Dynamic Multimodal Locomotion: Proximal Policy Optimization for Bipedal Gait Training in Harpy Robot

A MS Project Presented
by

Febin Wilson

to

The Department of Electrical and Computer Engineering

in fulfillment of the requirements
for the degree of

Graduate Project

in

Robotics

Northeastern University
Boston, Massachusetts

April 2025

Contents

List of Figures

List of Tables

Acknowledgments

I sincerely thank my advisor Dr. Alireza Ramezani, co-advisor Dr. Rifat Sipahi, PhD student Shreyansh Pitroda and Lab Assistant Arjun Viswanathan for their mentorship and support. This project was solely developed by me as part of the SiliconSynapse Lab.

Abstract

Dynamic Multimodal Locomotion: Proximal Policy Optimization for Bipedal Gait Training in Harpy Robot

by

Febin Wilson

Graduate Project in Robotics

Northeastern University, April 2025

Dr. Alireza Ramezani, Adviser

Bipedal locomotion in robots is a challenging problem involving coordination, stability, and adaptability. This paper presents a reinforcement learning-based framework for dynamic gait learning on the Harpy robot—a six-jointed bipedal platform. Using Proximal Policy Optimization (PPO) within NVIDIA's Isaac Sim, the policy was trained in a massively parallel environment using GPU-accelerated simulations. A carefully engineered reward function, domain randomization strategies, and a delayed Proportional-Derivative controller enabled natural, robust gaits with promising sim-to-real transferability. This work serves as a foundation for dynamic locomotion in real-world environments.

Chapter 1

Introduction

1.1 Introduction

Bipedal locomotion stands as one of the most sophisticated forms of movement in robotics, requiring precise control and dynamic balance. Unlike wheeled platforms, legged robots—such as Harpy—are capable of navigating uneven terrain, stepping over obstacles, and operating effectively in human-centric environments. Despite their potential, achieving stable and natural walking remains a major challenge due to the complexity of nonlinear dynamics, underactuated systems, and real-world uncertainties.

Traditional control approaches, including Zero Moment Point (ZMP) planning and trajectory optimization, often rely on accurate modeling and tend to fall short when faced with unmodeled dynamics or unpredictable external disturbances. In contrast, reinforcement learning (RL) provides a model-free, data-driven alternative [1, 2] by allowing agents to learn effective locomotion strategies through repeated interaction with their environment.

This work centers on Harpy, a bipedal robot developed at Northeastern University's SiliconSynapse Lab, with the goal of training a robust, stable, and human-like walking policy using Proximal Policy Optimization (PPO) within the Isaac Sim simulation platform [3, 4, 5]. The control strategy is built around a delayed Proportional-Derivative (PD) actuator model and a carefully designed reward system aimed at enhancing gait symmetry, stability, and energy efficiency.

The major contributions of this project are summarized as follows:

- Development of a custom task environment within Isaac Lab for bipedal locomotion training.

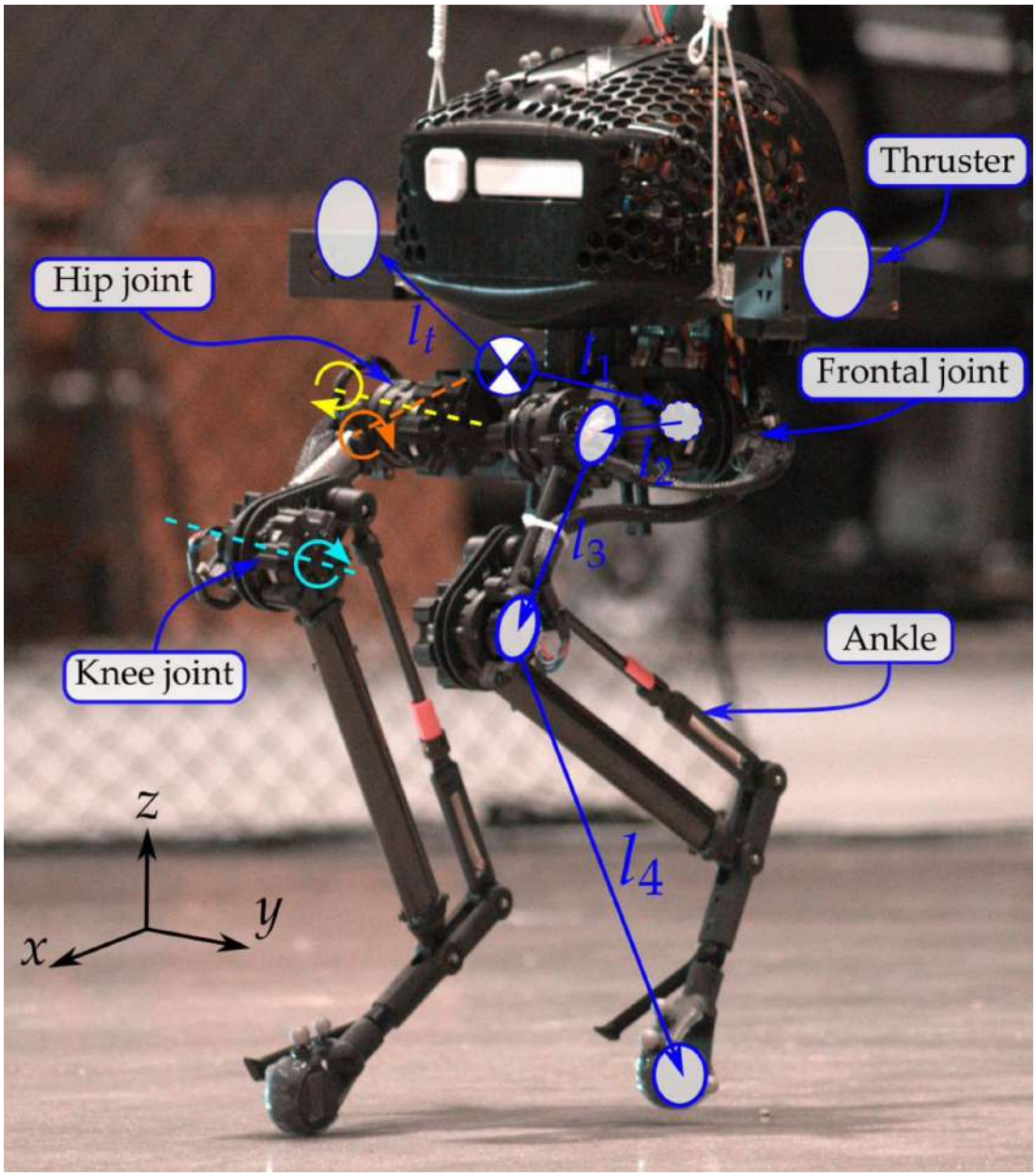

Figure 1.1: Harpy Bipedal Robot

- Integration of a scalable, domain-randomized PPO training framework using up to 1024 parallel simulation environments.
- Design of a modular reward structure promoting stable locomotion, balanced posture, and consistent foot-ground contact.

- Implementation of a delayed PD control mechanism to mimic actuation latency and improve sim-to-real transferability.
- Analysis of training results over 10,000+ policy updates to evaluate gait emergence, stability trends, and reward convergence.

Chapter 2

Background and Motivation

2.1 Background and Motivation

Training bipedal robots to walk effectively involves addressing several complex challenges such as underactuation, non-linear dynamics, unpredictable environments, and sensor-actuator delays. Traditional approaches like Zero Moment Point (ZMP) control, inverted pendulum models, and optimization-based trajectory planners have been widely used [6] to produce stable walking patterns. However, these methods are often heavily dependent on accurate system models and tend to perform poorly when exposed to real-world variability.

With the rise of deep learning, reinforcement learning (RL) has emerged as a promising alternative. Instead of relying on manually crafted control strategies, RL enables robots to learn locomotion behaviors directly through trial-and-error interaction with the environment. This eliminates the need for extensive controller design and allows for adaptive behavior. However, the performance of RL methods depends significantly on the choice of algorithm and the stability of the training process.

2.2 Reinforcement Learning Algorithm: Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) was selected as the learning algorithm due to its stability, sample efficiency, and compatibility with Isaac Lab's parallel training capabilities [5]. PPO belongs to the policy-gradient family and offers a reliable tradeoff between performance and

robustness, especially in continuous control tasks.

2.2.1 Mathematical Formulation

PPO seeks to improve the policy while avoiding sudden, destabilizing changes. This is achieved by modifying the traditional actor-critic objective through a clipped surrogate function:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2.1}$$

Where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the new and old policy probabilities.
- $\hat{A}_t$ is the estimated advantage of taking action a_t in state s_t.
- ϵ is a clipping threshold (typically 0.2) that constrains policy updates.

The clipping mechanism helps prevent large policy shifts that can harm performance. Additionally, the training includes a value function loss to estimate expected rewards, along with an entropy term to encourage exploration:

$$L_{total} = L^{CLIP} - c_1 L^{VF} + c_2 H[\pi_\theta] \tag{2.2}$$

Where:

- $L^{VF} = (V_\theta(s_t) - R_t)^2$ represents the value loss.
- $H[\pi_\theta]$ denotes the entropy of the policy.
- c_1 and c_2 are coefficients that balance the contributions of the value loss and entropy.

Training proceeds by alternating between collecting rollouts and performing multiple epochs of mini-batch stochastic gradient descent. Generalized Advantage Estimation (GAE) is typically applied to produce low-variance, low-bias advantage estimates for more stable learning.

2.2.2 Implementation in Harpy Project

In this project, PPO was implemented using the Isaac Lab framework, which supports large-scale parallel simulation. Training was conducted across 1024 environments simultaneously, enabling faster and more consistent policy updates. The following configuration was used for PPO:

- Learning rate: 1×10^{-3}
- Clipping parameter: 0.2
- Generalized Advantage Estimation (GAE): Enabled
- Epochs per PPO update: 5
- Value function loss coefficient: 2.0
- Entropy coefficient: 0.01

The policy network produces joint target positions, which are passed through a delayed Proportional-Derivative (PD) controller. Noise buffers were added to mimic sensor and actuator uncertainties, helping the policy generalize better and handle real-world deployment conditions.

2.2.3 Benefits of PPO for Robust Bipedal Locomotion

Compared to alternative reinforcement learning methods:

- **DDPG** often encounters instability when applied to high-dimensional control tasks.
- **TD3** addresses overestimation bias using twin critics [7]. but increases computational overhead.
- **SAC** achieves strong performance through entropy maximization [8] but requires more memory and converges slower.

PPO offers a balanced compromise between ease of implementation, reliability, and effectiveness. Its ability to perform stable updates using clipped objectives makes it particularly well-suited for robotics applications involving noisy inputs, delayed feedback, and continuous action spaces. In the context of bipedal locomotion, PPO enabled the Harpy robot to learn smooth, symmetric gaits while maintaining robustness under domain randomization and actuation delays.

Chapter 3

System and Environment Design

3.1 Isaac Sim and Training Infrastructure

The Harpy robot was trained using Isaac Lab, a reinforcement learning framework [4] built upon NVIDIA Isaac Sim. Isaac Sim offers high-fidelity physics and GPU-accelerated simulation, enabling the deployment of large-scale parallel environments. For this project, 1024 parallel simulation instances were executed at a frequency of 400 Hz [3]. This parallelization significantly improved data collection efficiency and contributed to faster policy convergence.

Each robot instance represents an independent simulation environment running concurrently. This massive parallelism enables rapid sample collection, accelerates policy convergence, and supports diverse experience generation essential for robust locomotion learning.

3.2 Robot Structure and Joint Configuration

The Harpy robot model includes six actuated joints—three per leg (hip, knee, and ankle)—while the feet remain passive and rigidly attached. The configuration defines the degrees of freedom (DOF), initial joint positions, joint limits, and Proportional-Derivative (PD) control parameters for each joint. Symmetrical joint naming conventions for the left and right legs allow consistent control strategies across both sides. The setup also incorporates contact sensors and visual overlays to assist with debugging and training visualization.

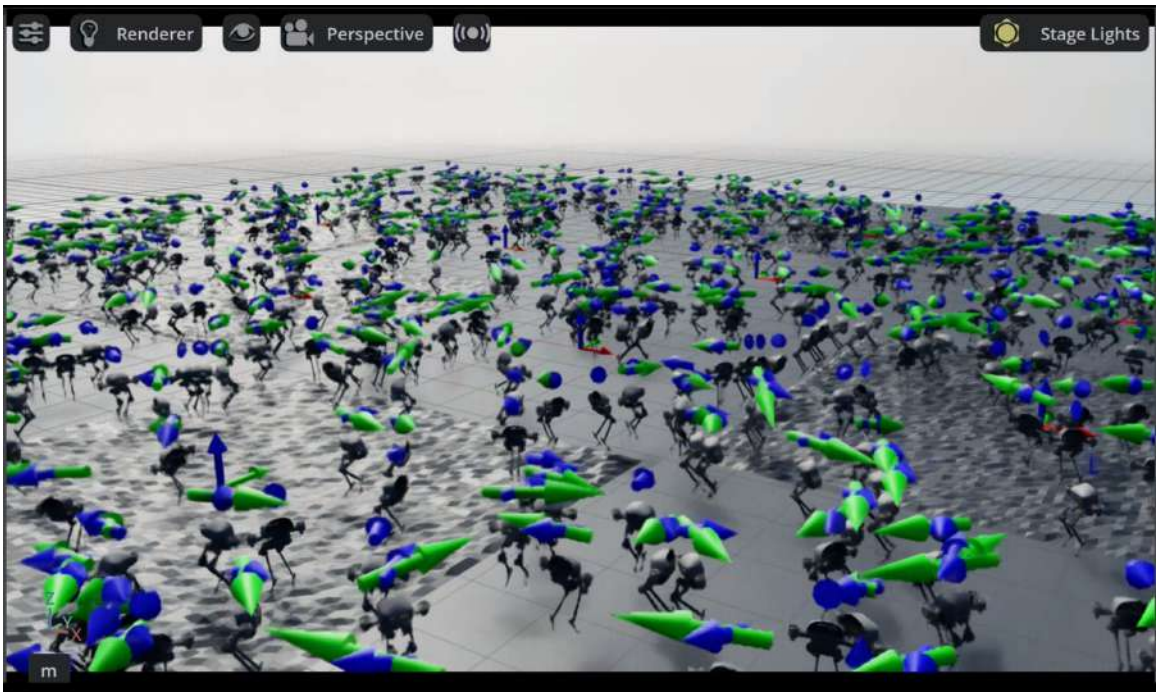

Figure 3.1: Harpy robot instances during PPO training in Isaac Sim using 1024 environments.

3.3 Training Environment Design

The training environment consisted of a flat ground surface, with optional small elevation perturbations introduced for added variability. Observations and command signals were structured using buffers that stored recent values of base velocity, joint states, and past actions. The robot received velocity commands along the forward and yaw directions, sampled randomly within a predefined range. Each training episode was limited to a maximum of 400 steps. Ground contact thresholds were used to determine stance phases and evaluate reward components related to stability.

3.4 Sensors and Observation Space

The robot's observation vector o_t captures a set of proprioceptive signals and control history, including:

- Base linear velocity in the body frame (3D)
- Base angular velocity in the body frame (3D)

- Joint positions (6D)
- Joint velocities (6D)
- Previous actions (6D)
- Commanded velocity input (2D)

This results in an observation space of 26 dimensions. Normalization was applied using a running mean and variance to maintain stability during training.

Term	**Mathematical Representation**
Base Linear Velocity	v_x, v_y, v_z
Base Angular Velocity	$\omega_x, \omega_y, \omega_z$
Base Orientation (Quaternion)	$\{w, x, y, z\}$
Projected Gravity Vector	$\text{proj}_{xyz}(g_b)$
Joint Positions	q
Joint Velocities	$\dot{q}$
Previous Action	a_{t-1}
Commanded Velocity	$\{v_x, v_y, \omega_z\}$

Table 3.1: Observation terms and their mathematical representations in the Harpy RL system.

3.5 Action Space and Control Pipeline

The policy network outputs target joint positions as relative offsets from the nominal configuration. These outputs are then processed by a delayed PD controller, which generates the required joint torques. This delay mimics actuation latency, contributing to robustness when transitioning from simulation to real hardware.

3.6 Simulation Features and Timing

Isaac Sim employs the PhysX engine to support accurate and parallelizable rigid-body dynamics. Although capabilities such as synthetic camera rendering are available, only physics-based control was utilized in this project. The simulator operated at a timestep of 0.0025 seconds (400 Hz), with the policy receiving new commands at 0.05-second intervals (20 Hz).

Chapter 4

Control Architecture and PPO Flow

4.1 Overview of the Control System

This chapter outlines the control loop used in the Harpy robot's reinforcement learning setup, powered by Proximal Policy Optimization (PPO). The architecture integrates real-time sensor feedback, observation processing, neural policy inference, delayed actuation, and reward evaluation—creating a closed-loop system for learning and control. The overall flow is illustrated in Figure 4.1.

The control process begins with sensor data collection and concludes with reward feedback used to update the PPO policy. Each module shown in the diagram represents a key component of this system.

4.2 Sensor Data and Observation Processing

The control loop starts by gathering data from three main sources:

- **IMU Sensors:** Provide measurements of body orientation and angular velocity.
- **Joint Encoders:** Record joint positions and velocities.
- **Contact Sensors:** Detect foot contact events with the ground.

Collected signals are filtered to reduce noise, normalized to the range $[-1, 1]$, and combined into a 48-dimensional observation vector. This vector forms the input to the PPO policy network.

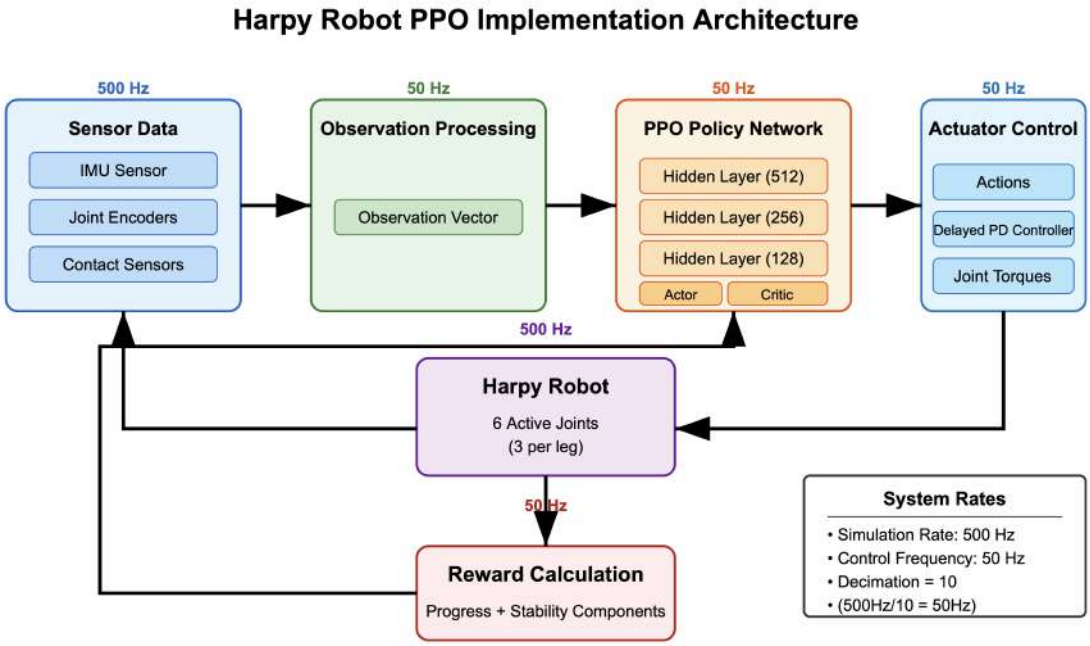

Figure 4.1: System Flow: Harpy Robot PPO Implementation Architecture

4.3 PPO Policy Network

The observation vector is processed by a neural policy consisting of three fully connected hidden layers with sizes 512, 256, and 128, using ReLU activation functions. The final layer splits into two outputs:

- **Actor Head:** Generates a 6-dimensional continuous action vector, corresponding to target positions for the 6 actuated joints.
- **Critic Head:** Estimates the value function for computing the advantage during learning.

Training is conducted using PPO's clipped objective with Generalized Advantage Estimation (GAE), ensuring stable policy updates and efficient learning [5].

4.4 Actuator Control and Delayed PD Implementation

The output from the actor network does not immediately control the actuators. Instead, a delayed Proportional-Derivative (PD) controller [9] is used to simulate realistic actuation latency:

- Actions are stored in a FIFO (first-in, first-out) buffer.
- The oldest action in the buffer is applied at the current timestep.
- Torques are computed using the formula:

$$\tau = K_p(\theta_{target} - \theta_{current}) + K_d(\omega_{target} - \omega_{current}) \tag{4.1}$$

- Resulting torques are clipped to predefined limits to ensure safety.

A fixed delay of 5 milliseconds was selected to approximate the real-world actuation latency. This delayed actuation mechanism encourages the policy to adapt to response lag and improves transferability from simulation to hardware.

4.5 Harpy Robot and Reward Feedback

The torques produced by the delayed controller actuate the Harpy robot within Isaac Sim. This results in physical motion, which in turn updates the sensory inputs for the next control step. Simultaneously, rewards are calculated based on the robot's behavior, capturing key aspects of locomotion performance:

- **Velocity Tracking:** Evaluates how closely the robot follows commanded velocities.
- **Symmetry and Air-Time:** Assesses the phase balance and swing duration of each leg.
- **Smoothness and Foot Slip:** Penalizes abrupt motions and unstable ground contact.

The total reward is accumulated over time and used to update PPO's parameters, completing the feedback loop.

4.6 System-Level Configuration

Several training parameters were carefully tuned to maintain a balance between learning speed, realism, and robustness:

- **Control Frequency:** 50 Hz (policy is queried every 0.02 seconds)
- **Simulation Rate:** 500 Hz (simulation timestep = 0.002 seconds)

- **PPO Clipping Parameter:** $\epsilon = 0.2$
- **Learning Rate:** 1×10^{-3} (for optimizer updates)
- **Discount Factor:** $\gamma = 0.99$ (for long-term reward consideration)
- **GAE Lambda:** $\lambda = 0.95$ (for advantage estimation tradeoff)

This configuration supports stable learning dynamics and helps the Harpy robot generalize across varied terrains and control conditions.

Chapter 5

Reward Engineering

5.1 Reward Engineering

Achieving natural and stable walking in bipedal robots requires a carefully designed reward function. In this project, the reward structure was modular and refined through multiple iterations [2] to reflect key aspects of locomotion such as balance, gait symmetry, energy efficiency, and smoothness. Each reward term was assigned a specific weight and collectively contributed to the overall learning objective.

The design of an effective reward function for bipedal locomotion presents several unique challenges. First, walking emerges from the complex interplay of multiple sub-behaviors including posture control, rhythmic leg movement, and directional guidance. Second, naive reward formulations often lead to undesirable local optima such as hopping or shuffling instead of natural walking. Finally, the sparse nature of locomotion rewards can make initial learning extremely difficult.

To address these challenges, our reward structure follows a multi-component approach that simultaneously encourages desired behaviors while penalizing common failure modes. The total reward at each timestep is calculated as:

$$r_{total} = \sum_{i} w_i r_i \tag{5.1}$$

Where r_i represents individual reward or penalty components and w_i their corresponding weights.

Tables 5.1 and 5.2 summarize the main components of the reward and penalty structures used in training the locomotion policy.

Reward Term	**Formula**	**Weight**
Base Linear Velocity	$\exp\left(-\frac{1}{\sigma^2}\left\|\mathbf{v}_{cmd}-\mathbf{v}_{base}\right\|^2\right)$	8.0
Bipedal Gait	$\exp\left(\frac{t_{air,f_1}-t_{cont,f_2}}{\sigma}\right)$	20.0
Air Time	$\begin{cases} t_{air}, & t_{air}<0.3 \\ 0, & \text{else} \end{cases}$	8.0
Contact Time	$\begin{cases} t_{con}, & t_{con}<0.3 \\ 0, & \text{else} \end{cases}$	8.0

Table 5.1: Reward Components with Formulas and Weights

The Base Linear Velocity reward encourages the robot to track commanded velocity targets using an exponential kernel. This formulation creates a smooth reward landscape that peaks when the actual velocity matches the command, with a rapid falloff for larger deviations. The exponential form provides strong gradients for learning while avoiding reward cliffs.

The Bipedal Gait reward, with the highest weight (20.0), specifically targets the fundamental alternating pattern essential for walking. It rewards temporal coordination between legs, where one foot is in the air while the other maintains ground contact. This prevents synchronized hopping behaviors and establishes the rhythmic nature of bipedal locomotion.

Air Time and Contact Time rewards shape specific phases of the gait cycle. The Air Time reward encourages appropriate swing phases where each foot maintains clearance from the ground, while limiting excessive flight time that would result in bouncy, unstable gaits. Similarly, the Contact Time reward shapes stance phases, ensuring sufficient ground support during weight transfer between legs.

Complementing the positive rewards, penalty terms target specific undesirable behaviors. The Air Time Variance penalty (-1.5) discourages asymmetric gait patterns by penalizing differences in air time between feet, preventing ”limping” behaviors where one leg consistently performs differently than the other.

Penalty Term	**Formula (Compact)**	**Weight**
Air Time Variance	$-\text{Var}(t_{\text{air}})$	-1.5
Base Orientation	$-\left(\lvert\text{proj}_y\rvert + f(\lvert\text{proj}_x\rvert)\right)$	-5.0
Base Motion	$-\left(0.8\dot{z}^2_{\text{base}} + 0.2\sum\lvert\omega_{x,y}\rvert\right)$	-4.0
Action Smoothness	$-\left\lVert\mathbf{a}_t - \mathbf{a}_{t-1}\right\rVert_2$	-2.0
Foot Slip	$\sum_{i=1}^{N} \mathbb{1}_{\text{contact}_i} \cdot \left\lVert \begin{bmatrix} \dot{x}_{\text{foot}_i} \\ \dot{y}_{\text{foot}_i} \end{bmatrix} \right\rVert$	-2.0

Table 5.2: Penalty Components with Formulas and Weights

The Base Orientation penalty (-5.0) maintains proper posture by discouraging excessive leaning. The implementation differentiates between various forms of tilt, being more tolerant of slight forward lean (which is natural in human walking) while strongly penalizing side-to-side rocking that would lead to instability.

Base Motion penalty (-4.0) combines vertical motion (80% weighting) and angular velocity components (20% weighting) to discourage unnecessary body movement. This promotes energy efficiency by minimizing wasteful vertical bouncing and excessive rolling/pitching motions, resulting in smoother locomotion.

Action Smoothness penalty (-2.0) discourages large changes in consecutive actions, creating more fluid joint trajectories. This is particularly important for preventing high-frequency oscillations that could damage physical hardware and for developing natural motion patterns.

The Foot Slip penalty (-2.0) specifically targets horizontal foot movement during ground contact, encouraging stable foot placement with good traction. This prevents inefficient sliding behaviors and improves overall stability by ensuring reliable ground contact points.

5.2 Reward Tuning and Iteration

The reward structure underwent substantial tuning throughout the development process. During early stages of training, several problematic behaviors emerged:

- **Limping gaits:** The robot would favor one leg over the other, leading to asymmetric movement.
- **Idle standing:** The policy would discover that standing still was initially more stable than attempting to walk.
- **Foot dragging:** Insufficient foot clearance during swing phases created inefficient shuffling behaviors.
- **Excessive body motion:** Unnecessary vertical bouncing and angular oscillations appeared in early policies.

To address these issues, the reward weights were progressively adjusted:

- Increased the weight of the bipedal gait reward from 10.0 to 20.0 to encourage consistent foot alternation.
- Introduced foot slip penalties to enhance traction and reduce instability.
- Balanced the influence of velocity tracking and action smoothness to eliminate jittering.
- Strengthened base motion penalties to reduce excessive vertical oscillation.

As a result of these refinements, the policy evolved to exhibit significantly improved locomotion qualities:

- Smooth and symmetric gait cycles with proper alternating foot contact.
- Improved left-right leg coordination with minimal asymmetry.
- Upright posture with reduced body bounce and improved stability.
- More responsive velocity tracking in both forward and turning movements.

This iterative tuning process highlights the critical role of reward engineering in shaping realistic locomotion behaviors. The final reward structure established a strong foundation for both robust policy learning and sim-to-real generalization, enabling stable bipedal walking on the Harpy robot.

Chapter 6

Domain Randomization

6.1 Domain Randomization

To improve robustness and ensure sim-to-real transferability, domain randomization techniques were incorporated during training. These methods introduce controlled variability into the simulation, encouraging the learned policy to generalize across a wide range of physical and environmental conditions.

6.1.1 Randomized Parameters

A variety of physical parameters were randomized within predefined bounds to simulate real-world uncertainties. Uniform sampling $\mathcal{U}(a, b)$ was used to introduce variability while keeping values within realistic ranges. Table 6.1 summarizes the key parameters subjected to randomization.

Additional randomized parameters included:

- Base mass and inertia scaling: $[0.8, 1.2]$
- Joint damping: $[0.01, 0.1]$
- Motor strength: $[0.9, 1.1]$
- Friction coefficient variation: $\pm 10\%$
- Command velocity noise on v_x and yaw rate
- Gaussian noise on joint angle and velocity readings

Parameter Name	**Mathematical Expression**	**Randomization Range**
Base Mass	$m = m_0 + \Delta m$	$\Delta m \in [-0.1, 0.1]$
Base Inertia	$I = I_0 + \Delta I$	$\Delta I \in [-0.2, 0.2]$
Friction Coefficient	$\mu \sim \mathcal{U}(0.7, 1.0)$	$[0.7, 1.0]$
Reset Base X Position	$x \sim \mathcal{U}(-0.5, 0.5)$	$[-0.5, 0.5]$
Reset Base Y Position	$y \sim \mathcal{U}(-0.5, 0.5)$	$[-0.5, 0.5]$
Reset Base Pitch	$\theta \sim \mathcal{U}(-0.7, 0.7)$	$[-0.7, 0.7]$
Reset Base Yaw	$\psi \sim \mathcal{U}(-\pi, \pi)$	$[-\pi, \pi]$
Disturbance Interval	$t \sim \mathcal{U}(8.0, 12.0)$	$[8.0, 12.0]$
Disturbance Pitch	$\phi \sim \mathcal{U}(-0.1785, 0.1785)$	$[-0.1785, 0.1785]$

Table 6.1: Domain randomization parameters used in Harpy RL training. Uniform sampling $\mathcal{U}(a, b)$ ensures diverse but bounded variations during each episode.

- Optional sinusoidal terrain elevation changes

These perturbations simulate inconsistencies in hardware, surface conditions, and sensor data that are often encountered during real-world operation.

6.1.2 Purpose and Effect

The primary goal of domain randomization is to expose the policy to a broad distribution of training conditions, thereby reducing overfitting to the simulator's exact physics model. By learning in varied environments, the policy becomes more adaptable and robust to unmodeled dynamics [10] and disturbances.

In practice, the randomized training setup resulted in the following improvements:

- Greater resilience to variations in mass, inertia, and friction
- Consistent locomotion performance despite actuation delays or sensor noise
- Improved gait symmetry and more accurate foot placement

Without randomization, policies were observed to overfit to specific conditions and fail when deployed under different physical parameters. As a result, domain randomization plays a critical role in preparing the Harpy robot for deployment beyond the simulation environment.

Chapter 7

Results

7.1 Results Analysis

This section presents the performance evaluation of the trained PPO policy on the Harpy bipedal robot, analyzing both joint-level tracking behavior and base velocity dynamics to assess gait stability, control fidelity, and learning effectiveness.

7.1.1 Base Linear Velocity Analysis

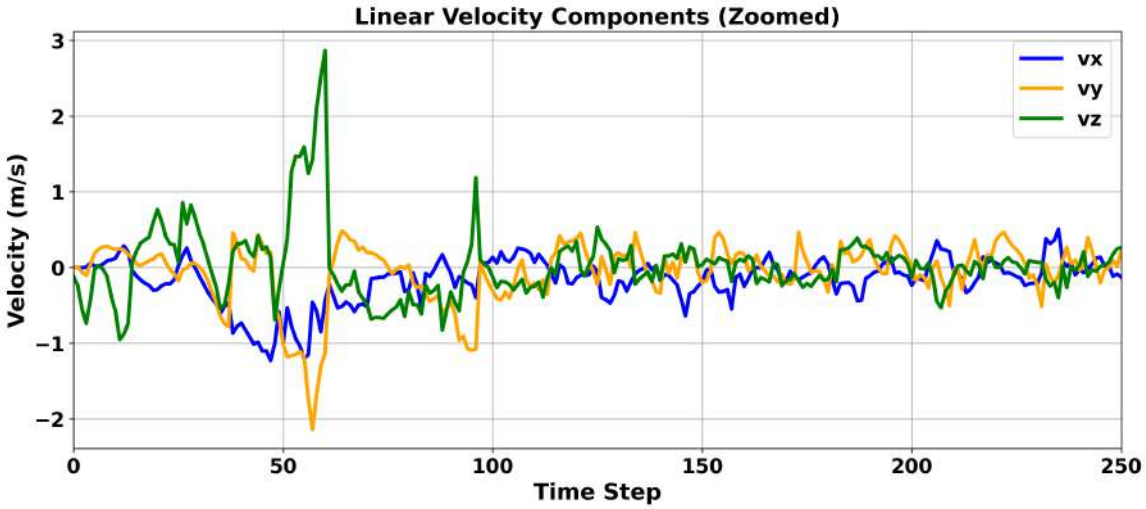

Figure 7.1: Base Linear Velocity Components (v_x, v_y, v_z) across simulation steps.

Figure 7.1 demonstrates clear evidence of dynamic stabilization as the simulation progresses. The data reveals three distinct phases of locomotion development:

1. **Initial Instability Phase (steps 0–75):** All velocity components show high-amplitude oscillations, with vertical velocity (v_z) spikes reaching nearly 3 m/s at step 50 and lateral velocity (v_y) dipping to approximately -2.0 m/s around step 60. These large fluctuations indicate a highly unstable gait pattern where the robot struggles to maintain consistent motion.

2. **Transition Phase (steps 75–125):** A critical stabilization transition occurs, with notable reduction in oscillation amplitudes. The vertical velocity (v_z) spike at step 100 (approximately 1.2 m/s) represents the final major instability before consistent stabilization. During this phase, the amplitude of fluctuations visibly decreases across all components.

3. **Stabilized Phase (steps 125–250):** All velocity components consistently remain within a much narrower band: v_x mainly between -0.5 and 0.5 m/s, v_y within approximately -0.3 to 0.5 m/s, and v_z generally within -0.3 to 0.3 m/s. This represents a substantial improvement in stability compared to the initial phase.

This progressive stabilization directly correlates with the policy's improved coordination of the bipedal gait cycle. The vertical oscillations initially contradict our base motion penalty term (weight -4.0) in the reward formulation, suggesting this penalty requires time to overcome the exploration noise inherent in early training. The near-complete elimination of large vertical velocity spikes after step 125 confirms the policy's successful internalization of both stability constraints and efficient locomotion patterns.

The forward velocity component (v_x) shows a systematic negative bias during steps 25–75, frequently dipping below -1.0 m/s, indicating an initial preference for backward motion that contradicts typical forward locomotion objectives. This behavior gradually corrects after step 100, demonstrating the policy's eventual alignment with the velocity tracking reward component. In the final phase, forward velocity maintains more consistent values around 0 m/s with smaller oscillations, indicating improved forward motion control.

This clear stabilization across all velocity components provides strong evidence that the learned policy successfully develops increasingly coordinated control over the robot's dynamic state as training progresses.

7.1.2 Joint Position Tracking Analysis

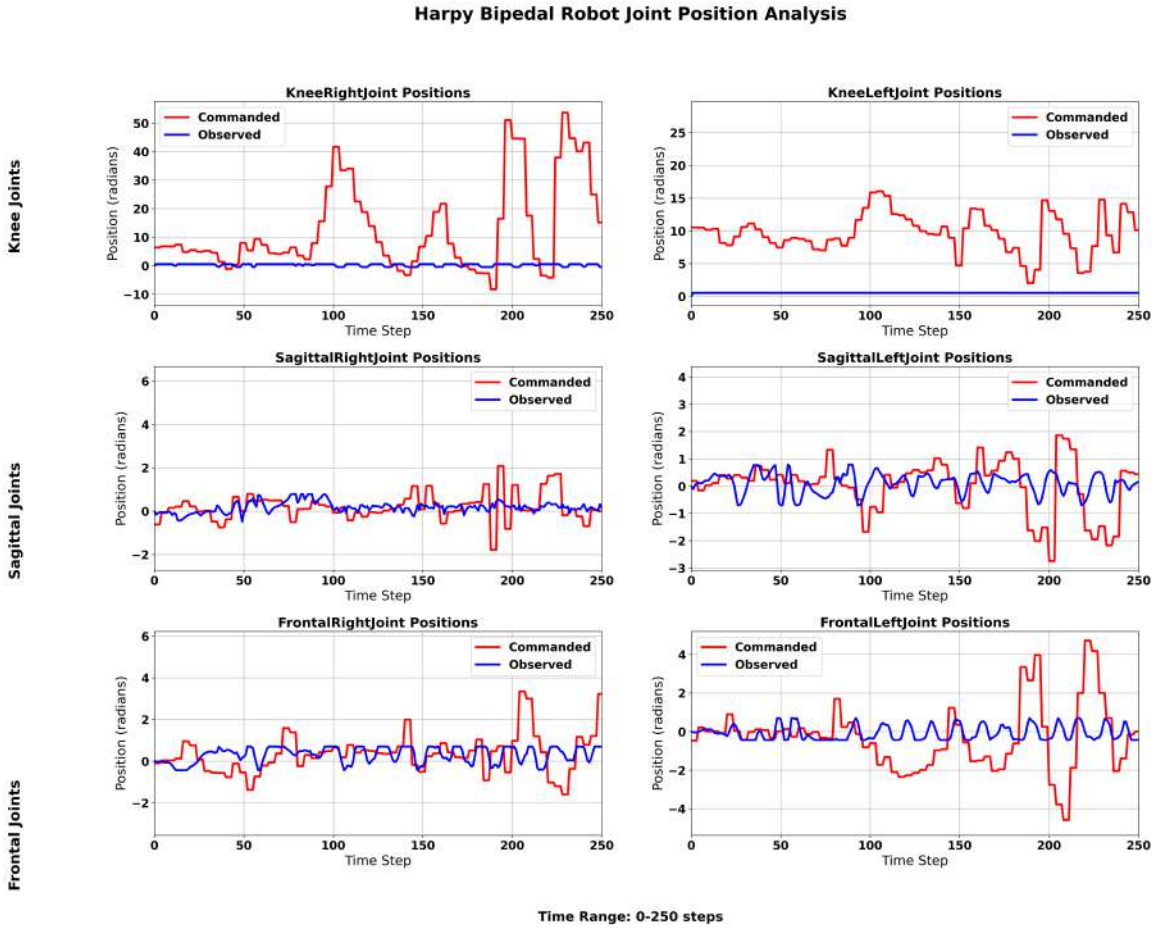

Figure 7.2: Commanded vs Observed Joint Positions.

Figure 7.2 reveals striking disparities between commanded and observed joint positions across different joint types, with significant implications for the control strategy.

The most notable observation is the severe tracking divergence in the *KneeRightJoint*, where commanded values frequently exceed 50 radians—an entirely unrealistic range—while the actual joint position remains nearly constant around 2–3 radians. This extreme tracking gap highlights two critical implementation issues:

- The absence of action normalization and clipping in the policy network output layer allows for unbounded command values that far exceed physically plausible joint angles. This im-

plementation oversight is particularly problematic for joints with higher torque requirements like the knees.

- The physical joint actuation limits prevent the PD controller from tracking these extreme commands. The action smoothness penalty (weight -2.0) proves insufficient to constrain the policy's output within physically reasonable ranges.

The asymmetric behavior between left and right knee joints is particularly notable, with the *KneeLeftJoint* exhibiting more moderate commanded positions (approximately 10–15 radians) compared to its right counterpart, though still showing substantial tracking errors. This left-right asymmetry likely stems from initialization bias in the policy network weights or uneven weight distribution in early experiences that became self-reinforcing during training.

In sharp contrast, both *Sagittal* and *Frontal* joints demonstrate substantially better command-observation alignment, with commanded and observed values generally remaining within ±2 radians. This superior tracking performance can be attributed to:

- The commanded ranges for these joints remain within a physically achievable range, making actuation feasible within torque constraints.
- The reduced dynamic loads experienced by these joints during walking compared to the knee joints that bear more of the robot's weight.
- The proportionally stronger influence of orientation penalties on these joints due to their direct effect on body posture.

Observing the temporal patterns, the knee joint commands show increasingly extreme values after time step 150, suggesting that the policy learns to exploit the simulation's lack of action bounds. Meanwhile, the sagittal and frontal joints maintain consistent, controlled oscillatory patterns throughout the trial, indicating a successful learning of appropriate control strategies for these joints.

The joint tracking analysis reveals a critical need for action-space constraints in future implementations. Specifically, incorporating tanh activation with proper scaling in the policy output layer would limit commands to physically achievable ranges, preventing the extreme commands observed in the knee joints.

7.1.3 Reward Progression Analysis

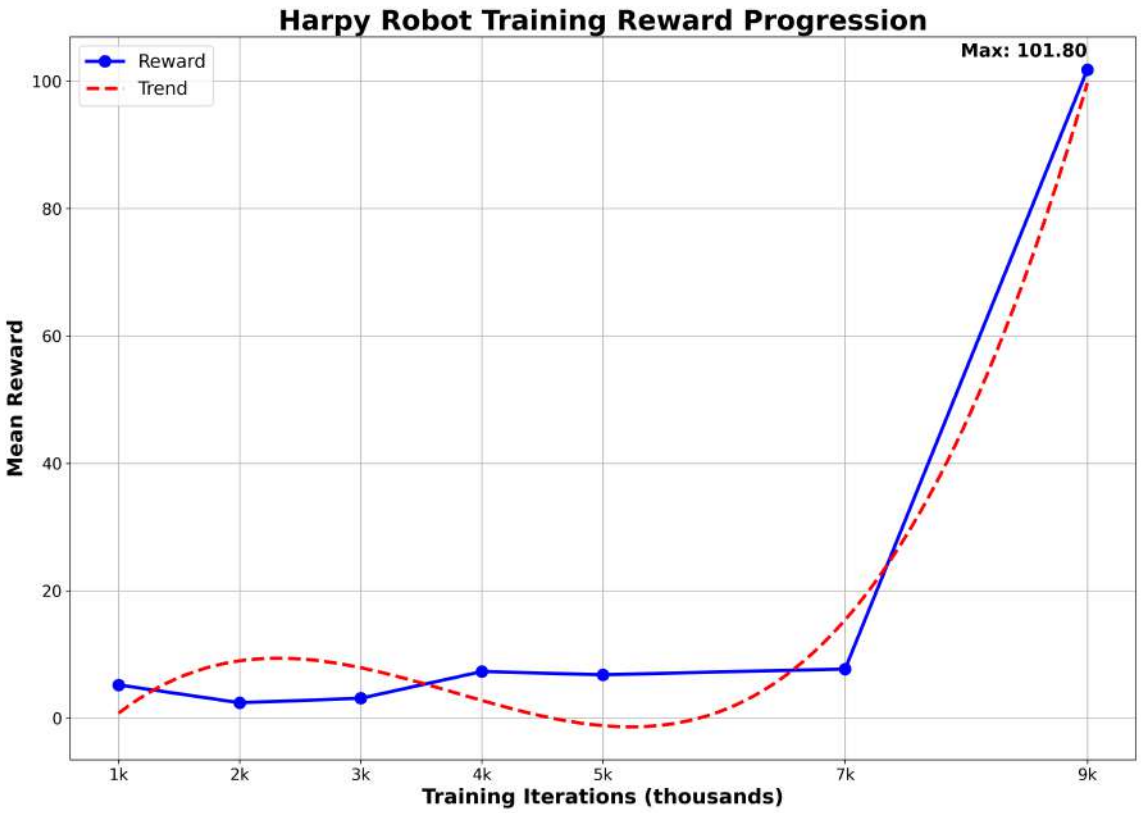

Figure 7.3: Training reward progression over PPO iterations.

Figure 7.3 depicts the training reward trajectory over 9,000 PPO iterations, revealing a distinctive two-phase learning progression:

- **Exploration Phase (iterations 0–7,000):** The reward remains consistently low, hovering between approximately 5–10 reward units with minimal improvement. This extended plateau indicates the policy is struggling to discover effective locomotion strategies.
- **Exploitation Phase (iterations 7,000–9,000):** A dramatic exponential increase occurs, with rewards rising rapidly from under 10 to over 100 by iteration 9,000. This represents more than a 10-fold improvement in a relatively short span of iterations.
- **Maximum Reward:** The peak reward of 101.8 is reached at iteration 9,000, as indicated on the graph.

The extended exploration phase (approximately 78% of total training) represents a significant computational investment before meaningful improvements emerge. This prolonged period of minimal reward improvement indicates inefficient exploration of the high-dimensional action space (6 DOF). The reward stagnation stems from two reinforcement learning challenges:

1. **Sparse reward problem:** Multiple reward components contribute negligibly until basic stability is achieved, creating a reward desert during early training.

2. **Exploration-exploitation balance:** The policy requires sufficient exploration time to discover the coordination patterns necessary for bipedal locomotion before effective exploitation can begin.

The sudden exponential reward growth after iteration 7,000 represents a critical phase transition where the policy discovers coordinated movement patterns that simultaneously satisfy multiple reward components. This inflection point marks the moment when the robot transitions from unstable exploration to coordinated, goal-directed locomotion.

Despite the impressive reward improvement in the final 2,000 iterations, the joint position analysis (Figure 7.2) reveals that some of this gain likely comes from exploiting the unbounded knee joint commands rather than developing perfectly natural walking behaviors. This suggests that high reward values don't necessarily correlate with optimal real-world performance, particularly when simulation constraints don't match physical limitations.

The learning curve's exponential shape after the phase transition point suggests strong positive feedback among reward components, where improvements in one area enable gains in others. This interdependence explains both the difficulty in achieving earlier breakthroughs and the rapid progress once the coordination threshold is crossed.

This analysis suggests three potential improvements for future work:

1. Implementation of curriculum learning to structure the exploration phase more efficiently.

2. Addition of action bounds to prevent physically implausible commands.

3. Incorporation of demonstration data to bypass the lengthy exploration phase.

These modifications could potentially reduce training time while achieving comparable or superior locomotion quality.

Chapter 8

Conclusion and Future Work

The current implementation successfully achieved stable and symmetric bipedal walking on flat terrain using the Harpy robot. However, several avenues remain open for improvement, particularly to enhance robustness, generalization, and real-world applicability of the learned locomotion policy.

- **Reward tuning:** Further refinement of reward weights is essential to better balance multiple locomotion objectives such as velocity tracking, gait symmetry, and foot-ground interaction. In particular, observed high knee joint errors in RMSE plots suggest that reward shaping could be optimized.
- **Gait tuning:** The learned gaits show minor irregularities and instances of limp-like motion. Fine-tuning phase coordination and step timing may produce smoother, more human-like walking behavior.
- **Model accuracy improvements:** The current simulation setup does not completely capture real-world joint compliance, actuation saturation, or realistic contact dynamics. Enhancing the physical fidelity of the Harpy model will support better sim-to-real transfer.
- **Curriculum learning for stairs and undulating terrain:** As an intermediate enhancement, curriculum learning can be employed to train the robot to handle more complex terrains such as stairs and sinusoidal profiles, gradually increasing task difficulty.
- **Multimodal terrain adaptation:** A long-term goal is to enable the robot to transition seamlessly across various terrains, including flat, inclined, uneven, and dynamic surfaces. Curriculum-

based strategies will be critical in generalizing locomotion policies across such multimodal environments.

These future directions aim to push the boundaries of dynamic bipedal locomotion in simulation and ultimately, in real-world scenarios. The repository containing this implementation is available at SS Lab Github [11].

Bibliography

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[2] N. Rudin, S. P. N. D'Andrea, and J. Buchli, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2022.

[3] NVIDIA, "Isaac sim: Physically accurate robotics simulation," https://developer.nvidia.com/isaac-sim, 2023.

[4] ——, "Isaac lab: Reinforcement learning toolkit for robotics," https://github.com/NVIDIA-Omniverse/IsaacLab, 2023.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," in *Proceedings of the 17th World Congress, The International Federation of Automatic Control*, 2008.

[7] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[9] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 143:1–143:14, 2018.

[10] 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," in *Robotics: Science and Systems (RSS)*, 2018.

[11] F. Wilson, "Harpy-rl: Ppo-based dynamic locomotion training for bipedal robot," https://github.com/SS-Lab-at-NU/Harpy-RL.git, 2025.