university of groningen

faculty of science and engineering

Learning Perceptive Bipedal Locomotion over Irregular Terrain

Bart van Marum

university of groningen / faculty of science and engineering

University of Groningen

Learning Perceptive Bipedal Locomotion over Irregular Terrain

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Dr. S.H. Kasaei (Artificial Intelligence, University of Groningen)
and
Dr. M. Sabatelli (Artificial Intelligence, University of Groningen)

Bart van Marum (s3005739)

June 29, 2023

Contents

Acknowledgments

I would like to express my deepest gratitude to Dr. Hamidreza Kasaei for his invaluable guidance and support throughout this project, as well as for his mentorship in navigating the academic world. I also wish to extend my appreciation to Dr. Mattia Sabatelli for his expert feedback and insight into the realm of reinforcement learning. Finally, I want to extend my heartfelt appreciation to my parents for their unending support, and to my girlfriend for her patience and understanding as she endured endless discussions about robotics and reinforcement learning.

Abstract

In this thesis we propose a novel bipedal locomotion controller that uses noisy exteroception to traverse a wide variety of terrains. Building on the cutting-edge advancements in attention based belief encoding for quadrupedal locomotion, our work extends these methods to the bipedal domain, resulting in a robust and reliable internal belief of the terrain ahead despite noisy sensor inputs. Additionally, we present a reward function that allows the controller to successfully traverse irregular terrain. We compare our method with a proprioceptive baseline and show that our method is able to traverse a wide variety of terrains and greatly outperforms the state-of-the-art in terms of robustness, speed and efficiency.

1 Introduction

1.1 Overview

Humans excel at executing complex dynamic maneuvers. Activities such as walking on a tightrope, performing parkour, or even just running fast on uneven terrain are tasks that well-trained humans can accomplish with relative ease. And it's not only us, many animals are also capable of performing complex dynamic maneuvers. Thus far, humanity has not been able to reproduce these skills artificially to the same level as observed in nature. Numerous attempts over the past decades to create robots capable of performing complex maneuvers have resulted in significant progress, yet many challenges remain.

In this thesis, we aim to advance the state-of-the-art in bipedal locomotion. We design and train a locomotion controller capable of navigating a wide variety of irregular terrains, while following a user command. To achieve this goal, we employ Reinforcement Learning (RL) to develop a locomotion policy. Unlike previous research in bipedal locomotion, our policy integrates both exteroceptive and proprioceptive sensory inputs directly into the neural network.

We demonstrate that our approach yields highly promising results. Our policy enables traversal over a diverse range of terrains with a high degree of robustness. We compare our method with a proprioceptive baseline, showing that our method outperforms the baseline in terms of robustness, speed, and efficiency. Additionally, we demonstrate our method's resilience to significant noise in the exteroceptive inputs.

1.2 Motivation

The development of a general-purpose humanoid robot holds transformative potential for society and the economy. These robots, if fully realized, can automate a significant portion of labor across various levels of supply chains, potentially reducing costs and elevating living standards. Moreover, with their ability to perform tasks that are currently constrained by capital, risk, or feasibility, humanoid robots can expand the scope of human endeavors.

As general-purpose platforms, humanoid robots are uniquely compatible with human-designed environments, allowing for seamless integration into existing infrastructures without necessitating costly modifications. However, designing a fully functional humanoid robot, particularly one capable of navigating irregular and unfamiliar terrains, remains a formidable challenge.

To address this, our work focuses on the development of a robust bipedal locomotion controller, advancing the capabilities of these robots and bringing us closer to realizing their full potential.

1.3 Research Questions and Contributions

The core aspects of this study that differ from past research are twofold: (1) the incorporation of exteroception for a bipedal locomotion controller, (2) and the traversal of irregular terrain, as opposed to flat or a singular terrain type. We argue that exteroception is fundamentally necessary for a policy that locomotes actively, contrasting with proprioceptive policies that locomote reactively. Furthermore, we argue that the ability to traverse irregular terrain is a necessary step towards a fully functional humanoid robot.

Therefore, this thesis primarily addresses the following research questions:

Q1. Will exteroception improve a bipedal locomotion policy despite the noise inherent in exteroceptive sensors?

Q2. Is it feasible to design a bipedal locomotion policy capable of successfully navigating a wide range of terrains?

1.4 Thesis Outline

The subsequent chapter, Background Literature, provides a comprehensive overview of the necessary background materials, exploring the general principles of Reinforcement Learning as well as relevant prior work in robotics. In Chapter 3, Methods, we delve into the methods underpinning this study, discussing in detail the reward function, the neural network architecture, and our training procedure. Chapter 4, Experimental Results, takes a close look at the experiments we conducted and the results we garnered. Lastly, in Chapter 5, Conclusion, we discuss our findings, draw conclusions from our study, and outline potential directions for future research.

2 Background Literature

In this chapter we will explain technical background and concepts that are vital to understand the rest of this study. Initially, we provide a general overview of Reinforcement Learning, followed by a more in-depth explanation of the specific RL algorithm utilized in this study: Proximal Policy Optimization [1]. For a more comprehensive overview of RL we refer the reader to the Reinforcement Learning book by Sutton et al. [2]. In the final section we explore related works in the field of robot locomotion.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a fundamental paradigm of Machine Learning. Unique to RL is its use of a reward-based system for updating the neural network model, in contrast to the loss difference between an output and a label often employed in other paradigms. This distinction allows for the training of policies applicable to tasks where traditional training data may be unavailable, such as in robot locomotion. The reward in RL is a scalar value, typically provided by an external entity such as a human operator or a simulation environment or another neural network. The model's performance is evaluated based on this reward, and the model parameters are updated to maximize the reward over time. Thus, RL centers on learning optimal actions within an environment to maximize a cumulative reward.

While the labeled data in supervised learning offers a direct means to train a model, RL shines in situations where such data is scarce or non-existent. For example, there may be no appropriate data available for robot locomotion because current robots cannot emulate human walking or because a robot's unique design prevents the use of existing data. In addition, supervised learning could limit the policy to the behavior demonstrated in the training data, precluding the development of novel behavior. Conversely, RL enables the creation of training methods that can give rise to new emerging behavior best suited to the environment.

However, RL is not without its challenges. The sparsity of the reward signal in RL necessitates extensive training over a large number of episodes before reaching useful policies. Furthermore, reward functions are often crafted manually, a process that can be time-consuming due to its iterative nature and extended feedback loop. It is not uncommon in practice for a single iteration of training an RL policy to take 24 hours or more, even on powerful hardware. This contrasts sharply with supervised learning of a similar architecture policy, which can be completed in a matter of minutes, provided the training data is available. Moreover, even minor modifications to the policy often require complete retraining, thus posing a significant limitation to RL's efficiency. Despite these challenges, RL has emerged as a potent tool for the development of robot locomotion policies. It particularly excels in terms of robustness, outperforming non-learning methods in this critical aspect — a vital attribute for any functional humanoid robot.

2.1.1 Agent and Environment

Fundamentally, RL can be conceptualized as an optimization problem, modeling the interactions between an agent and its environment, as illustrated in Figure 1. The agent represents the entity that interacts with the world, and the environment embodies the context or world in which the agent operates. The agent interacts with the environment by taking an action a_t, and the environment

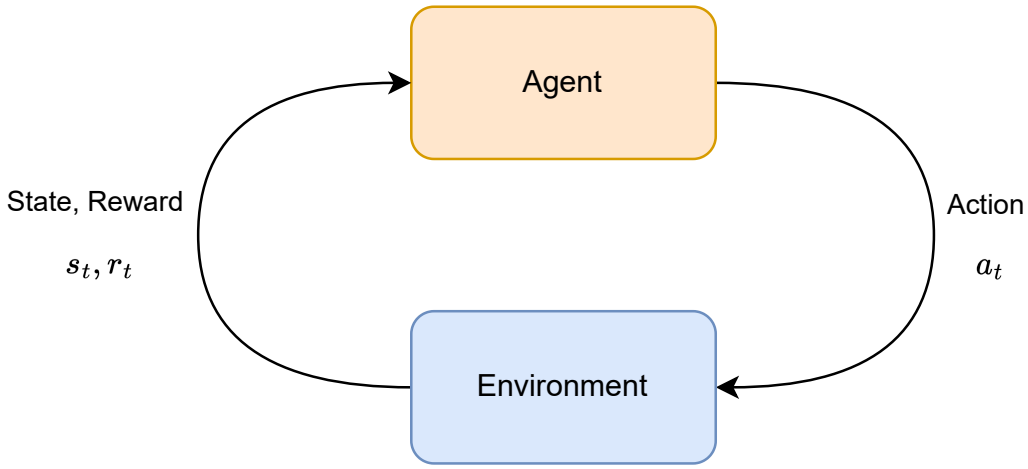


Figure 1: A diagram illustrating the interactions between an agent and its environment.

responds to such action by providing a reward r_{t+1} and a new state s_{t+1}. Subsequently, the agent then employs its policy π using the current state to determine the next action a_{t+1} to be taken. This cycle is repeated until some terminal state is arrived at. The primary goal of RL is to optimize the agent's policy such that the actions it takes maximize the cumulative reward over an episode.

2.1.2 Return

The cumulative reward over an episode is known as the return, and it is defined as the sum of all rewards from time step t to the end of the episode T:

$$R_t = \sum_{t'=t}^{T} r_{t'} \tag{1}$$

This definition of the return is known as the "undiscounted finite horizon return", as it represents the return in the context of a finite horizon, where all rewards within a given episode are summed without any discounting. However, optimizing for this return may not always be ideal, as it does not take into account the time at which the reward is received. For example, this approach equally weights a large reward received at a time far into the future and a large reward received in the next time step. This could potentially be problematic, as it assumes that the agent will always be able to reach future rewards, which may not always be the case due to uncertainties in the environment. To address this issue, future rewards can be discounted by introducing a discount factor γ and multiplying the reward at each time step by this factor:

$$R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \tag{2}$$

The discount factor γ is a value between 0 and 1, and it determines the extent to which future rewards are discounted. A value of 0 implies that only the immediate reward is considered, while a value of 1 means that all future rewards are considered equally. In practice, a value of 0.99 is often chosen, providing a balance between immediate and future rewards. This definition of the return is known as the "discounted infinite horizon return", as it considers an infinite number of future time steps while discounting their importance progressively.

2.1.3 Markov Decision Process

More formally, the RL problem can be defined mathematically as a Markov Decision Process (MDP). Different formulations exist, but one fundamental definition is the tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, \rho_0) \tag{3}$$

where:

- $\mathcal{S}$ denotes the state space, which is the set of all possible states that the agent can be in.
- $\mathcal{A}$ is the action space, which is the set of all possible actions that the agent can take.
- P is the transition probability function that, given the current state s_t and action a_t, outputs the probability distribution over the subsequent state s_{t+1}.
- R is the reward function, mapping the current state s_t and action a_t to the corresponding reward r_t.
- γ is the discount factor.
- ρ_0 is the initial state distribution, which is the probability distribution over the initial state s_0.

In this framework, the RL problem involves learning a policy that maximizes the discounted return. Navigating through the environment involves dealing with its inherent uncertainties, which is reflected in the transition probability function, P. This process, particularly in high-dimensional and continuous state-action spaces, can be a challenging task. In practice, the policy is often represented by a parameterized function, such as a neural network. The parameters of this function are iteratively adjusted to improve the expected return.

The above definition assumes full observability of the state, known as the Markov property. This means that the entirety of the environment's state is accessible to the agent at each time step, and that a future state is fully defined by the current state and the current action. However, this is not feasible in real-world scenarios. An agent, such as a robot, typically only has access to partial information about the environment's state through its sensors, such as gyros, joint positional encoders, and cameras. These sensors provide observations that represent only a subset of the full environment state and are often subject to noise. As such, they do not offer a complete picture of the environment's state.

Given these challenges, a real world robotics reinforcement learning problem is often better modeled as a Partially Observable Markov Decision Process (POMDP), which is a more realistic formulation for many practical scenarios. A POMDP is defined as the tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mathcal{O}, Z, \gamma, \rho_0) \tag{4}$$

with additional components:

- $\mathcal{O}$ is the observation space, which is the set of all possible observations that the agent can receive.

- Z is the observation probability function that, given the current state s_t and action a_t, outputs the probability distribution over the resulting observation o_t.

In a POMDP we do not directly operate on states $s \in \mathcal{S}$, but instead work with observations $o \in \mathcal{O}$, which are sampled from the observation probability function Z based on the underlying states $s \in \mathcal{S}$. POMPDs are harder to learn due to the partial observability. However, the challenges introduced by partial observability can be mitigated by incorporating past observations into the decision-making process, allowing the policy to maintain an understanding of the underlying state dynamics based on the sequence of observations. In this study, we use a policy with Long Short-Term Memory (LSTM), a type of recurrent neural network that is capable of retaining information from past observations, providing the agent with a form of memory to handle the complexities of the partially observable environment.

2.1.4 Policy

Two kinds of policies exist: deterministic and stochastic. A deterministic policy always maps a state to the same action, while a stochastic policy provides a probability distribution over possible actions. A stochastic policy π is a function that maps a state s to a distribution over actions, that we can sample from to obtain an action such that $a_t \sim \pi(\cdot|s_t)$. A stochastic policy is advantageous during the learning process as it facilitates exploration of different actions.

In RL the policy can be presented by any learnable function. However in the case of Deep Reinforcement Learning, which is what the methods in this study are based on, the policy is typically represented by a function approximator such as a neural network. Examples include architectures such as Multi-Layer Perceptrons (MLP) or Long Short-Term Memory (LSTM) networks [3]. Often the policy is denoted as $\pi_\theta(s)$, where θ represents the weights of the neural network.

2.1.5 The goal of Reinforcement Learning

The central objective of reinforcement learning is to find a policy that maximizes the expected return. Formally, this can be expressed as finding the optimal policy π^* that maximizes the expected return from all states, or equivalently, maximizes the expected value of the return for each state-action pair:

$$\pi^* = \arg\max_\pi E_\pi[R_t|s_t = s, a_t = a] \tag{5}$$

This formulation incorporates both immediate and future rewards, appropriately discounted according to the discount factor γ as discussed previously in section 2.1.2.

Value functions, including the state-value function $V(s)$ and the action-value function $Q(s,a)$, play a crucial role in achieving this goal. The state-value function under a policy π is the expected return from starting in a given state s and then following the policy thereafter. The action-value function is the expected return after taking an action a in state s that may or may not be sampled from the policy, and then following the policy thereafter starting from the next state:

$$V^{\pi}(s) = E_{\pi}[R_t|s_t = s] \tag{6}$$
$$Q^{\pi}(s,a) = E_{\pi}[R_t|s_t = s, a_t = a] \tag{7}$$

Bellman equations present a recursive relationship between the value of a state or state-action pair and the values of its successor states or state-action pairs:

$$V^{\pi}(s) = \underset{\substack{a\sim\pi \\ s'\sim P}}{E}[r(s,a) + \gamma V^{\pi}(s')] \tag{8}$$
$$Q^{\pi}(s,a) = \underset{s'\sim P}{E}\Big[r(s,a) + \gamma \underset{a'\sim\pi}{E}[Q^{\pi}(s',a')]\Big] \tag{9}$$

An interesting observation can be made here by comparing the value function and action value function for a given action, yielding the advantage function:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{10}$$

By subtracting the expected return from acting according to a policy π from the expected return from performing action a and then acting according to a policy π thereafter, we obtain the advantage function. This function represents the relative value of taking action a in state s, taking into account the longer-term effects on the trajectory.

2.1.6 Policy Gradient Methods

Policy Gradient Methods [4] belong to a larger family of model-free RL algorithms. These methods primarily focus on directly optimizing the policy parameters in order to increase the expected return of the policy. Formally, the policy parameters θ are updated in the direction of the gradient of the expected return $J(\pi_\theta)$:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta=\theta_k} \tag{11}$$

In this equation, α represents the learning rate, k indicates a particular iteration or version of the policy, and $J(\pi_\theta)$ corresponds to the expected return:

$$J(\pi_\theta) = E_{\pi_\theta}[R_t] \tag{12}$$

The Policy Gradient Theorem provides a method to compute the gradient of the expected return $J(\pi_\theta)$ with respect to the policy parameters θ:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right] \tag{13}$$

Intuitively, the above equation states that the gradient of the expected return is equal to the sum of the gradients of the log probability of the actions taken multiplied by the advantage function, meaning that the gradient is proportional to the advantage of the action taken. Thus the gradient update step will increase the probability of that action. Numerically, we can approximate this expectation by taking a mean over a batch of trajectories sampled from the policy π_θ.

2.1.7 Actor Critic Methods

The fundamental policy gradient method as described above directly optimizes the policy by calculating the gradient based on the expected return, which is computed using the return from the currently sampled trajectories. While this method is effective, it can yield high-variance estimates of the return, potentially leading to unstable policy gradient updates.

This variance can be mitigated by learning a value function that predicts the value of a state, rather than calculating it empirically. Methods employing this approach are referred to as actor-critic methods [5, 6], as they simultaneously learn both an actor (policy) and a critic (value function). In these methods, the critic is trained to predict the value of a state, while the actor is trained to maximize the expected return, using the critic's estimation for the state value. This strategy results in a lower variance for the policy gradient estimate, as the critic typically provides a more stable estimate of the state's value than the empirical return.

2.1.8 Proximal Policy Optimization

Proximal Policy Optimization (PPO) belongs to the group of actor critic policy gradient optimization methods and has been empirically proven to be highly effective in terms of speed and stability [1]. One of the limitations of many RL algorithms is the low sample efficiency. PPO attempts to address this issue by making more effective use of sampled data. The core idea of PPO is to take an as large as possible step given the current data, without overstepping and causing instability. Training for multiple epochs on a single batch of sampled trajectories can lead to instability, as the policy might diverge from the current data distribution. To address this, PPO limits the policy update to a certain range, using a clipped surrogate objective function. This clipped surrogate objective function is defined as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{14}$$

Here, $r_t(\theta)$ denotes the ratio of the probability of the action under the new policy to the probability under the old policy, given by:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{15}$$

The $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ operation ensures that the ratio $r_t(\theta)$ is bounded within $[1-\epsilon, 1+\epsilon]$. $\hat{A}_t$ is the estimated advantage at time t, and $\hat{\mathbb{E}}_t$ is the empirical expectation over time steps. The ϵ is a hyperparameter that determines the range in which the ratio of new to old policy is allowed to change.

The advantage estimation $\hat{A}_t$ is used to provide a direction for the policy update, giving more weight to advantageous actions and reducing the weight of disadvantageous actions.

To summarize, the PPO algorithm can be described in the following steps:

1. Collect trajectories by running the current policy in the environment.
2. Compute the returns and advantages of the collected trajectories.
3. Optimize the clipped surrogate objective using stochastic gradient ascent, using the returns and advantages as an estimate of the gradient.
4. Update the policy with the new parameters.
5. Repeat the process from step 1.

2.2 Related Work

Conventionally bipedal locomotion controllers are designed as explicit dynamical models [7, 8, 9]. However, these models often lack in robustness, do not generalize to new scenarios or terrains without explicit modelling, and are laborious and complicated to develop and maintain. Moreover, adding exteroceptive capabilities to such methods is not straightforward.

In recent years, the application of Reinforcement Learning (RL)[2] in the design of bipedal locomotion control systems has seen substantial growth, particularly for both simulated[10, 11] and real-world bipedal robots [12, 13, 14, 15, 16]. In this approach, the locomotion control policy is typically modeled as a neural network and trained to maximize a certain reward signal. This method has demonstrated robust performance, even in instances of motor malfunctions [13].

Many RL based approaches rely on the use of reference trajectories and imitation rewards to train a policy to produce a gait[12]. Such imitation rewards limit the policy to learn predetermined behaviour as defined by the reference trajectories, as well as require the existence of such trajectories, which is not the case for most gait-robot combinations. However, recent work suggests that it is possible to learn a wide variety of different bipedal gaits in a single neural network by using periodic reward functions instead of reference trajectories [14]. This approach allows designers to specify a desired gait in terms of ground time, air time and frequency.

Despite the great progress in recent years for neural network based bipedal locomotion controllers, most approaches are compatible with flat terrain only. Some successful attempts have been made to learn blind bipedal locomotion controllers for more challenging terrains [16]. However, these policies resort to more conservative foot trajectories with higher steps and are unable to avoid dangerous areas. Such blind strategies do not generalize well to a wide variety of unseen and irregular terrains and lack in efficiency, and therefore are not feasible for a fully capable humanoid robot.

In order for a legged locomotion controller to traverse any random previously unseen terrain robustly, it needs information about the world ahead. A controller that is based on proprioception only is limited to reactive and precautionary behaviour. Only a controller that has information about the world ahead can actively plan steps. Exteroceptive inputs are necessary, however using both proprioception and exteroception presents a challenge. Exteroceptive sensors such as cameras, Lidar, or

radar often produce spurious readings in cases such as reflection (water puddle), transparency (glass), deformation (snow), or fake obstacles (tall grass). This may lead locomotion policies based on such inputs to unnecessarily avoid certain areas or fail outright, and raises the question of how to handle disagreeing proprioceptive data.

The field of quadrupedal locomotion control has shown great progress in recent years in learning controllers for navigating challenging terrain [17, 18]. Most notably, [19] shows that using a recurrent belief encoder with an attention mechanism, a neural network policy is able to learn when to trust and when not to trust the exteroceptive data. This allows the locomotion controller to utilize exteroceptive data when it is most useful, and fall back to proprioceptive data when it is not. Our work extends these methods to the bipedal domain, resulting in a robust and reliable internal belief of the terrain ahead despite noisy sensor inputs.

Another effective approach to finding a robust locomotion controller is to use Model Predictive Control (MPC) [20]. MPC is a control method that uses a model of the system to predict future states and actions, and then optimizes a cost function over the predicted states and actions. MPC has been shown to be effective in perceptive locomotion control [21], however it is limited in its ability to handle unexpected situations or noisy inputs. In contrast, RL based methods are more robust and can handle unexpected situations more effectively. In this study we focus on RL based methods, as we believe that they are more suitable for the task of traversing a wide variety of terrains combined with noisy exteroceptive inputs.

3 Methods

In this chapter we explain and discuss the various methods used to train an exteroceptive bipedal locomotion policy that can navigate a wide variety of irregular terrains, following a user command. We explain our general setup, state and action representation, policy architecture, randomization processes and reward functions. To summarize, Figure 2 shows an overview of our approach.

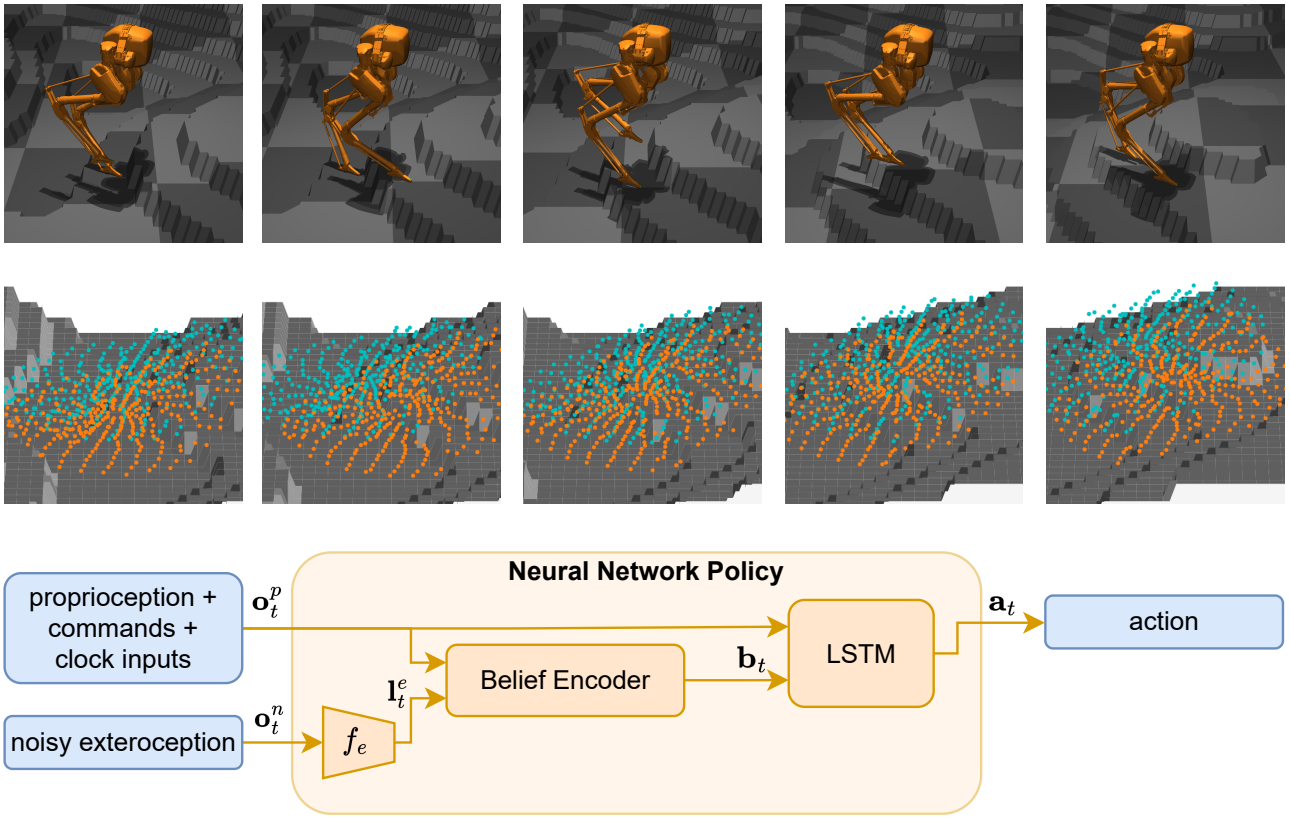
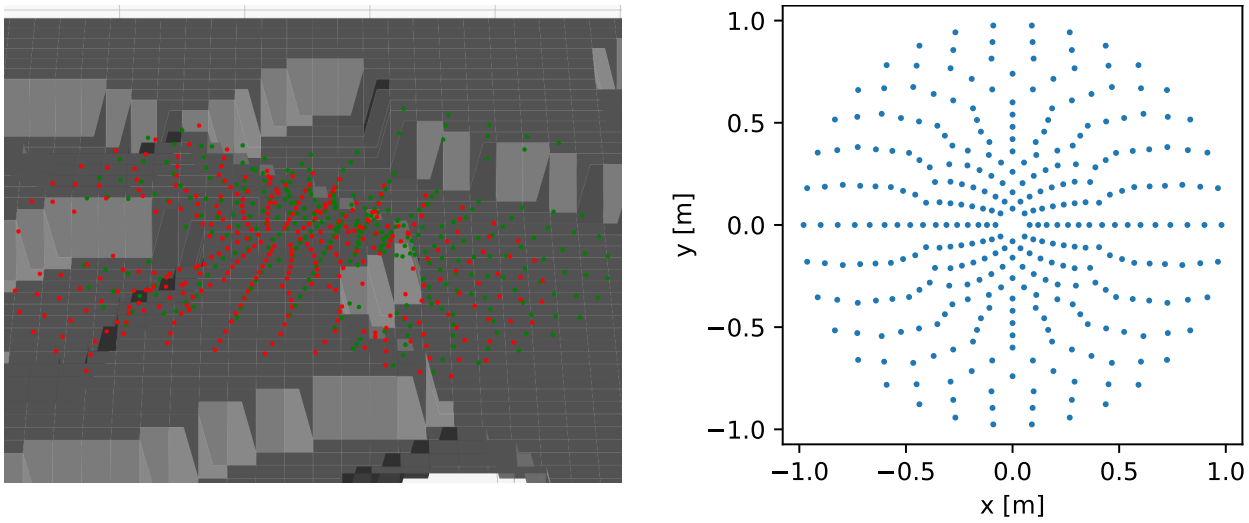


Figure 2: In this study we develop a bipedal locomotion control policy integrating both exteroception and proprioception that is able to traverse a wide variety of terrains. The first row shows Cassie walking over random terrain in the physics simulator. The second row shows noisy exteroceptive samples that are input to the policy at the same timesteps. The bottom shows the policy architecture during inference.

3.1 Learning Setup

The main goal is to develop a robust bipedal locomotion controller that is able to navigate irregular terrain while following a command. In order to do so we use privileged learning [22] to distill a policy that is able to work with potentially noisy and spurious exteroceptive observations. Previous work has shown that directly learning the desired behavior over rough terrain with RL does not converge within reasonable time budgets [17]. First a teacher policy with access to perfect, noise free observations is trained in simulation through reinforcement learning to traverse a wide range of different terrains. We then train a student policy to imitate the behaviour of the teacher policy, but without privileged information and noisy inputs. We use Proximal Policy Optimization (PPO) [1] to train our policies, as PPO has shown to yield good results in bipedal locomotion control [13, 14, 12].



(a) (b)

Figure 3: (a) A close up view of the exteroceptive simulator. The red dots represent the height sample for the right foot, and the green dots for the left foot. Samples are taken with the sampling pattern centered around the xy position of each foot and rotated to match pelvis orientation. (b) Shows a detailed plot of the pattern used to sample heights from the terrain.

3.2 State and Action Representation

We define three observations $\mathbf{o}_t^p, o_t^e, o_t^n$. Here $\mathbf{o}_t^p \in \mathbb{R}^{44}$ is the proprioceptive input, consisting of motor positions, motor speeds, joint positions, joints speeds, pelvis orientation, pelvis angular velocity, user commanded velocity v_c and clock inputs i_t. The user commanded velocity v_c is defined as the pair $(\mathbf{v}_{cmd}, \omega_{cmd})$ where $\mathbf{v}_{cmd} \in \mathbb{R}^2$ represents the commanded velocity in the x and y directions and ω_{cmd} represents the commanded angular velocity around the z axis. The clock input i_t is defined as the pair $\sin(2\pi(\phi))$ and $\sin(2\pi(\phi+0.5))$, where ϕ is defined as $\phi = (t \mod T)/T$, with t denoting the current timestep, and T a user defined gait period in terms of timesteps. Although it has been noted in past research [15] that clock inputs are necessary, we found in preliminary experiments that a policy without clock inputs learns a gait with no meaningful effect on the reward. However, we have not yet performed a thorough investigation on this matter.

Additionally, $o_t^e = (\mathbf{e}_t^l, \mathbf{e}_t^r)$ is the pair of noiseless exteroceptive observations for the left and right feet. To obtain the observations the terrain height is sampled with a sampling pattern centered at the location of the respective foot. The sampling pattern consists of 318 points spaced circularly. Figure 3 shows the height sampling taking place in the simulation environment.

Finally, $o_t^n = (\mathbf{n}_t^l, \mathbf{n}_t^r)$ is the pair of noisy exteroceptive observations. Sampling is similar to $\mathbf{o}_t^e$ however a noise is applied to the sampling pattern coordinates and sampled values. Further details on noise generation are discussed in section 3.5.

The action $\mathbf{a}_t \in \mathbb{R}^{10}$ represents the PD targets for the actuators in the robot model. Previous research has shown that PD targets are an effective action parametrization for learning locomotion [23]. The robot PD controller runs at 2 kHz, whereas actions are sampled from the policy at a rate of 40 Hz.

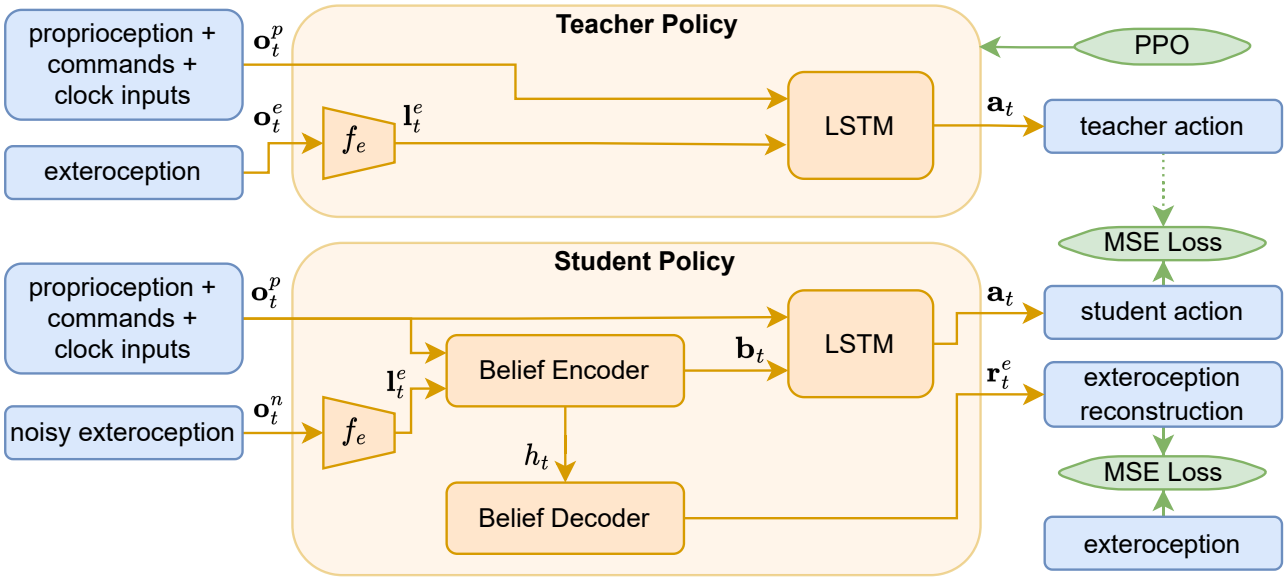


Figure 4: The top figure shows the teacher policy architecture which is trained with PPO. The bottom figure shows the student policy architecture which is trained to both imitate the action output of the teacher policy, and to denoise the noisy exteroceptive input.

3.3 Policy Architecture

The teacher and student policy architectures are illustrated in Figure 4. In this section, we will provide a detailed description of these architectures.

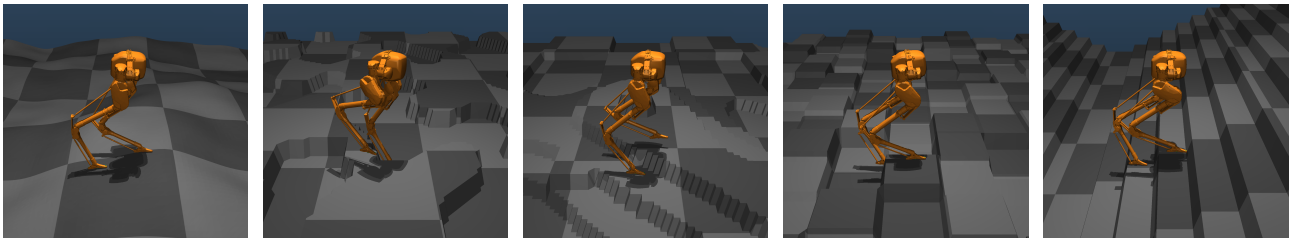
3.3.1 Teacher Policy

The teacher policy π^t consists of an exteroceptive encoder f_e and an LSTM [3]. The encoder f_e consists of 3 fully connected layers of size $\{256, 160, 96\}$ and the LSTM has two layers of 256 nodes. The teacher policy receives the observation $o_t^t = (\mathbf{o}_t^p, \mathbf{o}_t^e)$. The exteroceptive encoder f_e receives both exteroceptive observations $\mathbf{e}_t^l, \mathbf{e}_t^r$ that are in o_t^e and encodes them separately into the latent vectors $\mathbf{l}_t^{el}$ and $\mathbf{l}_t^{er}$ which are concatenated into the latent vector $\mathbf{l}_t^e \in \mathbb{R}^{200}$. The LSTM receives the concatenation of $\mathbf{o}_t^p$ and $\mathbf{l}_t^e$ and outputs an action $\mathbf{a}_t$.

3.3.2 Student Policy

The student policy π^s takes in the noisy observation $o_t^s = (\mathbf{o}_t^p, \mathbf{o}_t^n)$ and has a partially similar architecture as the teacher policy, using the same encoder f_e and LSTM. An added component is a recurrent belief encoder, which receives the exteroceptive latent vector $\mathbf{l}_t^e$ and the proprioceptive observation $\mathbf{o}_t^p$ and outputs a belief vector $\mathbf{b}_t \in \mathbb{R}^{192}$. The belief vector is then concatenated with the proprioceptive observation $\mathbf{o}_t^p$ and fed into the LSTM, which in turn outputs an action $\mathbf{a}_t$.

The main aspect of the student policy is the recurrent belief encoder, an approach introduced by [19], which is intended to take the proprioception and noisy exteroception and develop an internal representation of what the terrain looks like. In order to train this internal representation a belief decoder is added to the policy, which takes as input the hidden state of the recurrent belief encoder. The belief decoder outputs a reconstruction of the exteroceptive inputs, which is trained to minimize



(a) (b) (c) (d) (e)

Figure 5: The five different terrain modes used for training: (a) *hills*, (b) *edges*, (c) *quantized hills*, (d) *squares*, (e) *stairs*.

the difference with the noise free exteroceptive observation o_t^e. This method encourages the internal hidden state of the belief encoder to represent a representation of the outside world that is as accurate as possible, despite noisy inputs. Additionally, the belief encoding system is fitted with an attention mechanism, such that the policy is able to learn when exteroceptive data is not useful, and rely on proprioception instead to construct the belief. For more detail about the belief encoder and decoder we refer the reader to [19].

The student policy is trained with both an action imitation loss, and an observation reconstruction loss to encourage the internal belief representation of the outside world to be as accurate as possible.

3.4 Terrain Generation

We use a linear curriculum to ramp terrain generation intensity. The ramp starts after the policy has learned to walk on flat ground. All generated terrains are modelled as a height map in meters and multiplied with the curriculum factor $c_t \in [0, 1]$.

We define five different terrain modes, as shown in Figure 5. The first is *hills*, which is modelled as a sum of a low frequency and a higher frequency Perlin noise [24]. The generated values are normalized to a range $[0, 0.8]$. The second terrain mode is *edges*, which consists of a Perlin noise that has been quantized to two levels $\{0, h \sim \mathcal{U}(0.15, 0.25)\}$. The third mode is *squares* which consists of a grid of squares with sides $d \in [0.4, 0.6]$ of random height $h \in [0, 0.4]$. The fourth mode is *quantized hills* which is Perlin noise that has been quantized to discrete levels with a random step size $h \in [0.12, 0.18]$. The fifth and final mode of terrain generation is *stairs*, consisting of alternating ascending and descending staircases. To generate a staircase we randomly select a run $d \in [0.3, 0.4]$ and rise $r \in [0.1, 0.22]$ for 10 equal stairs.

3.5 Randomization

Although we only focus on simulation based experiments in this work, previous work has shown that policies trained in simulation are able to bridge the sim-to-real gap given proper domain randomization [15, 25, 26, 27]. Therefore we randomize joint damping, body part masses and friction coefficients at the start of each episode. We use the same parametrization as presented in [16].

Furthermore, we randomize the velocity command to expose the policy to a range of different velocity commands during training. At the start of each episode and at one random timestep during each episode a new velocity command is sampled. The probability distribution for the velocity commands

Table 1: Velocity Commands Randomization

v_x	v_y	ω_z	**Probability**
0	0	0	0.15
± 1	0	0	0.42
0	± 1	0	0.07
0	0	± 1	0.025
$\sim \mathcal{U}(-1,1)$	$\sim \mathcal{U}(-1,1)$	$\sim \mathcal{U}(-1,1)$	0.1

is shown in Table 1.

To obtain the noisy exteroceptive student observation o_t^n we apply a noise to the noise free exteroceptive teacher observation o_t^e. We sample noises for the sampling coordinates x, y and the sampled height values z at the episode, foot and timestep level. Additionally, we select random points on the height sampling pattern and apply a large noise to simulate outliers. For the parametrization of these noises we leverage three modes $\{nominal, offset, noisy\}$ defined in prior work by [19] who designed the nosing method to mimic noise in real exteroceptive sensors.

3.6 Reward

For the teacher policy to learn to follow arbitrary commands over arbitrary terrain we use a number of reward terms that are divided into three main categories: (1) *gait*, (2) *command following* and (3) *smoothness*. The full reward function we use is defined as:

$$\begin{aligned} r = (0.25r_{frc} + 0.25r_{vel} + 0.2) \cdot c_r + (r_{air} + 0.1r_{one}) \cdot (1 - c_r) \\ + 0.2r_{v,xy} + 0.2r_{\omega,z} + 0.05r_{lov} \\ + 0.05r_{fo} + 0.05r_{pm} + 0.05r_{po} + 0.025r_t + 0.025r_a \end{aligned}$$

of which the components are explained in the next subsections.

3.6.1 Gait rewards

We use a combination of two reward methods to learn a gait. The first is a clock based reward function that oscilates between swing and stance modes in a gait period, as introduced in [14]. In the swing phase the reward function will penalize foot forces, while in the stance phase foot velocity is penalized. By offsetting the clock functions for the left and right leg a gait can be learned. Figure 6 shows the gait clocks. The foot force reward component r_{frc} is defined as:

$$r_{frc} = \tanh(\pi F_l k_{frc,l}) + \tanh(\pi F_r k_{frc,r})$$

where F_r and F_l represent the normalized norm of the foot forces on the respective foot, and $k_{frc,l}$ and $k_{frc,r}$ are the respective foot force gait clocks. The foot velocity reward component r_{vel} is defined as:

$$r_{vel} = \tanh(\pi v_l k_{vel,l}) + \tanh(\pi v_r k_{vel,r})$$

where v_r and v_l represent the normalized norm of the foot velocities of the respective foot, and $k_{vel,l}$ and $k_{vel,r}$ are the respective foot velocity gait clocks.

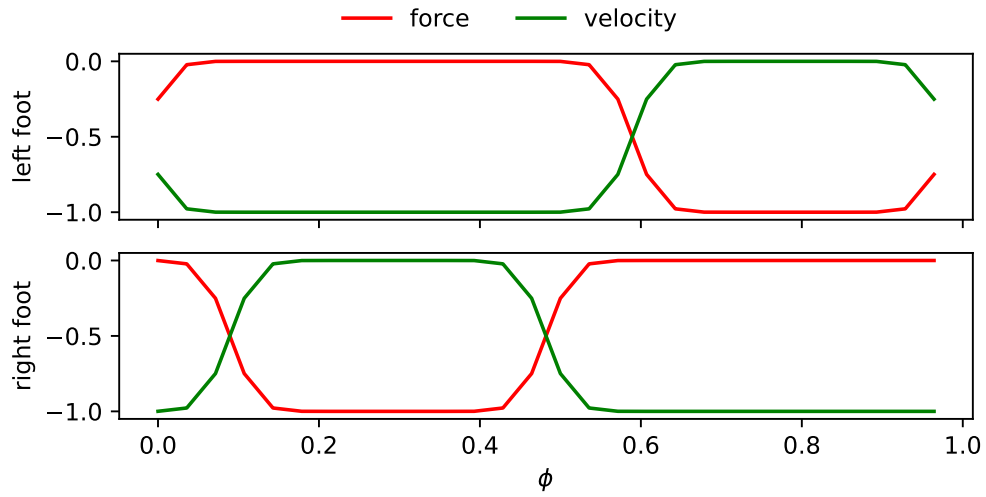


Figure 6: The gait clocks $k_{frc}(\phi)$ and $k_{vel}(\phi)$ for both feet. The stance phases of both feet are offset by $\phi/2$, but overlap slightly, producing a walking gait.

We find that this periodic reward function produces good quality gaits on flat terrain, however performance on rough terrain is less satisfactory. We hypothesize that this is due to the fixed cadence embedded in the gait clocks, restricting the policy to a fixed gait period, and that a more flexible reward function is better suited for rough terrain.

The second is a less constrained reward function that simply rewards foot airtime [18] for both feet. The foot airtime reward component r_{air} is defined as:

$$r_{air} = \sum_{f=0}^{2} (\mathbf{t}_{air,f} - 0.5) * \mathbb{1}_{first\ contact,f}$$

where $\mathbf{t}_{air} \in \mathbb{R}^2$ denotes the cumulative airtime during the swing phase of both feet, and $\mathbb{1}_{first\ contact} \in \mathbb{R}^2$ holds binary values indicating whether the swing phase is ended by contact. To prevent the policy from simply learning to jump, a component r_{one} is added to reward standing on one foot:

$$r_{one} = \mathbb{1}_{single\ contact}$$

where $\mathbb{1}_{single\ contact}$ is a binary value indicating whether the robot is standing on one foot.

We find that this less constraining reward function leads to more stable gaits on rough terrain, at the expense of slower convergence. Therefore, we start the training process with the clock based reward function on flat terrain, in order to capitalize on faster convergence. Once a satisfactory gait has been learned on flat terrain, we switch to the airtime based reward function by setting the reward curriculum factor c_r from 1 to 0. This allows us to utilize the higher robustness provided by the airtime based reward function, while still benefiting from the faster convergence of the clock based reward function. We find that once the gait has been learned on flat terrain, there is no significant impact on the speed of convergence due to the airtime reward anymore.

3.6.2 Command following rewards

We employ similar command following rewards as [17, 19] with the goal of maximizing velocity in a given direction. The velocity reward component $r_{v,xy}$ is defined as:

$$r_{v,xy} = \begin{cases} \exp(-2.5 \cdot \left\|\mathbf{v}_{xy}\right\|^2) & ||\mathbf{v}_{cmd}|| = 0 \\ 1 & \mathbf{v}_{cmd} \cdot \mathbf{v}_{xy} \geq 1 \\ \exp(-2 \cdot (\mathbf{v}_{cmd} \cdot \mathbf{v}_{xy} - 1)^2) & else \end{cases}$$

where $\mathbf{v}_{xy} \in \mathbb{R}^2$ represents the linear velocity in the xy plane. The angular velocity reward component $r_{\omega,z}$ is defined as:

$$r_{\omega,z} = \begin{cases} \exp(-5 \cdot \omega_z^2) & \omega_{cmd} = 0 \\ 1 & \omega_{cmd} \cdot \omega_z \geq 1 \\ \exp(-2 \cdot (\omega_{cmd} \cdot \omega_z - 1)^2) & else \end{cases}$$

where ω_z represents the pelvis angular velocity. The linear orthogonal velocity offset reward component r_{lov} is defined as:

$$r_{lov} = \exp(-5 \cdot \left\|\mathbf{v}_{xy} - \mathbf{v}_{cmd} \cdot \mathbf{v}_{xy}\right\|)$$

where $\mathbf{v}_{xy} \in \mathbb{R}^2$ again represents pelvis velocity in the xy plane. It is intended to penalize linear velocities orthogonal to the commanded velocity.

3.6.3 Smoothness rewards

The foot orientation reward component r_{fo} is defined as:

$$r_{fo} = \exp(-1.5 \cdot (\hat{\mathbf{z}} \cdot \boldsymbol{\psi}_{lf} + \hat{\mathbf{z}} \cdot \boldsymbol{\psi}_{rf})) \cdot (1 - c_t) + c_t$$

where $\hat{\mathbf{z}} \in \mathbb{R}^3$ represents the unit vector in the z direction and $\boldsymbol{\psi}_{lf}, \boldsymbol{\psi}_{rf} \in \mathbb{R}^3$ represent the foot orientation vectors pointing along the length of both feet. This reward component encourages the policy to keep the feet flat on the ground, but in any planar direction. Additionally, the reward is gradually shifted to a constant reward component as the curriculum is ramped up to allow the policy to adapt to terrains where a non flat position might be more beneficial. The pelvis motion reward component r_{pm} is defined as:

$$r_{pm} = \exp(-(v_z^2 + \omega_y^2 + \omega_x^2))$$

and is intended to penalize pelvis motions in directions not part of the command. The pelvis orientation reward component r_{po} is defined as:

$$r_{po} = \exp(-3 \cdot (|\psi_x| + \left|\psi_y\right|))$$

where ψ_x and ψ_y represent pelvis orientation and is intended to encourage the policy to keep the pelvis level. The torque reward component r_t is defined as:

$$r_t = \exp(-0.02 \cdot \overline{|\boldsymbol{\tau}|})$$

where $\boldsymbol{\tau}$ represents the torque vector exerted by the actuators, with the aim to reduce energy consumption. The action reward component r_a is defined as:

$$r_a = \exp(-5 \cdot \overline{|\mathbf{a}_t - \mathbf{a}_{t-1}|})$$

where $\mathbf{a}_t$ represents the action vector at time t and is intended to penalize large changes in the action vector in order to improve smoothness and stability.

4 Experimental Results

4.1 Simulation

Training and experimentation are performed in simulation on the Cassie robot [28]. We use the Mujoco [29] physics simulator with the *cassie-mujoco-sim* environment [30]. We do not have access to Cassie, however previous work has shown that policies trained in the *cassie-mujoco-sim* Mujoco environment are able to bridge the sim-to-real gap zero-shot given proper domain randomization [15, 25, 26, 27]. Additionally, the exteroceptive noising algorithm used in this work was designed to mimic real world noise and has been proven on real hardware [19]. Therefore we believe that the simulation-based results presented here are significant for real world applications.

4.2 Training

For teacher policy training the recurrent PPO algorithm from StableBaselines3 (SB3) [31] is used. In order to achieve a 7 times speedup in terms of timesteps per second we modified SB3 to use batches of whole sequences. The hyperparameters for PPO are listed in Table 2. Teacher policy training takes around 36 hours for 60×10^6 timesteps on a single 12-core V100 node.

The student policy is trained on a dataset of 10^6 timesteps sampled from the teacher policy, with the hyperparameters listed in Table 3. The student policy is implemented in PyTorch [32], and training takes around 1 hour on a high end laptop.

In order to compare and quantify the performance of our exteroceptive student policy we train a baseline policy π^b. This baseline policy observes proprioception $\mathbf{o}_t^p$ only and has the same architecture as the teacher policy, but without the exteroceptive encoder, similar to the current state-of-the-art [14, 16]. The baseline policy is trained on the same terrain and curriculum as the exteroceptive student policy.

Parameter	**Value**
max episode length	300
rollout buffer size	50000
learning rate	10^{-4}
batch size	8 seq.
epochs	5
clip range	0.2
GAE Lambda	0.95
gamma	0.99

Table 2: Teacher policy

Parameter	**Value**
max episode length	300
learning rate	0.001
batch size	12 seq.
epochs	100
optimizer	Adam

Table 3: Student policy

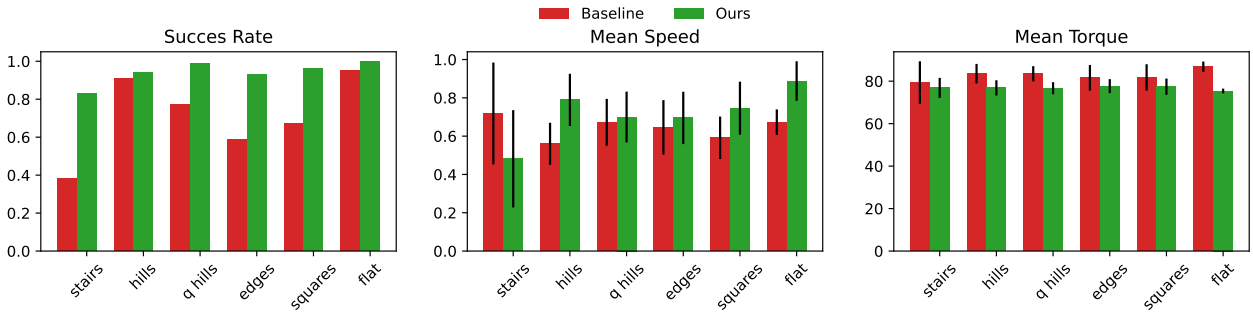


Figure 7: Policy performance metrics for the five different terrain modes as well as flat terrain. Error bars denote the standard deviation of the mean. Our exteroceptive policy achieves higher success rates and speeds as it can take more decisive actions, thanks to its ability to gather information about the environment in advance.

4.3 Experiments

We conduct a number of simulation based experiments to evaluate the performance of our exteroceptive policy against the proprioceptive baseline policy. In all experiments our policy has access to *nominal* noise exteroception, and is commanded to walk forward, unless otherwise mentioned. For all results 100 episodes were attempted and the results averaged. We include a supplementary video providing a visual representation of the experiments. [1]

4.3.1 Maximum speed over various terrains

We record mean speed, average actuator torque and the success rate for all terrains in the training curriculum. A success is defined as the episode completing at 300 timesteps without the robot falling over. The results are shown in Figure 7. Our policy outperforms the baseline policy on all terrains in terms of success rate, with near perfect scores on all terrains but stairs. Although the proprioceptive baseline is able to achieve a near 100% success rate on hills and flat terrain, its performance is worse on quantized hills, edges and squares. The largest outperformance is found in the stairs terrain, and upon further investigation we find that most failures occur during the descent of the staircase. This underperformance of the baseline policy is likely due to the large vertical change in the stairs, quantized hills, edges and squares terrains, which the robot is unable to anticipate due to the lack of exteroceptive information. To investigate this further we conduct a more detailed analysis of the staircase terrain in the next experiments.

Our exteroceptive policy is able to achieve higher speeds than the baseline policy on all terrains but stairs. This is due to the fact that the exteroceptive policy is able to take more decisive actions, as it can gather information about the environment in advance. The baseline policy is forced to resort to more conservative gaits, at slower speeds and lifting feet higher. The slow speed on stairs for our exteroceptive policy is caused by a combination of caution and the policy veering slightly off course during stair descent. This behaviour emerged from the training process and we hypothesize that, although undesirable, it is beneficial for the policy success, as it effectively lengthens the run of the stairs, making it easier to find a suitable foothold.

Applied torque is lower for our policy than the baseline policy on all terrains, indicating that less

[1] https://youtu.be/B3Qr-7ZZHZQ

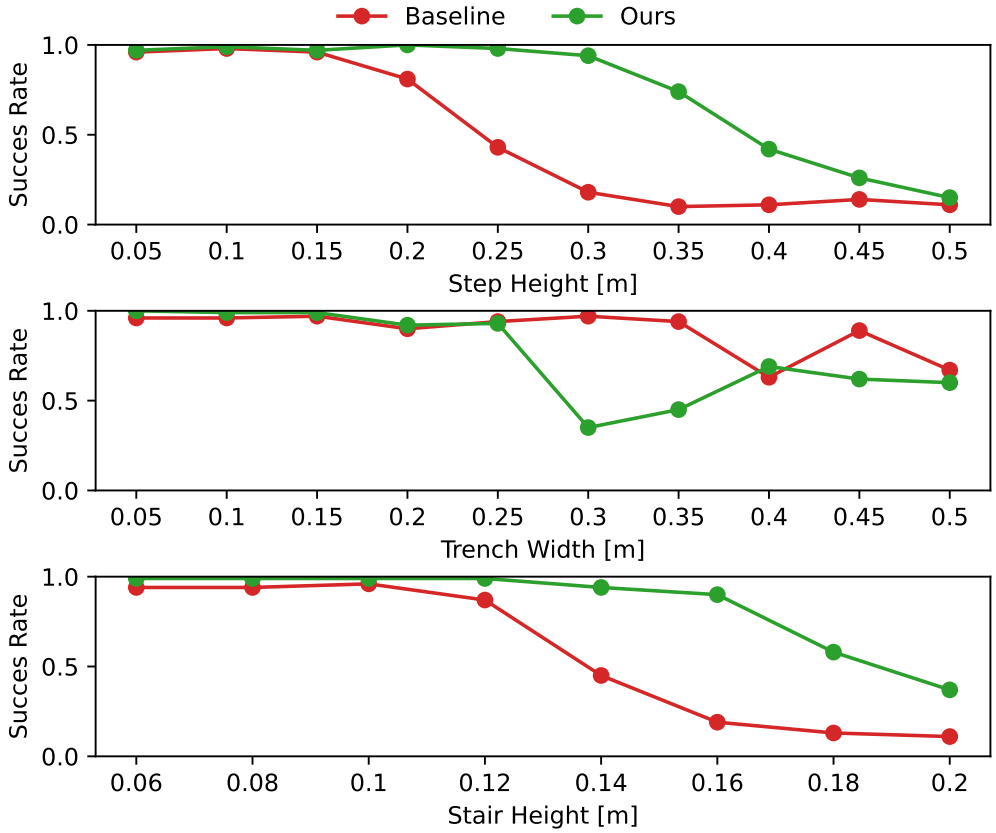


Figure 8: Success rate metrics for stepping over a single step (top), a trench (middle) and walking downstairs (bottom). In all cases the exteroceptive policy is able to traverse more challenging terrains than the proprioceptive baseline policy.

energy is consumed despite walking at higher speeds. Our exteroceptive policy thus generates more energy efficient gaits. We did not investigate whether the difference in energy consumption is large enough to offset the added computational cost of the exteroceptive encoder.

4.3.2 Success rate over a step

We command both policies to go forward over flat terrain with a 1 meter wide step of varying heights at 1 meter from the starting position and record the success rate. The results are shown in Figure 8. Our policy outperforms the baseline policy, reliably traversing over steps up to 30 cm in height, while the baseline policy starts to fail at 20 cm. This experiment clearly shows the advantage of exteroceptive information in the case of a step, as the policy can anticipate the step and take a more reliable action.

4.3.3 Success rate over a trench

We command both policies to go forward over a 50 cm deep trench of varying widths to gauge the effects of exteroception on the policy's ability to avoid dangerous areas. The results are shown in Figure 8. Surprisingly, the proprioceptive baseline is able to outperform our exteroceptive policy. We find that the exteroceptive policy does not avoid the trench, but tries to step inside. We hypothesize this is caused by the policy not encountering terrains with dangerous areas during training.

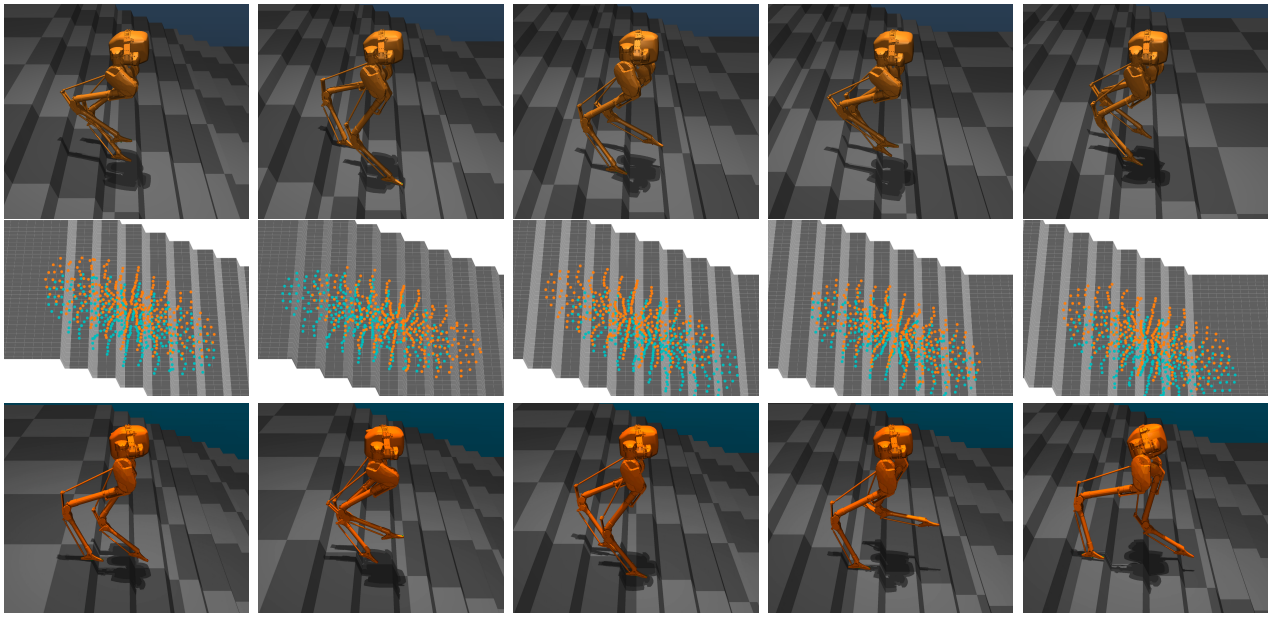


Figure 9: The top two rows shows our exteroceptive policy successfully walking down a staircase of ten 12 cm steps, with the second row showing the noisy exteroceptive inputs at the same timesteps. The bottom row shows the proprioceptive baseline policy attempting to walk down the same staircase, with less success.

4.3.4 Success rate for stair descent

As can be seen from the success rate results in Figure 7, the stairs terrain is most difficult for both policies. Specifically, most failures occur when the robot is descending stairs. We believe this is due to Cassie's morphology since the steep backwards incline of the legs make them interfere with the stairs. Effectively causing the useable portion of a stair run to be shorter and requiring more precision in foot placement. To investigate whether exteroception is beneficial in this environment we command both policies to go down staircases of 10 equal steps and vary the step heights while recording the success rates. All staircases use the same run of 35 cm. The results are shown in Figure 8. Our exteroceptive policy is able to walk down stairs with step heights up to 16 cm with near 100% success rate, while the baseline policy starts failing at 12 cm. This clearly shows the advantage of exteroception in this environment. Figure 9 shows both policies traversing down stairs of 12 cm step height.

4.3.5 Command Following

To gauge the capability of the policy to locomote over irregular terrain according to a user specified command we command the policy to walk in all directions over the squares terrain and record the speed. We choose the squares terrain for this experiment for its high density of changes in terrain height. Figure 10 shows the commanded linear and angular velocities and the policies' responses to them. The policies are able to follow the commands in the x direction over rough terrain accurately, albeit with a delay. Similar results apply for speed achieved in the y direction, however the maximum speed is lower than commanded. We believe this is due to the low range of motion available in the roll actuators of the robot hips, requiring smaller steps and limiting velocity. We found that results are similar on flat terrain, confirming that terrain is not the limiting factor. Lastly, both policies are able to accurately follow the commanded angular velocity over the squares terrain. These results clearly show that the policies have learned to locomote over irregular terrain according to a user specified

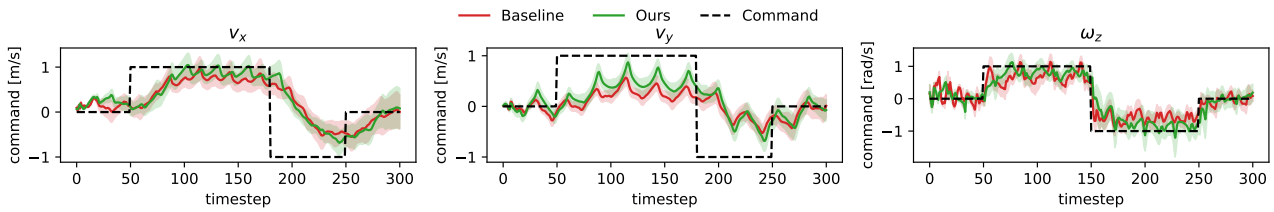


Figure 10: The policy is commanded to walk in all directions over the squares terrain, and the commanded and actual linear and angular velocities are recorded. All plots represent the mean velocity of 100 independent trials. The shaded area denotes the standard deviation of the mean.

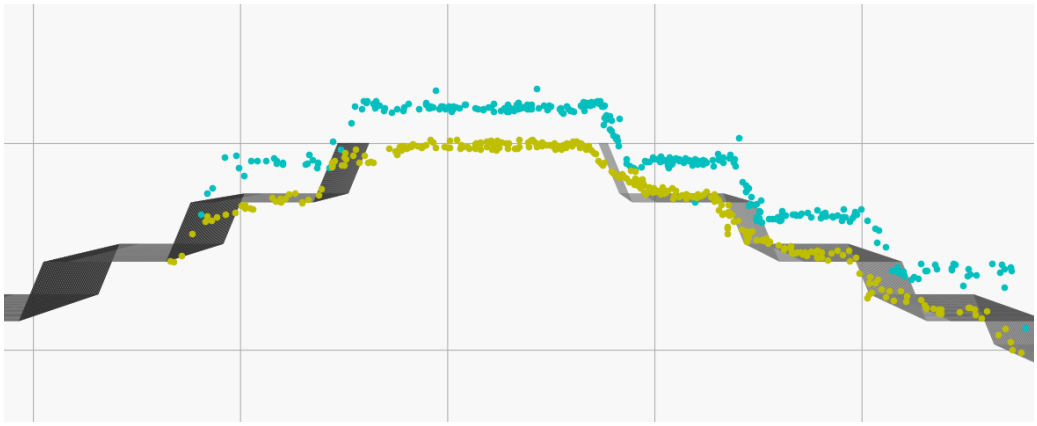


Figure 11: A side view of the sampled height map and reconstructed height map for the left foot during stair traversal. Cyan is the sampled noisy height map, yellow is the reconstruction produced by the belief decoder. The policy is able to denoise large offsets in the height map.

command, with our exteroceptive policy slightly outperforming in terms of speed.

4.3.6 Dealing with spurious exteroceptive inputs

An important aspect of the belief encoder system is the ability to interpret noisy exteroceptive inputs along with proprioception to form an accurate belief of the environment. In order to demonstrate this capability we show a view of the exteroceptive reconstruction produced by the belief decoder in Figure 11. Our policy is able to denoise large, sometimes alternating offsets in the height map. Additionally, the belief encoder is able to eliminate outliers, while keeping an accurate representation of the terrain.

5 Conclusion

5.1 Discussion

The field of bipedal locomotion has seen tremendous progress over the last few years. Learning-based locomotion policies have advanced the state-of-the-art in many aspects like robustness, capability and speed. These advances are critical in bringing humanity closer to the prospect of useful humanoid robots, which has the potential to propel our technological advancement forward by an enormous leap.

In this study we worked towards this goal by exploring the effects of exteroception on the performance of a bipedal locomotion policy. Concretely, we presented a methodology to learn a bipedal locomotion policy capable of utilizing noisy exteroceptive observations to successfully traverse irregular terrain, while following a user command. In order to do so we employed privileged learning to initially learn a noise free locomotion policy, that then acts as a teacher to train a student policy that can utilize noisy exteroceptive observations. We designed a curriculum, terrain and reward function to enable the policy to learn to locomote over irregular terrains.

We have shown that such a bipedal locomotion policy observing exteroception greatly outperforms a purely proprioception based bipedal locomotion policy when traversing irregular terrains, in terms of robustness, speed and energy efficiency. We also show that this does not come at the cost of the policy's ability to locomote over flat terrain. By ablating the exteroceptive element of the observation in our experiments, we confirm that the outperformance stems from the exteroceptive observations. Furthermore, by adding noise that was specifically engineered to mimic noise by real world sensors, we showed that outperformance is achievable despite the noise.

A critical component for a real world use case is the robustness of the policy, which has been a challenge for exteroceptive policies due to the inherent noise in exteroceptive sensors, causing instability and unreliability. By showing that our policy is robust to such noises we demonstrate the real world feasibility of our approach.

As demonstrated by the Trench Width Experiment (see 4.3.3), our approach's success hinges on the inclusion of test-time situations in train-time observations. The locomotion policy faces challenges from terrains and noise variations unseen during training, highlighting the importance of accurate representations of the test environment during the training phase.

Other limitations of this work encompass several factors. Firstly, our experiments were solely conducted in simulation. While the methods employed have been previously validated on real robots, it is imperative to verify the results on actual hardware to assess their real-world applicability.

Another limitation we observe is the low speed of iteration in experiments, caused by the multiple day training time of the policy. This severely slows the development of the policy, rewards and curriculum. Training a single iteration of the locomotion policies presented in this study can take between 24 and 48 hours to converge. Most of that time is needed, in order to determine whether the current iteration is bug free, and to evaluate alterations for subsequent iterations of the experiment. Adding to that, some time is spent in waiting queues on the shared compute cluster, and coding up new experiments. Overall, one can expect to train around 3-4 major iterations per week. Given the virtually infinite design space of these problems, this leads us to believe that there is much improvement to

be gained from simply increasing the speed of iteration, and more thoroughly exploring rewards, curricula and policy designs.

Another constraint witnessed in our experiments is the interaction of Cassie's legs with specific types of terrain, which impedes smooth locomotion particularly when descending stairs, or descending other forms of steep ledges. This constraint arises from the robot's morphology, as the steep backwards inclination of Cassie's legs causes them to collide with the ground on steep terrains. Although this is a limitation of the robot, it is also a limitation of the policy, as the policy is not able to sufficiently adapt to this morphology.

5.2 Summary of Main Contributions

In this study we contribute to the field of bipedal locomotion control with a demonstration that the integration of exteroception to a bipedal locomotion policy greatly increases its robustness, speed and energy efficiency, despite the noise that is inherent in real world sensors necessary for exteroceptive observations.

Additionally, we add to the field of bipedal locomotion control by developing a policy for locomoting over a wide variety of different terrains, as opposed to flat or predictable terrains such as is common in the field. We present a reward function and curriculum that allows us to train a policy that can successfully navigate a wide variety of terrains, including stairs, edges, squares, hills and flat terrain.

5.3 Future Work

The results presented in this study are promising, however there is still much work to be done before humanoid robots can be deployed in the real world. Extremely high reliability is required to deploy a humanoid robot in the real world on real world tasks. The author is not aware of papers that claim reliabilities on the order of 99.9% for bipedal locomotion. What would happen if we optimize our current methods for learning these locomotion polices to the theoretical maximum? Would we be able to achieve such reliabilities? If not, what is the limiting factor? Is it model size, reward function, RL, the physics simulator, the sensors, or something else? These are important challenges that need to be solved before humanoid robots are reliable enough to be deployed in the real world.

The study presented here is premised on the implicit assumption that computational resources on the robot are constrained, which puts an upper limit on the size of the neural network that can be deployed. However, this assumption may not be accurate, as the continuation of processor miniaturization leads to more potent and energy-efficient high-end chips. This advancement leads us to speculate on the possibilities that may unfold in scenarios where the robot is equipped with compute on the order of 2 PFLOPS [2] which is 3-4 orders of magnitude more than is currently present. Could such a leap in embodied processing capability facilitate the execution of large-scale transformer models on the robot's hardware? Would this, in turn, afford the policy a greater degree of spatial intuition? These are intriguing questions that may enable deployment of much more advanced policies on the robot in the future.

Directly related to that, future work could also explore methods to increase the speed of iteration

[2] 2 PFLOPS $\approx$ 1 H100 GPU, FP16

of the experiments. As discussed in the Discussion, the current approach is limited by its low speed of iteration. Potential avenues to achieve faster training could be faster hardware, or better utilization of the current hardware by optimizing implementations. Some promising methods include the use of faster simulations that run fully on the GPU, such as presented in [18], where speedups on two orders of magnitude are reported. Such optimizations will allow for more thorough exploration of the design space, and potentially lead to even more robust and capable policies.

Bibliography

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

[2] R. Sutton and A. Barto, *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series, MIT Press, 2018.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[5] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.

[7] *Spring-Mass Walking With ATRIAS in 3D: Robust Gait Control Spanning Zero to 4.3 KPH on a Heavily Underactuated Bipedal Robot*, vol. Volume 1: Adaptive and Intelligent Systems Control; Advances in Control Design Methods; Advances in Non-Linear and Optimal Control; Advances in Robotics; Advances in Wind Energy Systems; Aerospace Applications; Aerospace Power Optimization; Assistive Robotics; Automotive 2: Hybrid Electric Vehicles; Automotive 3: Internal Combustion Engines; Automotive Engine Control; Battery Management; Bio Engineering Applications; Biomed and Neural Systems; Connected Vehicles; Control of Robotic Systems of *Dynamic Systems and Control Conference*, 10 2015. V001T04A003.

[8] J. Reher, W.-L. Ma, and A. D. Ames, "Dynamic walking with compliance on a cassie bipedal robot," in *2019 18th European Control Conference (ECC)*, pp. 2589–2595, 2019.

[9] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*, pp. 4559–4566, 2019.

[10] X. B. Peng, *Developing locomotion skills with deep reinforcement learning*. PhD thesis, University of British Columbia, 2017.

[11] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph.*, vol. 37, jul 2018.

[12] 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.

[13] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," *CoRR*, vol. abs/2103.14295, 2021.

[14] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7309–7315, 2021.

[15] J. Siekmann, S. Valluri, J. Dao, L. Bermillo, H. Duan, A. Fern, and J. W. Hurst, "Learning memory-based control for human-scale bipedal locomotion," *CoRR*, vol. abs/2006.02402, 2020.

[16] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.

[17] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, oct 2020.

[18] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.

[19] 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, jan 2022.

[20] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[21] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," 2022.

[22] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," *CoRR*, vol. abs/1912.12294, 2019.

[23] 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*, SCA '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[24] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. Ebert, J. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," *Computer Graphics Forum*, vol. 29, no. 8, pp. 2579–2600, 2010.

[25] 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," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[26] H. Duan, A. Malik, M. S. Gadde, J. Dao, A. Fern, and J. Hurst, "Learning dynamic bipedal walking across stepping stones," 2022.

[27] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-real learning of footstep-constrained bipedal dynamic walking," 2022.

[28] Agility Robotics. Accessed: 2022-12-07.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[30] Agility Robotics, "cassie-mujoco-sim," *GitHub repository*, 2018.

[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.