TÉCNICO LISBOA

Motion Controller For Natural and Robust Humanoid Locomotion using Reinforcement Learning

Duarte Faria dos Santos

Report for the Second Cycle Integrated Project in

Electrical and Computer Engineering

Supervisor(s): Prof. Pedro Manuel Urbano de Almeida Lima
Prof. Rita Maria Mendes de Almeida Correia da Cunha

Examination Committee

Chairperson: Prof. Full Name 1
Supervisor: Prof. Full Name 2

January 2026

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Resumo

Inserir o resumo em Português aqui com um máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave.

Palavras-chave: palavra-chave1, palavra-chave2, palavra-chave3,...

Abstract

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords.

This thesis follows the study of a naturalized motion controller for robust humanoid locomotion

Asymetric Morphology

Sim2Sim Policy Validation and Sim2Real Deployment

Deployment Heavy Approach (Lots of Real Experimental Data)

Progressive Evolution of Complexity, Goal of Naturalized Locomotion, and Emphasis on an Energy Efficient Gait

Keywords: keyword1, keyword2, keyword3,...

Contents

List of Tables

List of Figures

Glossary

AMP	Adversarial Motion Priors
DDPG	Deep Deterministic Policy Gradient
DoF	Degrees of Freedom
DRL	Deep Reinforcement Learning
IL	Imitation Learning
ISR	Institute for Systems and Robotics
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
PID	Proportional Integral Derivative
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
TD3	Twin Delayed Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Optimization
URDF	Unified Robot Description Format

1

Introduction

Insert your chapter material here.

1.1 Motivation

Humanoid robotics has seen an unprecedented development boom in recent years, and the recent wave of availability of bleeding-edge humanoid robots for research and development (facilitated by an unprecedented financial push for mass production by the Chinese government) has finally made it possible for numerous top universities to be at the forefront of humanoid robot research.

Namely, bipedal locomotion has been one of the most concentrated challenges for humanoid robotics research in recent years, due to the inherent difficulty in developing these complex movements. This complexity has deemed it

Furthermore, an even more unexplored and complex system dynamic comes in the form of an asymmetric morphological structure in a humanoid robot (i.e., a humanoid robot where its arms manifest a big portion of the weight of the humanoid), which adds a ton more complexity, and again, furthers the challenges to the development of a natural locomotion controller

However, significant advances have been made

Humanoid robotics has seen an insurmontable development boom in recent years,

The recent wave of availability of bleeding-edge humanoid robots for research development has

made it acessible for numerous top universities to be on the forefront of humanoid robot research

In this, locomotion has been one of the most concentrated challenges in recent years, looking to

Relevance of the subject.

Home Robotics Introduction

Road to develop a motion model for humanoid robotics

The challenge of achieving a more natural humane movement

The insurmontable challenge to it: gap between the way a humanoid robot motor structure differs from the way humans move (much larger degrees of freedom,)

Importance of having more energy-efficient, smooth, unalarming movements in a humanoid

Utilizing the official deployment architecture, but modified to support our higher degree of freedom model

1.2 Topic Overview

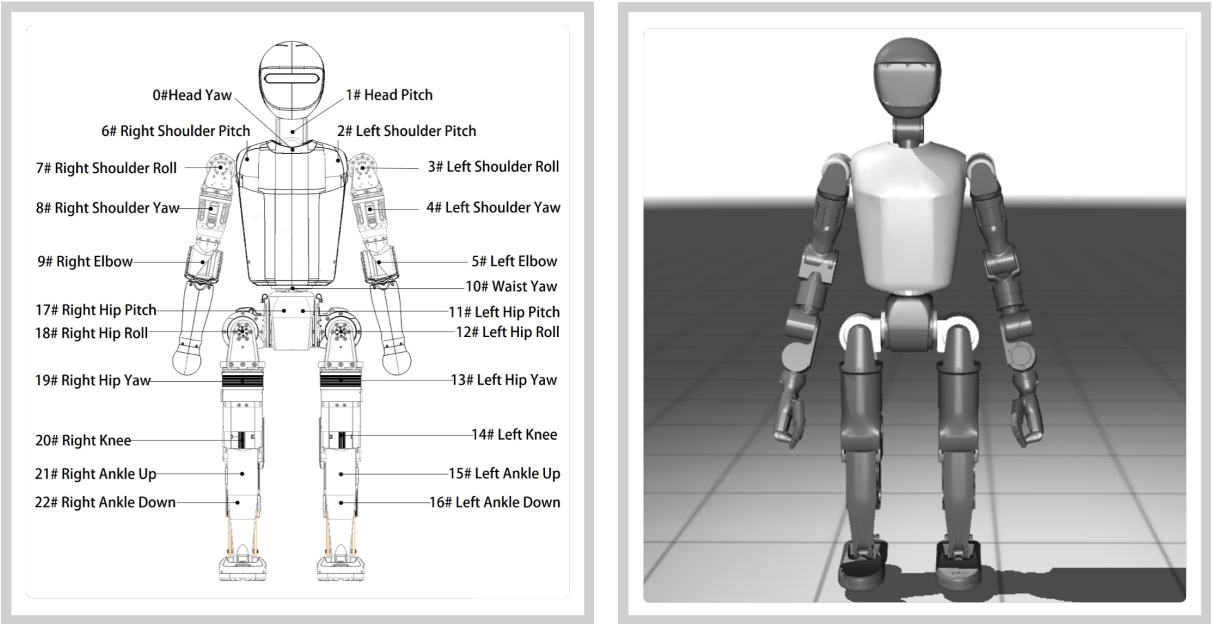
All of the experimental and developmental work reported in this thesis was carried out for the Booster T1[1], a humanoid robot acquired by the IRSG-ISR group in April 2025 to support the expansion of research into humanoid robotics at IST.

The Booster T1 line of manufactured robots is comprised of three distinct models. A 24-DOF model, as detailed in 1.1a, which features non-manipulative arm links, and serves as the primary and main basis model within the Booster T1 series. Due to its status as the central model, it benefits from the most comprehensive documentation and technical support, forming the foundation for most available resources. The remaining two variants enhance the complexity of the robot by incorporating three additional degrees of freedom per arm, enabling articulated arm motions for manipulation. Additionally, one model integrates a gripper end-effector, while the other employs a dexterous robotic hand. The latter configuration constitutes the specific model utilized in the present work (pictured in 4.4b).

Thus, this extended version of the humanoid robot model provides many additional degrees of freedom for complex grasps and manipulation behaviors, allowing for a wide range of research opportunities relating to humanoid manipulation within our research group at IST. However, the added complexity also introduces substantial additional mass and inertia on the upper body, which for a legged system that is already inherently delicate in ensuring whole-body stability, introduces a much greater challenge in developing dynamic and robust control for locomotion.

Additionally, as can be denoted in 4.4b, the arms of the robot take a significant portion of the robot's entire physical structure, reaching down to as low as the robot's knee joints, as well as contributing as one of the heaviest components in the joint space group by a considerable margin. This asymmetric morphology is unlike any streamlined humanoid robot carried out in current research and thus requires a great deal of additional consideration and adaptation to this relatively unknown research environment.

[1] Booster T1 Documentation: `https://booster.feishu.cn/wiki/DtFgwVXYxiBT8BksUPjcOwG4n4f`



(a) Booster T1 24-DOF Model with Joint Structure.

(b) Extended Manipulation Model in Simulation.

Figure 1.1: Showcase of Booster T1 Line of Models.

1.3 Objectives and Deliverables

Explicitly state the objectives set to be achieved with this thesis.

Also list the expected deliverables.

Primary Objective:

1.4 Thesis Outline

Briefly explain the contents of each chapter.

2

Background

This thesis

Work is about reinforcement learning and a future extension of it, so a proper contextualization of the state of the art is required for the subsequent work decisions

2.1 Core Principles of Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning that deals with sequential decision making. It aims to mimic how the human brain learns: through trial and error, given feedback from each action. Thus, for a certain **agent**, that acts as the decision-making entity, placed within an **environment**, that refers to everything that exists outside the agent and encompasses all the elements with which the agent interacts, actions are performed at each time instant, which may or may not alter the agent's representation of the state space, and this interaction between the agent and environment will yield a numerical **reward** that measures the desirability of the made decision in that state.

The main objective of RL is to develop a policy that will map an optimal action a within the agents action space $\mathcal{A}$, for any state s in the state space $\mathcal{S}$. This policy, denoted by π can represent the *probability* of selecting an action a when the agent is in state s, if defined as **stochastic** $\pi(s, a)$, represented as

$$\pi(s, a) = P(a|s), \tag{2.1}$$

or, when defined as **deterministic** $\mu(s)$, it will encode a specific action $a \in \mathcal{A}$ that is *always* taken when in state $s \in \mathcal{S}$ as denoted by

$$\mu(s) = a. \tag{2.2}$$

In the majority of reinforcement problems involving real hardware, a full state space is impossible to represent, so it is usually interchangeable with an observation space $o \in \mathcal{O}$ as a functional replacement, when the environment is fully observable.

2.1.1 Reward

Rewards play a central role in RL, as the actions rewarded and their assigned values directly determine the agent's performance. As a result, extensive research has explored methods to modify and structure rewards for the many possible tasks that employ RL. The most prevalent technique is reward shaping, which addresses the convergence of suboptimal policies caused by an exploitation of initial random or biased estimates during the initial stages of training, by constructing a dense reward signal that actively guides the optimization process, rather than providing feedback only upon task completion. This structured approach has been demonstrated to substantially improve agent performance [CITE], particularly in RL problems with highly complex goals.

The formulation of the reward function is the way RL-based approaches define how the policy is optimized for its intended objective, and must be conceptualized thoroughly to ensure the desired behavior. However, as the complexity of the state space, action set, environment, or intended goal increases, achieving effective policy optimization becomes more challenging [1] [2]. This added complexity results in greater computational demands and potential convergence issues during learning. Furthermore, when the objective involves highly specific or complex motions, designing an appropriate reward structure becomes particularly difficult [3], as it requires an accurate representation of every nuance of the desired movement to produce a smooth and natural behavior.

Policy optimization encompasses seeking to maximize a **cumulative return** that accounts for both the immediate reward received in the current state s, and all future rewards. This is key because often the immediate reward lacks the capacity to characterize how a present action will influence subsequent results. Furthermore, a greater emphasis should be put on the closest rewards, rather than those further in the future, so a **discount factor**, that controls the importance of future rewards, must be included. So by definition, the discounted cumulative return is written as

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.3}$$

where $r_t \in \mathbb{R}$ is the reward at time t, and $\gamma \in [0, 1)$ is the discount factor. Lower values of γ emphasize immediate gains, while values close to one encourage long-term optimization. And finally, since rewards are given by the environment at the next iteration, the time-step term for the reward in the sum is $t+k+1$.

2.1.2 Value Function and Q-Function

A framework must exist to quantify the desirability of occupying a particular state or executing a specific action within that state, evaluated through the lens of expected cumulative future returns, as presented in 2.3, achievable under a designated policy. This provides the formal definition of Value and Q-Functions in RL respectively.

As such, given a policy π, the **value function** quantifies the desirability of being in a given **state**

$$V_\pi(s) = \mathbb{E}_\pi \left[R_t | s_t = s \right]. \tag{2.4}$$

The **action-value** function, also commonly known as Q-function, differs by representing the value of a given **action** in a given **state**

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[R_t | s_t = s, a_t = a \right]. \tag{2.5}$$

By assigning value to all possible actions in a given state, algorithms that utilize the Q-function can greedily optimize a policy whose action yields the highest Q-values at any given state

$$\pi^*(s) = \arg\max_a Q_\pi(s, a), \tag{2.6}$$

while with the value function, a sequential sum over transition probabilities and rewards over the next time steps is required.

Given that the cumulative return R_t can be written as

$$R_t = r_t + \gamma R_{t+1}, \tag{2.7}$$

both functions can be expressed recursively using the Bellman equations:

$$V_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})] \tag{2.8}$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})]. \tag{2.9}$$

These equations are fundamental to many RL algorithms, as they form the mathematical foundation for policy optimization.

2.2 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) solves fundamental limitations with traditional RL methods regarding high-dimensional state spaces by leveraging deep neural networks as function approximators. Rather than storing separate values for each state-action pair, DRL uses neural networks to approximate value functions and policies, enabling agents to effectively process high-dimensional observations, such

as joint positions and velocities from a high DoF robot.

This development has unlocked new possibilities for solving complex robotic control problems through reinforcement learning, and made way to numerous state of the art algorithms to have been researched in recent years. Selecting the most appropriate DRL algorithm to implement and adapt is a pivotal decision for this work, requiring a careful match between the algorithm's strengths and the specific demands of the intended task.

2.2.1 TRPO and PPO

Trust Region Policy Optimization (TRPO) [4] is an **on-policy** DRL algorithm, which is characterized by a learning structure where policy updates rely solely on data or trajectories sampled by the current policy being updated. Thus, instances where the data generated by a bad policy would *contaminate* the policy optimization step can be avoided, however it comes at the cost of either utilizing less data than what would be possible when improving the policy, or requiring a very long time to generate a sufficient amount of data for every changing policy.

In terms of architecture, this algorithm upholds an **actor-critic** framework, a hybrid structure comprised of an actor network, that controls the agent's behavior and aims to refine a **stochastic** parametrized policy $\pi_\theta(a|s)$ by maximizing a policy gradient, alongside a critic network that actively learns a value function as a baseline to evaluate the performance of the actor, which will in turn reduce the variance of gradient estimates.

TRPO executes policy gradient optimization by maximizing a surrogate objective function $L(\theta)$, a method that diverges from standard policy gradient algorithms like REINFORCE [5], in the sense that additional measures are employed to ensure that the updated policy remains within a specific proximity to the previous policy. This is achieved via the following policy optimization step

$$\theta_{k+1} = \arg\max_\theta L_k(\theta) \tag{2.10}$$

$$L_k(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A_{\theta_k}(s,a)\right], \tag{2.11}$$

which averages over all states and actions sampled from the old policy θ_k, the probability ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$, which accounts for the inherent differences between the most recent policy and the one sampled from, along with the advantage function $A_{\theta_k}(s,a)$, which quantifies the difference between the estimated return and the old value baseline for that state. Additionally, the following constraint must always be kept:

$$D_{KL}^{max}(\pi_{\theta_k}||\pi_\theta) \leq \delta, \tag{2.12}$$

which accounts that the Kullback–Leibler divergence, specifically due to both policies being probabilistic distributions, between the previous and updated policy must remain close within a certain trust region of size δ.

Thus, policy improvement in TRPO is formulated as an approximate second-order optimization prob-

lem, because it must maximize the surrogate objective in 2.13 subject to the KL-divergence constraint in 2.12 at each policy update. This is implemented by using a Fisher approximation to the Hessian of the KL divergence and solving the resulting linear system for the search direction with a conjugate gradient algorithm, which introduces a substantial computational load at each optimization step.

Proximal Policy Gradient (PPO) [6] expands on TRPO and looks to mitigate these inherent computational demands, whilst maintaining the same algorithm structure.

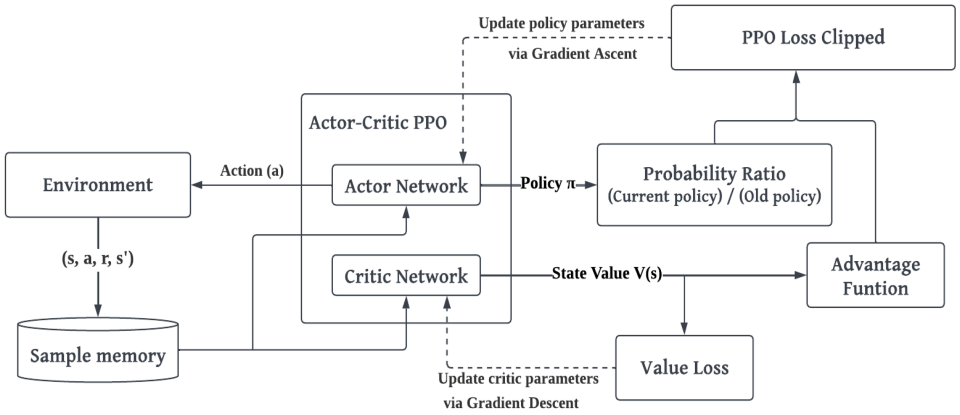


Figure 2.1: PPO Architecture. Adapted from [7].

The algorithm replaces the hard KL-divergence constraint with a modified first-order **clipped surrogate objective**, defined as

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\, \hat{A}_t,\ \text{clip}\left(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon\right) \hat{A}_t \right) \right], \tag{2.13}$$

where a clipped ratio $\text{clip}\left(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon\right)$ will restrict large deviations in the probability ratio, to reduce possibly unstable updates, but most notably removes the constraint in the optimization problem, and so heavily reduces computing time. This has made PPO a state-of-the-art, DRL algorithm widely used in most RL robotics applications.

2.2.2 DDPG and TD3

Deep Deterministic Policy Gradient (DDPG) [8] is an **off-policy** DRL algorithm, meaning that policy updates can rely on a portion of data or trajectories sampled by previous policies, making it a much more sample-efficient approach compared to on-policy algorithms.

It also holds an actor-critic architecture, just as the other previously described algorithms. However, the actor now optimizes a **deterministic** parametrized policy network, as it is built on the deterministic policy gradient theorem, which shows that for a deterministic policy $\mu_\phi(s)$, the gradient of the objective can be written as an expectation over states only, avoiding integration over actions, making gradient estimation more sample-efficient in high-dimensional continuous action spaces compared to naively estimating stochastic policy gradients that integrate over actions, which is updated by the following policy gradient

$$\nabla_\phi J \approx \mathbb{E}\left[\nabla_a Q_\theta(s,a)|_{s=s_t,a=\mu(s_t)} \nabla_\phi \mu_\phi(s)|_{s=s_t}\right]. \tag{2.14}$$

By requiring gradients of the return with respect to the action in order to update a deterministic policy, the critic learns an **action-state** value function to evaluate performance.

Both actor and critic networks hold a copy target network, which gets updated less frequently, and are used to calculate and solve a moving target problem by keeping the Bellman target stable while allowing the critic to learn.

The critic gets updated through TD error estimation, which calculates the loss between and a computed target y obtained from

$$y_t = r_{t+1} + \gamma Q_{\theta'}(s_{t+1}, \mu_{\phi'}(s_{t+1})) \tag{2.15}$$

$$L_k(\theta) = \mathbb{E}_{s,a\sim\mu_{\phi_k}}\left[(Q_\theta(s,a) - y)^2\right], \tag{2.16}$$

Once the critic produces reliable Q-value estimates, the actor improves by performing gradient ascent to maximize the critic's estimated value

The target networks receive periodic soft updates via **Polyak averaging**, which weighs the updated critic with the old target network parameters:

$$\phi^{'} \leftarrow \rho\phi^{'} + (1-\rho)\phi \tag{2.17}$$

$$\theta^{'} \leftarrow \rho\theta^{'} + (1-\rho)\theta \tag{2.18}$$

The algorithms' learning structure begets **overestimation bias** and .

Twin Delayed Deep Deterministic Policy Gradient (TD3) builds upon DDPG and reduces the impact of these deviations by introducing three main modifications. Clipped Double Q-Learning on target calculation which involves

Because of this,

Twin Delayed Deep Deterministic Policy Gradient was introduced by et al. in response to

Deep Deterministic Policy Gradient (DDPG) is an off-policy algorithm

Overestimation bias

Extension of DDPG

2.2.3 Behavior Cloning and Adversarial Motion Prior

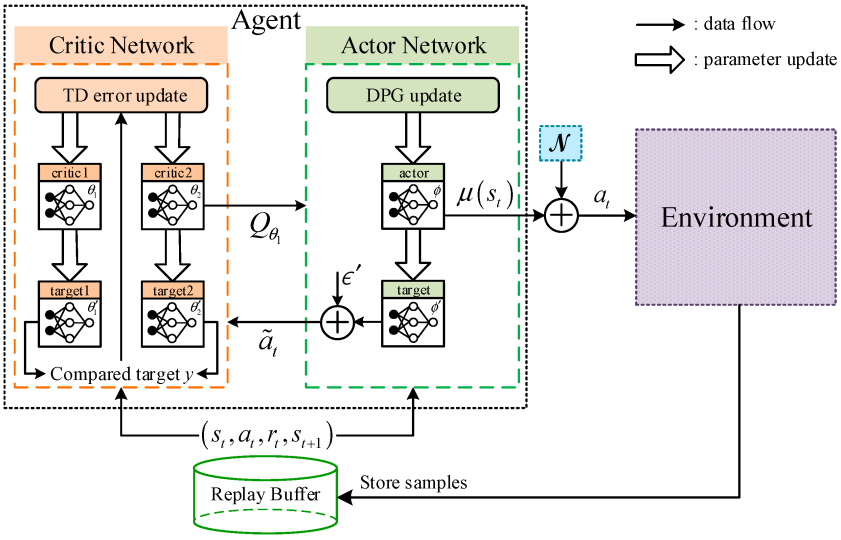


Figure 2.2: TD3 Architecture. Adapted from [9].

3

Literature Review

3.1 Model Based Controllers

Model-based solutions involve using a mathematical representation of the system as a model of its dynamics to predict its future behavior and optimize control actions accordingly.

Traditional model-based control approaches in robotics, such as PID control, struggle with multivariable systems [10], and lack predictive capabilities for complex dynamics, such as non-linear or underactuated systems, so they would require . Optimal control methods such as Linear-Quadratic Regulator (LQR) require offline computation of complex differential equations and cannot easily adapt to real-time constraints or changing operating conditions.

3.1.1 Model Predictive Control

For this, Model Predictive Control (MPC), has been widely regarded as an optimal model-based solution for controller development in robotics, with extensive work in robotic arm manipulation [11] and mobile robotics navigation [12] [13], because it can handle multiple inputs and outputs simultaneously, respect physical constraints on actuators and system states, and plan movements several steps ahead, adjusting its strategy in real time, making this approach enhance the system's ability to handle unforeseen disturbances and environmental changes.

However, its effectiveness depends greatly on the model's accuracy. In whole body bipedal locomo-

tion, the complexity of factors that have to be accounted for , are extremely large to be mathematically expressed. This is why models and dynamics are simplified, such as only the legs of the robot no whole body motion, or reduced system dynamics, a very non complex movement or gait, or just stability without locomotion. And sometimes run in simulation, because deployment on real hardware would not work with some of these simplifications, and also by the modeling not accounting for the plethora of possible disturbances and noise in real deployment. Additionally, all required computation is performed online, which scales tremendously with higher complexity systems, which also causes for a ton of simplifications to be made in order for the computation to be feasible in a real deployment on such a complex system. While in theory accounting for disturbances, any kind of irregular train, elevations and stairs, difficult motions, or push resistance are scenarios where an mpc based implementation will struggle very much, because of an amalgamation of impossible to model, and computationally heavy, especially for a bipedal humanoid robot.

This approach enhances the system's ability to handle unforeseen disturbances and environmental changes, though its effectiveness depends on the model's accuracy and computational resources available. Additionally,

However

showcase the advantages that it brings, mention that it must be performed fully online (maybe reference some more simple applications where it does not need to be fully online)

This has made it a staple in (reference use of MPC in simpler robotic problems or non robotic problems)

which optimize control signals in a receding horizon fashion, often utilize simplified dynamics models and relatively short time horizons to overcome this challenge

Regarding locomotion, some work has been employed in robotic arms and velocities bases (reference mpc in robotic locomotion), showcase the advantages that it brings, mention that it must be performed fully online (maybe reference some more simple applications where it does not need to be fully online) (show general architecture diagram). Cannot use the advantages of simulation (i.e. priviledged states, thus observations must be very complete), and note that the robustness performance is almost entirely dependent on the model formulation (reference this claim)

however state that when expanding the complexity to bipedal locomotion.

MPC and model based solutions on bipedal locomotion reference multiple works and their branching, their advantages and disadvantages. Why is it not suited for our work

Tends to be non whole body, alot of impossibilities in handling the control problem online Must run online, which means it will be to heavy to compute, so alot of simplifications or compromises have to be made. Gate switching, push resistance

Model based works –¿ just MPC ig, mas referir porquê só mpc. MPC faz trajectory optimization [14], como conciliar isto com remote velocities teleoperation ¡– referir porque é que isto é mau Ter de fazer online, por isso grande parte dos trabalhos operar em simulação (referir trabalho de locomoção bipedal mpc em simulação só) ou simplificar ou o hardware (referir trabalho só com uma perna, ou só com pernas), ou o behavior (trabalho com single gait, nenhum whole body motion, estabilização

dúbia). Finalmente, referir que trabalhos deste género não conseguem ser robustos a empurrões, terreno peripécio, escadas, ou qualquer tipo de outros environments, sem qualquer tipo de modelização extrema, e certificação de que o modelo está o mais bem definido quanto possível, que se torna quase impossível quando para locomoção whole body

3.2 Learning Based Controllers

Relevant works regarding quadruped locomotion have been the biggest staple in reinforcement learning based legged locomotion, for it's much more relative of ease of stability due to a higher count of ground contact, which make for quicker cases of convergence during policy development.

Allows to test a range of instability scenarios, mix different terrains or tasks.

Bipedal locomotion introduces a higher difficulty towards stabilization both when standing and while moving, which impacts training times and in some cases even the feasibility of conversion for complex tasks that require an extensive reward structure.

Reward shaping can reach a plateau for a perfectly coordinated whole body motion, particularly for a human like motion, for there are always intrinsic and slight behaviours that usually is very difficult to be manually planned in the reward structure.

Reinforcement learning based approaches on locomotion on quadrupeds. The inherent advantage in unstable environment of utilizing this learning approach. Most advances made on quadrupeds, dynamics change a lot for humanoids

4

Proposed Thesis and Preliminary Work

4.1 Proposed Thesis Structure

The proposed thesis structure is represented in Figure 5.1, and will involve a system based on **progression** along three locomotion objectives, followed by expanding the motion controller to handle tasks of even higher complexity.

The **standard locomotion** objectives will each build from one another, due to the work being transferable onward as an evolution in complexity. Thus, the first objective will relate to locomotion only involving **leg joint movement**, whilst maintaining the upper body at a fixed position. This simplifies the work framework process for an initial deployment, and not only allows for starting under the same groundwork as of the original booster motion model, allowing for a direct comparison towards improving it, but it is a good baseline in with to establish the core principles and development practices. It also encompasses the majority of the presented preliminary work and analysis.

As previously described in the Introduction section, stabilizing the upper-body during locomotion holds significant implications towards the robot's stability, regarding the specific joint configuration where the arms are placed under. More specifically, one of the additional reasons to select this as the first

objective was to allow for the opportunity of experimenting with **different arm configurations**, namely to rest the arms along with the torso, to ensure greater stability and a more natural movement, as opposed to the retraction of the arms that the official booster motion model requires, which adds considerable weight to the front of the robot and makes it more unstable in motion.

Following up on this, extending the locomotion to be performed as **whole-body** is the most clear progression from the previous objective, as it will involve having the upper-body shift weight along with the movement, allowing for more balance and an overall much more natural motion. It does however, introduce greater complexity to the learning process, and will present more significant challenges in terms of collision avoidance and safety when deploying on the real robot.

And from there, improvements in fluidity will look to be made by introducing **reference motions**, to hopefully capture nuanced characteristics in locomotion that cannot be hand-modeled in the reward structure, comprising then a third objective that will look to fine-tune the previously developed motion controller.

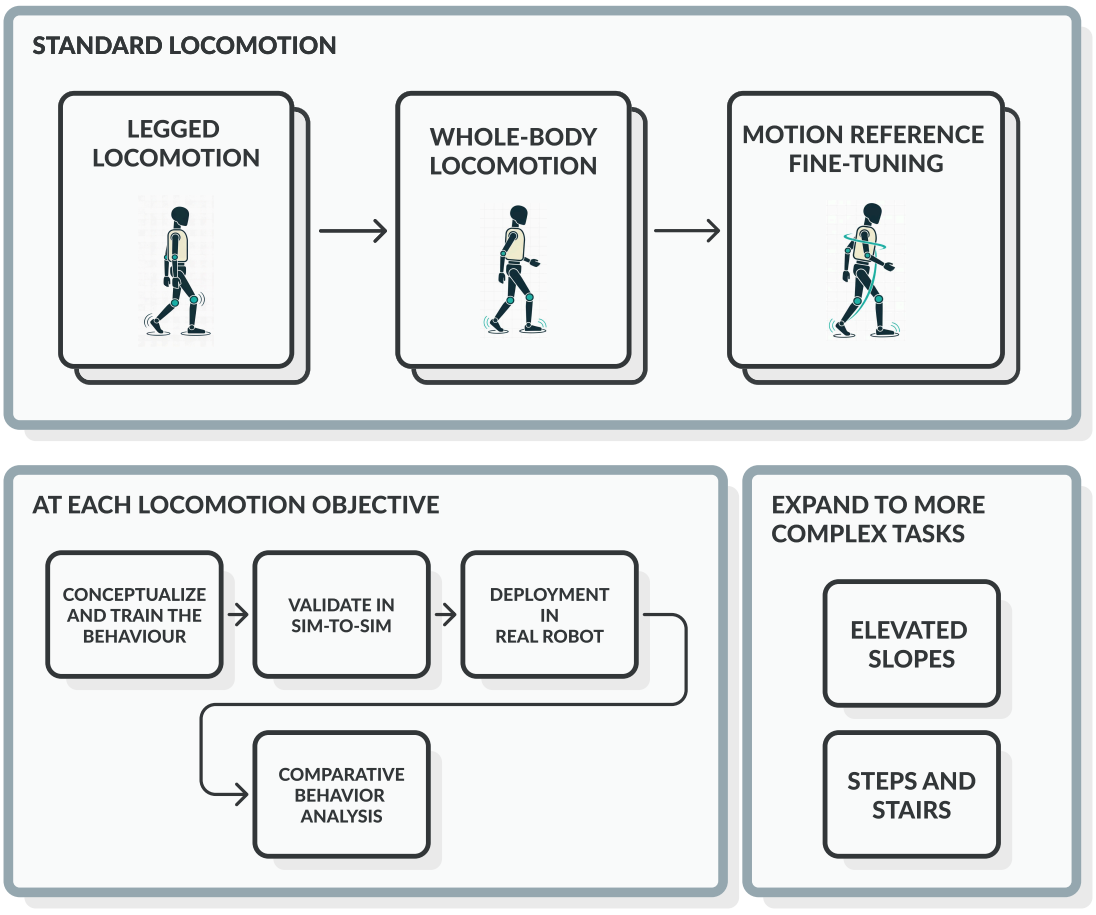


Figure 4.1: Proposed Thesis Structure.

Each objective will present a systematic work structure, to allow for a more direct comparison of each goal, and will be composed of:

- A conceptualization of the objective, including the formulation or modification of the learning network and reward structure, followed by training analysis.

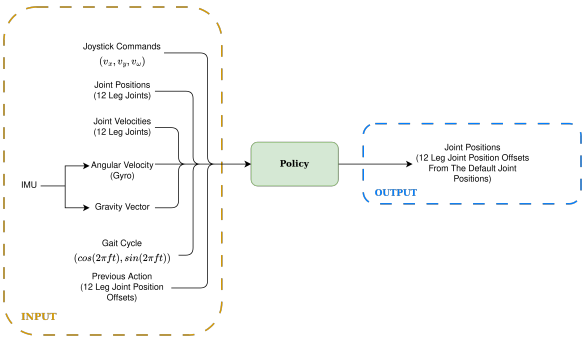
- A validation of the trained policy on a different simulator, to ensure its generalization and robustness through manually teleoperated stress tests.
- Deployment on the real robot, looking to replicate the same movements performed in simulation and confirm its transferability to ambient conditions in a real environment.
- Perform a comparative study at the joint level between the original booster motion model baseline and each developed objective, along with a sim-to-real analysis for each deployed policy.

And finally, as a natural progression towards building a better motion controller, more complex tasks like slopes and elevations, or steps and stairs will look to be incorporated, by creating new training environments and overhauling the reward structure to define fixed trajectory goals.

4.2 Legged Locomotion

Leg joint movement locomotion requires a policy that solely controls the leg motors of the robot, whilst keeping every other joint at a fixed position. For this, the policy will be simplified to only require observations of the leg joints, and in turn, only outputting joint positions of those leg motors.

As described in Figure 4.2a, the policy will receive as input a velocity vector (v_x, v_y, v_ω) as a command from a teleoperated joystick, the previous output action, the actual angular positions and angular velocities of the leg joints measured by their joint encoders, readings from the IMU at the robots torso indicating the gravity vector along the center of mass, and its angular velocity through the gyroscope of the IMU. Lastly, the policy also receives a gait cycle command, which is represented by a $(\cos(2\pi ft), \sin(2\pi ft))$ vector that encodes at which stage of the periodic walk cycle the state is currently in, which is essential for some of the behavior-specific rewards that are used.



(a) Developed Policy Structure.

Components	Dims	Actor	Critic
Commands	3	✓	✓
Gait Cycle	2	✓	✓
Gravity Vector	3	✓	✓
Angular Velocity	3	✓	✓
Joint Position	12	✓	✓
Joint Velocity	12	✓	✓
Previous Action	12	✓	✓
Body Accelerometer	3		✓
Base Linear Velocity	3		✓
Base Height	1		✓
Push Force	2		✓
Feet Air Time	2		✓
Feet Linear Velocity	12		✓

(b) Asymmetric Actor-Critic Structure.

Figure 4.2: Base Underlining of Both the Policy and Network Architecture.

During training, the actor network receives the readings from the IMU and joint encoders with added noise, whereas the critic network, serving as the evaluator, receives all of the actor's observations free of any noise, and is equipped with additional privileged information denoted in Figure 4.2b that is only

accessible in simulation and aids in policy optimization. These mostly include some useful state information for some of the employed rewards.

The utilized reward structure, detailed in Figure A.2, follows the general principles for locomotion of humanoid robots, which are positive reinforcement for not meeting the termination conditions, that occur when the robot falls or reaches a time limit, and for accurately tracking the specified velocity commands. It is then followed by penalizations when the base height and orientation of the torso does not meet the specified target value, which will encourage the robot's upper body to remain straight and at a stable position during motion. Excessive movements that would warrant a high torque, joint velocity, or acceleration are slightly penalized, along with nearing joint limits or collisions.

Lastly, some behavior-specific rewards are designed, with the aid of the gait cycle command, to promote foot swings according to the gait frequency, penalizations for rolling or slipping of the ankles, as well as maintaining each leg at a set distance when idle.

The learning structure follows the off-policy TD3 algorithm adapted to speed up training with parallelized environments [15] [16] and support the modified robot model.

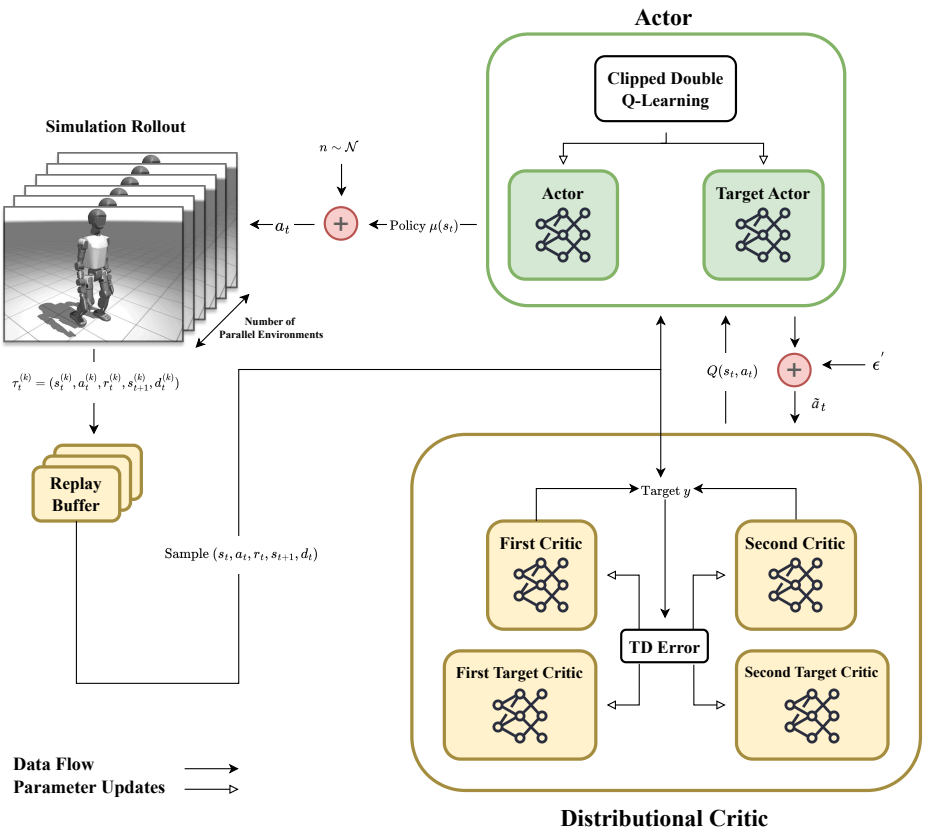


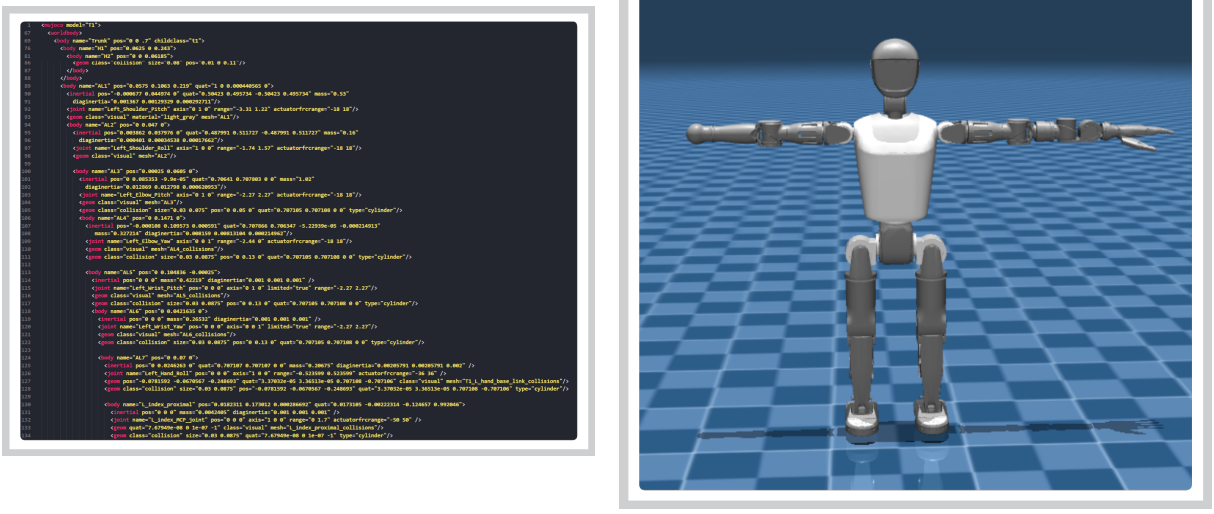
Figure 4.3: Utilized Network for the First Objective.

At each global step, in-simulation transitions across the parallel environments are collected and collectively stored into the replay buffer, which is sized *buffer size* $\times$ *num environments*, so that it becomes comprised of an even distribution of samples collected from every environment. During initial steps, the algorithm will operate solely on data collection, gathering samples from environment steps without

network parameter updates. Due to an employed high count of parallel environments, just after a couple initial steps the buffer is sufficiently large enough for learning to start, and from then on, at every environment step, batches are sampled by the replay buffer are used to update both critic networks multiple times per global step, with their corresponding target networks and the delayed actor being updated half as frequently. Using batches instead of single samples greatly reduces the extreme noise for such a high dimensional state and action space, and by optimizing the distributional critic networks multiple times between each environment step, it greatly increases sample efficiency for learning in such a complex space to be feasible in a reasonable time-frame.

Training was performed using MuJoCo Playground [17] as the simulator, an extension of MuJoCo [18] that enables . It requires the robot model file to be in the XML format, rather than the standard unified robot description format (URDF) that other robotics simulators use.

As previously mentioned in subsection 1.2, the majority of official provided resources are made for the simplified model in Figure 1.1a, and this includes existing XML and URDF files for the robot model. This required replacing the arm attachment below the elbow for the extended manipulation arm parts, whose meshes had to be extracted from the official Universal Scene Description (USD) file[1], and the XML hierarchical code structure had to be rewritten to rejoint the additional DoF correctly.



(a) Excerpt of the Modified XML Model.

(b) In Simulation Comparison of Changes Made.

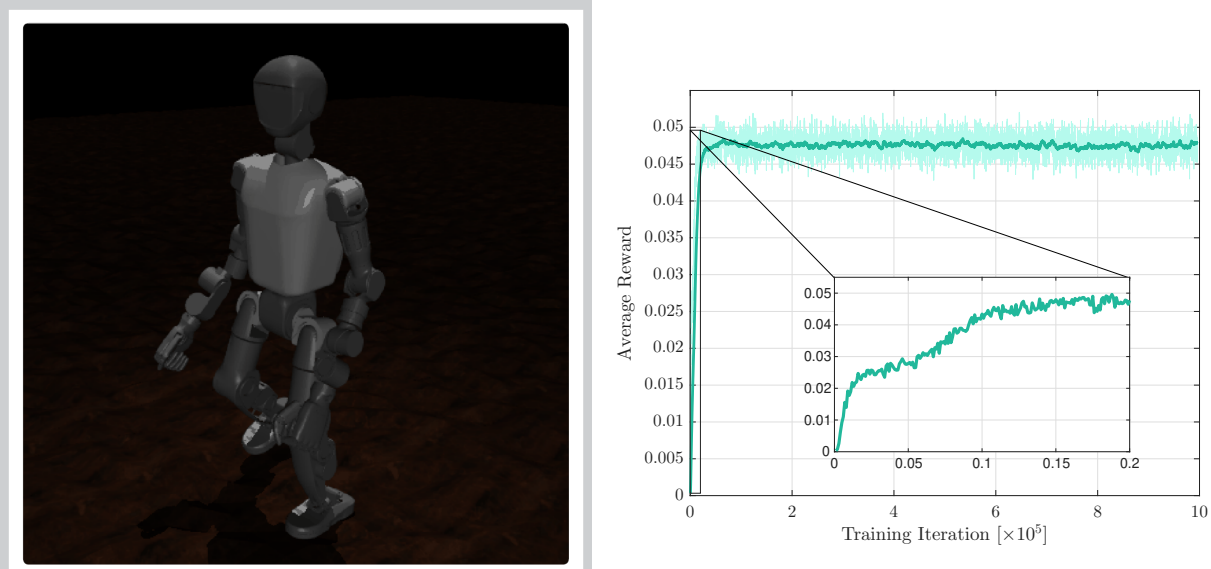
Figure 4.4: Results of the Work on Restructuring the XML Model.

The changes made were necessary to have an accurate representation in simulation of the actual robot model in which these trained policies will be deployed, as the modified physical dynamics inherent in this joint expansion drastically change the learning process, and thus enabling the training environment to become a much more accurate approximation of the real deployment conditions.

[1] Booster T1 Documentation: `https://booster.feishu.cn/wiki/DtFgwVXYxiBT8BksUPjcOwG4n4f`

4.3 Training Results

For this first objective, training with parameters detailed in Table A.1 for a fixed retracted arm configuration, which was chosen due to this being the configuration on the official booster motion model, are shown in Figure 4.5.



(a) Training Environment for the Adapted Robot Model.

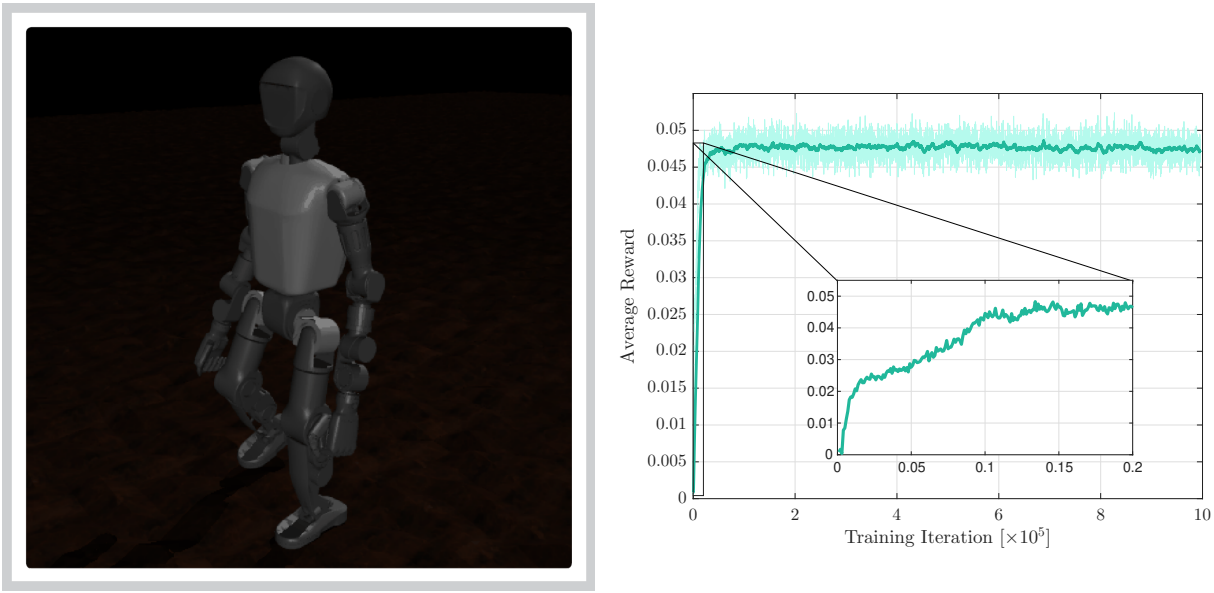
(b) Averaged Reward Obtained During Training.

Figure 4.5: Main Training Results for the Retracted Arms Configuration.

Total training time took approximately 12 hours on a single RTX 5080 GPU, for a total of one million iterations. The averaged reward across all parallel environments per global step stabilized at around the 8 $\%$ of the total training sequence (around the 8×10^4th iteration), and from then on oscillated around the same plateau. The employed reward structure in Figure A.2 alludes to the quick rise evidenced in these initial iterations as the byproduct of the robot no longer easily falling, or colliding with itself, as these termination conditions hold the biggest impact on measured rewards due to trajectories being cut early.

During the initial subsequent iterations after plateauing, several undesirable effects, such as endless micro-movements when standing still and slight instabilities when decelerating to a stop, manifested themselves in the extracted model. These get somewhat ironed out, evidenced by the continuous decay of the loss functions of the estimated networks in A, however due to most of these nuanced behaviors either not holding enough weight to be prioritized, or not being possible to structure into the reward function, remnants of such inconsistencies still can sometimes randomly appear from one iteration to the other, which was why it was important to run training for such an added extended period after the supposed stabilization, allowing for a manually prorated analysis of the most robust model.

The other section of results shown in Figure 4.6 relate to the same exact training environment but with modifications made to the robot model to rest its arms nearer to the center of mass. It was hypothesized that this improved weight distribution would show a faster convergence during the initial steps



(a) Training Environment for the Adapted Robot Model.

(b) Averaged Reward Obtained During Training.

Figure 4.6: Main Training Results for the Rested Arms Configuration.

where initial stability is the first learned characteristic, but the results show to be very similar in behavior between both configurations, despite showing higher reward values at each instance during the ascent, even if slightly.

4.4 Validation Results

Each of the periodically saved policies throughout training for both configurations are transferred to a validation environment in MuJoCo, following the official framework provided by the robot manufacturers [19], but modified to work on our actual robot model and also for the robot to be manually teleoperated via a joystick controller, just as in a real-life setting.

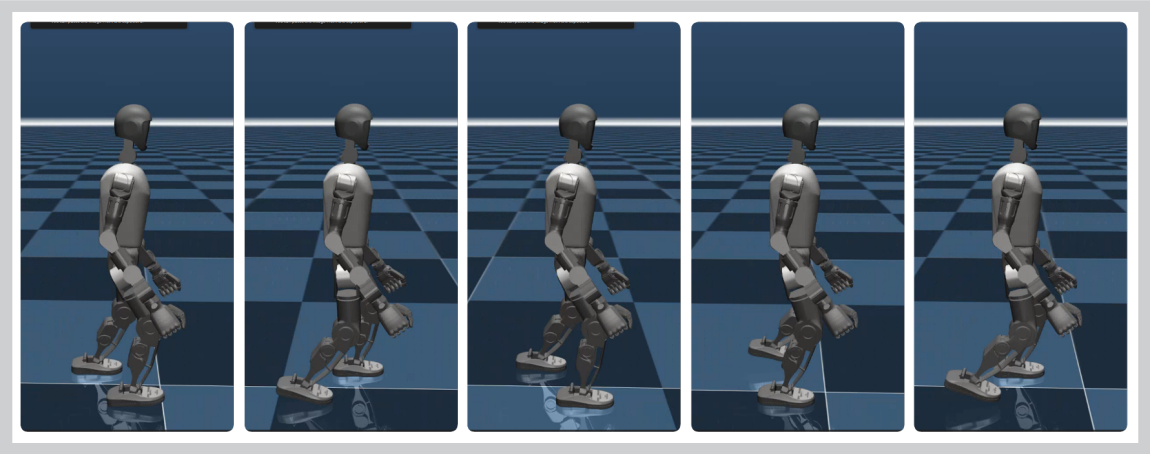


Figure 4.7: Validation Results for the Retracted Arms Configuration.

This was how the prorated selection of the best model to deploy was made, as it allowed for performing manual stress tests on the controller to ensure complete stability when transferring to real deploy-

ment.

Figures 4.7 and 4.8 respectively showcase the models saved at the 9.5×10^5 th and 8×10^5 th iteration of their respective training environments, and were selected on the basis of exhibiting no unusual behaviors and never losing their balance across the many employed control experiments.

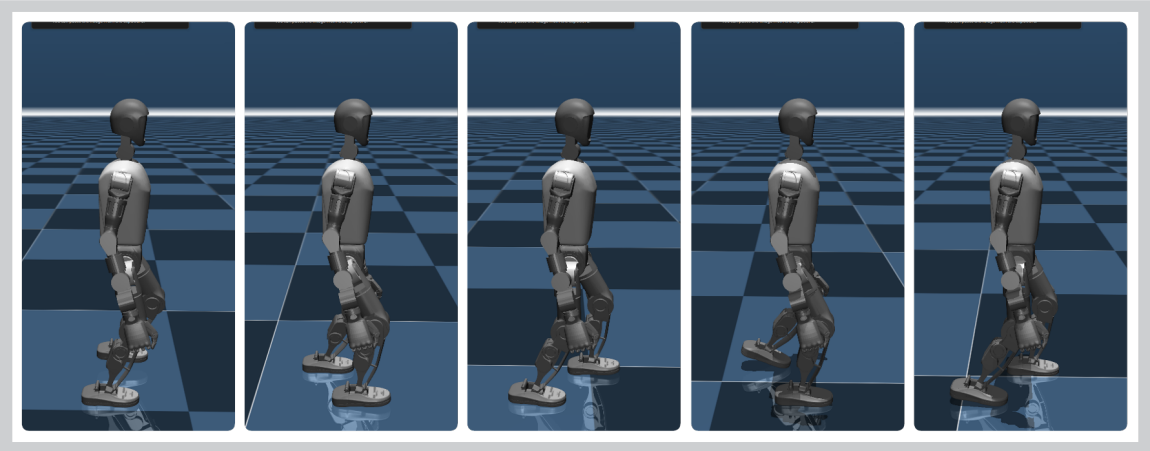


Figure 4.8: Validation Results for the Rested Arms Configuration.

Both policies were also validated in other simulators such as Isaac Sim and Webots to ensure broad generalization, in the sense of confirming that the MuJoCo physics engine was not being exploited by the policy, and it was observed that the validation results in MuJoCo present the nearest approximation to the robot's behavior in a real deployment.

4.5 Real Robot Deployment Results

During deployment, the robot's sensors and actuators are controlled through the official middle ware of the robot manufacturer, **Booster SDK**[2], which acts similar to frameworks like ROS, where measurements can be accessed by subscribing to a topic, and motors can be controlled by publishing the desired joint position command.

Illustrated in Figure 4.9, policy deployment on the real robot involves utilizing the `/joint_ctrl` service to publish the joint positions that serve as the policy output to be converted internally to a control signal for the robot to execute, and the `/low_state` topic to obtain live measurements of both the joint positions and velocities recorded by each joint encoder and the data read from the IMU.

The deployment script will run simultaneously through threading inference calculation of the output, and publishing through the SDK. Lastly, `T1.yaml` houses a specified stiffness and damping configuration across all joints, that is the same utilized in training, and specifies the desired initial joint positions and links the used .pt policy model.

Running the deployment involves two separate modes: an initial one where the robot starts from its base retracted arms configuration to a specified initial joint position in the yaml file, before the policy is loaded, so it is crucial that the movement made from the retracted arm configuration to the desired initial

[2] `https://github.com/BoosterRobotics/booster_robotics_sdk`

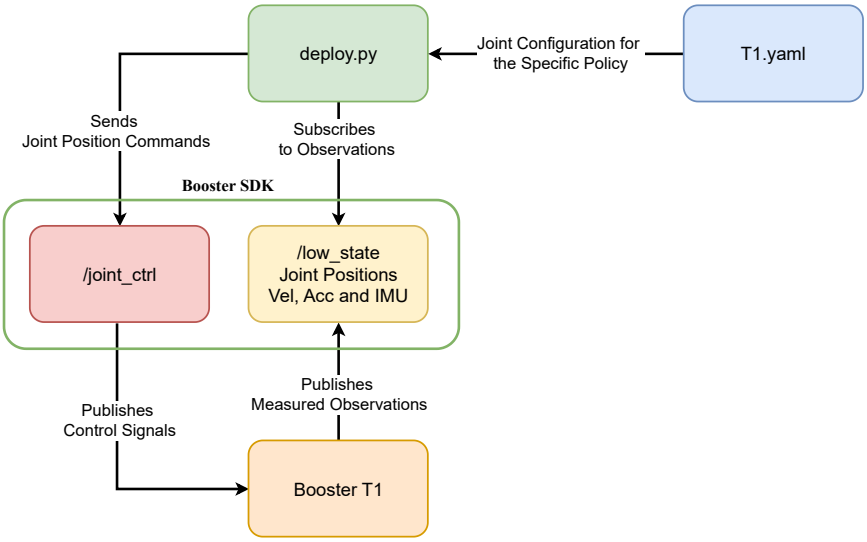


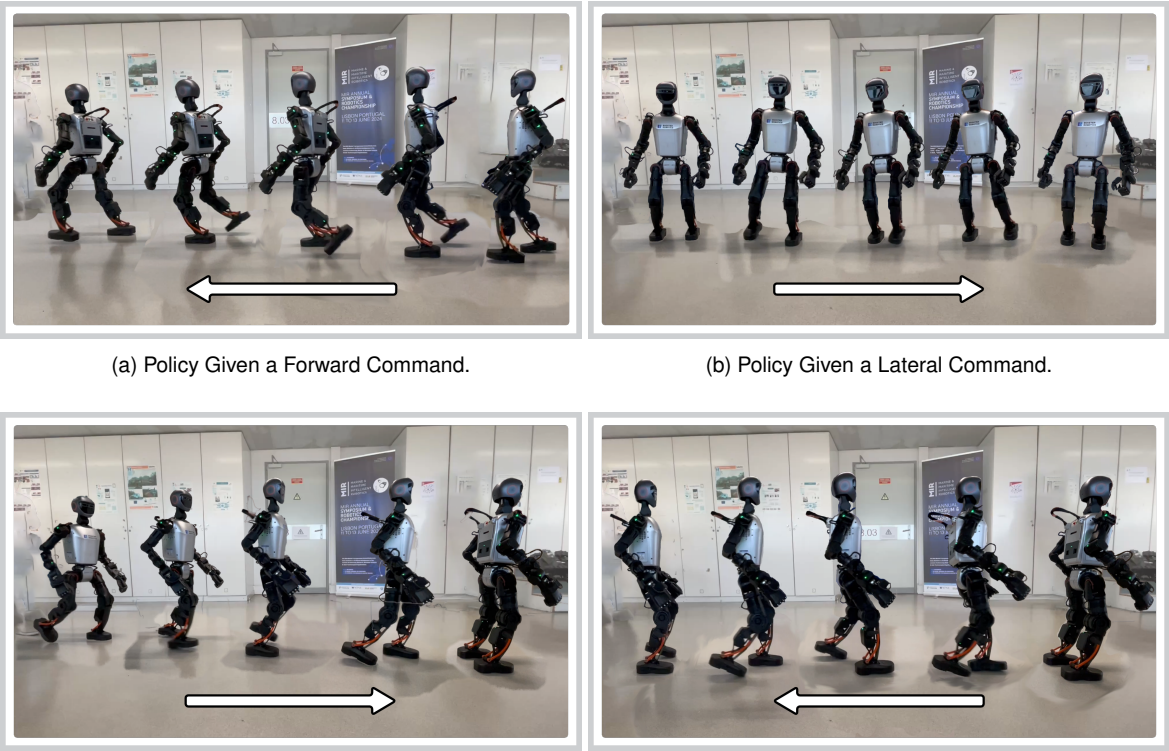
Figure 4.9: Real Robot Deployment Architecture.

joint position is not drastic enough to cause the robot to lose balance, because as the policy is not yet loaded the robot cannot adjust himself to regain stability. This is why the initial configuration tested was one that maintained the retracted arms position, as to serve as the initial safest test.

After entering the specified initial positions, the second mode involves loading the policy and thus being able to control the robot through teleoperation.

Words are made here Words are made here Words are made here Words are made here Words are made here

[3]First Recorded Trial Run: `https://youtu.be/K0Rxt2i_bao`
[4]Second Recorded Trial Run: `https://youtu.be/B5VMVE8vY2U`



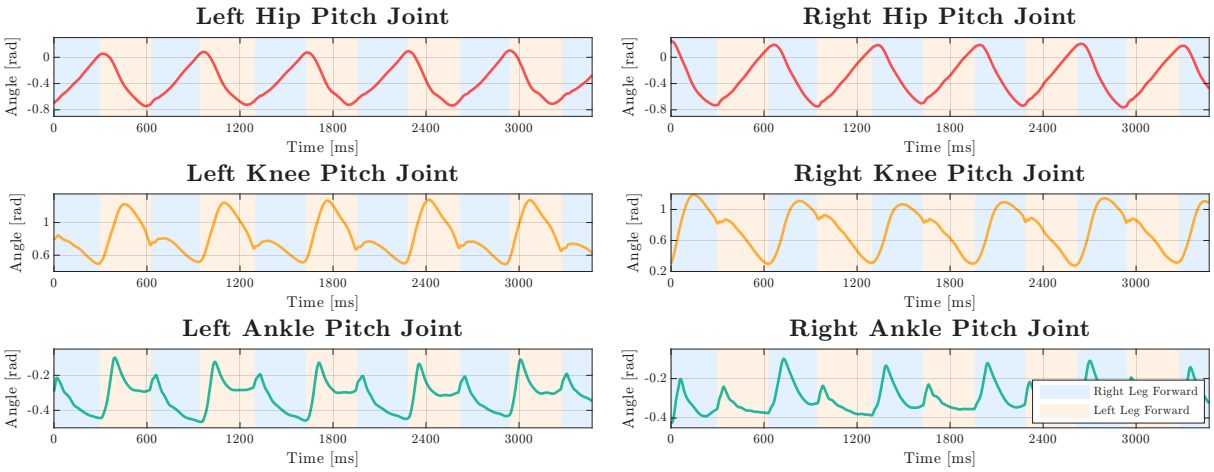
(a) Policy Given a Forward Command.

(b) Policy Given a Lateral Command.

(c) Policy Given a Forward Command.

(d) Policy Given a Backwards Command.

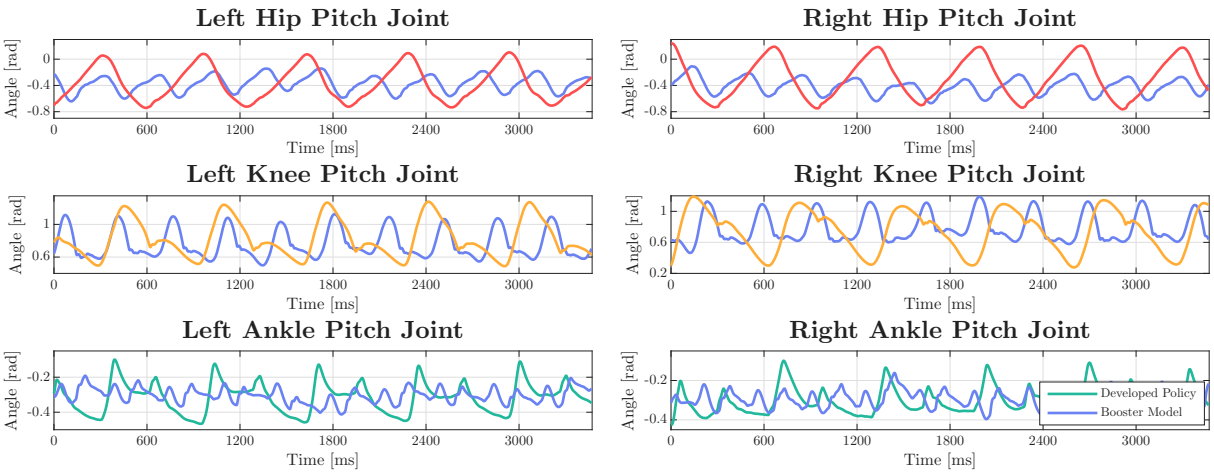
Figure 4.10: Deployment Results of the Retracted Arms Configuration Policy[3]



(a) Left Leg Joint Angles in a Forward Walk Cycle.

(b) Right Leg Joint Angles in a Forward Walk Cycle.

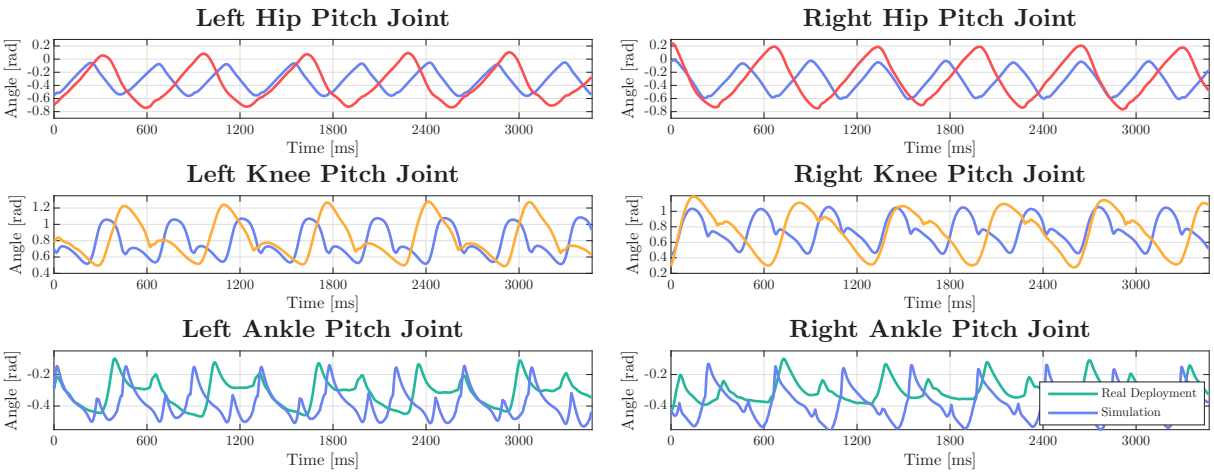
Figure 4.11: Logged Joint Positions during Deployment.



(a) Left Leg Joint Angles in a Forward Walk Cycle.

(b) Right Leg Joint Angles in a Forward Walk Cycle.

Figure 4.12: Developed Locomotion Policy *vs* Original Booster Model Positions.



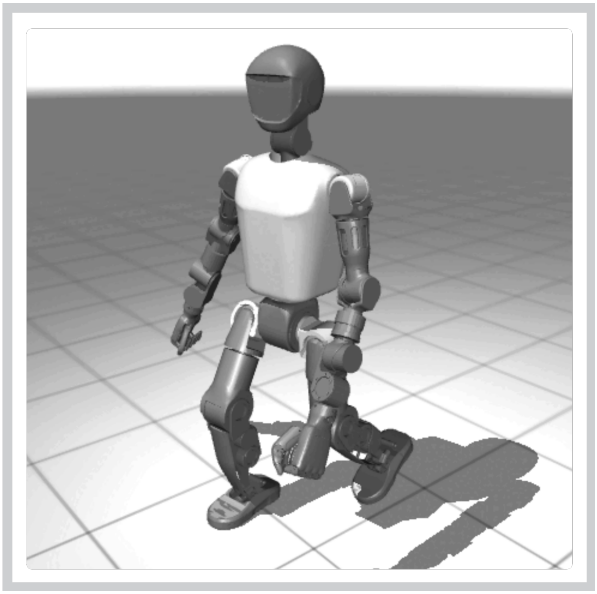
(a) Left Leg Joint Angular Positions in a Forward Walk Cycle.

(b) Right Leg Joint Angular Positions in a Forward Walk Cycle.

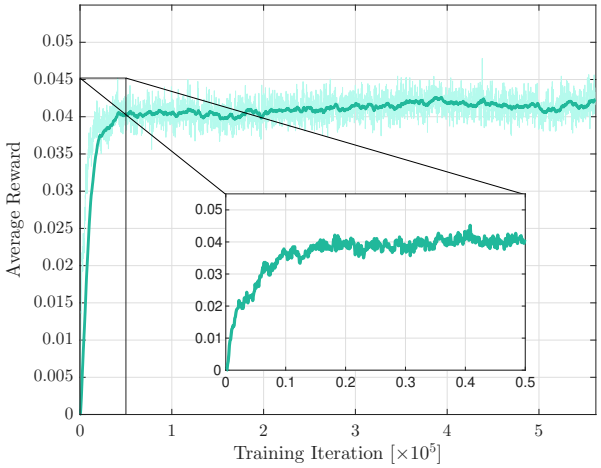
Figure 4.13: Sim2Real Gap Positions.

Figure 4.14: Deployment Results of the Rested Arms Configuration Policy[4]

4.6 Preliminary Whole Body Training Results



(a) Training Environment for the Second Objective.



(b) Averaged Reward Obtained During Training.

Figure 4.15: Preliminary Training Results for the Second Objective.

Description of the baseline problem.

4.7 Preliminary Whole Body Validation Results

Analysis of the baseline solution.

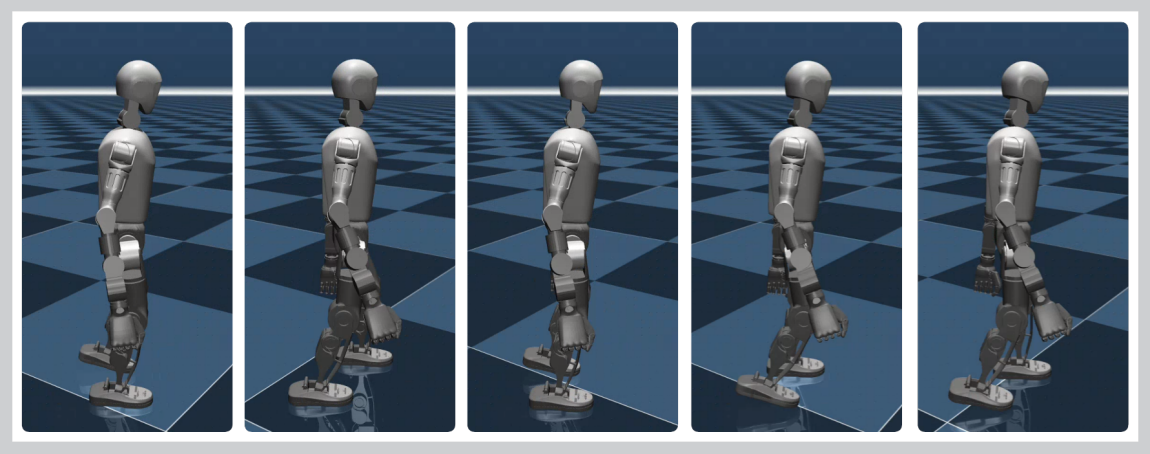


Figure 4.16: Validation Results for the Second Objective.

5

Conclusions

Insert your chapter material here.

5.1 Achievements

The major achievements of the present work.

5.2 Future Work

A few ideas for future work.

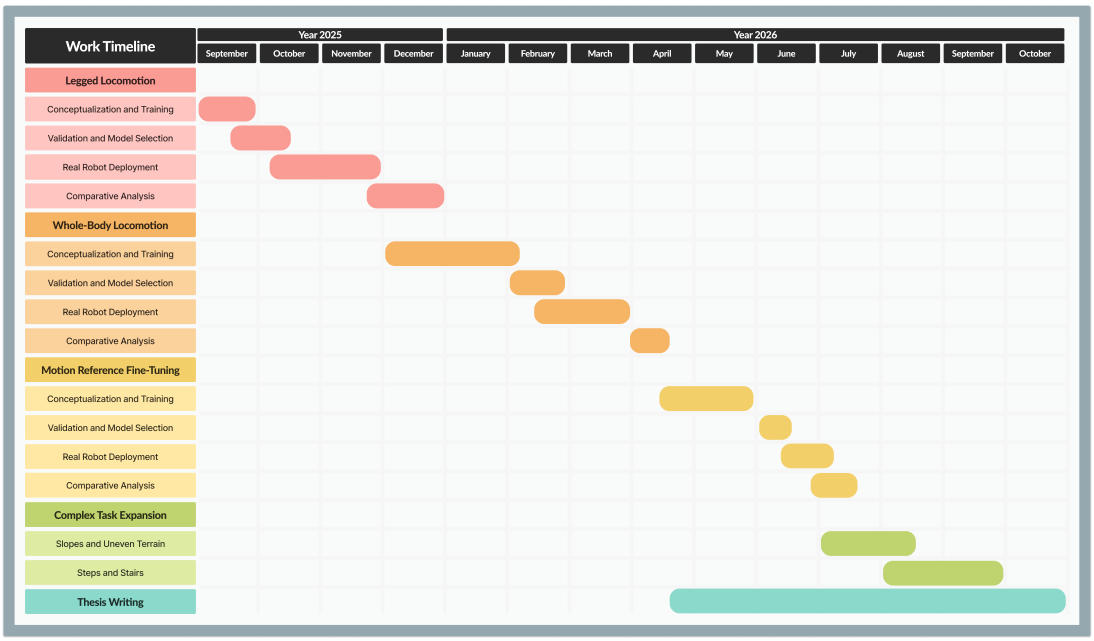


Figure 5.1: Gantt Chart Exhibiting the Planned and Scheduled Dissertation Work.

Bibliography

[1] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[2] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[3] 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)*, 37(4): 1–14, 2018.

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

[5] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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

[7] L. Tan, L. Sun, B. Cao, J. Xia, and H. Xu. Research on weighted energy consumption and delay optimization algorithm based on dual-queue model. *IET Communications*, 18(1):81–95, 2024.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[9] S. Liu, Z. Yang, Z. Zhang, R. Jiang, T. Ren, Y. Jiang, S. Chen, and X. Zhang. Application of deep reinforcement learning in reconfiguration control of aircraft anti-skid braking system. *Aerospace*, 9 (10):555, 2022.

[10] S. W. Sung and I.-B. Lee. Limitations and countermeasures of pid controllers. *Industrial & engineering chemistry research*, 35(8):2596–2610, 1996.

[11] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger. Data-driven model predictive control for trajectory tracking with a robotic arm. *IEEE Robotics and Automation Letters*, 4(4):3758–3765, 2019.

[12] G. Klančar and I. Škrjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and autonomous systems*, 55(6):460–469, 2007.

[13] M. Spahn, B. Brito, and J. Alonso-Mora. Coupled mobile manipulation via trajectory optimization with free space decomposition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12759–12765. IEEE, 2021.

[14] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, pages 292–299. IEEE, 2013.

[15] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[16] Y. Seo, C. Sferrazza, H. Geng, M. Nauman, Z.-H. Yin, and P. Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

[17] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

[18] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[19] Y. Wang, P. Chen, X. Han, F. Wu, and M. Zhao. Booster gym: An end-to-end reinforcement learning framework for humanoid robot locomotion. *arXiv preprint arXiv:2506.15132*, 2025.

[20] B. Robotics. *T1 Instruction Manual*. https://booster.feishu.cn/wiki/XAS3wv4IwiSiXXkDbMrceE6UnHc, 2025.

A

Additional Results

A.1 Detailed Structure for the Legged Locomotion Policies

Table A.1: Network and Training Parameters for Both Legged Locomotion Training Policies.

Network Parameters	**Value**
Actor Number of Hidden Layers	512
Critic Number of Hidden Layers	1024
Actor Learning Rate	3×10^{-4}
Critic Learning Rate	3×10^{-4}
Training Parameters	**Value**
Number of Environments	1024
Number of Iterations	1000000
Buffer Size	8192
Batch Size	32768
Number of Evaluation Environments	1024
Discount Factor (γ)	0.97
Velocity Command Limit x and ω	$[-1.0, 1.0]$
Velocity Command Limit y	$[-0.8, 0.8]$
Domain Randomization	**Distribution**
Floor Friction	$\mathbf{u} \sim \mathcal{U}(0.2, 0.6)$
Link Mass	$\mathbf{u} \sim \mathcal{U}(0.98, 1.02)$
Torso Mass	$\mathbf{u} \sim \mathcal{U}(-1.0, 1.0)$
Initial Joint Position Jitter	$\mathbf{u} \sim \mathcal{U}(-0.05, 0.05)$
Joint Stiffness	$\mathbf{u} \sim \mathcal{U}(0.7, 1.3)$
Joint Damping	$\mathbf{u} \sim \mathcal{U}(0.7, 1.3)$

Table A.2: Utilized Reward Structure for Both Legged Locomotion Training Policies.

Components	**Equations**	**Weights**
Survival	1	0.25
Velocity Tracking x	$\exp\big(-(v_x^{\text{cmd}} - v_x)^2/\sigma\big)$	1.0
Velocity Tracking y	$\exp\big(-(v_y^{\text{cmd}} - v_y)^2/\sigma\big)$	1.0
Velocity Tracking ω	$\exp\big(-(\omega_z^{\text{cmd}} - \omega_z)^2/\sigma\big)$	2.0
Base Height	$(h - h^{\text{target}})^2$	-20.0
Orientation	$\|g_{xy}\|^2$	-5.0
Torque	$\|\tau\|^2$	-1×10^{-4}
Torque Tiredness	$\|\tau/\tau_{\max}\|^2$	-0.5×10^{-2}
Power	$\max(\tau \cdot \dot{q}, 0)$	-1×10^{-4}
Linear Velocity z	v_z^2	-2.0
Angular Velocity xy	$\|\omega_{xy}\|^2$	-0.2
Joint Velocity	$\|\dot{q}\|^2$	-1×10^{-4}
Joint Acceleration	$\|\ddot{q}\|^2$	-1×10^{-7}
Base Acceleration	$\|\dot{v}\|^2 + \|\dot{\omega}\|^2$	-1×10^{-4}
Action Rate	$\|a_t - a_{t-1}\|^2$	-0.5
Joint Position Limit	$1_{q>q_{\max}} + 1_{q<q_{\min}}$	-1.0
Collision	$n_{\text{collision}}$	-10.0
Feet Swing	$1_{\text{feet swing}} \cdot 1_{\text{swing period}}$	3.0
Feet Slip	$1_{\text{feet stance}} \cdot \|v_{\text{feet}}\|^2$	-0.1
Feet Yaw	$\|\psi_{\text{feet}} - \psi_{\text{base}}\|^2$	-0.1
Feet Roll	$\|\phi_{\text{feet}}\|^2$	-1.0
Feet Distance	$\max(d_{\text{ref}} - d_{\text{feet}}, 0)$	-10.0

A.2 Extensive Results of Training: Arms Retracted Configuration

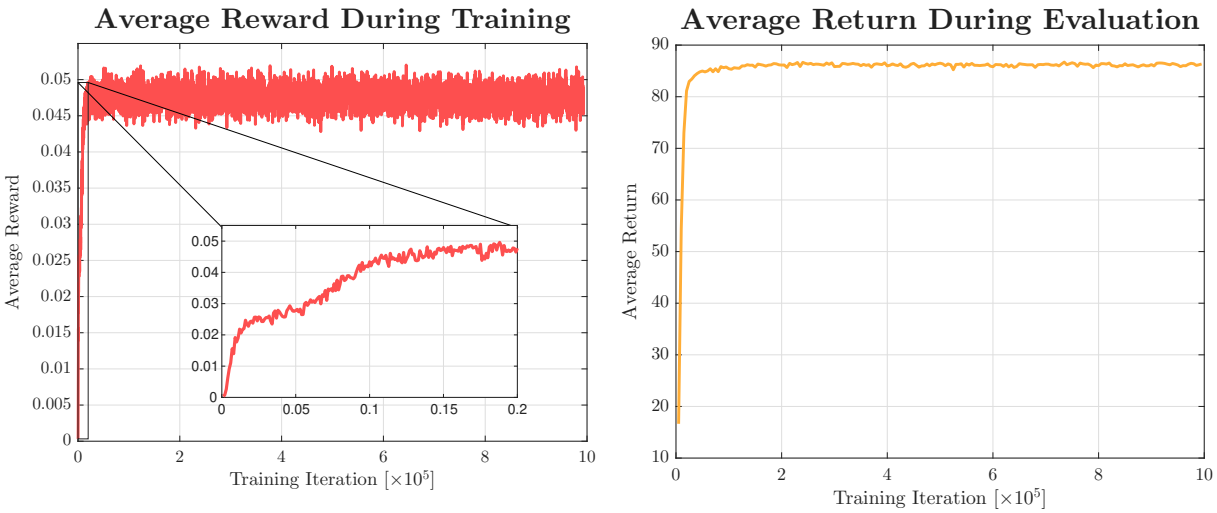


Figure A.1: Additional Results During Training for the Average Reward and Return.

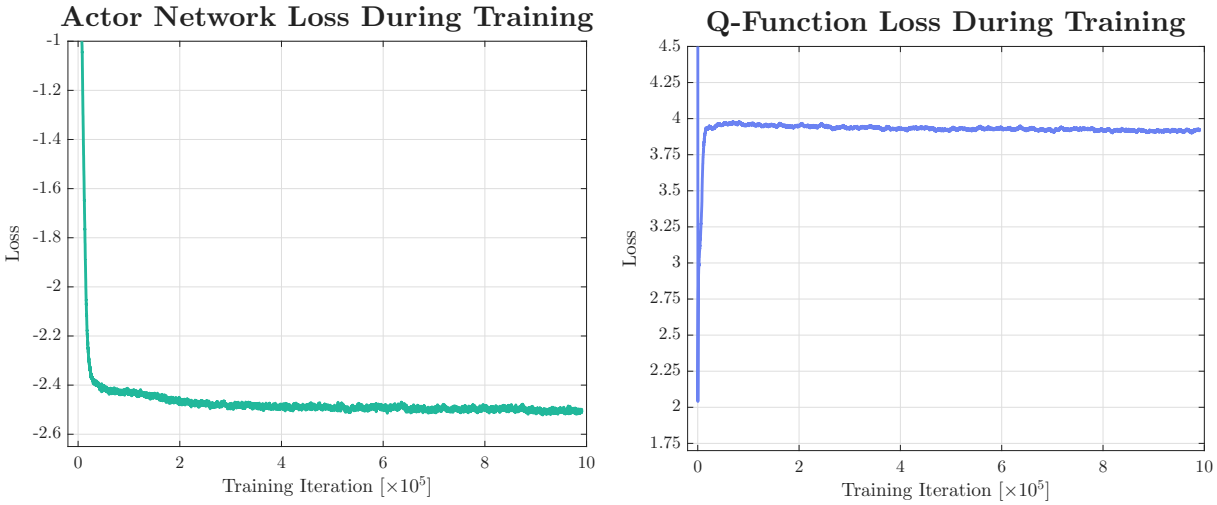


Figure A.2: Loss Functions of the Actor-Critic Architecture of the Training Algorithm.

A.3 Extensive Results of Training: Arms Rested Configuration

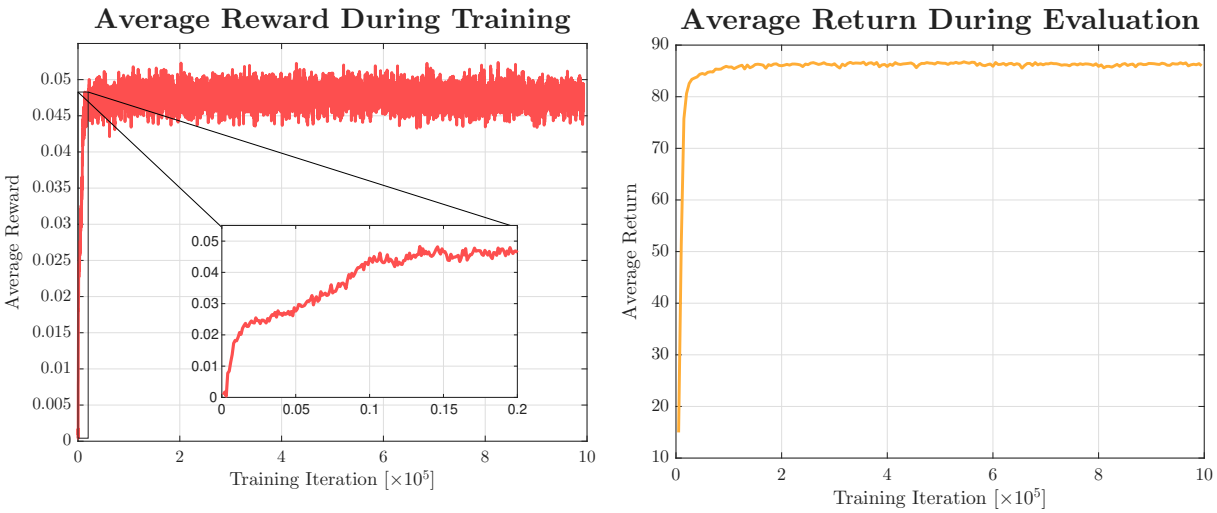


Figure A.3: Additional Results During Training for the Average Reward and Return.

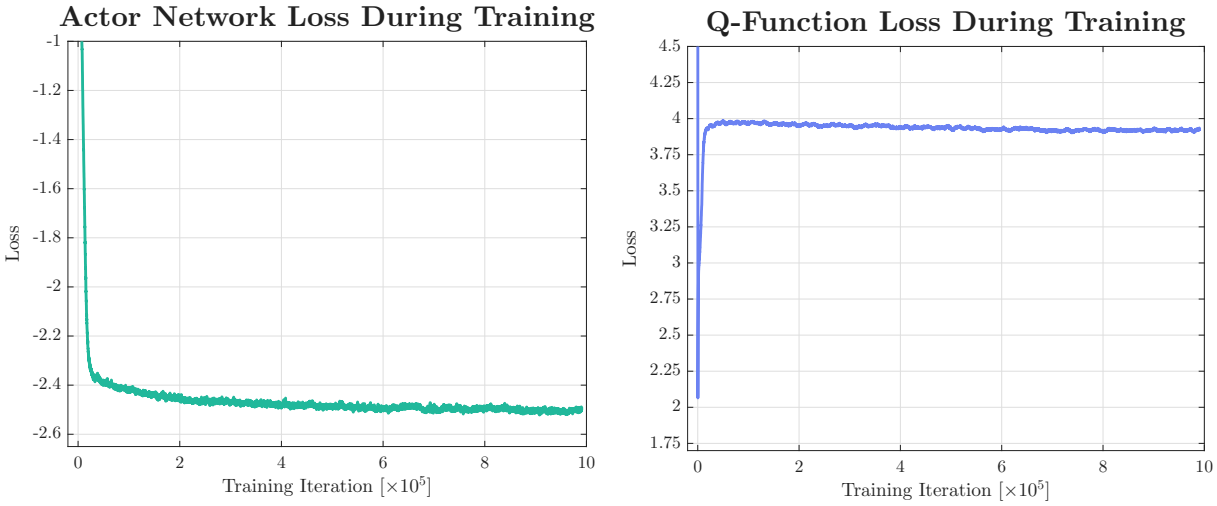


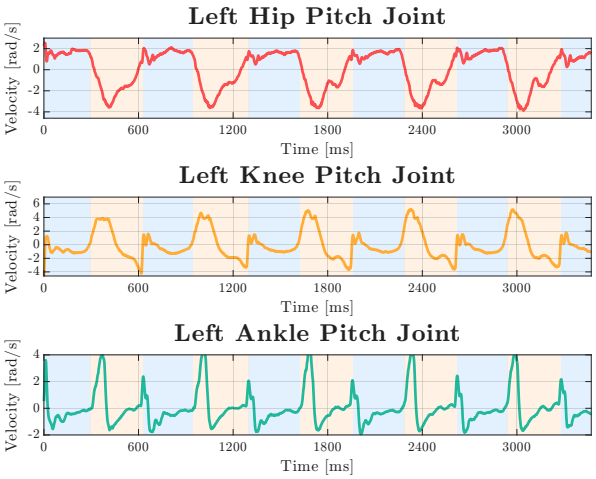
Figure A.4: Loss Functions of the Actor-Critic Architecture of the Training Algorithm.

A.4 Joint Space of the Robot Model

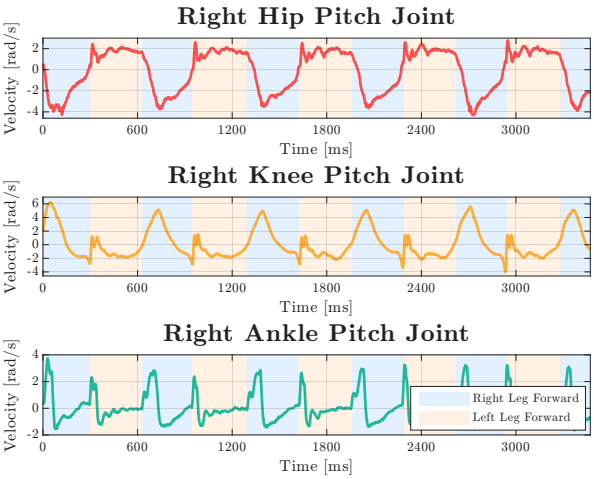
Table A.3: Joint Angle Limits in Radians for Upper and Lower Body, as Detailed in [20]

Joint	**Minimum Angle Limit** [rad]	**Maximum Angle Limit** [rad]
Head Yaw	-1.01	1.01
Head Pitch	0.31	0.82
Left Shoulder Pitch	-3.28	1.19
Left Shoulder Roll	-1.64	1.54
Left Elbow Pitch	-2.23	2.23
Left Elbow Yaw	-2.09	0.03
Right Shoulder Pitch	-3.28	1.19
Right Shoulder Roll	-1.54	1.64
Right Elbow Pitch	-2.23	2.23
Right Elbow Yaw	-0.03	2.09
Waist Yaw	-1.01	1.01
Left Hip Pitch	-2.06	2.06
Left Hip Roll	-0.37	1.54
Left Hip Yaw	-1.01	1.01
Left Knee Pitch	0.00	2.15
Left Ankle Pitch	-0.40	0.85
Left Ankle Roll	-0.42	0.78
Right Hip Pitch	-2.06	2.06
Right Hip Roll	-1.54	0.37
Right Hip Yaw	-1.01	1.01
Right Knee Pitch	0.00	2.15
Right Ankle Pitch	-0.40	0.85
Right Ankle Roll	-0.42	0.78

A.5 Real Deployment Velocities

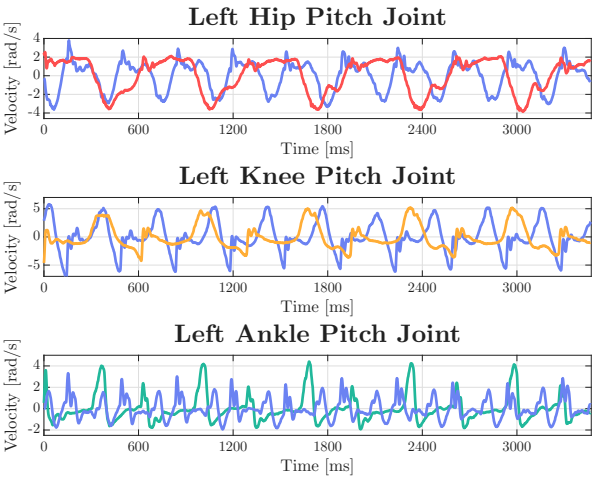


(a) Left Leg Joint Angular Velocities in a Forward Walk Cycle.

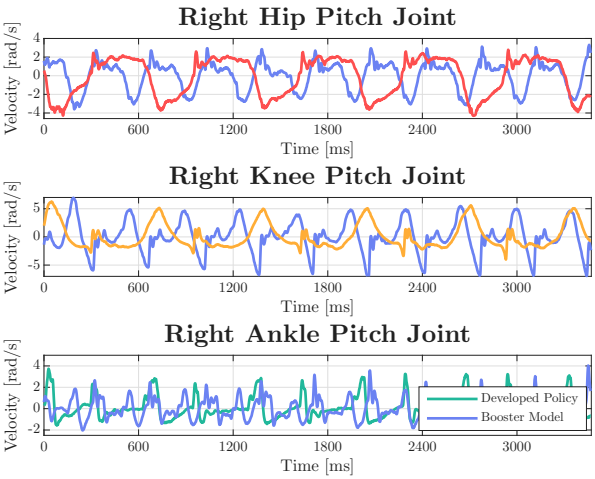


(b) Right Leg Joint Angular Velocities in a Forward Walk Cycle.

Figure A.5: Developed locomotion policy deployment on the real robot velocities.

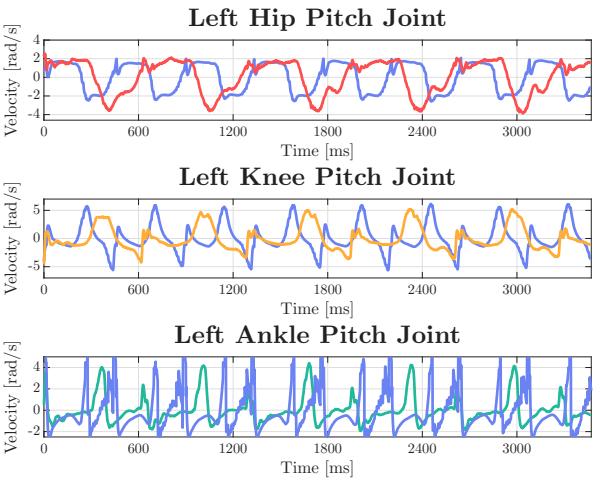


(a) Left Leg Joint Angular Velocities in a Forward Walk Cycle.

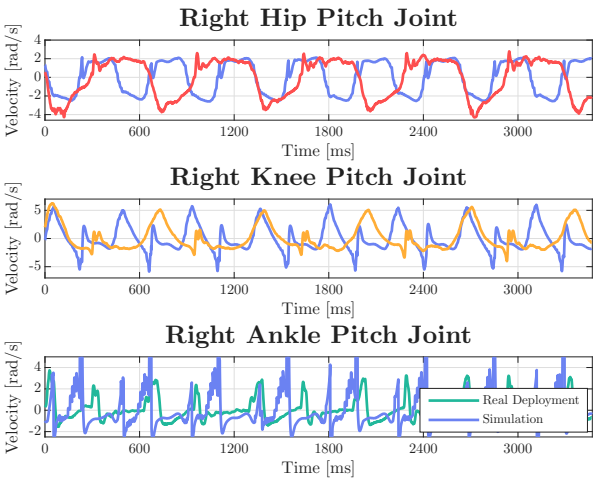


(b) Right Leg Joint Angular Velocities in a Forward Walk Cycle.

Figure A.6: Developed locomotion policy *vs* original booster model velocities.



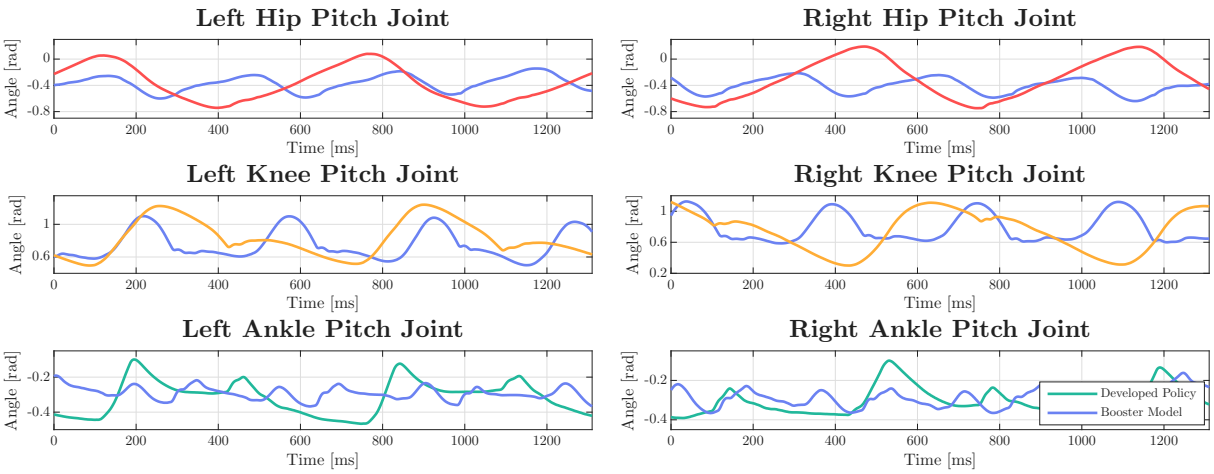
(a) Left Leg Joint Angular Velocities in a Forward Walk Cycle.



(b) Right Leg Joint Angular Velocities in a Forward Walk Cycle.

Figure A.7: Sim2Real gap velocities.

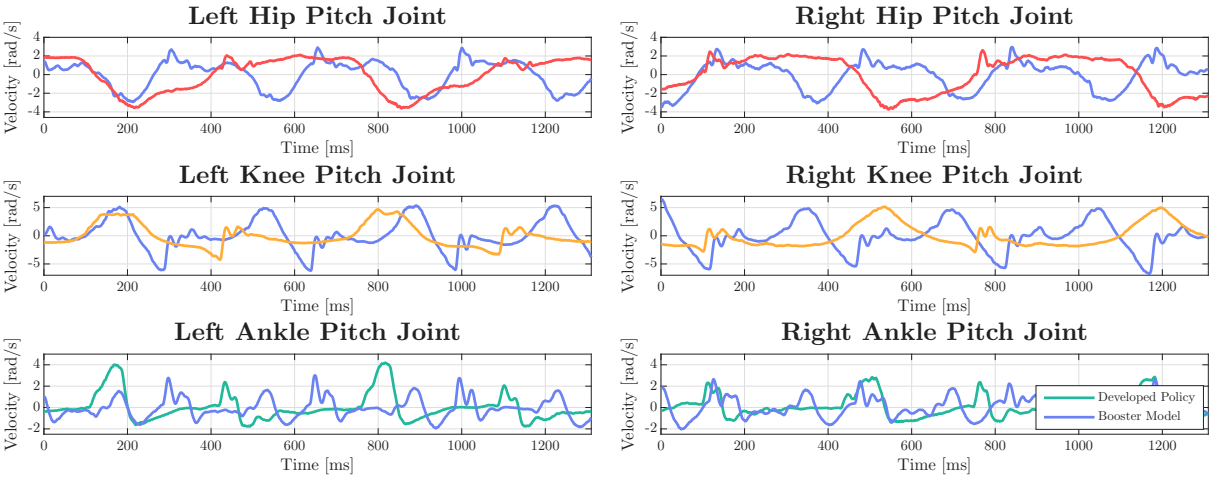
A.6 Detailed Look into Developed Policy *vs* Original Booster Model



(a) Left Leg Joint Angles in a Forward Walk Cycle.

(b) Right Leg Joint Angles in a Forward Walk Cycle.

Figure A.8: Zoomed-in Developed locomotion policy *vs* original booster model positions.

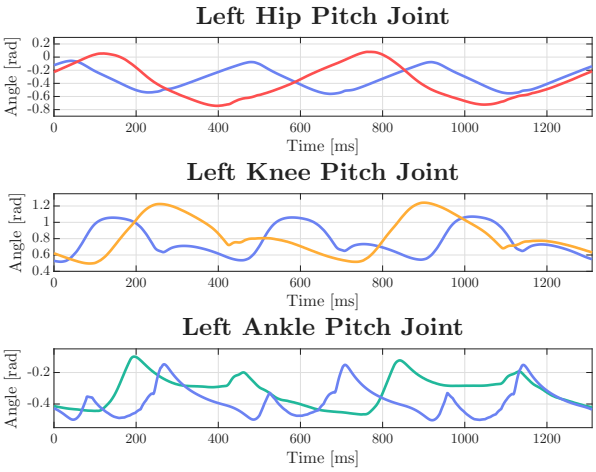


(a) Left Leg Joint Angular Velocities in a Forward Walk Cycle.

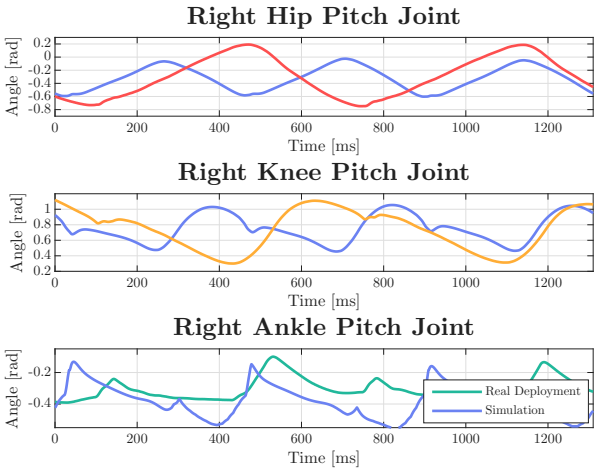
(b) Right Leg Joint Angular Velocities in a Forward Walk Cycle.

Figure A.9: Zoomed-in Developed locomotion policy *vs* original booster model velocities.

A.7 Detailed Look into Sim-to-Real Gap

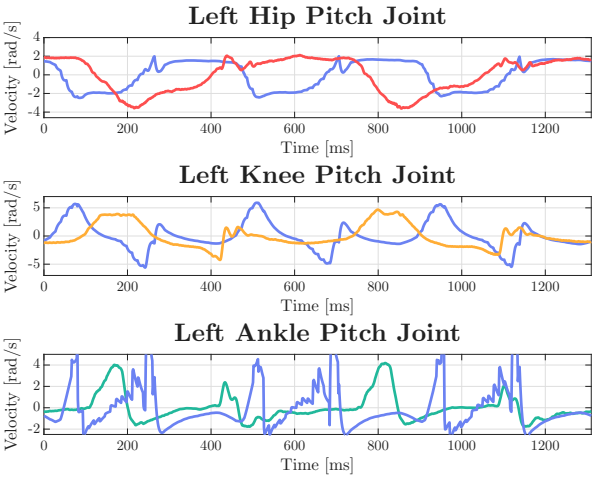


(a) Left Leg Joint Angular Positions in a Forward Walk Cycle.

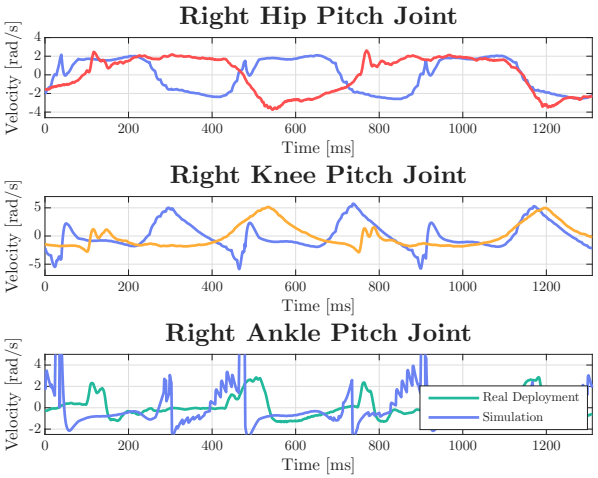


(b) Right Leg Joint Angular Positions in a Forward Walk Cycle.

Figure A.10: Zoomed-in Sim2Real gap positions.



(a) Left Leg Joint Angular Velocities in a Forward Walk Cycle.



(b) Right Leg Joint Angular Velocities in a Forward Walk Cycle.

Figure A.11: Zoomed-in Sim2Real gap velocities.