracking error.

State Terms
$s_R = \mathbf{e}_z^B$
$s_v = \mathbf{R}_B^T \mathrm{v}_B$
$s_\omega = \mathbf{R}_B^T \omega_B$
$s_j = [\mathrm{q}_{jt}^T \ \ \Delta\mathrm{q}_{jt-1}^T \ \ \Delta\mathrm{q}_{jt-2}^T \ \ \Delta\mathrm{q}_{jt-3}^T \ \ \Delta\mathrm{q}_{jt-4}^T \ \ \Delta\mathrm{q}_{jt-5}^T \ \ \Delta\mathrm{q}_{jt-6}^T$ $\dot{\mathrm{q}}_{jt}^T \ \ \dot{\mathrm{q}}_{jt-1}^T \ \ \dot{\mathrm{q}}_{jt-2}^T \ \ \dot{\mathrm{q}}_{jt-3}^T \ \ \dot{\mathrm{q}}_{jt-4}^T \ \ \dot{\mathrm{q}}_{jt-5}^T \ \ \dot{\mathrm{q}}_{jt-6}^T]^T$
$s_a = \mathrm{q}^*_{j\,t-1}$
$s_c = \mathbf{c}^*$

The design of our state space is motivated by prior works on quadrupedal locomotion using deep RL [60, 62, 66]. These have shown that the introduction of a history of joint position tracking errors and joint velocities enables implicit modeling of system dynamics which facilitates emergence of system-adaptive behavior robust to variations in dynamics. We will further discuss the importance of this history in Section 6.7.6.

The robot information included in the MDP state space can be directly extracted from the on-board sensors. This is based on the assumption that a state estimator is available to integrate and filter sensory data. In this work, we employ the architecture proposed in [103] to concurrently learn a base linear velocity estimator $\mathbf{E}_\mathrm{v}$, parameterized as an MLP, to generate $s_v \in \mathbb{R}^3$ from the partially observable state information $\mathbf{s}_b \in \mathbb{R}^{189}$. The estimator, $\mathbf{E}_\mathrm{v}$, comprises two dense layers of dimensions $\{128, 128\}$ with LeakyReLU activation. All the other measurements are directly accessible from on-board sensors to a sufficiently accurate degree and do not require further filtering or processing. The output s_v of the linear base velocity estimator $\mathbf{E}_{\mathrm{v}}$ is then concatenated to $\mathbf{s}_b$ and forwarded to the locomotion policy π_θ which is parameterized as a MLP with two dense layers of dimensions $\{512, 512\}$ with LeakyReLU activation.

The action is given by the tuple $\mathbf{a}_b := \langle \mathrm{q}_j^* \rangle$ and represents the desired joint positions for each of the twelve actuators on the robot. The actions produced by π_θ are scaled by 0.25 to limit the motion of the upper limbs, by 0.5 for the lower limbs and are clipped between $\frac{\pi}{4}$ and $-\frac{\pi}{4}$ to avoid exploring states too far from the nominal joint configuration q^N.

The actions modified as above are then sent to a custom PD controller, the latter is in charge of computing the torque and it also takes care of modelling communication delays, static and dynamic friction, as shown in Equation (6.2). The nominal values used are $K_p = 30$, $K_d = 0.5$, $\tau_{max} = 20$, $delay \in [0, 7]$, $f_s = 0.2$ and $f_d = 0.01$.

6.4.2 Training

The PPO algorithm [104] was adopted for training using clipped loss and GAE (Generalized Advantage Estimation), the hyperparameters used are: learning rate=1e-4, epochs=8, ϵ=0.2, γ=0.998, λ=0.95, episode length=4 s, mini batch=4500. Similarly to [60, 62], we adopt curriculum learning to improve the robustness of the controller. It is important to note that the rewards do not depend on the curriculum, since it affects only the magnitude of external perturbations and the degree of randomization of the environment. Where the curriculum factor k_f is computed as in [60], such that every learning iteration j the curriculum factor $k_{f_{j+1}} = (k_{f_j})^{k_i}$, starting from the k_{f_0} and a common $k_i = 0.999$. In more details, we split the training in two different phases with their respective curriculum factors: before 6000 training iterations $k_{f_0}^1 = 0.05$ and this allows the policy π_θ to exploit the perfect simulation environment and quickly learn bipedal locomotion. After 6000 iteration

	Nominal	**Std**
Mass	10	$0.2 \cdot Nominal$
Knee joint pos	-0.2	0.02
P gain	30	5
D gain	0.5	0.15
Torque limits	20	2.5
Friction	0.4	0.2
Gravity	-9.81	0.5
Static friction	0.2	$0.3 \cdot Nominal$
Dynamic friction	0.01	$0.3 \cdot Nominal$
External force	0	4
Orientation perturbation	0	$\frac{\pi}{18}$

Table 6.2: Domain randomization parameters

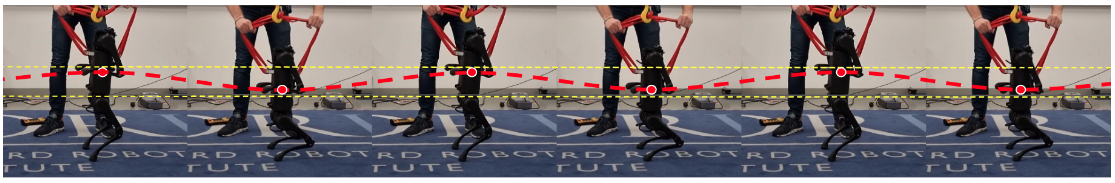


Figure 6.2: Squat controller in action.

the curriculum learning starts, $k_{f_0}^2$ is updated and π_θ is exposed to increasingly more difficult environments, by randomizing base mass, knee joint position, PD gains, torque limits, friction, gravity, static friction, dynamic friction, external forces and base orientation. During this phase, at every reset of the environment, the new value each of parameters in Table 6.2 is computed as : $p_{new} = p_{nominal} + p_{std} \cdot p^* \cdot k_f^2$, where p^* is sampled from a normal distribution $X \sim \mathcal{N}(0, 1)$.

Lastly, after 12000 training iterations the learning rate decays by a factor of ten every 1000 iterations and such that the π_θ settles to the desired behavior.

The terminal state, responsible of resetting the environments, is triggered when the episode is completed or when the robot touches the ground with parts different from the two feet f_{HR} or f_{HL}.

6.4.3 Reward

The training is guided by the reward function composed by the weighted sum of the terms in Table 6.3.

	Definition	Weight
Base orientation	$exp(-1(\cos^{-1}(\mathbf{e}_z^B \cdot \mathbf{e}_z^H)\| - \pi/2)^2)$	$c_1 = 4$
Base linear velocity	$\phi(v^*_{H_{x,y}}, v_{H_{x,y}}, 10)$	$c_2 = 22$
Base angular velocity	$\phi(\omega^*_{H_z}, \omega_{H_z}, 10)$	$c_3 = 22$
Action smoothness	$\|\|\mathrm{q}^*_{j_t} - \mathrm{q}^*_{j_{t-1}}\|\|^2$	$c_4 = -2$
Feet clearance	$\sum_{n=2}^{n<4}(0.1 - f_{n z})^2$	$c_5 = -1500$
Feet slip	$\sum_{n=2}^{n<4} \|\|f_{n\dot{x},\dot{y}}\|\|^2$	$c_6 = -25$
Joint position	$\|\|q_j - \mathrm{q}_j^N\|\|^2$	$c_7 = -2$
Joint velocity	$\|\|\dot{q}_j\|\|^2$	$c_8 = -0.3$
Joint torque	$\|\|\tau_j\|\|^2$	$c_9 = -1$
Feet swing duration	$\sum_{n=2}^{n<4} \max(\min(\mathbf{t}_n^{air}, 0.5), 0.4)$	$c_{10} = 10$
Jumping gait	$\mathbf{t}_2^{air} \cdot \mathbf{t}_3^{air}$	$c_{11} = -3$
Contact transition	$\sum_{n=2}^{n<4}(contact_{t-1}^n == contact_t^n)\ ?\ 0 : 1$	$c_{12} = -0.5$

Table 6.3: Reward term definitions for the bipedal MDP, where $\phi(x, t, s) := e^{-s||x-t||^2}$.

6.5 Ablation study

In the following section we detail on some of the decisions taken during the training of the bipedal policy.

6.5.1 Hardware feasibility of the bipedal configuration

In consideration of the potential stress that bipedal locomotion can place on motors, particularly the knee joints, we opted to first test the feasibility of the approach by evaluating the machine's ability to stand on two legs before conducting any hardware validation with the learned policy. To achieve this, we developed an IK-based bipedal controller that enabled the Unitree A1 to perform squats while on its rear legs. The controller utilized a sinusoidal function to generate the desired height of the CoM (h_{CoM}), which was then used in conjunction with the IK to compute the required joint angles for tracking. Specifically, the height of the CoM was defined as $h_{CoM} = 0.3 + A\sin(2\pi f t + \Delta\theta)$, where A, f, and $\Delta\theta$ correspond to the amplitude, frequency, and phase shift of the oscillation, respectively. The desired joint angles for the rear HFE and KFE joints were calculated using the equations in Equation (6.4), with reference to the vertices depicted in Figure 6.4a.

$$\begin{aligned} AB = BC \Rightarrow HC = AH \quad &\text{(specific to Unitree A1)} \\ q_{HFE} = acos(\frac{AH}{AB}) + \frac{\pi}{2}& \\ q_{KFE} = -2 \cdot acos(\frac{HC}{BC})& \end{aligned} \tag{6.4}$$

We then compared the torques recorded during our hardware experiments using the sqat-controller with the torques measured during simulation testing of the Deep-RL trained policy. Our results indicated that, on average, the policy consumed less torque than the squat controller, while exerting the same maximum torque, as shown in Figure 6.3.

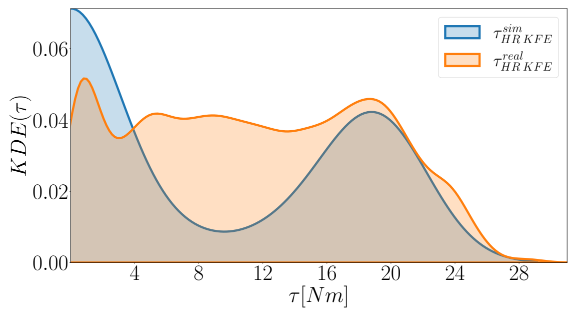


Figure 6.3: The Kernel Density Estimations (KDEs) of the τ_j^{real}, recorded with the squat controller, and τ_j^{sim} obtained executing π_θ overlap, the π_θ consumes overall less torque.

To further assess the differences between joint velocities in simulation and reality, we suspended the robot and generated a chirp signal to control one of the rear leg's knee joints. We compared the measured velocities in both simulation and hardware in the frequency domain, and found no visible anomalies, as illustrated in Figures 6.4b and 6.4c, except for those caused by the sim-to-real gap, such as the ideal/real PD controller and the inertia of the limbs.

Based on these comparative tests of joint torques and velocities between hardware and simulation, we concluded that the hardware is capable of exerting the necessary torques and velocities required by the controller trained in simulation.

6.5.2 Randomization curriculum

To train the policy, we followed the settings outlined in Section 6.4. We conducted six experiments in total, three of which started the randomization curriculum from

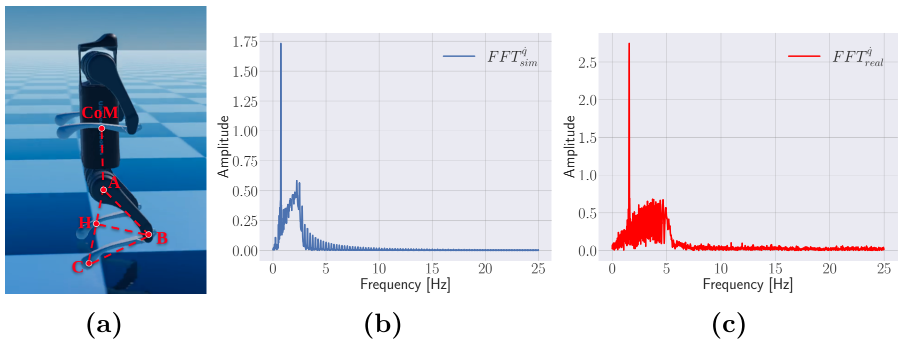


Figure 6.4: Figure 6.4a Vertices positioning for the IK in Equation (6.4). Figures 6.4b and 6.4c Joint velocity sim-real comparison.

the beginning, and three which started after 6000 iterations. We analyzed the results of both sets of experiments in terms of the average total reward and its standard deviation, as shown in Figure 6.5. The figure clearly demonstrates that beginning the randomization curriculum from the start of the training significantly compromises the final performance with the current settings.

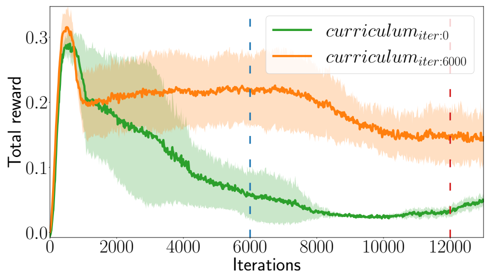
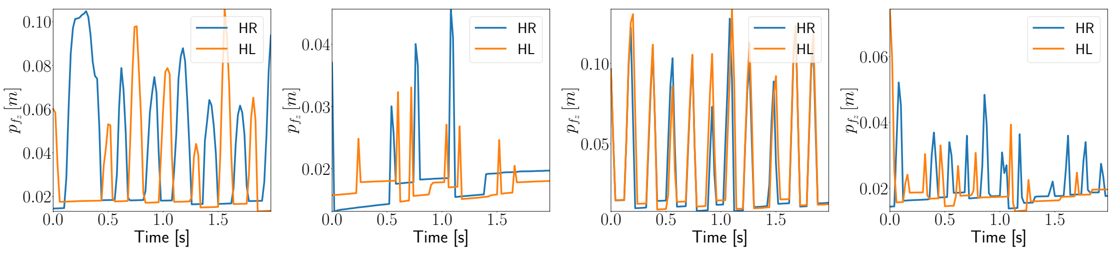


Figure 6.5: In this picture we show how curriculum scheduling affect the training. We observed that starting the curriculum (with the settings in Section 6.4) after the policy already converged to a meaningful solution (blue dashed line, 6000 iterations) determines a higher total reward and better locomotion behaviors compared to starting with the same curriculum from the beginning of the training. Moreover the red dashed line represents the point from which the learning rate starts decaying (12000 iterations).

6.5.3 Reward ablation

Many of the rewards in Table 6.3 have been previously adopted to train quadruped robot controllers [60, 62, 65, 66]. However, in this work, we introduce three rewards that are particularly useful for bipedal locomotion. These rewards include the foot swing promotion reward, which encourages the lifting of the feet and helps to avoid



(a) No ablation **(b)** $c_{10} = 0$ **(c)** $c_{11} = 0$ **(d)** $c_{12} = 0$

Figure 6.6: Reward ablation: setting some of the rewards to zero affects stepping height, stepping frequency, and a hopping behavior. We demonstrate this statement in the plots above, here each of the policy was trained till convergence.

dragging, the contact transition reward, which penalizes frequent transitions to different contact states, and the jumping reward, which helps to avoid hopping gaits.

To investigate the contribution of each of these reward terms, we trained four different policies. The first policy includes all the reward terms presented in Table 6.3. In the second policy, only the foot swing duration reward is set to zero, and in the third policy, only the contact transition reward is set to null. The same procedure is repeated in the fourth policy, but only for the jumping gait reward.

We compared the feet height in the policy trained with the full set of rewards (Figure 6.6a) with the different ablations to appreciate the contribution of each reward term. As shown in Figure 6.6b, the foot swing promotion reward significantly promotes lifting of the feet. Similarly, the contact transition reward effectively reduces the frequency of transitioning between contact states, as seen in Figure 6.6c. Lastly, the jumping reward successfully discourages hopping behavior, as demonstrated in Figure 6.6d.

6.5.4 Role of the upper limbs

Previous research on bipedal locomotion using quadruped robots [98] has involved keeping the upper limbs fixed to the base during locomotion to reduce controller complexity. In contrast, our study allows for the policy π to actively control the upper limbs. To investigate the role of the arms during locomotion, we conducted tests where we blocked the use of the arms. We found that on flat ground - the environment on which the robot was originally trained - the robot did not heavily rely on the arms.

However, when we repeated the experiment on rough terrain (as shown in Figure 6.7), an environment that the robot had never experienced during training, we observed that the arms were crucial for maintaining balance and tracking the commanded velocities without falling. This observation is supported by the results shown in Figures 6.15d and 6.15h. When the arms were locked in the q^N position, the robot fell to the ground, as shown by the spikes in the measured velocities in Figure 6.8.

Overall, our results suggest that the arms play an important role in maintaining stability during bipedal locomotion, particularly in challenging environments.

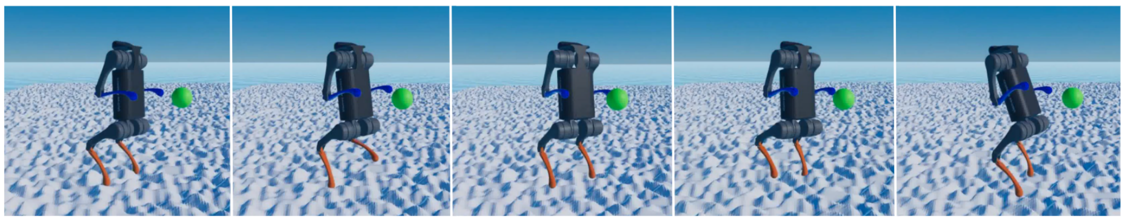


Figure 6.7: The robot was tested on rough terrain to assess the role of the upper limbs in maintaining balance during locomotion.

6.6 Results

This study introduces an omni-directional controller designed for quadruped robots with point-feet walking on two legs. To demonstrate the controller's robustness to changes in the simulation environment, we conducted experiments with variations beyond the randomization domain used during training. These variations included changes in the PD gains and delay in the impedance controller, torque limits, friction coefficient between the feet and the ground, gravity acceleration, mass of the robot's base, and position of the knee joints along the thighs.

We evaluated the controller's performance by analyzing two metrics: success rate, defined as the ratio of completed experiments without falling to the total number of experiments, and velocity tracking error, defined as the difference between the desired and measured base velocity. As shown in Figure 6.9, the controller

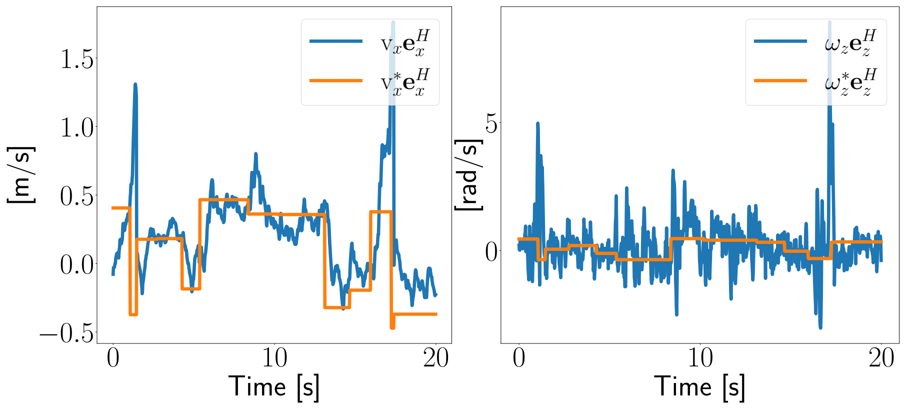


Figure 6.8: The base velocities shown above are recorded during bipedal locomotion on rough terrain with locking the arms of the robot in the nominal position. Without using the arms to balance the robot fell on the ground, and this can be inferred from the tall spikes in the measured velocities. Conversely, in the same rough terrain settings, but with the freedom of moving the arms, the robot successfully traversed the terrain without falling Figures 6.15d and 6.15h.

exhibited robustness not only within the training domain but also to variations outside it, including rough terrain locomotion (see Figure 6.7).

The only parameter that significantly affected the controller's performance was the communication delay, which was injected as described in Equation (6.2). However, we do not consider this to be a major problem as the maximum communication delay measured on the hardware was 10 [ms], while the 7 time-steps (at 50 [Hz]) used to randomize the delay itself amount to 14 [ms], roughly 40% more than the expected maximum communication delay.

6.6.1 Sim-To-Real Transfer

After having verified the correspondence between the performances required by the policy during locomotion in simulation and the capabilities of the hardware, the robustness to environment variations of the policy in different simulation environments, the controller was finally deployed on the real robot.

6.6.2 Adopting domain randomization

Despite the extensive domain randomization described in Table 6.2, successful tests verifying joint torque and velocity matching between hardware and software in

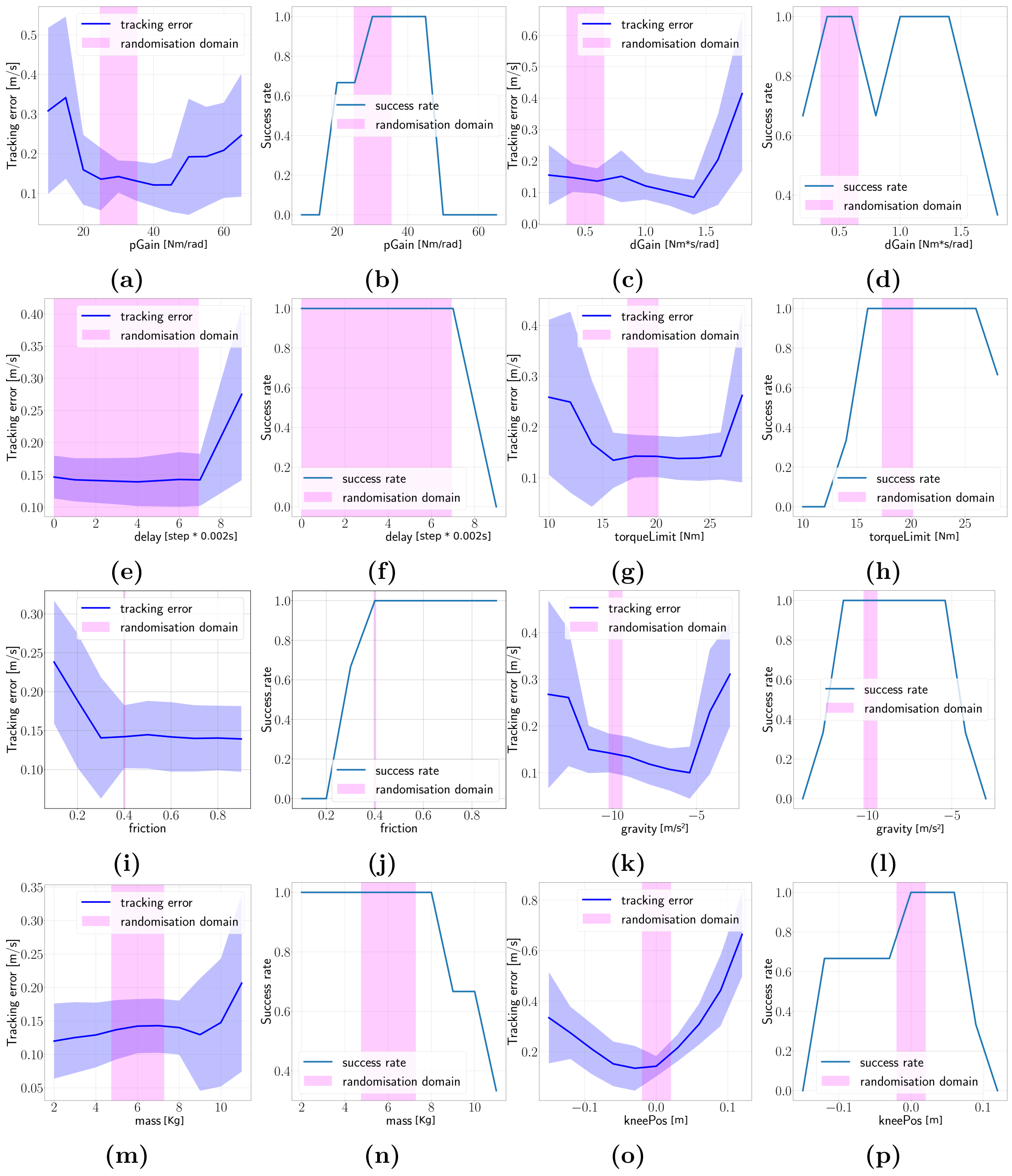


Figure 6.9: Success rate and tracking error under varioations of the environment.

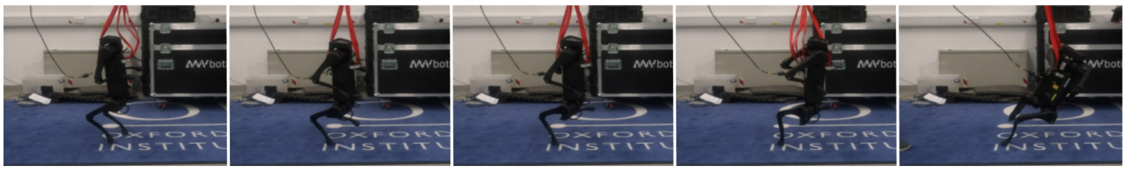


Figure 6.10: Deployment of the policy trained with domain randomization on the hardware resulted in the robot incorrectly placing its left hind leg, indicating a failure to recognize the current state and make the correct decision. The experiment lasted for a few seconds.

Section 6.5.1, and robustness demonstrated on rough terrain in Figure 6.8, the robot fell to the ground a few seconds after starting the controller, as illustrated in Figure 6.10. Upon analyzing the video, it became evident that the robot lifted its left hind foot but failed to step with it, resulting in a loss of balance and ultimately leading to a fall. This suggests a failure to recognize and understand the current state and make the correct decision.

Our hypothesis is that in a dynamic situation like walking on two legs with a quadruped robot, the many simplifications imposed by the simulator may have led to the robot's fall. For example, the feet are deformable and not rigid, the slings negatively affect the base's motion, and the motors might not repeatedly exert high impulsive torques because they overheat. Additionally, other sources of noise, such as domain randomization, could shift the distribution of state observed by the policy in simulation far from the true distribution experienced on the hardware.

6.6.3 Adopting ERFI

To simplify the study and reduce uncertainty in the controller, we decided to adopt ERFI-50 instead of domain randomization, which, based on our previous experience [2], can account for many disturbances on the real machine. This approach reduced the parameters to randomize to only $\tau^{lim}o_j$ and $\tau^{lim}r_j$.

Surprisingly, the new policy trained with ERFI-50 achieved eleven steps (consistently in several attempts) before falling, as shown in Figure 6.11.

We conducted a thorough investigation to determine the potential reasons for the discrepancy between simulation and hardware, such as verifying whether the IMU in a vertical configuration is not subjected to gimbal-lock and ensuring that the

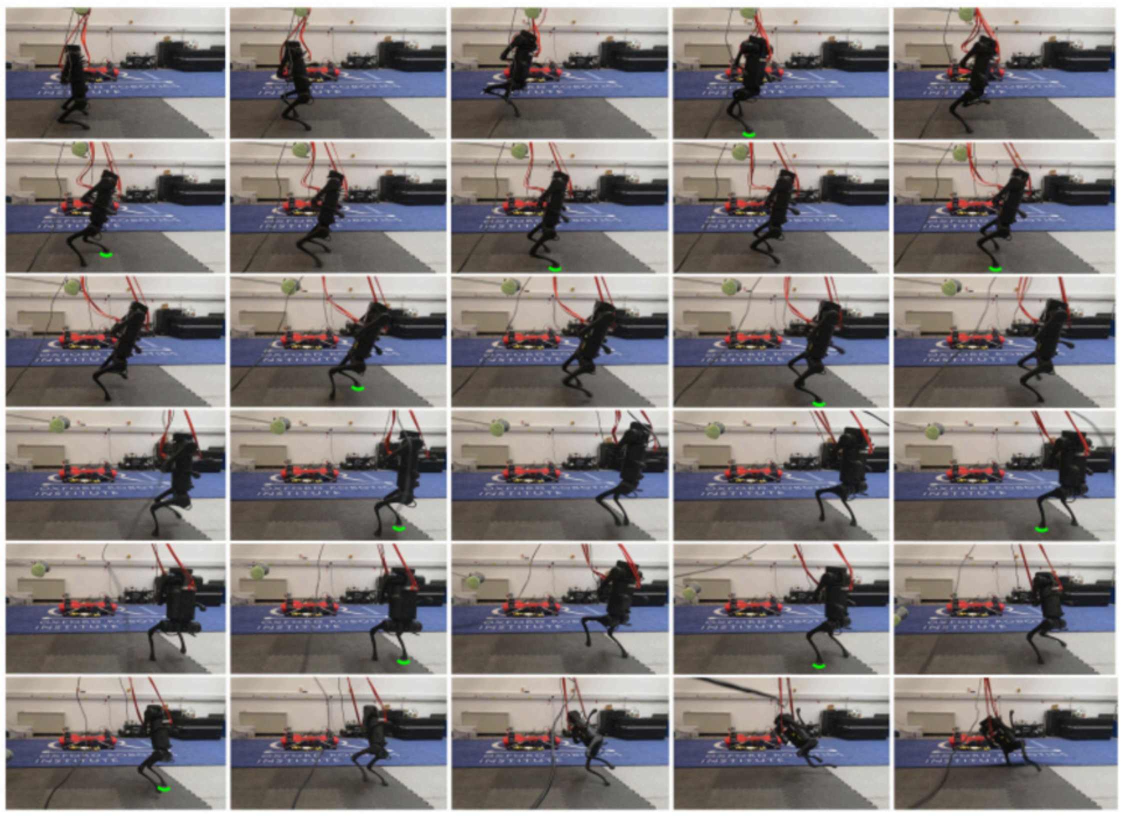


Figure 6.11: Bipedal policy deployed on the Unitree A1 robot, the green lines underneath the feet indicate the steps taken. Note that the slings are kept loose during the whole experiment.

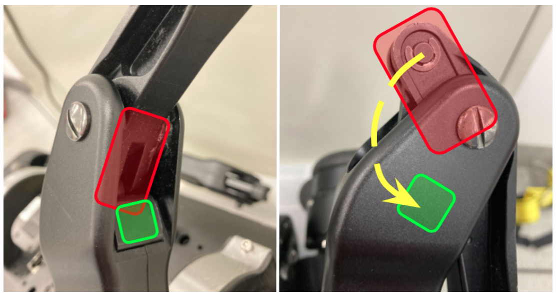
rotations are correctly measured. Based on our analysis, we found that everything was correctly represented and comparable with the simulator.

6.7 Discussion

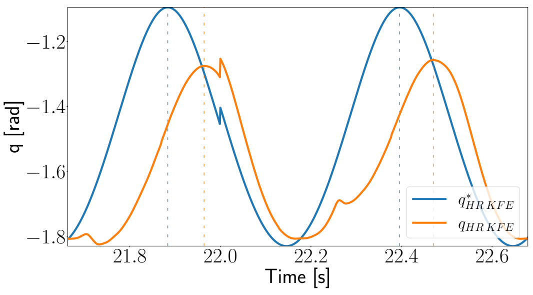
In this section we address further details that may have affected the deployment of the bipedal policy on the quadruped robot.

6.7.1 Hardware joint position limits

The Unitree A1 robot's $q_{FR\,KFE}$, $q_{FL\,KFE}$, $q_{HR\,KFE}$, and $q_{HL\,KFE}$ joints are limited by a physical stopper between [-2.7, -0.916] rad. This constraint is illustrated in Figure 6.12a and requires the nominal joint configuration q^N of the lower limb knees to be bent to ensure that the robot's motion adheres to the policy $q^* \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ without interfering with the stoppers. As a result, the knee joints are crouched

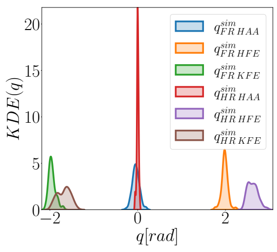


(a)

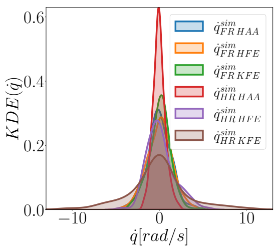


(b)

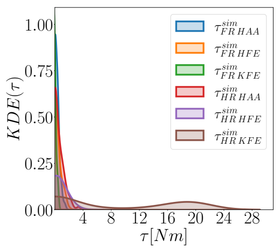
Figure 6.12: In Figure 6.12a Unitree A1's knee: the green block identifies the stopper, while the red one the articulation that could hit it during the motion. To avoid this physical constraint the final leg configuration is more bent, with the effect of generating higher torques during the motion. While in Figure 6.12b we show the time difference between desired and measured knee joint position, the delay observed amounts to roughly 10 [ms].



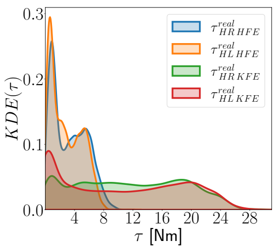
(a)



(b)



(c)



(d)

Figure 6.13: Most of the joints are involved in the locomotion, including the upper limbs Figure 6.13a. The $\dot{q}_j$ in Figure 6.13b are below the hardware limits ($21\,\mathrm{rad\,s^{-1}}$). The KDE of τ_j Figure 6.13c shows that the most loaded motors are the knees of the hind limbs, followed by the hips. While Figure 6.13d shows the KDE of τ_j of the hardware after running the squat controller 3 min, the highest torque that the system exerts before the performance degrades is 20 N m.

more than desired, leading to the knee motors exerting higher torque.

The distribution of joint positions, velocities, and torques during locomotion is shown in Figures 6.13a to 6.13c, while Figure 6.13d illustrates the distribution of torques observed on the physical machine.

6.7.2 Absence of abduction/adduction motors

In bipedal configuration, the abduction/adduction motion is absent, as demonstrated in Figure 6.14. The figure displays the current motor configuration of the robot, alongside a schematic representation. Additionally, the figure showcases the required ideal configuration of the motors, which necessitates an extra joint (*HAA 2*) to ensure successful abduction/adduction motion.

For quadruped robots on two legs, the standard twelve joint configuration presents significant difficulties in positioning the feet close to the projection of the CoM on the ground. This challenge is compounded when a reward system discouraging deviations from the nominal joint configuration is required.

To emphasize the significance of the *HAA 2* joint, it is worth noting that all robots specifically designed for walking on two legs, including Boston Dynamics Atlas, Cassie [105], Digit [25], MIT Humanoid [106], Hume & Mercury [107], and Talos [108], are equipped with it.

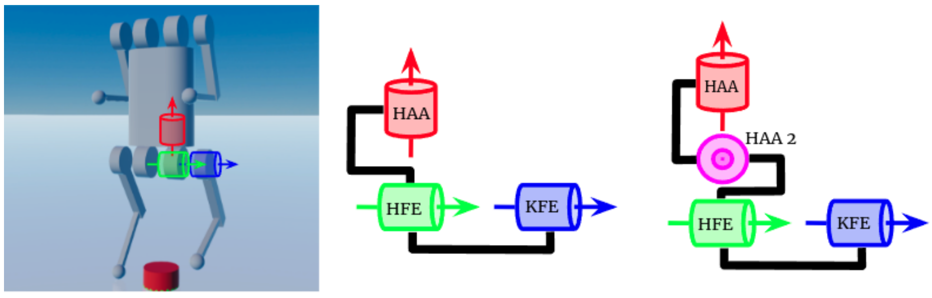


Figure 6.14: The figure on the left and in the centre show the current configuration of the motors in Unitree A1. This configuration does not allow for the abduction/adduction motion, instead on the right figure we show how adding a new *HAA 2* motor would allow these movements.

6.7.3 Limiting knee joint velocity

Bipedal robots without abduction/adduction joints face increased instability, as they have less time to maintain balance when a foot is off the ground. This is due to the foot's striking point being far from the projection of the robot's CoM on the ground. To compensate for this, the robot's stepping frequency increases, and this behavior is further amplified because the policy aims to minimize the time during which the feet are on the ground, thus reducing the total torque at the knees.

Hence, to enhance stability and to reduce the total torque expended over time, the policy applies high velocity (> 20 [rad/s]) to the knees for a fraction of a second. This improvement comes at a small cost of increasing the reward penalty for high joint velocity during a brief period of the episode. Ultimately, the only solution to prevent this behavior from the policy was to set hard limits on the maximum motor velocities (to 15 [rad/s]) via the simulator's clipping mechanism.

6.7.4 Communication delay

Using the data collected from the squat controller experiment, we observed a delay of approximately 10 ms, as presented in Figure 6.12b. To address this issue, we introduced a random actuation delay ranging from 0 to 0.014 sec.

To assess the policy's ability to handle this artificial delay, we examined its robustness, which is demonstrated in Figure 6.9f using the impedance controller equation in Equation (6.2).

6.7.5 Velocity command tracking

In this study, we examine the impact of domain randomization and environmental variations on a policy's ability to track desired linear and angular velocities, as shown in Figure 6.15. We compare the performance of a policy trained without any domain randomization (overfitting to the simulation environment), as depicted in Figures 6.15a and 6.15e, with a policy trained with domain randomization, as depicted in Figures 6.15b and 6.15f. We observed that the policy trained with domain randomization has a higher tracking error, potentially due to more cautious movements, since used to a less predictable and more challenging environment. This environment characterized by random changes in mass, knee joint position, PD gains, and other parameters listed in Table 6.2 after each training iteration.

Nevertheless, caution is extremely valuable in challenging conditions, as demonstrated when the experiments are carried out on rough terrain. When the policy trained without domain randomization is tested on such terrain, it constantly falls to the ground, as evidenced by the spikes in the measured velocities in Figures 6.15c and 6.15g. In contrast, the policy trained with domain randomization can successfully walk on such terrain without falling, as depicted in Figures 6.15d and 6.15h, and it demosntrates a tracking error on rough terrain comparable to that on flat ground.

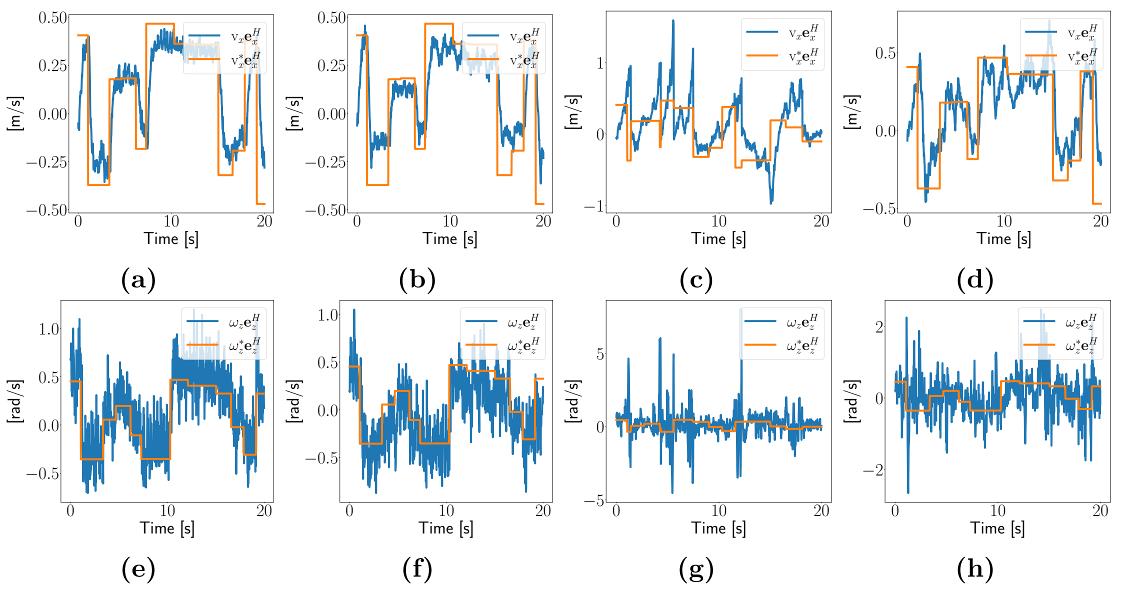


Figure 6.15: Comparing the base velocity tracking without domain randomization on flat ground (Figures 6.15a and 6.15e), with domain randomization on flat ground (Figures 6.15b and 6.15f), without domain randomization on uneven ground (Figures 6.15c and 6.15g), and finally with domain randomization on uneven ground (Figures 6.15d and 6.15h). Sudden spikes in the measured velocities signify that the robot fell on the ground, as in Figures 6.15c and 6.15g.

6.7.6 Long history in the observations

In this study, we employed an observation space similar to that used in [60], with the main difference being the length of the joint position error and velocity history. While [60] used a history encompassing three time steps (t_0, t_{-1}, and t_{-2}), we extended the history to include the past seven time steps (t_0, t_{-1}, t_{-2}, ..., t_{-6}). The joint history length was empirically determined in [60], and their conjecture was that it allowed for the detection of ground contact.

To investigate the role of different observations in the π network, we generated a saliency map by propagating gradients from the outputs to the inputs over several time steps. The saliency maps for $q_{HR\,HFE}$ and $q_{HL\,HFE}$ displayed a periodic vertical pattern, which was time-shifted between the legs. We were able to demonstrate that this pattern perfectly related to the contact state of each leg, as shown in Figure 6.16.

We also investigated the role of histories of joint positions error and velocities in locomotion using the saliency map in Figure 6.16. Specifically, the green and light-blue rectangles in the figure represent the joint error history and joint velocity history, respectively. The last rows of each rectangle, corresponding to the oldest observation in the joint history (t_{-6}), were frequently illuminated, indicating that the network heavily relies on information from the distant past to locomote.

Increasing the joint history length by one time step increases the input size by 12 and the number of weights by 3072 in the current MLP architecture. Hence, for future analyses, it would be interesting to investigate recurrent architectures and specifically how far back in the past they look.

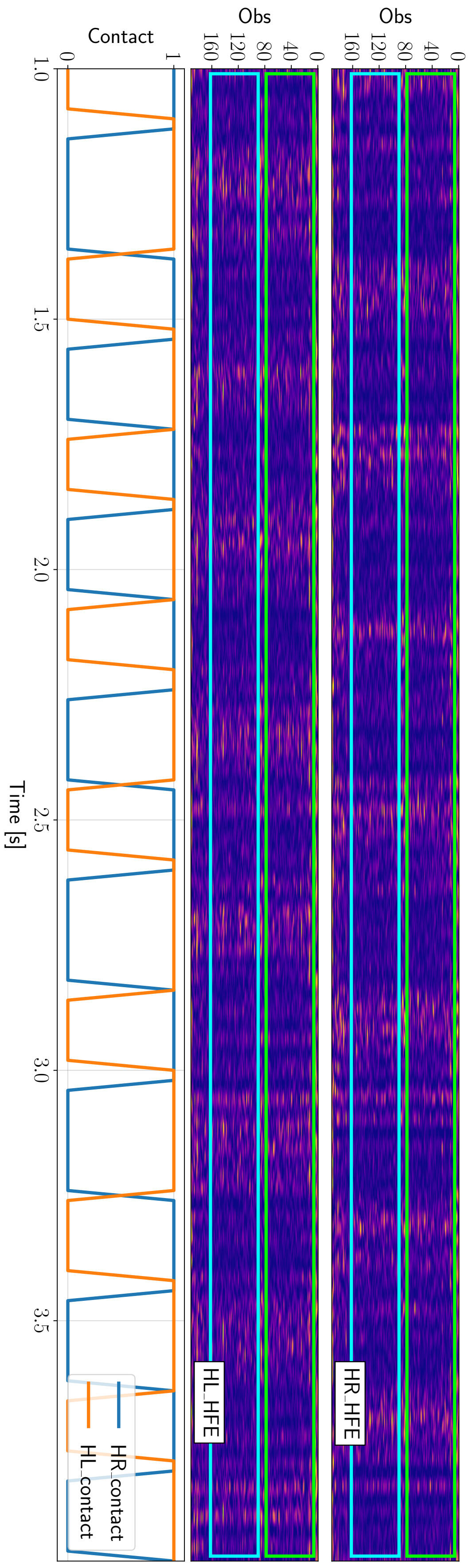


Figure 6.16: This saliency map shows a periodic pattern that corresponds to the contact state of the legs. Moreover, the green and light-blue rectangles represents the joint error history and the joint velocity history, parts of the observations. The last rows of each rectangle light up often, this is very important because they are the oldest memory of the joint history (t_{-6}), and this means that the network heavily rely on very past information to properly locomote.

7 Conclusion

Contents

7.1 Summary of contributions

Legged robots are highly versatile machines that offer superior mobility compared to other robotic systems. However, their complex controllers are difficult to design and this currently limits their capabilities. Classic methods require engineers to distill their knowledge into the controllers, which can be time-consuming and limiting when approaching dynamic tasks in unknown environments. Conversely, learning-based methods gather knowledge from data, and have the potential to unlock the versatility of legged systems. In this thesis, I followed the latter path to address several challenges in locomotion.

In the first paper [1] I proposed a novel approach for incorporating feedback into a fully differentiable CPG formulation adopting NN and applying Deep-RL to jointly learn the CPG parameters and MLP feedback. The outcome is the

CPG-Actor architecture, which enables end-to-end training of coupled CPG and an MLP for sensory feedback using Deep-RL. The main contributions of this paper are the fully differentiable approach and the novel method for capturing the recurrent state of the CPGs without unrolling them over time, which results in roughly twenty times better training performance compared to previous state-of-the-art methods. The paper provides insights into the impact of training on the distribution of parameters in both the CPGs and MLP feedback network, as well as qualitative results and insights into the baseline and proposed methods. The effectiveness of this approach was validated through simulation on an ANYmal B hopping leg. While the technique is versatile, its extension to diverse domains (e.g. full-body locomotion) is reserved for future research endeavors.

Due to the complexity of high-dimensional problems, such as locomotion, physics simulators are typically used for training Deep-RL control policies. However, there is often a significant gap between the simulated training domain and the physical target domain, known as the reality gap. To address this, I proposed a new method called ERFI [2], which randomizes only two parameters to allow the sim-to-real transfer of locomotion controllers. ERFI effectiveness is compared to its predecessor, RFI, variations of the same method, and standard domain randomization. Simulation experiments showed that ERFI improved success rates significantly for varying masses of the base and for attaching a manipulator arm to the robot during testing. Moreover, ERFI achieved competitive performance when compared to standard randomization techniques, while requiring tuning only a fraction of the parameters. The method's efficacy is further demonstrated by successfully deploying perceptive and blind policies on to the physical ANYmal C and Unitree A1 quadrupeds.

To overcome the sim-to-real gap when transferring controllers to the real world various strategies have been proposed, including identifying relevant properties of the robot and introducing sensory noise during training. In this context, ERFI handles system and actuation uncertainty demonstrating state-of-the-art sim-to-real performances with minimal parameters tuning. However, despite its effectiveness ERFI does not explicitly encompass modelling noise in observations. To solve this issue I proposed a new method called Roll-Drop. Roll-Drop introduces dropout during rollout, improving the robustness of Deep-RL policies to observation noise by only tuning a single parameter. Rool-Drop was evaluated on a Unitree A1 quadruped robot, demonstrating an 80% success rate when tested with up to 25% noise injected in the observations.

While quadruped robots provide higher mobility than wheeled vehicles, they are not as capable as their natural counterparts, which can use two-legged locomotion and fore limbs for manipulation. Bipedal locomotion on point feet with quadruped robots is a challenging scenario for current Deep-RL control architectures due to the limited number of joints and restricted kinematics per leg. To tackle this challenge, I proposed adopting model-free controllers to enable omni-directional bipedal locomotion on point feet with a quadruped robot without any hardware modification or external support. The study seeks to explore the capabilities and limitations of modern Deep-RL methods by developing a controller that is compliant with the quadruped's hardware requirements and capable of navigating challenging environments. Despite the limitations posed by the quadruped's hardware, the study considers this a perfect benchmark task to unlock new skills and future avenues for the legged robotics community.

7.1.1 Key takeaways

Robotics presents challenges, characterized by intricate debugging and time constraints. It's crucial to approach projects with the expectation that issues will arise.

When I embarked on the work described in Chapter 6, prior attempts at achieving bipedal locomotion on a quadruped robot with point feet had proven ineffective, and motivated by recent strides in Deep-RL applied to quadrupedal robots, I delved into adapting this approach for bipedal locomotion.

My initial assumption was that successfully developing a bipedal policy in simulation would seamlessly translate to a real-world implementation, something I had already accomplished with quadrupeds. In this context, Chapter 6 meticulously outlines the steps taken to achieve eleven successful bipedal steps, but the following speculations delve into the factors contributing to its partial success, potentially benefiting all those intrigued by this endeavor.

As evident in Figure 6.10, the policy lacks a coherent understanding of optimal stepping timing, resulting in the robot's instability. While transitioning from standard domain randomization to ERFI, as demonstrated in Figure 6.11, partially mitigated this issue, I believe that the policy's comprehension of the system state remains inadequate. This assertion finds validation in Figure 6.11, where it becomes evident that the robot continues accelerating forward instead of leaning back for decelerating.

As already mentioned, debugging was a challenge, further exacerbated by the minimal and sparse real-world data, which made comprehending the differences between simulation and reality exceedingly complex. In retrospect, a physical support structure could have facilitated repeatability and failure analysis – a resource that was regrettably absent.

Regarding the policy's architecture, it is my conviction that a more substantial history of states would have been beneficial. This could encompass states that had previously lacked a history, such as base rotations, or not present, such as base accelerations. The significance of memory becomes evident in Figure 6.16, illustrating the role of prior observations in determining current actions. A potential remedy might involve employing recurrent networks instead of conventional feed-forward ones. However, given the time constraints, pursuing all these avenues was unfeasible.

In conclusion, our work offers valuable insights for achieving bipedal locomotion on a quadruped robot with point feet. From addressing the simulation-reality gap with ERFI, to enhancing memory in architecture, our findings provide actionable directions. By applying these lessons, the broader robotics community can strive to replicate the success demonstrated in simulation on the hardware.

7.2 Considerations

Current perceptive locomotion controllers, whether model-based [109] or learning-based [65], rely on an elevation map of the surrounding environment. However, this method reduces the rich diversity of the world to a uniform "carpet" of varying smoothness, based on the adopted filters and the presence of artifacts like grass. This carpet lacks important information about friction, softness, density, and other properties of the ground, and the robot assumes it to be solid. As a result, locomotion controllers must rely heavily on reactive skills to detect and respond to unfavorable conditions.

Recent advances in learning-based methods powered by Deep-RL have shown superior robustness compared to classic approaches, but this robustness mainly stems from reactive skills rather than a deeper understanding of the environment. However, recent attempts to overcome the limitations of model-free Deep-RL, such as the use of world models to speed up training and improve planning in the policy [110], have shown promise. Nevertheless, this approach still requires a considerable amount of data, since it demonstrated poor gait performance.

Multitasking presents a challenge, as learning multiple skills with a single policy can be difficult. One approach that has shown success is combining multiple expert policies to address specific tasks [111], such as omni-directional locomotion and fall recovery. However, it's important to note that this method may become increasingly complex and computationally expensive as more skills are added, due to the need of duplicating expert policies.

Adversarial training has emerged as a highly effective approach to mitigate these problems, as demonstrated in [80]. Complex behaviors can be represented by comparing samples from the policy with those from experts using a discriminator.

Despite significant progress in the last five years, current robotics controllers are still largely limited to locomotion tasks, where a carpet representation of the world is sufficient to achieve some utility. However, these limitations restrict their applications, as legged robots are currently only used to carry laser-scanners or cameras and have little to no interactions with the environment.

7.2.1 Future Work

The potential impact of legged systems on people's daily lives is vast. To achieve this impact, it is essential to begin by understanding the complexity of the environment in which these systems will operate. This understanding requires abandoning the simplification of the environment's representation as a carpet, and instead representing it in its full complexity.

Representing the environment's complexity solely through images is challenging, as it would require a vast amount of data to build a comprehensive model. One promising approach is to construct a graph representation of the world, starting from the robot, as demonstrated in [112]. This approach provides a more detailed, compact, and understandable representation of reality.

In addition to the graph representation, it is crucial to define rewards in a convenient and unique manner. While AMP-based approaches are useful, they do not take full advantage of the vast corpus of online information, such as videos, which could greatly enhance the learning of new skills without the need for specific task controllers.

Another essential tool for achieving success are Large Language Models (LLMs), which, although currently auto-regressive, can be grounded in the visuo-physical experience of robots, Figure 7.1. By doing so, they can be instrumental in representing and recycling complex skills while also serving as a natural interface with humans.

In summary, to make robots intelligent and achieve the full potential of legged systems, it is crucial to represent the environment's complexity accurately, define rewards uniquely, utilize the vast corpus of online information, and take advantage of language to structure and comprehend the information consumed.

In my pursuit of making robots intelligent, I hope that the reflections above will serve as useful guidelines for success.

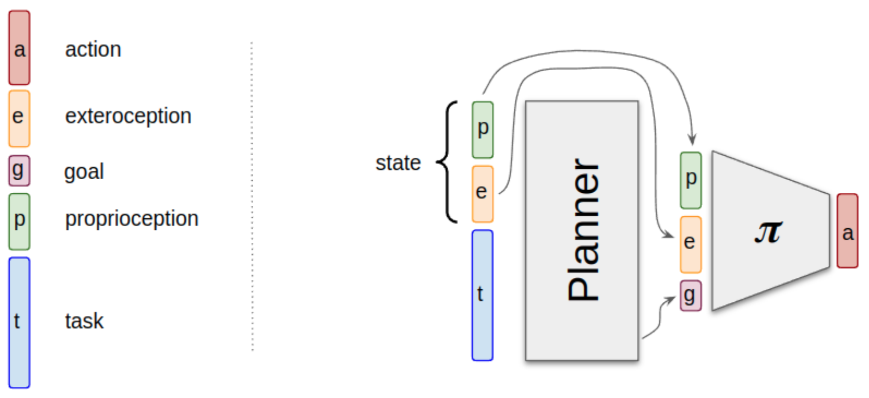


Figure 7.1: To recycle skills already learned and exploit the vast corpus of videos available online, the control architecture adopted in Chapter 4 needs to be expanded with planning capabilities. This planner would allow for abstract and long term reasoning. Moreover, since natural language understanding is required to interface with humans, it is likely that LLMs would be adopted for this task. This planner would take as inputs the state of the robot (proprioceptive and exteroceptive), the long term task to fulfill, and it would output short term goals that the policy π would be in charge of executing.

Appendices

A
Learning locomotion at low frequency

Contents

A.1 Overview of contribution

The use of legged systems enables agile motions in complex and unstructured environments. This requires control solutions that can recover from unexpected perturbations, adapt to system and environmental dynamics, and execute safe and reliable locomotion. Feedback-based control systems have been successful in achieving dynamic locomotion behaviors, but these systems require high-frequency actuation commands to minimize motion tracking errors and address external disturbances. In contrast, animals can demonstrate agile locomotion despite sensory noise and sensorimotor latencies associated with nerve conduction, electro-mechanical and force generation delays, which limit their motion control frequency.

In [113] the authors present a case study of a house cat that exhibits a low locomotion frequency, indicating that high-frequency feedback-based decision-making may not be critical for locomotion over challenging terrains. The authors investigate this discrepancy between biological and mechanical systems and propose a parallel compliant joint system and leg-length controller to realize actuation response similar to that of animal muscle-tendon units.

The ANYmal C quadruped is equipped with series elastic actuators SEAs that offer high compliance but trade-off controllability for compliance. In comparison, quasi-direct drives offer better actuation command tracking performance with lower control latencies, enabling highly dynamic locomotion. This work explores the question of whether robots can perform robust and dynamic locomotion at low motion control frequencies, inspired by the performance of animals.

Blind and perceptive control strategies for the ANYmal C quadruped are developed and evaluated for robust and dynamic locomotion over flat and uneven terrain. Related work includes model-free data-driven Deep-RL and model-based techniques that have been successful in achieving dynamic and complex locomotion. However, these approaches often employ finite-order motion parameterization, which inhibits the discovery of optimal behaviors. In contrast, motions executed by Deep-RL policies that map robot state information to desired joint states are not constrained by motion primitives, making Deep-RL particularly suitable for this work.

Overall, this work investigates the potential for robots to achieve agile and robust locomotion using low-frequency control strategies inspired by animals, and develops control solutions that enable safe and reliable locomotion in complex and unstructured environments.

A.2 Integrated manuscript

Learning Low-Frequency Motion Control for Robust and Dynamic Robot Locomotion

Siddhant Gangapurwala, Luigi Campanaro and Ioannis Havoutis

***Abstract*— Robotic locomotion is often approached with the goal of maximizing robustness and reactivity by increasing motion control frequency. We challenge this intuitive notion by demonstrating robust and dynamic locomotion with a learned motion controller executing at as low as $8\,\mathrm{Hz}$ on a real ANYmal C quadruped. The robot is able to robustly and repeatably achieve a high heading velocity of $1.5\,\mathrm{m\,s^{-1}}$, traverse uneven terrain, and resist unexpected external perturbations. We further present a comparative analysis of deep reinforcement learning (RL) based motion control policies trained and executed at frequencies ranging from $5\,\mathrm{Hz}$ to $200\,\mathrm{Hz}$. We show that low-frequency policies are less sensitive to actuation latencies and variations in system dynamics. This is to the extent that a successful sim-to-real transfer can be performed even without any dynamics randomization or actuation modeling. We support this claim through a set of rigorous empirical evaluations. Moreover, to assist reproducibility, we provide the training and deployment code along with an extended analysis at `https://ori-drs.github.io/lfmc/`.**

I. Introduction

Legged systems can execute agile motions by leveraging their ability to reach appropriate and disjoint support contacts, thereby enabling outstanding mobility in complex and unstructured environments. This, however, requires control solutions that are able to recover from unexpected perturbations, adapt to variations in system and environment dynamics, and execute safe and reliable locomotion. For feedback-based control systems, taking a corrective control action as soon as a sensory signal is detected allows for minimizing motion tracking errors while offering high reactivity to address external disturbances and modeling inaccuracies. This design motivation has been employed for achieving dynamic locomotion behaviors in [1], [2], [3] through generation of low-level actuation commands at frequencies ranging from 400 Hz to 1 kHz.

In contrast, animals are able to demonstrate remarkably agile locomotion in spite of sensory noise [4] and considerable sensorimotor latencies [5] associated with nerve conduction, electro-mechanical, and force generation delays [6] which limits their motion control frequency. The sensing and actuation delays for a medium-sized 20 kg dog, for example, can be approximately 58 ms of which 23.2 ms are required to process sensory feedback, generate an actuation signal, and deliver electro-mechanical commands [7]. The remaining delay corresponds to the ramp-up time for achieving peak muscle force. For a 40 kg animal, the total delay is estimated to be 67 ms with processing, generation and delivery latency of 30.4 ms.

The authors are with Dynamic Robots Systems (DRS) group, Oxford Robotics Institute, University of Oxford, UK. Email: {siddhant, luigi, ioannis}@robots.ox.ac.uk

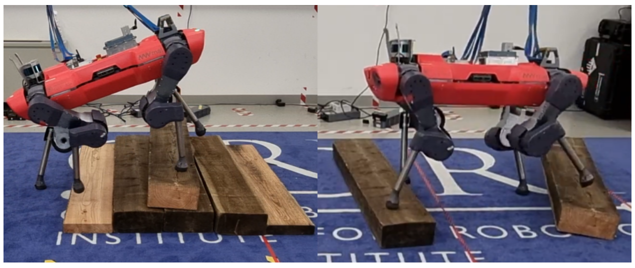


Fig. 1. ANYmal C quadruped walking over uneven terrain with a motion control frequency of 8 Hz.

In [8], *Ashtiani et al.* present an example in which a house cat exhibiting a locomotion frequency of 5 Hz [9] is sensor-blind for half its stance-phase. This duration corresponds to the entirety of the muscle force ramp-up time suggesting that high-frequency feedback-based decision-making may not be critical for locomotion over challenging terrains. *Ashtiani et al.* investigate this discrepancy between biological and mechanical systems and propose a parallel compliant joint system along with a leg-length controller to realize actuation response similar to that of animal muscle-tendon units. This is based on the motivation that elastic actuation allows for self-stability [10].

In this regard, the ANYmal C quadruped [11] houses series elastic actuators (SEAs) that offer high compliance making the system robust to impacts. SEAs, however, trade-off controllability for compliance [12]. In comparison, quasi-direct drives offer better actuation command tracking performance with lower control latencies enabling highly dynamic locomotion [13]. This makes it possible for the Mini Cheetah to sprint at $3.74\,\mathrm{m\,s^{-1}}$ [14] while for the ANYmal C, *Miki et al.* reported heading and lateral velocities of up to $1.2\,\mathrm{m\,s^{-1}}$ even with an extremely robust locomotion controller [15].

This work explores and presents our findings, alluding to a bio-inspired control design choice: *if animals can perform robust and dynamic locomotion at low motion control frequencies, can robots do so too?*

For this, we develop blind and perceptive control strategies for the ANYmal C quadruped and evaluate its performance for robust and dynamic locomotion over flat and uneven terrain as shown in Fig. 1.

A. Related Work

Model-free data-driven deep reinforcement learning (RL) enables obtaining control solutions that have the potential to thoroughly utilize the system capabilities of current robots. This property has been leveraged for learning agile and dynamic robotic locomotion skills to perform blind bipedal traversal over stairs [16], quadrupedal locomotion over challenging terrains [17] and even robust quadrupedal state recovery [18].

Model-based techniques have also demonstrated dynamic and complex locomotion [19], [20], [21]. A combination of model-based and model-free methods have also been proposed [22], [23], [24], [25]. These approaches, however, often employ finite-order motion parameterization which inhibits the discovery of optimal behaviors. In contrast, motions executed by RL policies that map robot state information to desired joint states are not constrained by motion primitives. This makes RL particularly suitable for our task of obtaining motion control policies operating at frequencies as low as 5 Hz. In such a case, the optimal behavior is not bounded by carefully tuned model-based controllers. Instead, the objective of finding the appropriate style to achieve dynamic and stable locomotion is addressed by the RL agent.

Despite the significant progress in RL for robotic locomotion, there remains an inconsistency in the design motivations for much of the proposed control architectures. In [26], *Hwangbo et al.* train an RL locomotion policy to map robot states to desired joint positions. This policy is queried at 200 Hz and the authors especially note that introducing a history of joint states into the RL state space is essential to obtain a locomotion policy. *Rudin et al.*, however, train a locomotion policy at 50 Hz without utilizing joint state history [27]. The obtained policy is transferable to the real platform even with access to only the current proprioceptive state. In [28], *Duan et al.* also show bipedal locomotion at 40 Hz without utilizing joint state history. We study these differences and observe that at higher motion control frequencies, the controller is more sensitive to the actuation dynamics compared to at lower-frequencies. This allows low-frequency policies to operate as motion planners as opposed to motion controllers. We detail upon our findings in Section IV of this manuscript.

While we discussed control solutions for dynamic and robust behavior, it is worth mentioning that a rich body of work also focuses on bio-inspired mechanical designs [29], [30], [31]. Although this is beyond the scope of our current work, we believe it serves as an important reminder that control intelligence and mechanical design complement each other [32].

B. Contribution

Our main contribution with this work is presenting that low-frequency motion control is sufficient to perform robust and dynamic locomotion. We further show that dynamics randomization or actuation modeling may not even be necessary for successful sim-to-real transfer. We additionally provide a comparative analysis of motion control policies trained and deployed at a range of frequencies. We believe this work will provide an important reference to the robotic control research community with regards to design motivations for developing custom control solutions.

We additionally highlight our contributions relating to sharing of training and deployment code, the low-frequency motion control (LFMC) framework. We provide a RaiSim [33] based optimized implementation for training locomotion policies for ANYmal C at various motion control frequencies. We additionally provide minimal deployment code in both C++ and Python. For the Python version, we also provide an option to choose between the RaiSim and PyBullet [34] simulation engines. We hope this encourages reproducibilty and allows colleagues to easily perform benchmarking against our approach.

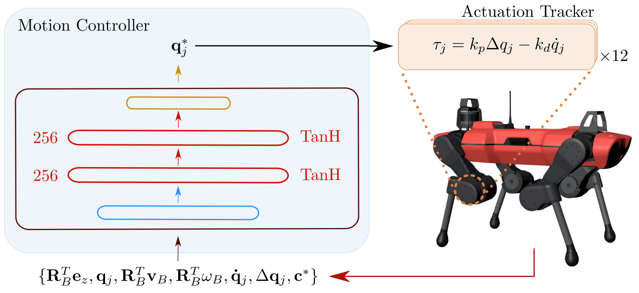

Motion Controller
$\mathbf{q}_j^*$
256
TanH
256
TanH
$\{\mathbf{R}_B^T\mathbf{e}_z, \mathbf{q}_j, \mathbf{R}_B^T\mathbf{v}_B, \mathbf{R}_B^T\omega_B, \dot{\mathbf{q}}_j, \Delta\mathbf{q}_j, \mathbf{c}^*\}$
Actuation Tracker
$\tau_j = k_p\Delta q_j - k_d\dot{q}_j$
×12

Fig. 2. Control architecture of our proprioceptive locomotion framework comprising a motion controller and an actuation tracker.

II. Preliminaries

A. System Model

We model a quadrupedal robot as a floating base B with four attached limbs. The robot state is measured and expressed in a global reference frame where the position is written as $\mathbf{r}_B \in \mathbb{R}^3$. The orientation is represented as the rotation matrix $\mathbf{R}_B \in SO(3)$. Each limb comprises three rotational joints. The angular joint positions are denoted by the vector $\mathbf{q}_j \in \mathbb{R}^{12}$. The linear and angular base velocities are represented as $\mathbf{v}_B \in \mathbb{R}^3$ and $\omega_B \in \mathbb{R}^3$ respectively.

The joint control torques $\boldsymbol{\tau}_j \in \mathbb{R}^{12}$ actuate the quadrupedal system and, in this work, are computed using the impedance control model,

$$\boldsymbol{\tau}_j = k_p \Delta\mathbf{q}_j - k_d \dot{\mathbf{q}}_j. \tag{1}$$

Here, k_p and k_d represent tracking gains and $\Delta\mathbf{q}_j = \mathbf{q}_j^* - \mathbf{q}_j$ where $\mathbf{q}_j^*$ denotes the desired joint positions.

B. Control Architecture

Our control architecture comprises a high-level *motion controller* and a low-level *actuation tracker*. This design is motivated by prior works on RL for robotic locomotion [26], [35], [36]. The motion controller, executed at a deployment frequency f_m, processes robot state information to generate desired joint states. The actuation tracker, executed at a frequency f_a where $f_a \geq f_m$, tracks these desired joint states by generating $\boldsymbol{\tau}_j$ using the model described in Eq. 1.

We model the motion controller policy as a multi-layer perceptron (MLP), π_θ. Here, θ represents the network parameters. The policy, $\pi_\theta : \mathbf{s} \mapsto \mathbf{a}$, maps the input state tuple

$\mathbf{s}$ to actions $\mathbf{a} \in \mathbb{R}^{12}$. The tuple $\mathbf{s}$ comprises observations that can be accessed on the real robot. Since we perform comparative analysis of different types of policies, the dimensionality of $\mathbf{s}$ depends on the individual motion control policy. We discuss this in the following subsection.

Each of the policies outputs an action tuple, $\mathbf{a} := \langle \mathbf{q}_j^* \rangle$, representing the desired joint positions and is based on the motivation that a low-impedance joint position control can offer improved training and control performance over torque control [37].

C. Motion Control Policies

We represent the motion control policies as π_θ where θ denotes the parameters of a generic motion controller. To refer to specific policies, we introduce the notation

$$\pi_{M:H}^{f_t}$$

where f_t is the motion control frequency at which the policy was trained, M is the mode of operation which can either be b (for blind) or p (for perceptive), and $H \in \mathbb{R}$ represents history length of joint states introduced in the state tuple $\mathbf{s}$. The joint state history is recorded at a frequency of f_j with a corresponding time step t_j. In this work, the joint state recording frequency $f_j \geq f_m$. In this work, we use $f_j = 200\,\mathrm{Hz}$ which we obtained empirically as part of [24].

As an example, $\pi_{b:2}^{8}$ represents a *blind* motion control policy trained at 8 Hz. The state space of $\pi_{b:2}^{8}$ also contains $\mathbf{q}_j$ and $\dot{\mathbf{q}}_j$ at joint recording steps $t_j - 1$ and $t_j - 2$, corresponding to a history length of 2.

For brevity, we omit f_t while referring to a class of motion control policies with the same operation mode and history length. For blind policies, $\pi_{b:0}$, with no joint state history, the state tuple $\mathbf{s}_{b:0} \in \mathbb{R}^{48}$ is defined as

$$\mathbf{s}_{b:0} := \langle \mathbf{R}_B^T \mathbf{e}_z, \mathbf{q}_j, \mathbf{R}_B^T \mathbf{v}_B, \mathbf{R}_B^T \omega_B, \dot{\mathbf{q}}_j, \Delta \mathbf{q}_j, \mathbf{c}^* \rangle,$$

where $\mathbf{e}_z = [0,0,1]^T$ represents the vertical z-axis and $\mathbf{c}^* \in \mathbb{R}^3$ comprises the desired heading velocity, lateral velocity and yaw rate commands represented in the base frame. The objective of the motion control policies is thus to track user-generated desired velocity commands.

The state space of perceptive policies $\pi_{p:0}$ is written as $\mathbf{s}_{p:0} \in \mathbb{R}^{235}$. $\mathbf{s}_{p:0}$ augments $\mathbf{s}_{b:0}$ with robo-centric terrain information $\mathbf{T} \in \mathbb{R}^{17 \times 11}$ observed between $[-0.8, 0.8]\,\mathrm{m}$ along the heading axis and $[-0.5, 0.5]\,\mathrm{m}$ along the lateral axis with a resolution of 0.1 m. The perceptive state space design is based on [27].

The joint state history augments the state space dimensionality by $H \times 24$. For a blind policy with history length of 4, $\pi_{b:4}$, its state tuple $\mathbf{s}_{b:4} \in \mathbb{R}^{144}$ is written as

$$\mathbf{s}_{b:4} := \langle \mathbf{s}_{b:0}, \mathbf{q}_{t_j-1}, \mathbf{q}_{t_j-2}, \mathbf{q}_{t_j-3}, \mathbf{q}_{t_j-4} \rangle.$$

Here, $\mathbf{q}_{t_j}$ represents the joint state tuple comprising joint positions and joint velocities recorded at time step t_j. The control architecture, including the dense neural network policy architecture, is illustrated in Fig. 2.

III. Methodology

A. Training

We represent the problem of legged locomotion as a sequential Markov decision process (MDP) [38] with continuous state and action spaces. With regards to RL, our goal is to obtain a policy, or a class of policies, that maximizes the expected cumulative discounted return,

$$J(\pi) \doteq \underset{\mathcal{T} \sim \pi_\theta}{\mathrm{E}} \left[\sum_{t=0}^{N} \gamma^t R \right], \tag{2}$$

where $\gamma \in [0,1)$ represents the discount factor and $\mathcal{T}$, dependent on π_θ, denotes a finite-horizon trajectory with episode length N. Our reward function, R, comprises several reward terms that allow for efficient and stable tracking of reference base velocity commands. We use the proximal policy optimization (PPO) [39] strategy to train each of our policies. Our training approach, including the reward function, has been derived from prior works [26], [35], [27].

While our method is quite standard, training several policies for different motion control frequencies requires tuning of individual reward terms and hyperparameters such as γ. For example, for an episodic length of 1 s, executing a policy at 200 Hz would imply collection of forty times more samples than for a 5 Hz policy. Consider another example. The half-life of γ can be given by,

$$n_{\gamma_{0.5}} = \frac{\log 0.5}{\log \gamma} \approx \frac{-0.3}{\log \gamma}. \tag{3}$$

For $\gamma = 0.98$, the half-life would correspond to 34 control steps. For a 200 Hz policy, this is equivalent to 0.17 s while for a 5 Hz policy, this represents a duration of 6.8 s.

To ensure consistency across different training frequencies, we denote the duration of $n_{\gamma_{0.5}}$ in seconds as opposed to control steps. For a training frequency f_t, the discount factor can then be computed by

$$\gamma = exp\left(\frac{\log 0.5}{f_t \times n_{\gamma_{0.5}}} \right). \tag{4}$$

In our training setup, we use $n_{\gamma_{0.5}} = 3\,\mathrm{s}$. For an episodic length $N = 1\,\mathrm{s}$, we ensure the batch size per policy iteration remains the same for every control frequency. For this, we perform parallel data collection wherein the number of parallel environments are scaled up to fit the desired batch size, $b_s = f_t \times n_{env}$. For $b_s = 48\mathrm{k}$, we use $n_{env} = 240$ for $f_t = 200\,\mathrm{Hz}$, and $n_{env} = 9600$ for $f_t = 5\,\mathrm{Hz}$.

To avoid retuning the reward function, we compute the returns at each simulation step, t_s as opposed to each control step t_m. Normally, $t_s \leq t_m$ and we use $t_s = 0.0025\,\mathrm{s}$ in this work. While this largely addresses exhaustive reward function tuning, we observed that reward terms representing deviation from nominal joint configuration and action smoothness needed to be slightly tuned for individual frequencies to achieve visually similar locomotion behavior. We provide the different training configurations on the project website[1].

[1]`https://ori-drs.github.io/lfmc`

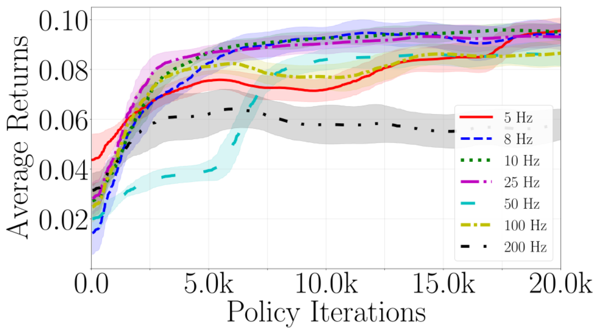


Fig. 3. Average returns for each of the trained policies $\pi_{b:0}$.

We train each of the $\pi_{b:0}$ policies for 20 k iterations. The iteration time is dependent on f_t and varies between 0.4 s (for $f_t = 25\,\text{Hz}$ and $f_t = 50\,\text{Hz}$) to 1.5 s (for $f_t = 5\,\text{Hz}$ and $f_t = 200\,\text{Hz}$). on a standard desktop computer housing an 8-core 3.6 GHz Intel i9-9900K and an Nvidia RTX 2080Ti. The returns plot for each of the trained policies is shown in Fig. 3. The policies trained at low-frequencies (8 Hz, 10 Hz and 25 Hz) converge a lot faster (¡10k iterations) compared to high-frequency policies. Note, $\pi^5_{b:0}$ suffers from poor reactivity and is therefore harder to train.

Note that, we do not perform any dynamics randomization (DR) while training the blind policies. Although we do use an actuator network [26] to model the real actuation dynamics, in Section IV, we show that introducing the actuator network during training may not even be necessary for LFMC.

B. Evaluation

We follow the narrative of bio-inspired low-frequency motion control (LFMC) and discuss the following key observations and reasoning in Section IV.

- LFMC policies are less sensitive to actuation dynamics under the assumption that actuation settling time [40] is less than control step time (Section IV-A).
- LFMC policies do not perform implicit modeling of system dynamics necessary for predictive control at high frequencies. Instead, LFMC policies can operate as motion planners (Section IV-B). To support this, we visualize the policy network Jacobians in Fig. 7.
- Since LFMC policies operate as motion planners, they show more robustness to variations in system dynamics. This is based on the assumption that the low-level actuation tracker stably and reliably tracks the motion plans. We show this to be the case in Fig. 9.
- LFMC policies are faster to train (Fig. 3).

To support these points, we evaluate the performance of each of the individual blind $\pi_{b:0}$ policies in RaiSim with unstructured rough terrain generated using Perlin noise [41] with maximum extrusion of 0.15 m. This is shown in Fig. 4. Our motivation for this setup is twofold: (1) the terrain noise introduces randomness allowing us to measure a probability distribution and (2) the unexpected perturbations highlight the reactivity of each of the policies.

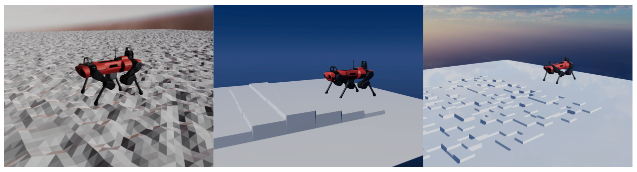


Fig. 4. RaiSim simulation set up for evaluation of blind and perceptive locomotion policies with ANYmal C traversing terrains comprising unstructured ground, stairs and bricks.

We introduce success rate (SR) as a performance metric defined as,

$$\text{SR} = 1 - \frac{N_e}{N_T} \tag{5}$$

where N_e refers to the number of episodic rollouts that were terminated early due to an invalid robot state and N_T represents the total number of rollouts. In this work, we use $N_T = 100$. For each rollout, we randomize the base heading direction. This randomization occurs with the same seed across each of the policies. An invalid robot state is defined by the criteria: (1) $\arccos\left(\mathbf{R}_{B_{3,3}}\right) > 0.4\pi$ which relates to base orientation, (2) self-collisions, or (3) collision of the robot base with ground.

We also train and compare $\pi^{10}_{b:4}$ and $\pi^{200}_{b:4}$ to show that joint state history is relevant for modeling system dynamics and is essential for high-frequency motion control as presented in [26]. This, however, is not the case for LFMC.

To demonstrate robustness on uneven terrain, we evaluate the performance of perceptive locomotion policies on terrains comprising rough ground, stairs and bricks as shown in Fig. 4. Our evaluation method for perceptive locomotion policies is based on the setup introduced in [24].

IV. Results

This section presents the key results in support of our contribution. We provide an extended analysis on the project website.

A. Intuitive Reasoning

Figure 5 (top) illustrates a toy example of a 1 DoF PD controller tracking sinusoidal set points updated at 5 Hz and 200 Hz for $k_p \in \{50, 65, 80, 95\}$ and $k_d = 2$. For the 5 Hz update frequency, the joint trajectories converge to very similar states before a new set point is generated. Note that, we use $k_p = 80$ and $k_d = 2.0$ for deploying our policies on to the real robot. For a 5 Hz controller, this implies the sensory readings at each update step are less effected by actuation tracking dynamics in comparison for higher update frequencies such as 200 Hz. This implies that, for an effective control behavior, the 200 Hz policy requires observability of the actuation dynamics.

We hypothesize that LFMC allows for operation as a planner and refer to it as *motion planning hypothesis*. This makes low-frequency motion controllers robust to actuation dynamics under the assumption that the low-level actuation controller stably tracks the generated joint states. Figure 5 (bottom) illustrates why the stable tracking is necessary. For

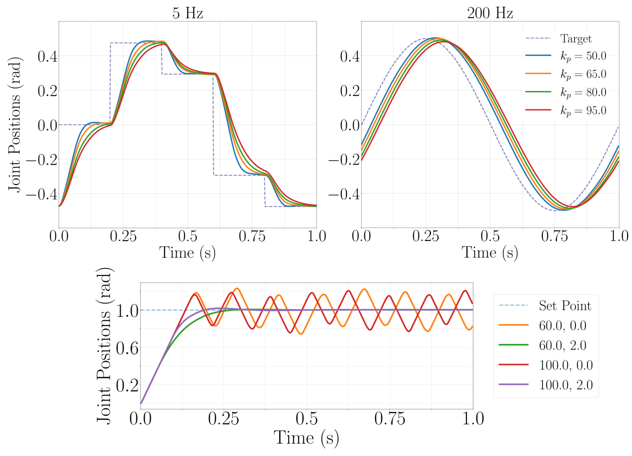


Fig. 5. *Top*: Tracking of sinusoidal set points updated at 5 Hz and 200 Hz for various position tracking gains. *Bottom*: Step responses observed for $k_p \in \{60, 100\}$ and $k_d \in \{0, 2\}$ for a series elastic actuator present on the ANYmal C quadruped.

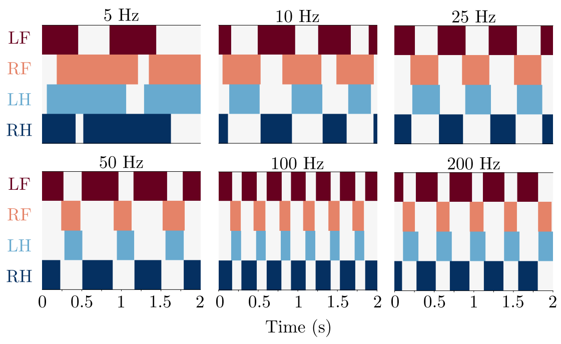


Fig. 6. Gait sequences for various $\pi_{b:0}$ motion control policies. The colored regions represent stance phase.

cases of under or over-damping, the motion controller may be required to adapt to the settling state even with low-frequency control.

B. Qualitative and Behavioral Analysis

We test the motion planning hypothesis by transferring the trained motion control policies on to the real ANYmal C quadruped. We observe extremely aggressive actuation tracking for $\pi^{200}_{b:0}$ resulting in vibrations at the rotary joints. This aggressive behavior is reduced with lower-frequency policies and no vibrations are recorded for $\pi^{25}_{b:0}$ and lower. We suspect that, since no dynamics randomization (DR) was performed during training, and due to imperfect actuation modeling, high-frequency policies overfit to the simulation domain affecting sim-to-real transfer. Note, while we are able to transfer $\pi^{5}_{b:0}$ onto the real robot, we are only able to stably execute low-velocity motions. The 5 Hz policy suffers from poor reactivity and is unable to execute recovery actions in unstable states.

We also observe interesting behavior with regards to the stance (foot-in-contact) and swing (foot-not-in-contact) phase duration. Low-frequency policies exhibit larger stance and swing phases compared to high-frequency policies (Fig. 6). We expected this to be an artifact of the scaling of action smoothness reward term (which penalizes large deviations between current and previous actions) with variations in motion control training frequencies. This, however, was not the case when we introduced joint state history ($N \geq 4$) into the state space. *Hwangbo et al.* hypothesized that the joint state history implicitly modeled contact detection [26]. While this has been consistent with our analysis of observing the absolute of policy Jacobians, $|d\pi_\theta(s)/ds|$, as presented in [17], we also observed that high-frequency control policies are more dependent on joint state history than low-frequency policies. We posit that joint state history improves the domain observability through implicit encoding of actuation dynamics [42] and is therefore more relevant for high-frequency policies.

We further investigate this for $\pi^{10}_{b:4}$ and $\pi^{200}_{b:4}$. Figure 7 illustrates the mean of the policy Jacobians recorded for a duration of 2 s. Dark blue regions suggest higher gradients, implying larger dependency. Compared to 10 Hz, the 200 Hz policy requires more observations to execute the same task relating to larger dependency on system dynamics. Interestingly, the gradients for joint velocities are quite low for $\pi^{10}_{b:4}$ while $\pi^{200}_{b:4}$ utilizes joint velocities more than joint positions.

For high-frequency $\pi_{b:0}$, we suspect the fast contact switching behavior occurs due to partial observability of the system dynamics. In our experience, we have observed this to be the case for poorly designed state spaces. This, however, needs further investigating. For the 200 Hz policy, the increase in stance and swing phase duration, compared to 100 Hz policy, is due to poor tracking with lower stability.

C. Robustness Analysis

One of our main objectives with this work is to demonstrate robustness with low-frequency motion control. Figure 9 shows that $\pi^{10}_{b:0}$ performs better than most of the

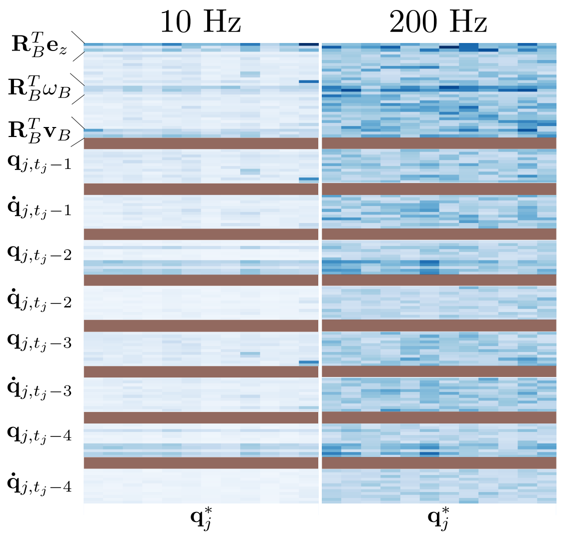


Fig. 7. Visualization of the mean of network Jacobians recorded for $\pi^{10}_{b:4}$ and $\pi^{200}_{b:4}$ for 2 s. Dark blue regions correspond to high gradients whereas white corresponds to zero gradients. The brown regions separate different observations and have only been included for visual aid. Note that, joint state history is sampled at 200 Hz for both the policies. Sampling joint state history at 10 Hz for $\pi^{10}_{b:4}$ resulted in near-zero gradients for history terms.

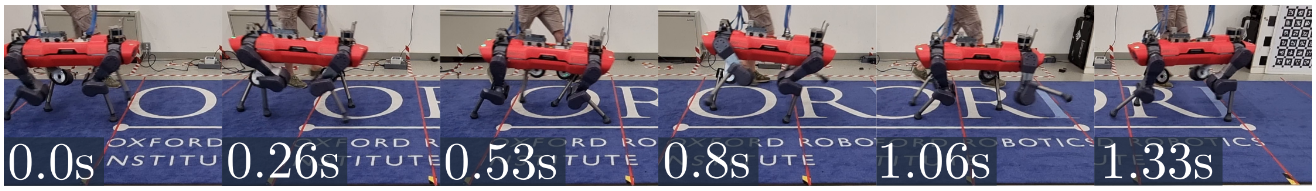


Fig. 8. Motion control policy trained and deployed at 8 Hz stably tracking heading base velocity of $1.5\,\mathrm{m\,s^{-1}}$.

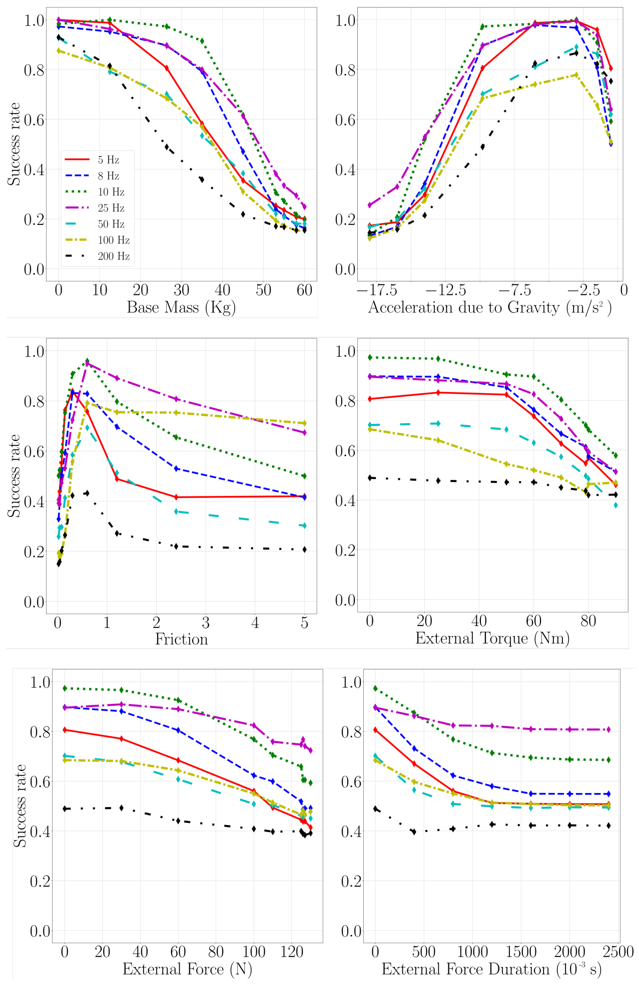


Fig. 9. Success rate observed for various motion control policies, $\pi_{b:0}$, for different perturbations and dynamics parameters.

policies. $\pi^{25}_{b:0}$ offers both high robustness and reactivity whereas $\pi^{8}_{b:0}$ falls behind $\pi^{10}_{b:0}$ due to inadequate reactivity for traversal over rough terrain. We also investigate robustness to actuation latencies and show that LFMC policies offer higher robustness than high-frequency policies (Table I).

D. Dynamic Locomotion

Based on our robustness analysis, we perform qualitative evaluation on the real robot. We demonstrate high-speed dynamic locomotion with $\pi^{8}_{b:0}$. As shown in Fig. 8, we are able to achieve a heading velocity of approximately $1.5\,\mathrm{m\,s^{-1}}$ traversing a distance of 2 m in roughly 1.33 s.

We also trained a perceptive locomotion policy $\pi^{8}_{p:0}$ based on [27] for locomotion over uneven terrain. We show that 8 Hz motion control is sufficient for robust traversal over considerable obstacles (wooden railway sleepers) and steps (both up and down) as presented in Fig. 1.

TABLE I

MAXIMUM ACTUATION DELAY THAT $\pi_{b:0}$ POLICIES CAN BE ROBUST TO RIGHT BEFORE FAILURE MEASURED AT A RESOLUTION OF 5 ms.

Training Frequency (Hz)	**5**	**10**	**25**	**50**	**100**	**200**
Latency (ms)	90	90	65	50	30	20

We further compare the behavior of policies trained with and without DR. The DR parameters are based on [35] and have been provided on the project website. This is shown for 10 Hz and 200 Hz perceptive policies in Table II. As expected, DR allows for better performance over uneven terrain. Introduction of joint state history is not as effective as doing both, introducing joint state history and DR. Joint state history and DR allow for better observabiliy of environment interactions, necessary for uneven terrains [17], while also encouraging generalizability to unseen domains.

We are also able to demonstrate transfer on to the physical system with policies trained without the actuator network. The behavior is stable, however, not as smooth as policies trained with the actuator network. We detail upon the extended analysis on the project website and summarize our evaluation in the overview video.

TABLE II

SUCCESS RATES OF 10 Hz AND 200 Hz PERCEPTIVE POLICIES MEASURED FOR 100 RUNS EACH.

	10 Hz				200 Hz			
	$\pi_{p:0}$	$\pi_{p:0}$ (DR)	$\pi_{p:4}$	$\pi_{p:4}$ (DR)	$\pi_{p:0}$	$\pi_{p:0}$ (DR)	$\pi_{p:4}$	$\pi_{p:4}$ (DR)
Rough	0.94	0.94	0.94	0.95	0.87	0.92	0.93	0.94
Stairs	0.86	0.93	0.92	0.94	0.52	0.59	0.88	0.95
Bricks	0.66	0.71	0.69	0.80	0.58	0.62	0.64	0.76

V. CONCLUSION

This work aims to start the discussion within the robotics community about the role and benefit of high- versus low-frequency motion control, especially within the context of learning-based approaches. From biological studies we know that animals can perform robust and dynamic locomotion at low motion control frequencies and, with this work, we showed how robots can achieve this too.

We demonstrated dynamic and robust quadrupedal locomotion with as low as 8 Hz of motion control frequency. We further provided empirical evaluations to support our claim that motion control policies trained at low-frequencies do not require dynamics randomization or actuation modeling to perform a successful sim-to-real transfer.

References

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[2] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, July 2018.

[3] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *Conference on Robot Learning*. PMLR, 2022, pp. 773–783.

[4] A. A. Faisal, L. P. Selen, and D. M. Wolpert, "Noise in the nervous system," *Nature reviews neuroscience*, vol. 9, no. 4, pp. 292–303, 2008.

[5] H. L. More, J. R. Hutchinson, D. F. Collins, D. J. Weber, S. K. Aung, and J. M. Donelan, "Scaling of sensorimotor control in terrestrial mammals," *Proceedings of the Royal Society B: Biological Sciences*, vol. 277, no. 1700, pp. 3563–3568, 2010.

[6] H. L. More, S. M. O'Connor, E. Brøndum, T. Wang, M. F. Bertelsen, C. Grøndahl, K. Kastberg, A. Hørlyck, J. Funder, and J. M. Donelan, "Sensorimotor responsiveness and resolution in the giraffe," *Journal of Experimental Biology*, vol. 216, no. 6, pp. 1003–1011, 2013.

[7] H. L. More and J. M. Donelan, "Scaling of sensorimotor delays in terrestrial mammals," *Proceedings of the Royal Society B*, vol. 285, no. 1885, p. 20180613, 2018.

[8] M. S. Ashtiani, A. Aghamaleki Sarvestani, and A. Badri-Spröwitz, "Hybrid parallel compliance allows robots to operate with sensorimotor delays and low control frequencies," *Frontiers in Robotics and AI*, p. 170, 2021.

[9] J. E. Bertram, A. Gutmann, J. Randev, and M. Hulliger, "Domestic cat walking parallels human constrained optimization: optimization strategies and the comparison of normal and sensory deficient individuals," *Human Movement Science*, vol. 36, pp. 154–166, 2014.

[10] R. Blickhan, A. Seyfarth, H. Geyer, S. Grimmer, H. Wagner, and M. Günther, "Intelligence by mechanics," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 199–220, 2007.

[11] E. Ackerman, "Anybotics introduces sleek new anymal c quadruped," *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/anybotics-introduces-sleek-new-anymal-c-quadruped*, 2019.

[12] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1. IEEE, 1995, pp. 399–406.

[13] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[14] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[15] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[16] 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.

[17] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

[18] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, p. eabb2174, 2020.

[19] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4730–4737.

[20] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.

[21] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2464–2470.

[22] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[23] Z. Xie, G. Berseth, P. Clary, J. W. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," *CoRR*, vol. abs/1803.05580, 2018. [Online]. Available: http://arxiv.org/abs/1803.05580

[24] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," 2020.

[25] ——, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, 2021.

[26] 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*, vol. 4, no. 26, p. eaau5872, 2019.

[27] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[28] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1276–1282.

[29] A. A. Saputra, N. Takesue, K. Wada, A. J. Ijspeert, and N. Kubota, "Aquro: A cat-like adaptive quadruped robot with novel bio-inspired capabilities," *Frontiers in Robotics and AI*, vol. 8, p. 562524, 2021.

[30] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[31] S. Coyle, C. Majidi, P. LeDuc, and K. J. Hsia, "Bio-inspired soft robotics: Material selection, actuation, and design," *Extreme Mechanics Letters*, vol. 22, pp. 51–59, 2018.

[32] R. Pfeifer, M. Lungarella, and F. Iida, "The challenges ahead for bio-inspired'soft'robotics," *Communications of the ACM*, vol. 55, no. 11, pp. 76–87, 2012.

[33] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[34] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[35] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[36] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4955–4961.

[37] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[38] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[40] S. N. Vukosavic, *Electrical machines*. Springer Science & Business Media, 2012.

[41] K. Perlin, "Improving noise," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 681–682.

[42] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Learning Low-Frequency Motion Control for Robust and Dynamic Robot Locomotion
Publication Status	Published
Publication Details	Gangapurwala S., Campanaro L., Havoutis I., Learning Low-Frequency Motion Control for Robust and Dynamic Robot Locomotion, 2023

Student Confirmation

Student Name:			
Contribution to the Paper	Analyzed the performance of the approach proposed, compared it against other methods, revised the manuscript, and ran experiments on the robot.		
Signature Luigi Campanaro		Date	11/04/2023

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Dr. Ioannis Havoutis		
Supervisor comments The contributions are as above		
Signature Ioannis Havoutis	Date	19/04/2023

This completed form should be included in the thesis, at the end of the relevant chapter.

References

[1] Luigi Campanaro et al. “CPG-Actor: Reinforcement Learning For Central Pattern Generators”. In: *Towards Autonomous Robotic Systems: 22nd Annual Conference, TAROS 2021, Lincoln, UK, September 8–10, 2021, Proceedings*. Lincoln, United Kingdom: Springer-Verlag, 2021, pp. 25–35. URL: `https://doi.org/10.1007/978-3-030-89177-0_3`.

[2] Luigi Campanaro et al. *Learning and Deploying Robust Locomotion Policies with Minimal Dynamics Randomization*. 2022. URL: `https://arxiv.org/abs/2209.12878`.

[3] Luigi Campanaro et al. “Roll-Drop: accounting for observation noise with a single parameter”. In: *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. Ed. by Nikolai Matni, Manfred Morari, and George J. Pappas. Vol. 211. Proceedings of Machine Learning Research. PMLR, 15–16 Jun 2023, pp. 718–730. URL: `https://proceedings.mlr.press/v211/campanaro23a.html`.

[4] Marc H. Raibert, Jr H. Benjamin Brown, and Michael Chepponis. “Experiments in Balance with a 3D One-Legged Hopping Machine”. In: *The International Journal of Robotics Research* 3.2 (1984), pp. 75–92. eprint: `https://doi.org/10.1177/027836498400300207`. URL: `https://doi.org/10.1177/027836498400300207`.

[5] R.R. Playter and M.H. Raibert. “Control Of A Biped Somersault In 3D”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 1992, pp. 582–589.

[6] M. Raibert, M. Chepponis, and H. Brown. “Running on four legs as though they were one”. In: *IEEE Journal on Robotics and Automation* 2.2 (1986), pp. 70–82.

[7] Y. Sakagami et al. “The intelligent ASIMO: system overview and integration”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. 2002, 2478–2483 vol.3.

[8] Gabe Nelson et al. “PETMAN: A Humanoid Robot for Testing Chemical Protective Clothing”. In: *Journal of the Robotics Society of Japan* 30.4 (2012), pp. 372–377.

[9] C Semini et al. “Design of HyQ – a hydraulically and electrically actuated quadruped robot”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225.6 (2011), pp. 831–849. eprint: `https://doi.org/10.1177/0959651811402275`. URL: `https://doi.org/10.1177/0959651811402275`.

[10] Claudio Semini et al. “Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max”. In: *IEEE/ASME Transactions on Mechatronics* 22.2 (2017), pp. 635–646.

[11] MARCO HUTTER et al. “STARLETH: A COMPLIANT QUADRUPEDAL ROBOT FOR FAST, EFFICIENT, AND VERSATILE LOCOMOTION”. In: *Adaptive Mobile Robotics*, pp. 483–490. URL: `https://www.worldscientific.com/doi/abs/10.1142/9789814415958_0062`.

[12] Christian Gehring et al. “Control of dynamic gaits for a quadrupedal robot”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 3287–3292.

[13] Christian Gehring et al. “Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot”. In: *IEEE Robotics & Automation Magazine* 23.1 (2016), pp. 34–43.

[14] Marco Hutter et al. “ANYmal - a highly mobile and dynamic quadrupedal robot”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 38–44.

[15] Marco Hutter et al. “Towards a Generic Solution for Inspection of Industrial Sites”. In: *Field and Service Robotics*. Ed. by Marco Hutter and Roland Siegwart. Cham: Springer International Publishing, 2018, pp. 575–589.

[16] Marco Tranzatto et al. “CERBERUS in the DARPA Subterranean Challenge”. In: *Science Robotics* 7.66 (2022), eabp9742. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.abp9742`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.abp9742`.

[17] Sangok Seok et al. “Design principles for highly efficient quadrupeds and implementation on the MIT Cheetah robot”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 3307–3312.

[18] Hae-Won Park, Patrick Wensing, and Sangbae Kim. “Online Planning for Autonomous Running Jumps Over Obstacles in High-Speed Quadrupeds”. In: *Proceedings of Robotics: Science and Systems*. Rome, Italy, July 2015.

[19] Hae-Won Park, Sangin Park, and Sangbae Kim. “Variable-speed quadrupedal bounding using impulse planning: Untethered high-speed 3D Running of MIT Cheetah 2”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 5163–5170.

[20] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. “Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6295–6301.

[21] Gerardo Bledt and Sangbae Kim. “Extracting Legged Locomotion Heuristics with Regularized Predictive Control”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 406–412.

[22] Christian Hubicki et al. “ATRIAS: Design and validation of a tether-free 3D-capable spring-mass bipedal robot”. In: *The International Journal of Robotics Research* 35.12 (2016), pp. 1497–1521. eprint: `https://doi.org/10.1177/0278364916648388`. URL: `https://doi.org/10.1177/0278364916648388`.

[23] Jonah Siekmann et al. "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning". In: *CoRR* abs/2105.08328 (2021). arXiv: `2105.08328`. URL: `https://arxiv.org/abs/2105.08328`.

[24] Jonah Siekmann et al. "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 7309–7315.

[25] Guillermo A. Castillo et al. "Robust Feedback Motion Policy Design Using Reinforcement Learning on a 3D Digit Bipedal Robot". In: *CoRR* abs/2103.15309 (2021). arXiv: `2103.15309`. URL: `https://arxiv.org/abs/2103.15309`.

[26] C. Dario Bellicoso et al. "Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2261–2268.

[27] Alexander W. Winkler et al. "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1560–1567.

[28] David Surovik et al. "Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles". In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 16–18 Nov 2021, pp. 1509–1518. URL: `https://proceedings.mlr.press/v155/surovik21a.html`.

[29] Marc H. Raibert and Ernest R. Tello. "Legged Robots That Balance". In: *IEEE Expert* (1986).

[30] S. Kajita et al. "Biped walking pattern generation by using preview control of zero-moment point". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. Sept. 2003, 1620–1626 vol.2.

[31] Mrinal Kalakrishnan et al. "Fast, robust quadruped locomotion over challenging terrain". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2665–2670.

[32] Dong Jin Hyun et al. "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT Cheetah". In: *The International Journal of Robotics Research* 33.11 (2014), pp. 1417–1445. eprint: `https://doi.org/10.1177/0278364914532150`. URL: `https://doi.org/10.1177/0278364914532150`.

[33] Victor Barasuol et al. "A reactive controller framework for quadrupedal locomotion on challenging terrain". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 2554–2561.

[34] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[35] Yuval Tassa, Tom Erez, and Emanuel Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 4906–4913.

[36] Michael Neunert et al. “Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds”. In: *CoRR* abs/1607.04537 (2016). arXiv: `1607.04537`. URL: `http://arxiv.org/abs/1607.04537`.

[37] Farbod Farshidian et al. “An efficient optimal planning and control framework for quadrupedal locomotion”. In: IEEE, May 2017. URL: `https://doi.org/10.1109%2Ficra.2017.7989016`.

[38] Oliwier Melon et al. “Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9805–9811.

[39] Carlos Mastalli et al. *Agile Maneuvers in Legged Robots: a Predictive Control Approach*. 2022. arXiv: `2203.07554 [cs.RO]`.

[40] Auke Jan Ijspeert. “Central pattern generators for locomotion control in animals and robots: A review”. In: *Neural Networks* 21.4 (2008), pp. 642–653.

[41] J. Yu et al. “A Survey on CPG-Inspired Control Models and System Implementation”. In: *IEEE Transactions on Neural Networks and Learning Systems* 25.3 (Mar. 2014), pp. 441–456.

[42] Auke Jan Ijspeert et al. “From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model”. In: *Science* 315.5817 (2007), pp. 1416–1420.

[43] Ludovic Righetti and Auke Jan Ijspeert. “Pattern generators with sensory feedback for the control of quadruped locomotion”. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 819–824.

[44] Mostafa Ajallooeian et al. “Modular Control of Limit Cycle Locomotion over Unperceived Rough Terrain”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013*. Tokyo, 2013, pp. 3390–3397.

[45] A. Kamimura et al. “Automatic locomotion pattern generation for modular robots”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. Sept. 2003, 714–720 vol.1.

[46] Stéphane Bonardi et al. “Automatic Generation of Reduced CPG Control Networks for Locomotion of Arbitrary Modular Robot Structures”. In: *Proceedings of Robotics: Science and Systems* (2014).

[47] Alexander Spröwitz et al. “Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot”. In: 32.8 (2013), pp. 932–950. eprint: `https://doi.org/10.1177/0278364913489205`. URL: `https://doi.org/10.1177/0278364913489205`.

[48] M. Ajallooeian et al. “Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 3321–3328.

[49] Sébastien Gay, José Santos-Victor, and Auke Ijspeert. “Learning robot gait stability using neural networks as sensory feedback function for Central Pattern Generators”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 194–201.

[50] Dai Owaki and Akio Ishiguro. "A Quadruped Robot Exhibiting Spontaneous Gait Transitions from Walking to Trotting to Galloping". In: *Scientific Reports* 7.277 (2017).

[51] Takahiro Fukui et al. "Autonomous gait transition and galloping over unperceived obstacles of a quadruped robot with CPG modulated by vestibular feedback". In: *Robotics and Autonomous Systems* 111 (2019), pp. 1–19.

[52] Richard S. Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Denver, CO: MIT Press, 1999, pp. 1057–1063.

[53] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: `http://arxiv.org/abs/1509.02971`.

[54] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.

[55] David Silver et al. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 387–395. URL: `https://proceedings.mlr.press/v32/silver14.html`.

[56] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: `1312.5602`. URL: `http://arxiv.org/abs/1312.5602`.

[57] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *CoRR* abs/1712.01815 (2017). arXiv: `1712.01815`. URL: `http://arxiv.org/abs/1712.01815`.

[58] Christopher Berner et al. "Dota 2 with Large Scale Deep Reinforcement Learning". In: *CoRR* abs/1912.06680 (2019). arXiv: `1912.06680`. URL: `http://arxiv.org/abs/1912.06680`.

[59] OpenAI et al. "Solving Rubik's Cube with a Robot Hand". In: *CoRR* abs/1910.07113 (2019). arXiv: `1910.07113`. URL: `http://arxiv.org/abs/1910.07113`.

[60] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (2019), eaau5872. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.aau5872`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.aau5872`.

[61] Joonho Lee, Jemin Hwangbo, and Marco Hutter. "Robust Recovery Controller for a Quadrupedal Robot using Deep Reinforcement Learning". In: *CoRR* abs/1901.07517 (2019). arXiv: `1901.07517`. URL: `http://arxiv.org/abs/1901.07517`.

[62] Joonho Lee et al. “Learning quadrupedal locomotion over challenging terrain”. In: *Science Robotics* 5.47 (2020), eabc5986. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.abc5986`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.abc5986`.

[63] Chuanyu Yang et al. “Multi-expert learning of adaptive legged locomotion”. In: *Science Robotics* 5.49 (2020), eabb2174.

[64] Nikita Rudin et al. “Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning”. In: *Proceedings of the 5th Conference on Robot Learning.* Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, Aug. 2022, pp. 91–100. URL: `https://proceedings.mlr.press/v164/rudin22a.html`.

[65] Takahiro Miki et al. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science Robotics* 7.62 (2022), eabk2822. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.abk2822`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.abk2822`.

[66] Siddhant Gangapurwala, Alexander Mitchell, and Ioannis Havoutis. *Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion.* 2020. URL: `https://arxiv.org/abs/2002.09676`.

[67] Xue Bin Peng et al. “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization”. In: *CoRR* abs/1710.06537 (2017). arXiv: `1710.06537`. URL: `http://arxiv.org/abs/1710.06537`.

[68] OpenAI et al. “Learning Dexterous In-Hand Manipulation”. In: *CoRR* abs/1808.00177 (2018). arXiv: `1808.00177`. URL: `http://arxiv.org/abs/1808.00177`.

[69] Siddhant Gangapurwala et al. “RLOC: Terrain-Aware Legged Locomotion Using Reinforcement Learning and Optimal Control”. In: *IEEE Transactions on Robotics* 38.5 (2022), pp. 2908–2927.

[70] Zhaoming Xie et al. *Dynamics Randomization Revisited:A Case Study for Quadrupedal Locomotion.* 2020. URL: `https://arxiv.org/abs/2011.02404`.

[71] Ashish Kumar et al. “RMA: Rapid motor adaptation for legged robots”. In: *Robotics: Science and Systems.* 2021.

[72] Ananye Agarwal et al. *Legged Locomotion in Challenging Terrains using Egocentric Vision.* 2022. arXiv: `2211.07638 [cs.RO]`.

[73] Jonah Siekmann et al. *Learning Memory-Based Control for Human-Scale Bipedal Locomotion.* 2020. arXiv: `2006.02402 [cs.RO]`.

[74] David Ha and Jürgen Schmidhuber. “World Models”. In: (2018). URL: `https://zenodo.org/record/1207631`.

[75] Philipp Wu et al. “DayDreamer: World Models for Physical Robot Learning”. In: *6th Annual Conference on Robot Learning.* 2022. URL: `https://openreview.net/forum?id=3RBY8fKjHeu`.

[76] Danijar Hafner et al. “Learning Latent Dynamics for Planning from Pixels”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2555–2565. URL: `https://proceedings.mlr.press/v97/hafner19a.html`.

[77] Danijar Hafner et al. *Mastering Atari with Discrete World Models*. 2022. arXiv: `2010.02193 [cs.LG]`.

[78] Jack Clark and Dario Amodei. Dec. 2016. URL: `https://openai.com/research/faulty-reward-functions`.

[79] Xue Bin Peng et al. *Learning Agile Robotic Locomotion Skills by Imitating Animals*. 2020. URL: `https://arxiv.org/abs/2004.00784`.

[80] Eric Vollenweider et al. *Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning*. 2022. URL: `https://arxiv.org/abs/2203.14912`.

[81] Jonas Buchli, Ludovic Righetti, and Auke Jan Ijspeert. “Engineering entrainment and adaptation in limit cycle systems”. In: *Biological Cybernetics* 95.6 (Dec. 2006), pp. 645–664.

[82] Younggil Cho, Sajjad Manzoor, and Youngjin Choi. “Adaptation to Environmental Change Using Reinforcement Learning for Robotic Salamander”. In: 12.3 (2019). URL: `https://doi.org/10.1007/s11370-019-00279-6`.

[83] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[84] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[85] Guillaume Bellegarda and Auke Ijspeert. “CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 12547–12554.

[86] Anna Lisa Ciancio et al. “Hierarchical reinforcement learning and central pattern generators for modeling the development of rhythmic manipulation skills”. In: *2011 IEEE international conference on development and learning (ICDL)*. Vol. 2. IEEE. 2011, pp. 1–8.

[87] Yutaka Nakamura et al. “Reinforcement learning for a biped robot based on a CPG-actor-critic method”. In: *Neural networks* 20.6 (2007), pp. 723–735.

[88] Mostafa Ajallooeian et al. “Modular control of limit cycle locomotion over unperceived rough terrain”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 3390–3397.

[89] Mostafa Ajallooeian et al. “Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 3321–3328.

[90] Eugene Valassakis, Zihan Ding, and Edward Johns. “Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5372–5379.

[91] Ananye Agarwal et al. “Legged Locomotion in Challenging Terrains using Egocentric Vision”. In: *6th Annual Conference on Robot Learning*. 2022. URL: `https://openreview.net/forum?id=Re3NjSwf0WF`.

[92] Geoffrey E. Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *CoRR* abs/1207.0580 (2012). arXiv: `1207.0580`. URL: `http://arxiv.org/abs/1207.0580`.

[93] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[94] Shimon Whiteson et al. “Protecting against evaluation overfitting in empirical reinforcement learning”. In: *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 2011, pp. 120–127.

[95] Chiyuan Zhang et al. “A Study on Overfitting in Deep Reinforcement Learning”. In: *CoRR* abs/1804.06893 (2018). arXiv: `1804.06893`. URL: `http://arxiv.org/abs/1804.06893`.

[96] Hae-Won Park, Patrick M. Wensing, and Sangbae Kim. “Jumping over obstacles with MIT Cheetah 2”. In: *Robotics and Autonomous Systems* 136 (2021), p. 103703. URL: `https://www.sciencedirect.com/science/article/pii/S0921889020305431`.

[97] Takahiro Miki et al. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science Robotics* 7.62 (2022), eabk2822. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.abk2822`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.abk2822`.

[98] Shenggao Li et al. “Quadruped Robot Hopping on Two Legs”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7448–7455.

[99] Chen Yu and Andre Rosendo. “Multi-Modal Legged Locomotion Framework With Automated Residual Reinforcement Learning”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10312–10319.

[100] Yuxiang Yang et al. “Fast and Efficient Locomotion via Learned Gait Transitions”. In: *CoRR* abs/2104.04644 (2021). arXiv: `2104.04644`. URL: `https://arxiv.org/abs/2104.04644`.

[101] Yecheng Shao et al. “Learning Free Gait Transition for Quadruped Robots Via Phase-Guided Controller”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1230–1237.

[102] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. “Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6295–6301.

[103] Gwanghyeon Ji et al. “Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4630–4637.

[104] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2015. URL: `https://arxiv.org/abs/1506.02438`.

[105] Jacob Reher, Wen-Loong Ma, and Aaron D. Ames. “Dynamic Walking with Compliance on a Cassie Bipedal Robot”. In: *CoRR* abs/1904.11104 (2019). arXiv: `1904.11104`. URL: `http://arxiv.org/abs/1904.11104`.

[106] Matthew Chignoli et al. “The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors”. In: *CoRR* abs/2104.09025 (2021). arXiv: `2104.09025`. URL: `https://arxiv.org/abs/2104.09025`.

[107] Donghyun Kim et al. “Dynamic Locomotion For Passive-Ankle Biped Robots And Humanoids Using Whole-Body Locomotion Control”. In: *CoRR* abs/1901.08100 (2019). arXiv: `1901.08100`. URL: `http://arxiv.org/abs/1901.08100`.

[108] O. Stasse et al. “TALOS: A new humanoid research platform targeted for industrial applications”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 689–695.

[109] Fabian Jenelten et al. “TAMOLS: Terrain-Aware Motion Optimization for Legged Systems”. In: *IEEE Transactions on Robotics* 38.6 (Dec. 2022), pp. 3395–3413. URL: `https://doi.org/10.1109%2Ftro.2022.3186804`.

[110] Philipp Wu et al. *DayDreamer: World Models for Physical Robot Learning*. 2022. URL: `https://arxiv.org/abs/2206.14176`.

[111] Chuanyu Yang et al. “Multi-expert learning of adaptive legged locomotion”. In: *Science Robotics* 5.49 (2020), eabb2174. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.abb2174`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.abb2174`.

[112] Tingwu Wang et al. “Nervenet: Learning structured policy with graph neural networks”. In: *Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada*. Vol. 30. 2018.

[113] Milad Shafiee Ashtiani, Alborz Aghamaleki Sarvestani, and Alexander Badri-Spröwitz. “Hybrid parallel compliance allows robots to operate with sensorimotor delays and low control frequencies”. In: *Frontiers in Robotics and AI* (2021), p. 170.