UNIVERSITY OF CALIFORNIA

Los Angeles

Learning Bipedal Locomotion
Using Central Pattern Generators and Deep Reinforcement Learning

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Computer Science

by

Avalon Vinella

2024

ABSTRACT OF THE THESIS

Learning Bipedal Locomotion Using Central Pattern Generators and Deep Reinforcement Learning

by

Avalon Vinella
Master of Science in Computer Science
University of California, Los Angeles, 2024
Professor Demetri Terzopoulos, Chair

We present a biomechanics-based framework for the locomotion of a muscle-actuated human model that is driven by Central Pattern Generators (CPGs). Our CPG system directly generates the activation signals of 22 lower body muscles to reproduce the oscillatory patterns of locomotive bipedal stepping. We employ a dual-module architecture that trains a CPG Tuner network and a Reflex Controller to jointly adjust the muscle signals during simulation in order to adapt to the challenges of 3D bipedal locomotion. These modules are trained simultaneously using the Soft Actor-Critic (SAC) reinforcement learning algorithm. Our CPG system achieves stable, realistic walking and we observe promising results toward task-driven locomotion and adjustable gaits.

The thesis of Avalon Vinella is approved.

Chenfanfu Jiang

Aditya Grover

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2024

To my family and friends

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Professor Demetri Terzopoulos for his guidance in my thesis and in my computer graphics education. I came to UCLA with the express goal of exploring the field of computer graphics and animation, and I am grateful for the opportunity that Demetri gave to me to experience and contribute to his novel research.

I also want to express my gratitude to Wuyue Lu, who graciously allowed me to collaborate with him on his research. He was extremely welcoming and willing to introduce me to his dissertation topic, and I cannot thank him enough for working with me. I thank Haotian Yi for sharing his ideas, work, and time with us as well; I thoroughly enjoyed our time together as a little research group. Additionally, I thank the previous students of the UCLA Computer Graphics & Vision Lab, without whom this work would not have been possible.

Thank you to my committee members, Professors Chenfanfu Jiang and Aditya Grover, for reviewing and supporting my thesis.

Lastly, thank you to my parents for their never-ending support and care.

CHAPTER 1

Introduction

Simulating locomotion is a pertinent problem in a wide array of fields, including computer animation, robotics, and biomechanics. In particular, dynamic and autonomous bipedal locomotion is an important skill for artificial humans, and there are various approaches to effective locomotive controllers depending on the specific use case. This thesis considers the problem of bipedal locomotion within the scope of biomimetically animating a biomechanically simulated musculoskeletal model.

Deep Reinforcement Learning (DRL) is a highly suitable approach for the task of learning locomotion, as it adopts an informed trial-and-error method inspired by natural learning. A DRL model learns to execute a task through user-engineered rewards based upon observations of the state at each step. For a dynamic human model, the DRL model may control movement through several outlets, such as the joints or muscles. In our case, we would also like to consider physiological systems that may contribute to or assist human locomotion. Central Pattern Generators (CPGs) are spinal circuits that produce rhythmic movements irrespective of cerebral control or peripheral movement feedback. In simulated biomechanical systems, we can treat CPGs as a system of oscillators that directly control muscle activation.

In this thesis, we utilize deep reinforcement learning to produce oscillation-based bipedal locomotion of a biomechanical human model through the tuning of central pattern generators.

1.1 Motivation and Contributions

The problem of simulated locomotion is already a well-explored field, especially within the context of deep learning. However, while the inclusion of CPGs in such models is less prevalent, the physiological system offers several merits.

Our primary motivation in integrating CPGs into conventional reinforcement learning techniques is to emphasize anatomical fidelity and biomechanical accuracy. Unlike simpler articulated bodies driven by joint motors, a muscle-actuated model is meant to more closely mimic the anatomical basis and dynamics of actual human movement. Thus, we strive to generate locomotion using a physiologically rooted system. Our investigation not only works to improve the learned gaits from a visual or aesthetic perspective, but also to elucidate how the biological CPGs may contribute to human locomotion in real life.

An added benefit of using CPGs to enhance the realism of the simulation is improved explainability. Pure mimic-based or reflex-based control will often operate upon a "black box" paradigm in that, beyond how one formulates the reward mechanism, it is difficult to determine what training data the model is leveraging to improve. By introducing an additional layer of control via the CPG oscillators, through examination of the CPG signals that determine the appropriate muscle activations, we can observe how the model adjusts the CPG weights and parameters to activate muscles so as to best achieve its goal. This gives us valuable insight into specific oscillatory patterns that are desirable and most effective toward successful locomotion.

Lastly, using CPGs as a low-level muscle control mechanism yields the potential for simple and user-accessible tuning and control of locomotion. As Chapter 2 and Chapter 6 will explore, CPG generated gaits can often be modified via a few simple parameters, such as frequency and amplitude, rather than requiring additional resources to train a new model to reproduce a different gait.

Our research develops previous work in the field of CPG-driven locomotion in multiple novel directions. First, we use the CPG as a method of generating the action, not as a target or reference pattern as some previous works do; the CPG directly drives the final locomotion of our musculoskeletal model. Similarly, we output muscle activation signals rather than joint torques, which is closer to how a biological CPG would operate. Our humanoid model is biomechanically-based and it simulates muscles on top of traditional articulated dynamics. Our model is also generally more complex than the models used in prior work, which often resemble simple quadrupeds. This raises additional issues of balance and a larger action space, which make stable bipedal locomotion more challenging.

1.2 Overview

The remainder of this thesis is organized as follows:

Chapter 2 explores related work in the animation, robotics, and biomechanics fields.

Chapter 3 details the implementation of our learning-based CPG system.

Chapter 4 introduces our proposed dual-module locomotion controller, which is comprised of a CPG Tuner and Reflex Controller, along with the method in which we train them.

Chapter 5 specifies the simulation details of our experiments.

Chapter 6 reports experiments that we performed in formulating our models, including phase regulation and task-driven training.

Chapter 7 further elaborates on our design decisions and compares our methods to alternative approaches to tackling similar problems.

Finally, Chapter 8 presents our conclusions, the limitations of our work, and promising avenues for further exploration.

CHAPTER 2

Related Work

While various studies have demonstrated the existence of CPGs among certain quadrupedal species, it has not yet been proven that they are also present in humans and, if so, how they contribute to bipedal locomotion. Nevertheless, in simulated biomimetic systems, schemes that integrate CPGs have shown advantages in ease of control and robustness over purely reflex-based systems. To contextualize our work and justify the relevance of CPGs, we review related studies on locomotion and CPGs from the domains of physiology, robotics, and computer animation in both the biological and artificial contexts.

2.1 Biomechanical Human Animation

In contrast to animating human models via the direct application of torque at joints, biomechanical animation enables the dynamic generation of motion through the modeling of contractile muscle forces that actuate the skeleton realistically. Human bodies are highly complex, and realistic muscle-actuated models require both musculoskeletal detail and sophisticated controllers to act as the motor center of the "brain". In view of the complexity of human anatomy, the problem of biomechanical human animation is often broken into smaller musculoskeletal groups; previous works have focused specifically on the face (Lee et al., 1995; Kähler et al., 2002; Sifakis et al., 2005), neck (Lee and Terzopoulos, 2006), upper body (DiLorenzo et al., 2008; Zhou, 2019; Zordan et al., 2004; Lee et al., 2009), hands (Sueda et al., 2008; Tsang et al., 2005; Nierop et al., 2008), and legs (Wang et al., 2012; Dong et al., 2002; Delp et al., 1990). A few researchers have successfully animated the biomechanical structure of the entire body Nakamura et al. (2005); Si et al.

(2015); Zhou (2019).

Control of physics-based human models take many approaches. Faloutsos et al. (2001) developed composable controllers to perform a variety of dynamic full-body movements by transitioning between key poses. Nakada et al. (2018) and Zhou (2019) leverage deep neural networks to learn cooperating neuromuscular motor controllers for each of the 6 musculoskeletal complexes—torso, cervicocephalic, 2 arms, and 2 legs. Specifically in the realm of locomotion, Song and Geyer (2015) used a reflex-based approach in which a "spinal" layer responds to the body state and environment and a "supraspinal" layer handles the gait control. Reinforcement learning is also a popular approach to the control of muscle-actuated locomotion, which we will explore further in Section 2.3.

2.2 Central Pattern Generators

CPGs on their own are a compelling area of study as a potential source of locomotive impulses inherent to some vertebrates. Minassian et al. (2017) explored the uncertain existence and role of CPGs in human locomotion through the review of biological studies and modeled biomechanical systems, providing arguments for and against their presence. It is difficult to prove with certainty that the CPG circuits actually contribute specifically to human locomotion; nevertheless, the authors posit that the stepping reflex demonstrated by infants may be mediated by CPGs that are later adapted into the adult locomotion system. This concept of a pipeline of locomotive development beginning with CPG signals is analogous to our RL approach, inasmuch as we begin with a basic CPG system that is iteratively tuned and combined with supporting systems to produce stable bipedal locomotion. Most relevant to this work, the authors note that in comparing modeled bipedal locomotive systems, those that used a CPG were more resistant to perturbation and generally were simpler models in that the speed and gait was easily changeable through the adjustment of a few parameters. Similarly, Ijspeert (2008) outlines the benefits of using CPGs in robotics and further describes CPG design considerations. In addition to the aforementioned stability, robustness, and ease of adjustment, he also highlights the scaling capabilities of CPG based systems, including the integration of sensory feedback signals and learning algorithms.

Ijspeert (2001) and Ijspeert et al. (2007) thoroughly investigate the implementation of CPG systems in simulated and physical salamander robots. They designed CPGs for the body and limbs of an artificial salamander such that it could successfully swim in water and trot on land subject to the tonic states of the muscle actuators and the external forces. In addition to the limbs, the sinusoidal motion of the body also contributes to both terrestrial and aquatic locomotion. Similar to an RL approach, they employ a genetic algorithm to learn optimal CPG parameters according to a fitness function. Control is realized by applying a “drive” to the system that modulates the oscillation frequency and, in the robotic implementation, determines the type of gait.

Using a comprehensive biomechanical human model, Si et al. (2015) apply CPGs to control the more complex problem of realistic simulated human swimming. Their CPGs output the desired muscle lengths, which are then fed into a PD feedback loop to synthesize the activation signal to achieve that length. Their virtual swimmer learns to mimic various swimming styles using incremental locally weighted regression (ILWR) to extract the fundamental frequency of key-framed reference swimming motions, as in (Gams et al., 2009). In this method, different swimming styles yield different learned weights for the CPG parameters. Once learned, a higher-level voluntary controller can easily switch between and alter the swimming style and speed of the model. However, their external factors accounted only for the forces between the water and the deformable skin and flesh of a body that floats rather stably; thus, their CPG controller alone will not suffice for terrestrial locomotion, which requires continual adjustments for balance.

2.3 Locomotion Using Reinforcement Learning

Lee et al. (2019) use DRL to control the muscle actuators of a complex biomechanical model. Notably, their scheme is able to simulate a fair number of muscles in the musculoskeletal system. Essentially, they solve an underspecified inverse kinematics problem in which the muscle activations are determined based on the desired joint movements specified by reference data; here, they utilize DRL to skip the computation of a complex dynamic numerical system.

Peng et al. (2018) achieve physically-informed movements based upon kinematic data through a task-oriented RL model. Their strategy utilizes a neural network that maps pairs of body states and goals to a distribution over the action space. Their larger RL model then learns which actions will best mimic motion-captured or key-framed reference data while also achieving various task objectives, such as moving in specified directions or traversing uneven terrain. This approach is not biomechanically based, but it nevertheless demonstrates the success of applying RL to learn realistic movement.

2.4 Integrating CPGs and RL

Especially in the robotics field, there have been more explorations of the use of CPGs in RL models. Bellegarda and Ijspeert (2022) use DRL to learn the oscillation parameters of the CPG of a physical quadraped robot. A key experiment in their approach is the level of sensory information that should be included in the model's observation space. They found that the overall position and velocity, foot contact, and CPG states were sufficient to generate locomotion that was robust to significant perturbation. Unlike our work, however, their CPG is not learned or adjusted over an episode, and is instead treated as another observational body space.

More similar to our approach of learning a CPG, Campanaro et al. (2021) use CPGs as actors in actor-critic RL feedback loops. They employ a multilayer perceptron (MLP) model that processes sensory data to feed into the CPGs to generate natural and robust gaits. The weights of the MLP and CPG are trained together to maximize symbiosis within the system and encourage the best utilization of each component.

CHAPTER 3

CPG Implementation and Control

As previously described, CPGs are spinal circuits that induce periodic motor patterns in vertebrates. We utilize these oscillatory circuits to directly produce the muscle activation signals that will actuate the simulated biomechanical human model.

3.1 Dynamical System Model

We formulate CPGs as nonlinear dynamical systems as in (Gams et al., 2009), since this approach allows us to model the CPG after an existing oscillatory signal.

The trajectory of the system y represents the output signal of the CPG. The oscillation of the system around an anchor point g is characterized as follows:

$$\dot{z} = \Omega \left(\alpha_z(\beta_z(g-y)-z) + \frac{\Sigma_{i=1}^{N} \Psi_i w_i r}{\Sigma_{i=1}^{N} \Psi_i} \right) \tag{3.1}$$

$$\dot{y} = \Omega z \tag{3.2}$$

$$\Psi_i = \exp(h(\cos(\Phi - c_i) - 1)). \tag{3.3}$$

The intermediate variable $z(t)$ describes the first-order derivative of the trajectory y, and Φ is the phase of the signal. Here, Ω is the fundamental frequency; as we are modeling stepping as a periodic motion, we can consider Ω as a constant $\frac{2\pi}{T}$, where T is the period either extracted from the reference motion or specified by a goal task. As in (Gams et al., 2009) and (Si et al., 2015), we set the positive constants $\alpha_z = 8$ and $\beta_z = 2$ to ensure critical damping such that we establish monotonic variation around the anchor point g. N denotes the number of Gaussian-like periodic kernel functions Ψ_i of width h. Again, we follow Gams et al. (2009) and Si et al. (2015) and use $N = 25$ and $h = 2.5N$ for all simulations, with c_i equally spaced between 0 and 2.5π over N intervals. The amplitude control parameter is r, which is initially set to 1.0.

3.2 CPG Learning

We desire a trajectory modeled after a reference pattern generated from the muscle patterns of reflex-based locomotion. To this end, we initialize the weights w_i of the CPG system using Incremental Locally Weighted Regression (ILWR) (Vijayakumar and Schaal, 2000) to minimize the quadratic error

$$J_i = \sum_{t=1}^{P} \Psi_i(t)(f_{\text{targ}}(t) - w_i r(t))^2, \tag{3.4}$$

in reference to the target data

$$f_{\text{targ}} = \frac{1}{\Omega^2}\ddot{y}_{\text{demo}} - \alpha_z \left(\beta_z (g - y_{\text{demo}}) - \frac{1}{\Omega}\dot{y}_{\text{demo}} \right), \tag{3.5}$$

formulated using the reference muscle activations y_{demo}.

As specified by Gams et al. (2009), locally weighted regression can be performed in a batch or by incrementally minimizing J_i to accommodate an input signal stream. We choose the latter in order to reduce the effects of noise and perturbations that may be present in our reference signal. In this scheme, we perform incremental regression using recursive least squares with a forgetting factor of λ; note that if $\lambda = 1$, batch and incremental regression will learn the same weights w_i. Depending on the reference data we use, we set $\lambda \in [0.95, 0.99]$, since a forgetting factor less than 1 encourages a bias toward the most recently seen data.

Error J_i is minimized by iteratively updating the kernel weights as follows:

$$w_i(t+1) = w_i(t) + \Psi_i P_i(t+1) r(t) e_r(t), \tag{3.6}$$

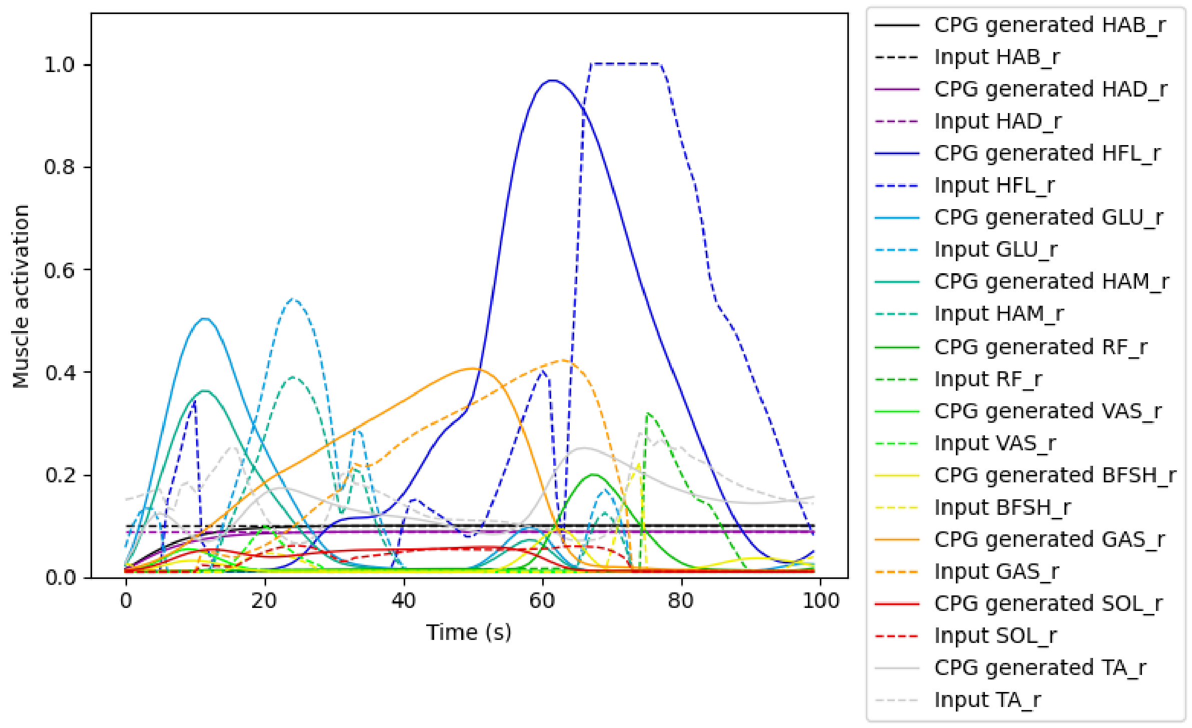


Figure 3.1: One cycle of input data and the learned CPG generated activations for some right side muscles in a 2D environment. Dashed lines indicate the target, while solid lines are the corresponding learned trajectories.

where

$$P_i(t+1) = \frac{1}{\lambda}\left(P_i(t) - \frac{P_i(t)^2 r(t)^2}{\frac{\lambda}{\Psi_i} + P_i(t)r(t)^2}\right) \tag{3.7}$$

and

$$e_r(t) = f_{\text{targ}}(t) - w_i(t)r(t). \tag{3.8}$$

Here, P_i is the inverse covariance matrix as introduced by Söderström and Ljung (1983), and e_r is the weight learning error, which evaluates how well the current weights approximate the target data. We begin regression with $w_i = 0$ and $P_i = 1$.

In Figure 3.1, we see that the system captures the overall shape of different input signals while smoothing noise in the data.

During RL training, the model will tune the CPG's parameters using the modulation signals φ and the vector ρ, where φ modulates the phase velocity to adjust the frequency and each ρ_i will modulate amplitude and thereby the strength of the output signal. The non-linear behavior of the CPG ensures smooth modulations, and we can modify Equation 3.1 to include the modulators

$$\dot{z} = \Omega_{\text{tuned}} \left(\alpha_z(\beta_z(g - y) - z) + \frac{\Sigma_{i=1}^{N} \Psi_i w_i \rho_i r}{\Sigma_{i=1}^{N} \Psi_i} \right), \tag{3.9}$$

where

$$\Omega_{\text{tuned}} = (1 + \varphi) \times \Omega_{\text{original}}. \tag{3.10}$$

This dynamical CPG design allows for effective learning from reference signals as well as smooth subsequent adjustments, which best suits our online training needs.

CHAPTER 4

Learning Locomotion

Simply learning and reproducing a periodic signal via our CPG system is insufficient to drive stable bipedal locomotion. The imitated reference signal is merely a starting point to encourage our locomotion controller to emulate a stepping pattern, but it must be further refined in order to adapt to time-dependent body states and environments. To this end, we make use of deep reinforcement learning to iteratively improve our controller's performance and robustness.

4.1 Soft Actor-Critic

We employ a Soft Actor-Critic (SAC) DRL model as introduced by Haarnoja et al. (2018) as the learning network for tuning our CPG. At its core, it operates utilizing an actor-critic architecture in which an evaluation policy (the critic) iteratively judges a value function (the actor) such that the actor is incentivized to converge to an optimal function via stochastic gradient descent. The off-policy nature of SAC encourages diverse behaviors and the actor's broad exploration of the continuous action space by augmenting the standard maximum reward learning objective with an entropy maximization term. Appendix A presents the formulation and algorithmic details.

4.2 Dual-Module CPG and Reflex Control

We propose a learning network that consists of a dual-module architecture in which a parallel CPG Tuner and Reflex Controller are trained and combined such that both are

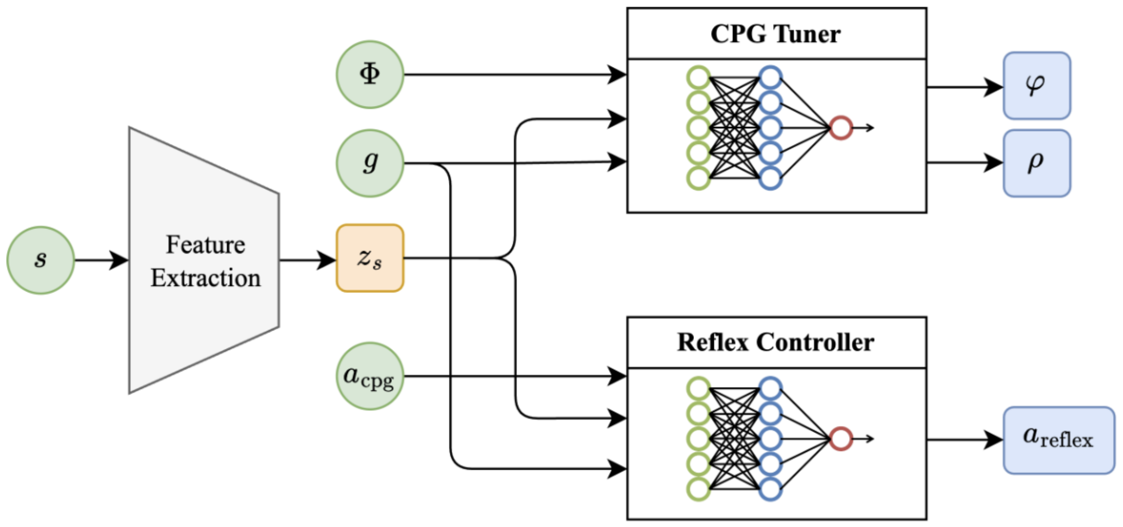


Figure 4.1: The dual-module architecture of our locomotion controller. Our feature extraction transforms the body state s into a latent space representation z_s, which is then fed to both the CPG Tuner and Reflex Controller. Φ is current phase of the CPG, g is a goal presentation, and a_{cpg} are the current outputs of the CPG, which are muscle activations. The CPG Tuner outputs adjustments φ and ρ to the signal's frequency and amplitude, respectively, while the Reflex Controller outputs muscle activations a_{reflex}.

actors within the SAC policy. We prefer to isolate the two controllers so that the user is able to enforce balance between their contributions to the policy; in some cases, the model may learn to rely solely on the reflex controller, which is undesirable. We want to ensure that the CPG is being properly and effectively utilized in the network while being supported by the reflex controller. The intention of this scheme is for the CPG to act as the voluntary control over the locomotive patterns.

Both modules have access to a latent space representation of the body state, as well as a goal representation. To most simply encourage a locomotive motion, we provide the model a reference motion sequence, which we generated using the reflex-based controller introduced by Song and Geyer (2015) in a 2D environment (further details are provided in Chapter 5).

This SAC actor outputs a concatenation of the outputs of the CPG Tuner and Reflex Controller, as detailed in the following sections.

4.2.1 CPG Tuner

The CPG Tuner network outputs changes to the signal's frequency φ and amplitude ρ as described in Section 3.2. In cases where the signals for individual muscles should be learned separately, the signals will share a fundamental frequency, but will allow for independent tuning of their respective amplitudes.

At the beginning of training, the CPG kernel weights are randomly initialized. Over the length of the episode, the model learns to adjust the CPG weights and parameters to best maximize the reward function. At each training checkpoint, which occurs at user-specific intervals, the learned CPG weights at the end of the preceding episodes are averaged to serve as the new initial weights; this will effectively learn better initial weights to make more efficient use of each episode's learning.

4.2.2 Reflex Controller

The Reflex Controller additionally sees the current CPG's output signals in its observation space, which it utilizes to output muscle activations based on how the bipedal model should reflexively respond to the environment in order to maintain balance. Its behavior is learned directly by the SAC policy; over the course of training, it learns how to keep the CPG-driven skeleton from falling or otherwise failing.

4.3 Reward Design

The key to the reinforcement learning approach is the design of the reward function; the model can only properly learn given the right components that lead to reward or penalty. We carefully choose what we consider to be a successful or failed attempt at locomotion and evaluate the model's performance appropriately. A core piece of our reward is the mimicking of the reference sequence; the model is rewarded on its similarity in position and velocity to the reference. We crucially penalize the model for early termination; that is, falling, or straying too far from the reference sequence. In our scheme, these two factors are the main drivers toward locomotion. Unlike other DRL approaches, we do not expressly reward the model for moving in the horizontal plane. We also include penalties to enforce smoothness, such as joint constraints and action regularization. The resulting reward function for each training step for our base model is then formulated as

$$r = w_m r_m + w_{\text{clip}} p_{\text{clip}} + w_{\text{smooth}} p_{\text{smooth}} + w_{\text{early}} p_{\text{early}}, \tag{4.1}$$

where p_{clip} penalizes activation outputs outside of the rang [0, 1], p_{smooth} is a consolidations of penalties associated with the regularization of the total action space, p_{early} is the penalty for early termination, and the mimic reward is defined as

$$r_m = w_q r_q + w_u r_u + w_e r_e, \tag{4.2}$$

where r_q is the reward for joint positional and rotational similarity, r_u for joint positional and angular velocity similarity, r_e for positional similarity of the end effectors, and w_q, w_u, and w_e are their associated weights.

4.4 Fine-Tuning

We train a baseline model with the reward function as described in Section 4.3. The training session is intended to teach the model what the overall pattern of locomotion should look like and how the CPG tuner and reflex controller should be adjusted to achieve that motion. In practicality, however, we do not want the model to replicate only the reference sequence; instead, we would like to be able to extrapolate the reference movement to achieve similar but altered gaits. To achieve this, we modify the reward function in a fine-tuning process that continues after a significant number of training steps. The particulars of varying the fine-tuning reward functions are explored in Chapter 6.

CHAPTER 5

Simulation

To conduct our muscle-driven experiments, we primarily use OpenSim (Delp et al., 2007), an open source software system for biomechanical modeling, simulation, and analysis. This chapter summarizes the practical details of our implementation, including the models and simulation methods.

5.1 OpenSim

The OpenSim API was specifically developed for the simulation and analysis of biomechanical motion and thus adequately suits our needs for muscle-driven dynamics. We modify the OpenSim-compatible `Gymnasium` learning environment used in the 2017–2019 NeurIPS "Learn to Move" competitions (Towers et al., 2023; Kidziński et al., 2018) in concert with the `Stable Baselines3` implementation of SAC (Raffin et al., 2021).

5.1.1 Biomechanical Model

OpenSim uses a custom model format, in which external and internal forces are prespecified within the model file. We use a simplified model presented by Kidziński et al. (2018), which simulates a total of 22 lower body muscles (11 for each side of the body) and 14 rotational degrees of freedom at the joints. Appendix B provides descriptions of each included muscle. For simplicity, the upper body of our model is arm-less and does not articulate. Muscles are modeled as Hill-Type actuators that connect bones and exert a spring-type force at its attachment points as in Thelen (2003). Joint constraints are also represented as rotational spring forces that limit rotation beyond realistic thresholds.

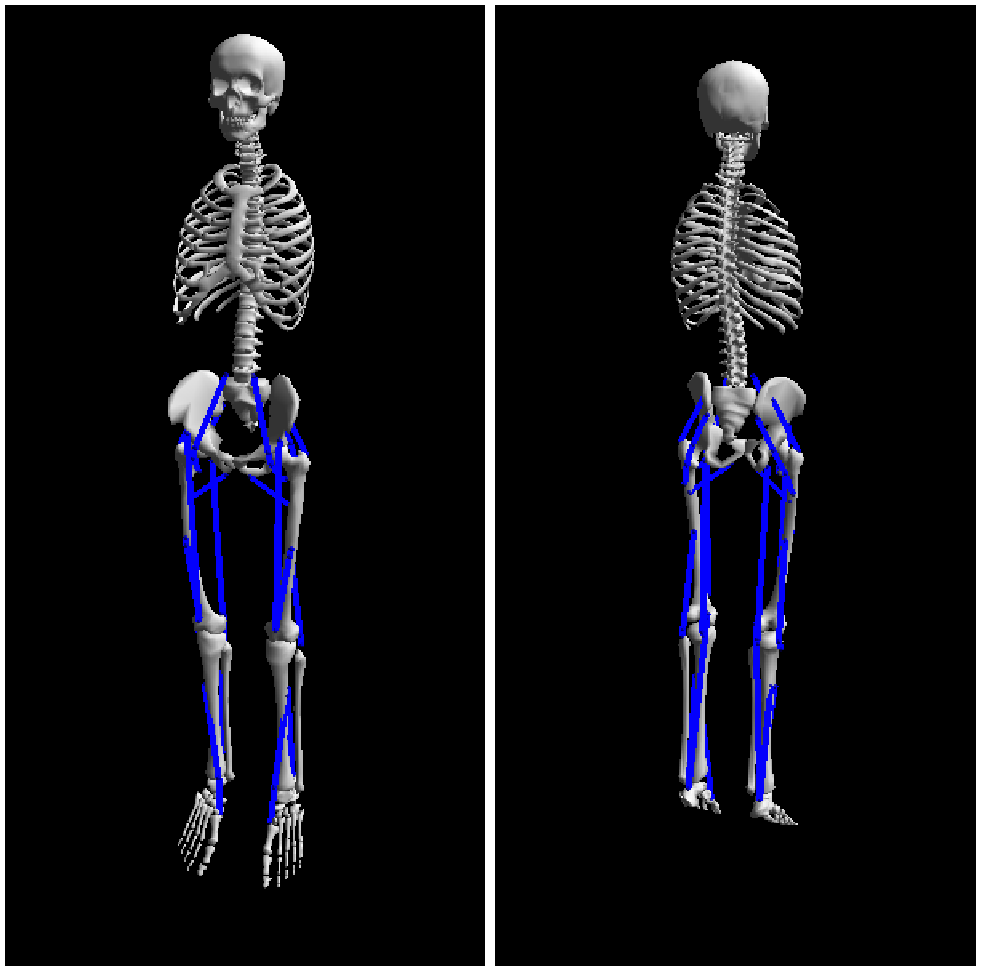


Figure 5.1: Front and back views of our OpenSim model created by Kidziński et al. (2018), with 22 muscle actuators in the lower body and 14 degrees of freedom.

Each episode of simulation initializes the model using the same initial pose, which is taken from a timestep of the reference sequence that we determined to be the most conducive to our CPG signal. After sampling poses from the reference data, our chosen initial pose induced the longest lasting locomotion using only the unmodulated CPG outputs.

5.1.2 Muscle Control and Simulation

We tune a CPG system in which each muscle has its own corresponding CPG signal, as previously detailed in Chapter 3. Every muscle is activated according to a linear combination of the outputs of the CPGs and Reflex Controller, both of which are direct muscle activations.

At each step, we perform forward dynamics based on the activations of each muscle to drive the skeleton. Once each muscle is actuated, the musculotendinous forces are calculated according to the Hill-Type model, the specifics of which are provided in Appendix C. Once the internal and external forces are calculated, Runge-Kutta-Merson integration is used to update the joints and bone rigid bodies as appropriate. We use an integration accuracy of 5e-5 and a step size of 0.01.

5.1.3 2D vs 3D Environments

We employ two models in our OpenSim trials; one that restricts movement in the lateral plane, and one that does not. The restricted 2D environment ensures that the biomechanical model will not turn or lean to the sides. This increases the likelihood of the model staying upright, thus making locomotion easier. We perform some of our experiments in this 2D space to examine how various factors affect the CPG, and later develop our controllers in the more challenging 3D environment.

Another challenge of the 3D space is the reduced usefulness of the reference sequence, which is recorded in the 2D space. Simply mimicking the reference will not produce stable locomotion in this approach, so it is essential that the model learns and relies on an effective CPG pattern. The model may also need to rely more heavily on the reflex controller to maintain stability.

CHAPTER 6

Experiments and Results

Our experiments are conducted as fine-tuning trials in the OpenSim framework, in which we continue training on top of a pretrained baseline model as described in Section 4.4. The training and testing environment runs for ten seconds of simulation time, and we simultaneously run ten instances during training.

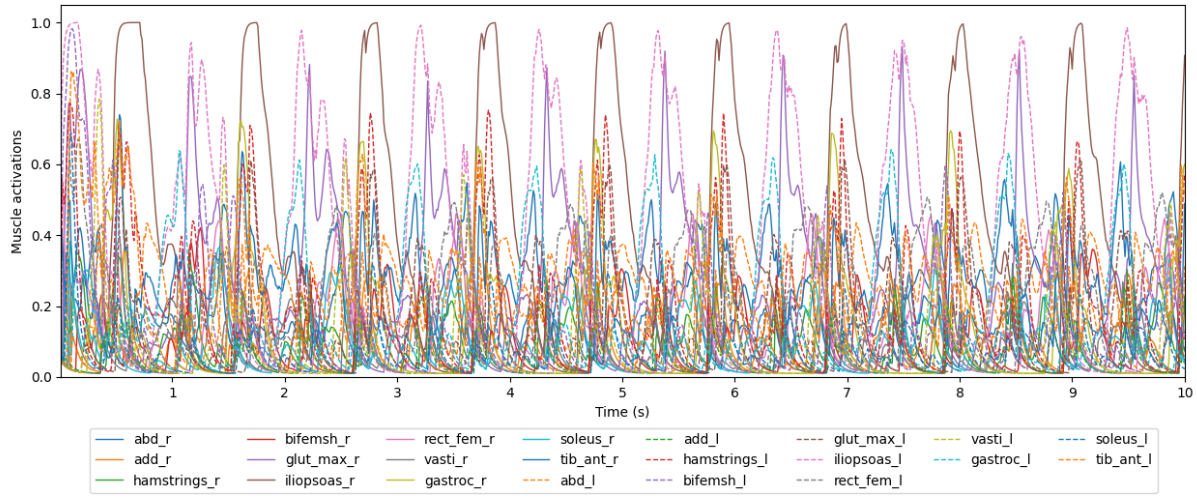
6.1 Baseline Model

We evaluate the experimental results reported in this chapter relative to the performance of a baseline model, which was trained for 14.8 million training steps. We reference its muscle activations, CPG phase, and footsteps in Figure 6.1 for comparison. Note that the left and right footsteps are uneven, where the left foot typically maintains contact with the ground for longer. This model does not consistently walk upright for the entire testing episode.

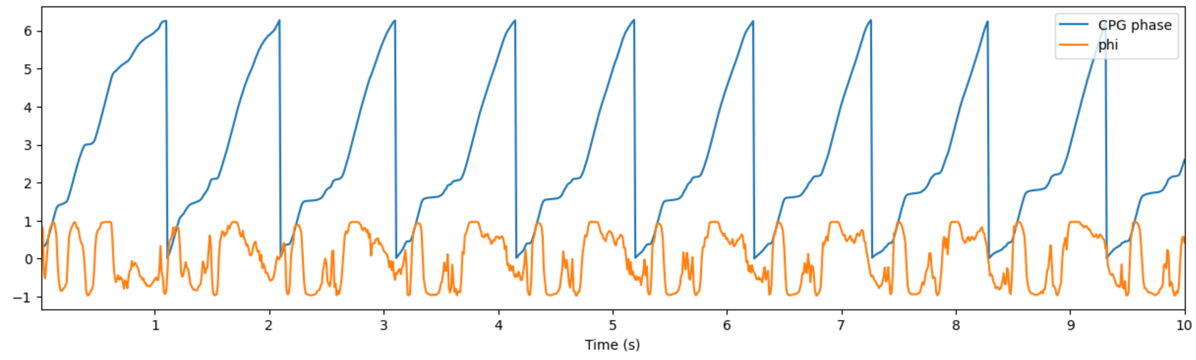
6.2 Time-Aware vs Timeless

In our initial formulation, we gave the RL model access to time observations. However, after removing this access, we found remarkable improvements in generating a steady gait. We believe that the model was learning a sequence of poses based on the observed time and ignoring the CPG information, which ultimately made the biomechanical locomotion model less robust and unable to walk more than a few steps without falling. Making a timeless model ensures that it is predicting muscle activations based on the body and

(a)



(b)



(c)

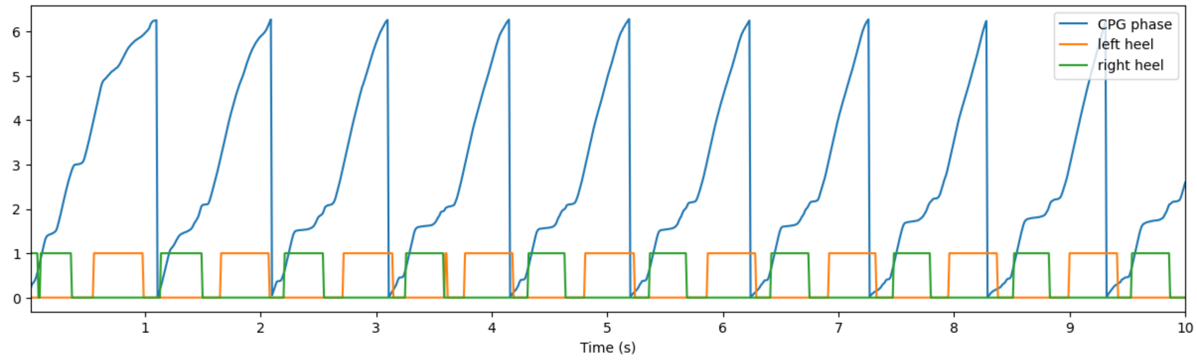


Figure 6.1: 10 seconds of simulation in the evaluation environment of the baseline 3D model. (a) The muscle activations of the right and left lower body muscles. (b) The CPG's phase, as well as φ, the frequency tuning variable. (c) The occurrence of each footstep within cycles of the CPG.

CPG states instead of abstract markers.

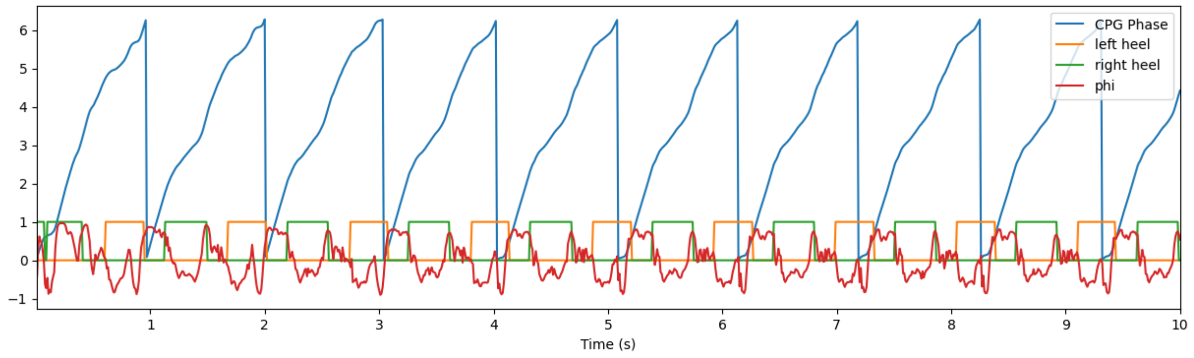
6.3 Phase Regulation

As the baseline model continues to step forward within an episode, we observed that the CPG would slowly fall out of phase with its expected fundamental frequency, generally via a slight increase in frequency that compounds over the episode. It is unclear whether this stems from similar behavior in the reference motion or from another source. However, in the 2D environment, we are able to mitigate this deviation by adding an appropriate penalty to the reward function. When the CPG approaches the end of a cycle, it is penalized according to the distance from the expected phase based on its initialized phase. Note that it is not necessarily desirable to limit the entirety of the cycle to the expected phase, as different phases of the stepping motion may benefit from taking more or less time. With this penalty, the resulting phase is much more similar to the expected phase, ensuring that the basic oscillatory property of the CPG is well utilized, as shown in Figure 6.2a.

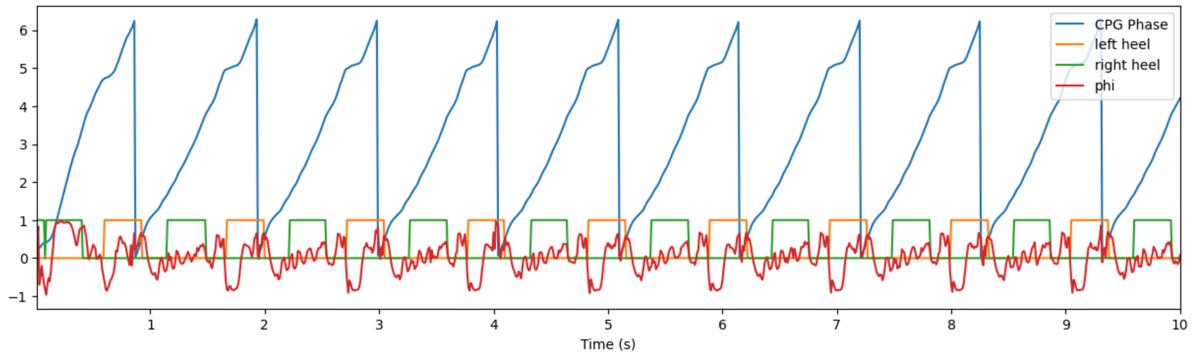
However, it must be certain that the CPG signal does not fall out of phase with the expected cycles of the model's footsteps. If this were to occur, the muscle activations supplied by the CPG would no longer be appropriate for the model's current locomotion phase, and the model will ultimately fall. Thus, we additionally penalize the angular distance of the stepping's phase from the CPG's phase. The isolated effects of this penalty are shown in Figure 6.2b. This will encourage more stable locomotion over longer episodes.

The combination of the two penalties, however, seemed both to shift the CPG phase away from the expected period and to generate rougher phase patterns (Figure 6.2c). Additionally, we noted that in our 3D experiments, the time dependent phase regulation did not have a strong impact on the CPG output signals, which were more inconsistent. We believe this indicates that due to the increased challenge of balance in the lateral direction, the model either prioritizes the output of the reflex controller over the CPG signal or fluctuates the signal's frequency too much for the regulation to have a distinct

(a)



(b)



(c)

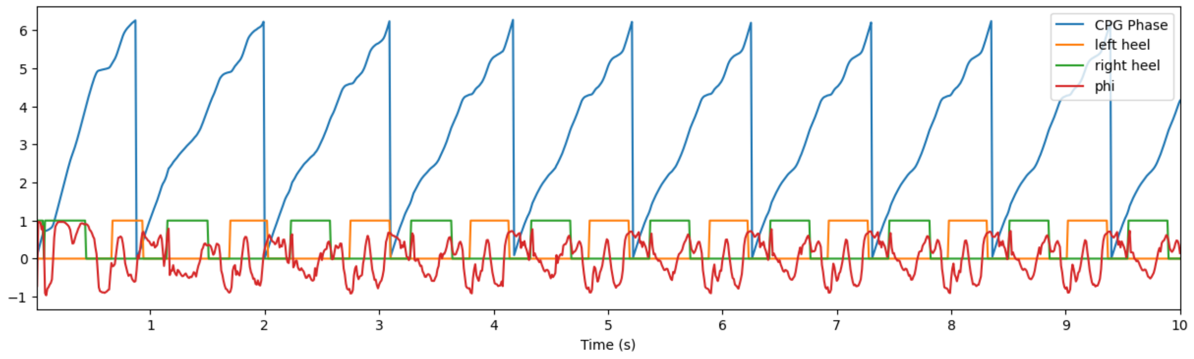


Figure 6.2: CPG phase, modulator φ, and footsteps using (a) time-based phase regulation, (b) footstep-based phase regulation, and (c) a combination of the two.

impact. In our later 3D experiments, we thus employ only the footstep-based phase matching.

In practice, we also found that the first contact of the right foot does not occur when the CPG's phase is zero, but instead slightly later. This may be due to the initial, more irregular modulations that occur at the very beginning of simulation during which the CPG and reflex networks are adjusting from the initial pose. This is acceptable in terms of realism, as the right step can include the "pre-swing" motion before contact. We therefore define the phase "start" to be 0.1 in our later experiments.

6.4 Task Driven Locomotion

Ultimately, we would like our model to be able to emulate slightly different gaits depending on user input. In these cases, we include goal-driven terms in our reward function such that our model can adjust CPG parameters in order achieve different speeds and directions.

As the model must move further away from its reference sequence in order to realize these velocity-based goals, we must significantly modify the reward function to increase generalizability.

First, the reward for matching the end effector positions is heavily diminished. These positions are described in global space and will necessarily differ when the velocity of the model is changed. Nevertheless, we found that excluding them from the reward function entirely resulting in the model adopting non-stepping and non-cyclical motions to achieve specific velocities, such as jumping forward. Therefore, we instead lower their impact on the reward. Similarly, we ignore global positions and translational velocities in the goal sequence, which effectively removes the pelvis translational position from the mimic reward. To support turning, we ignore the yaw of the pelvis. We also partially disable early termination and its corresponding penalty, as it partly judges based on the difference of joint and end effector positions compared to the reference sequence.

To support different speeds, we also sample the reference sequence by aligning the phase of its footsteps to the phase of the CPG, rather than time-based indexing. This modification not only assists in task-driven locomotion, but generally makes our model more robust, as perturbations that might disrupt the model's normal stepping will not throw it out of sync with the reference motion.

Finally, we include the reward for reaching the desired velocity. For our purposes, we only care about movement in the horizontal plane; the height of the body is free to change as it does in natural human gaits. The current body state is compared the task defined in the observation space, and it is appropriately rewarded for its similarity:

$$r_{\text{task}} = \exp(-w_{\text{task}}(\|v_t - v_{\text{goal}}\|_2)^2), \tag{6.1}$$

where the y component of v_t has been set to 0.

We separately examine trials in which the trajectory of the goal velocity is changed and its speed. Potential explanations and implications of these results are further discussed in Section 7.1.

6.4.1 Turning

We note that in our baseline, the model will drift toward the left side, seemingly due to a consistent imbalance in the length of the left and right footsteps. We find that when we set the goal velocity to be straight ahead (forward in the x direction), this issue is mitigated, as the model is incentivized to travel forward.

Turning proves to be a more difficult task. Given that the mimic sequence only moves in one direction, the mimic reward is heavily reduced if the goal direction is changed, as the end effector positions are referenced globally. Still, the model will adjust toward other directions along the horizontal plane. For example, when the goal velocity is set to [1.5, 0.0, -1.5], the model will shift toward the right, as shown in Figure 6.3. It is noteworthy that the model does not turn to align its pelvis toward the goal direction, but rather adjusts the width of its footsteps such that the model "sidesteps" toward the

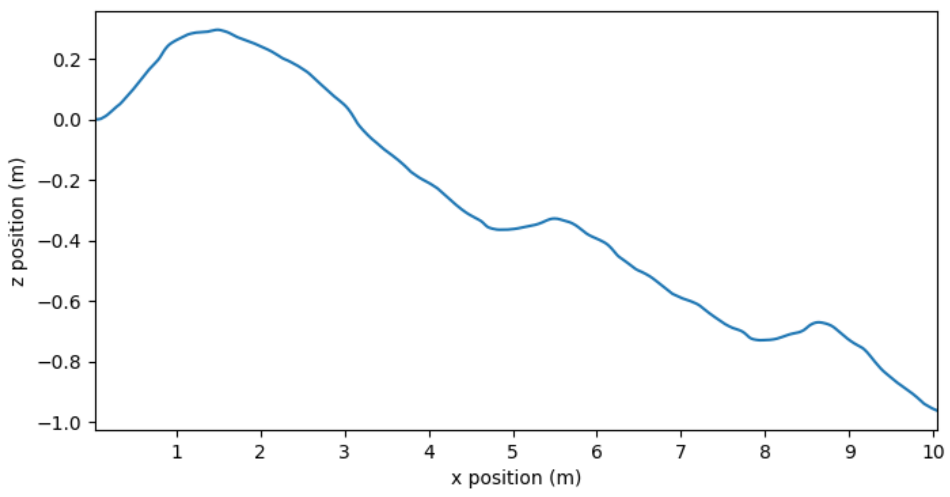


Figure 6.3: Horizontal position of the biomechanical model over an episode with a goal velocity of [1.5, 0.0, -1.5].

Goal speed	Achieved speed (mean)	Distance from goal speed
3D Baseline (no task)	1.410	-
0.5	1.332	0.832
1.0	1.377	0.377
1.5	1.476	0.024
2.5	1.519	0.981

Table 6.1: Resulting speeds of models trained to achieve a specific speed in the forward direction, averaged over a testing episode.

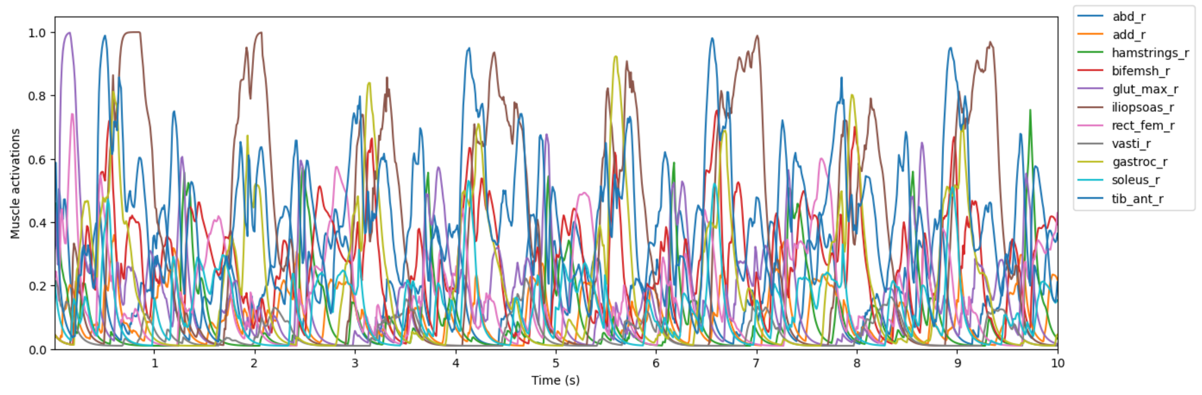
correct direction; in order to move to the right, the left footstep are narrower and the right footsteps are wider. This is demonstrated by the muscle activation in Figure 6.4a and Figure 6.4b, in which the right abductors contract more than the left abductors, and vice versa for the adductors. Figure 6.4c also reveals that the right footsteps take longer, which confirms these observations.

At more extreme angles, the model ultimately ignores the goal direction in favor of increasing the mimic reward.

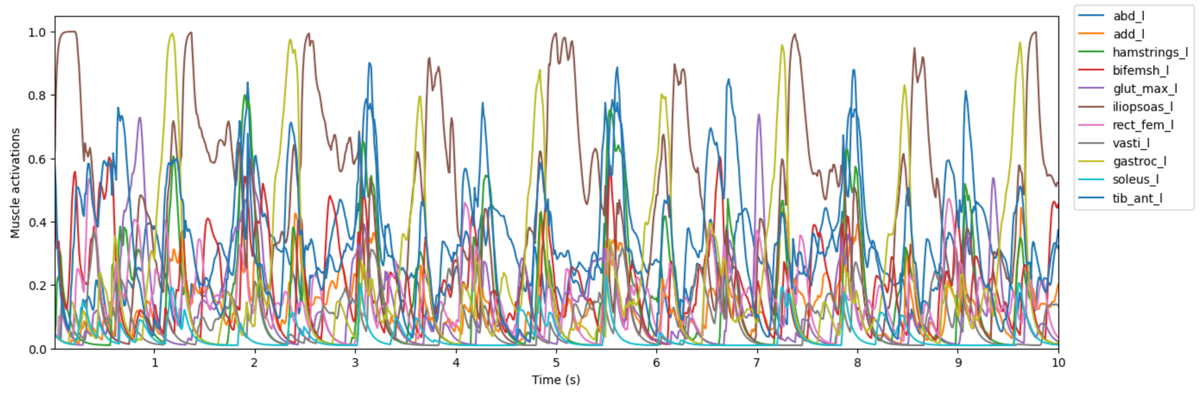
6.4.2 Speed Changes

Our model is more resistant to changes in speed than changes in direction. Table 6.1 reports the achieved average speeds of locomotion compared to the goal speed. While the

(a)



(b)



(c)

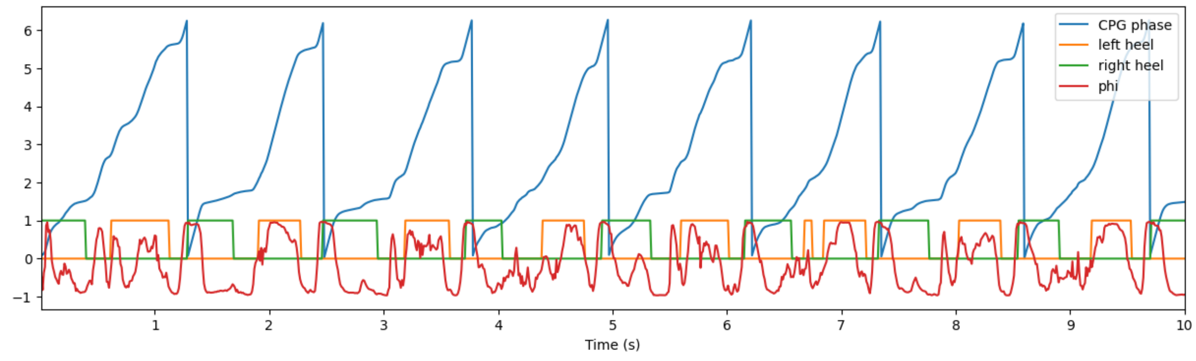


Figure 6.4: The muscle activations of the (a) right side, (b) left side, and (c) CPG phase and footsteps of the biomechanical model with a goal velocity of [1.5, 0.0, -1.5].

speeds are shifted slightly toward the desired trajectory, the changes are minimal. The further the goal speed is from the baseline speed, the less correct the final model will be.

CHAPTER 7

Discussion

7.1 Generalizability

Our locomotion controller depends on reference motion sequences during locomotion, which ultimately reduces its generalizability in terms of the range of motions it can produce. We saw some success in changing directions and speeds in Section 6.4; however, they are hardly as extensive as we believe a CPG-driven locomotion controller is capable of. The greatest impediment to our expected results is likely the dependence on a reference motion sequence, as it limits the types of behavior that will be well-rewarded during training despite our reward modifications. A few approaches and adjustments to the current fine-tuning methods that may improve performance require further experimentation.

First, an appropriate balance of rewards and penalties must be found. This could potentially be learned by another network for efficiency. In our experiments, rewarding the goal task too heavily or removing the mimic reward encouraged the model to utilize the CPG less and forgo a stepping pattern altogether. On the other hand, a low task reward meant the model would not vary too far from the learned trajectories of the reference data. The most straightforward way to amend the issue would then be to find better-balanced weights that encourage the continued use of the CPG while modifying it in an appropriate manner.

Another possible solution might be to restrict the model in some way that preserves the overall shape of the CPG signal. Perhaps this could be introduced as a penalty that compares the proposed trajectory with the original learned signal. This might allow our CPG tuner to learn more mappings for more extreme yet stable modulations of its parameters while observing the general activation patterns in the original gait. On the other hand, this may make the model less robust against environmental perturbations that would otherwise require unexpected changes in the CPG signal in order to maintain stability.

Lastly, we could consider a more brittle approach that involves adjusting the raw reference motion data such that it matches the desired task trajectory. This would require no adjustment to our current reward function or architecture. However, this is a rather undesirable option as it both adds overhead to the task-driven model and undercuts the value of the CPG as a robust source of locomotive control signals.

In exploring the generalizability of our model, we furthermore consider an alternative method of adjusting the gait that does not involve retraining or fine-tuning the model.

7.1.1 Zero-shot Gait Changes

Several publications on CPG modeling explore the ability to achieve varying velocities of a type of gait only by changing the high-level CPG parameters of frequency and amplitude. We explore how our model handles this approach by manually adjusting the CPG's parameters at test time. This exploration comes with the caveat that our reflex controller is fairly brittle in that it was trained to mimic one specific gait; in many of these experiments, it is not able to adjust to the potentially unseen states created by an altered CPG signal. Were it trained with a variety of CPGs or reference sequences, it may be able to better support the direct modulation of CPG parameters.

By manually adjusting the frequency, we would expect a change in speed. In the timesteps before the reflex controller fails and the model falls, we sometimes observe shorter footsteps in terms of the amount of time the heels remain in contact with the ground during a step. As shown in Figure 7.1, it does not consistently adjust the speed as expected; nevertheless, there seems to be somewhat of a correlation between the frequency and speed, particularly when comparing the maximum velocities. For example, multiplying the fundamental frequency by 1.5 slows the model down in terms of its average

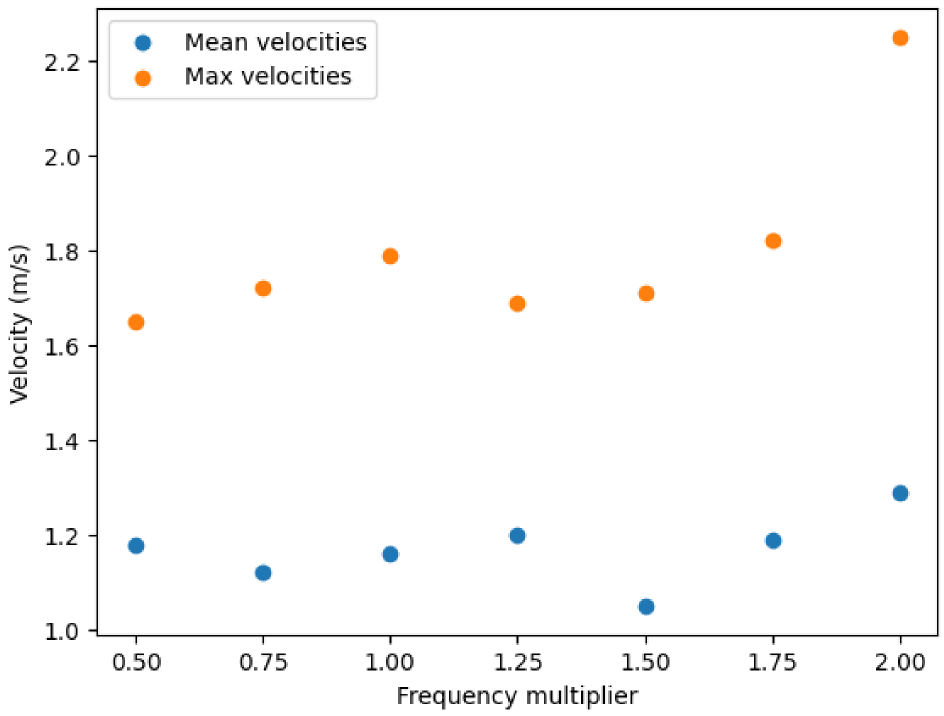


Figure 7.1: Achieved velocities of trained models tested with modulated frequencies. Both the mean velocity over the episode and the maximum velocity are reported.

Frequency multiplier	0.5	0.75	1.0	1.25	1.5	1.75	2.0
Episode length	4.0	3.7	10.0	4.0	7.0	9.1	4.7

Table 7.1: The episode lengths of models with modulated frequencies, which estimate the amount of time the model performed standard stepping motions before falling or otherwise failing.

speed, whereas most other multipliers greater than 1 will increase the speed. We also note that if the frequency is too low, the CPG tuner is not able to match the footstep cycle to the CPG; when using 0.5 as the multiplier, two cycles of footsteps occurred during one CPG cycle instead of just one. This may explain why higher frequency trials tended to last longer before falling down, as is reported in Table 7.1.

Amplitude modulation less directly affects gait style, as it determines the magnitude of the muscles' activations. Theoretically, this would only change the amount of energy exerted during the stepping and the overall stiffness of the muscles but would not have a particularly strong visual effect. Considering that our baseline model already applies

Amplitude multiplier	Mean velocity	Max velocity
0.5	1.18	1.65
0.75	1.12	1.72
1.0	1.16	1.79
1.25	1.20	1.69
1.5	1.05	1.71
1.75	1.19	1.82
2.0	1.26	2.25

Table 7.2: Achieved velocities of trained models tested with modulated amplitudes. Both the mean velocity over the episode and the maximum velocity are reported.

maximum activation to certain muscles during some phases of the step, increasing the amplitude throws the relative muscle activities out of balance and causes the model to fail almost immediately; thus, we only report results for reducing the amplitude during test time. The models in all such trials lasted for the entire ten second episode.

While amplitude modulation is intuitively less useful in designing gait trajectories, it does reveal which muscles the reflex controller employs the most to maintain stable locomotion. For example, although the activation signals from the CPG should all be less than 1 when the amplitude is manually scaled down, we observe that the hip flexor group consistently reaches maximum activation in all trials. We also notice in Table 7.2 that the model's velocity is impacted by amplitude variation, as the maximum velocity achieved tends to increase with the amplitude multiplier. This makes sense, as gaits with overall lower muscle activity would be unable to achieve higher speeds.

As we only learn one signal for the lower body activations, we are unable to easily adjust each side independently; this behavior might result in turning, but our proposed model does not offer such flexibility.

7.2 Usefulness of the CPG

In several of our experiments, we have observed that the model learns a bias toward the reflex controller's output rather than effectively tuning and utilizing the CPG. Naturally, this leads to a question of the usefulness of the CPG in the first place, given that many have achieved successful biomechanical locomotion without it. We argue that despite some obstacles, CPGs still offer many benefits in biomechanical locomotion. First, in many of these cases, we cannot be sure if the undesirable results are due to a fundamental unsuitability of employing CPGs, or simply a problem of getting trapped in local optima during learning. When we encounter such an issue, we are generally able to reformulate our rewards and architecture to avoid them. Second, stable, extendable, and adaptable locomotion would be much more difficult to achieve just from a single reference sequence; we see this even in our own results in Section 6.4 and Section 7.1.1. Still, we are able to facilitate small modifications to the reference gait, which would likely be more difficult without a CPG. CPGs allow for an analysis of fundamental stepping patterns that enables the model to extrapolate beyond the original reference data. Lastly, the main motivation behind our approach is to attain physiological realism. Other DRL policies may outperform our model in terms of successful persistent locomotion, but might not necessarily produce the realistic gaits without excessive motion data to mimic.

CHAPTER 8

Conclusion

8.1 Contributions

In this work, we proposed a dual-module deep reinforcement learning architecture that employs Central Pattern Generator oscillator systems and a reflex-based controller to produce locomotion in a biomechanical human model. We train a CPG using an existing oscillatory signal extracted from a reference synthesized motion sequence. Unlike some related efforts in the literature, the CPG is the backbone of our locomotive control and produces the muscular activation signals used to drive the model. During the online Soft Actor-Critic learning loop, a CPG tuning module adjusts high-level CPG parameters to adapt to the environment. The CPG output signals are then supported by a reflex control module that prevents the model from losing balance. Each network is trained in combination to learn efficient generation of stable bipedal locomotion that is able to walk forward for the entirety of our test episodes.

While our model cannot boast the generalizability and controllability of some CPG-based systems, it shows limited but promising results toward task-driven locomotion and may show greater success after further experimentation and development. Nevertheless, bipedal locomotion is a complex problem, and our efforts have yielded solid results in learning one type of gait using our architecture, which can surely be scaled to increase the range of its output.

8.2 Future Directions

This research has provided a proof of concept toward the feasibility of CPG-based bipedal locomotion. Many improvements can be made to our current method and results, including computational optimization and the increased complexity and biological accuracy of the model. In regard to the CPG system, we could train multiple CPG systems for different lower-body muscle groups, including the separation of the left and right sides. This might allow us to achieve a more diverse set of gaits with easier user control. Using models with more articulation, especially in the upper body, would be another natural next step to enhance the realism of our model. We might also consider training different CPGs for other bipedal gaits, such as jogging or running, which likely vary too greatly from walking to reuse the CPG signals learned in our work.

As discussed in Section 7.1, task-driven locomotion via either fine-tuning or zero-shot CPG adjustments would be a promising route of further investigation, as it would bolster the robustness of the model and increase its generalizability, both of which are properties of the CPG that should be thoroughly exploited.

APPENDIX A

SAC Learning Algorithm

As described by Haarnoja et al. (2018), SAC explores a continuous action space as a Markov decision process $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the continuous state space and $\mathcal{A}$ is the action space. The probability of transitioning from a state $\mathbf{s}_t \in \mathcal{S}$ to state $\mathbf{s}_{t+1} \in \mathcal{S}$ via action $\mathbf{a}_t \in \mathcal{A}$ is $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$. The reward $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is determined per transition to state $\mathbf{s}$ using action $\mathbf{a}$.

Stochastic policies are favored by augmenting the objective function, which is usually the expected sum of rewards, with the expected entropy $\mathcal{H}$ of the policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ over $\rho_\pi(\mathbf{s}_t)$, the state marginal:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t)\sim\rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right], \tag{A.1}$$

where α is a temperature parameter that control the overall stochasticity of the optimal policy.

Rather than explicitly performing alternating evaluations and policy improvements, SAC alternates between optimizing the Q-function, the learned mapping of expected rewards given an action in a state and the policy, using stochastic gradient descent; these are approximated by functions parameterizing the state value function $V_\psi(\mathbf{s}_t)$, a soft Q-function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$, and an tractable policy $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$.

The state value function, approximating the soft value, is optimized to minimize the squared residual error

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}} \left[\frac{1}{2} \left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t\ \mathbf{s}_t)] \right)^2 \right] \tag{A.2}$$

with the estimated gradient

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t)(V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)), \tag{A.3}$$

where $\mathcal{D}$ is the distribution of previously sampled states and actions. Note that a function approximator for the state value is not strictly required, but its inclusion can stabilize training.

The soft Q-function is trained to minimize the soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right] \tag{A.4}$$

with the gradient

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t)(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V_{\bar{\psi}}(\mathbf{s}_{t+1})), \tag{A.5}$$

where

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\psi}}(\mathbf{s}_{t+1})\right] \tag{A.6}$$

and $V_{\bar{\psi}}$ is a target value network such that $\bar{\psi}$ is an exponentially moving average of value network weights. This network also stabilizes training.

Lastly, the policy parameters are learned through minimizing the Kullback-Leibler divergence as follows:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\cdot|\mathbf{s}_t)\middle\|\frac{\exp(Q_\theta(\mathbf{s}_t, \cdot))}{Z_\theta(\mathbf{s}_t)}\right)\right]. \tag{A.7}$$

Since the Q-function is the target density, Equation A.7 is reparameterized using a neural network transformation

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t), \tag{A.8}$$

where ϵ_t is a noise vector sampled from some fixed distribution $\mathcal{N}$. Furthermore, the

Algorithm 1: Soft Actor-Critic

```
1  Initialize parameter vectors $\psi, \bar{\psi}, \theta, \phi$
2  for each iteration do
3      for each environment step do
4          $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
5          $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
6          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}$
7      end
8      for each gradient step do
9          $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
10         $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
11         $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
12         $\bar{\psi} \leftarrow \tau\bar{\psi} + (1-\tau)\bar{\psi}$
13     end
14 end
```

partition function Z_θ is independent of ϕ and can thus be omitted. We then have that

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t)|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t)) \right], \tag{A.9}$$

with the approximated gradient

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) + (\nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)) \nabla_\phi f_\phi(\epsilon; \mathbf{s}_t). \tag{A.10}$$

Here, π_ϕ is defined implicitly in terms of f_ϕ.

Algorithm 1 is the final SAC optimization algorithm, in which we alternate between sampling the environment and performing stochastic gradient descent. Note that it employs two Q-functions with the parameters θ_i for $i \in \{1, 2\}$ that are trained independently, the minimum of which is used for the state value and policy gradients. This mitigates positive bias during policy improvement and speeds up training (Hasselt, 2010; Fujimoto et al., 2018).

APPENDIX B

Biomechanical Model Details

The OpenSim model we use, provided by Kidziński et al. (2018) as described in Chapter 5, includes 11 symmetric lower body muscles, which are detailed in Table B.1 with their corresponding functions. Note that some are not individual muscles, but rather muscle groups that are modeled as a singular muscle with the same functionality.

Muscle Name	Variable	Primary Functions
Hip abductors	HAB	Hip abduction (away from the vertical midline)
Hip adductors	HAD	Hip adduction (toward the vertical midline)
Short head of the biceps femoris	BFSH	Knee flexion
Gastrocnemius	GAS	Knee flexion and ankle extension (plantarflexion)
Gluteus maximus	GLU	Hip extension
Biarticular hamstrings	HAM	Hip extension and knee flexion
Hip flexors	HFL	Hip flexion
Rectus Femoris	RF	Hip flexion and knee extension
Soleus	SO	Ankle extension (plantarflexion)
Tibialis anterior	TA	Ankle flexion (dorsiflexion)
Vasti	VAS	Knee extension

Table B.1: Muscles in the biomechanical OpenSim model with their functions and variable names as they appear in plots.

APPENDIX C

Hill-Type Muscle Actuators

The prevailing model for biomechanical muscle simulation is the Hill-type actuator model, first introduced by Hill (1938), which models muscle as a uniaxial contractile actuator. The total force enacted by a skeletal muscle attached to bones via tendons is the sum of an active contractile force due to muscle activation and a passive nonlinear elastic restoring force. This can be modeled as an contractile element (CE) in line with a series element (SE), combined with a parallel element (PE). The former pair reflect the active muscle contraction and passive deformation, respectively, while the latter represents the passive tendon deformation.

Thelen (2003) provides the following implementation of the Hill-type musculotendinous actuator that is used in our OpenSim model. To calculate the length of the muscle, we solve for and integrate the muscle fiber velocity

$$\dot{l}_{\mathrm{CE}} = \big(0.25 + 0.75\, a(t)\big) V_{\max} \frac{f_{\mathrm{CE}} - a(t) f_{\mathrm{AL}}}{b}, \tag{C.1}$$

where $a(t)$ is the muscle activation, $V_{\max}$ is the maximum contraction velocity, and f_{AL} is the active force length curve, which is chosen to be a Gaussian function. The active muscle force

$$f_{\mathrm{CE}} = a(t) f_{\mathrm{AL}}(l_{\mathrm{CE}}) f_V(\dot{l}_{\mathrm{CE}}), \tag{C.2}$$

is that produced by the contractile element, where l_{CE} and $\dot{l}_{\mathrm{CE}}$ are the length and velocity of the muscle fibers. Denominator b in Equation C.1 is a piecewise continuous parameter that differs from the muscle shortening ($f_{\mathrm{CE}} \leq a(t) f_{\mathrm{AL}}$) to the muscle lengthening

$(f_{\mathrm{CE}} > a(t) f_{\mathrm{AL}})$ states:

$$b = \begin{cases} a(t) f_{\mathrm{AL}} + f_{\mathrm{CE}}/A_f & f_{\mathrm{CE}} \leq a(t) f_{\mathrm{AL}}; \\ \dfrac{(2 + 2/A_f)\big(a(t) f_{\mathrm{AL}} F_{\mathrm{len}} - f_{\mathrm{CE}}\big)}{F_{\mathrm{len}} - 1} & f_{\mathrm{CE}} > a(t) f_{\mathrm{AL}}, \end{cases} \tag{C.3}$$

where F_{len} describes the maximum normalized muscle force that can be generated when the muscle is lengthening. The OpenSim implementation further adds linear extrapolation in the cases of concentric $(f_{\mathrm{CE}} < 0)$ and eccentric $(f_{\mathrm{CE}} > 0.95 F_{len})$ contractions to ensure that the force velocity curve is invertible.

The force velocity function f_V in Equation C.2 describes the shape of the force velocity curve, which is calculated using the standard muscle equilibrium equation

$$\cos\phi\big(a(t) f_{\mathrm{AL}}(l_{\mathrm{CE}}) f_V(\dot{l}_{\mathrm{CE}}) - f_{\mathrm{SE}}(l_{\mathrm{CE}})\big) - f_{\mathrm{PE}}(l_T) = 0, \tag{C.4}$$

where l_T denotes the tendon length, f_{SE} and f_{PE} are the forces of the series and parallel elements, respectively, and ϕ is the pennation angle, describing the angle between the tendon, which is representative of the overall direction of the muscle, and the muscle fibers. Solving for f_V yields

$$f_V(\dot{l}_{\mathrm{CE}}) = \frac{\dfrac{f_{\mathrm{SE}}(l_T)}{\cos\phi} - f_{\mathrm{PE}}(l_{\mathrm{CE}})}{a(t) f_{\mathrm{AL}}(l_{\mathrm{CE}})}. \tag{C.5}$$

REFERENCES

Bellegarda, G. and Ijspeert, A. (2022). CPG-RL: Learning central pattern generators for quadruped locomotion. *IEEE Robotics and Automation Letters*, 7(4):12547–12554. 7

Campanaro, L., Gangapurwala, S., De Martini, D., Merkt, W., and Havoutis, I. (2021). CPG-Actor: Reinforcement learning for central pattern generators. In Fox, C., Gao, J., Ghalamzan Esfahani, A., Saaj, M., Hanheide, M., and Parsons, S., editors, *Towards Autonomous Robotic Systems*, pages 25–35, Cham. Springer International Publishing. 7

Delp, S., Anderson, F., Arnold, A., Loan, P., Habib, A., John, C., Guendelman, E., and Thelen, D. (2007). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54:1940 – 1950. 16

Delp, S., Loan, J., Hoy, M., Zajac, F., Topp, E., and Rosen, J. (1990). An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering*, 37(8):757–767. 4

DiLorenzo, P., Zordan, V., and Sanders, B. (2008). Laughing out loud: Control for modeling anatomically inspired laughter using audio. *ACM Trans. Graph.*, 27:125. 4

Dong, F., Clapworthy, G., Krokos, M., and Yao, J. (2002). An anatomy-based approach to human muscle modeling and deformation. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):154–170. 4

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001). Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 251–260, New York, NY, USA. Association for Computing Machinery. 5

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. 37

Gams, A., Ijspeert, A. J., Schaal, S., and Lenarčič, J. (2009). On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 27(1):3–23. 6, 8, 9

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290. 12, 35

Hasselt, H. (2010). Double Q-learning. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc. 37

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle. *Proceedings of The Royal Society B: Biological Sciences*, 126:136–195. 39

Ijspeert, A., Crespi, A., Ryczko, D., and Cabelguen, j.-m. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315:1416–20. 6

Ijspeert, A. J. (2001). A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84(5):331–348. 6

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653. Robotics and Neuroscience. 5

Kidziński, Ł., Mohanty, S. P., Ong, C. F., Hicks, J. L., Carroll, S. F., Levine, S., Salathé, M., and Delp, S. L. (2018). Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In Escalera, S. and Weimer, M., editors, *The NIPS '17 Competition: Building Intelligent Systems*, pages 101–120, Cham. Springer International Publishing. 16, 17, 38

Kähler, K., Haber, J., Yamauchi, H., Seidel, H.-P., and Spencer, S. (2002). Head shop: Generating animated head models with anatomical structure. *Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation, ACM SIGGRAPH, 55-64 (2002).* 4

Lee, S., Park, M., Lee, K., and Lee, J. (2019). Scalable muscle-actuated human simulation and control. *ACM Trans. Graph.*, 38(4). 6

Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2009). Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.*, 28(4). 4

Lee, S.-H. and Terzopoulos, D. (2006). Heads up! biomechanics modeling and neuromuscular control of the neck. *ACM Trans. Graph.*, 25:1188–1198. 4

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 55–62, New York, NY, USA. Association for Computing Machinery. 4

Minassian, K., Hofstoetter, U. S., Dzeladini, F., Guertin, P. A., and Ijspeert, A. (2017). The human central pattern generator for locomotion: Does it exist and contribute to walking? *Neuroscientist*, 23(6):649–663. 5

Nakada, M., Zhou, T., Chen, H., Weiss, T., and Terzopoulos, D. (2018). Deep learning of biomimetic sensorimotor control for biomechanical human animation. *ACM Trans. Graph.*, 37(4). 5

Nakamura, Y., Yamane, K., Fujita, Y., and Suzuki, I. (2005). Somatosensory computation for man-machine interface from motion-capture data and musculoskeletal human model. *IEEE Transactions on Robotics*, 21(1):58–66. 4

Nierop, O., Helm, A., Overbeeke, K., and Djajadiningrat, T. (2008). A natural human hand model. *The Visual Computer*, 24:31–44. 4

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018). DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4). 7

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8. 16

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2015). Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph.*, 34(1). 4, 6, 8, 9

Sifakis, E., Neverov, I., and Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.*, 24(3):417–425. 4

Söderström, T. and Ljung, L. (1983). *Theory and Practice of Recursive Identification*. The MIT Press. 10

Song, S. and Geyer, H. (2015). A neural circuitry that emphasizes spinal feedback generates diverse behaviours of human locomotion. *The Journal of Physiology*, 593(16):3493–3511. 5, 13

Sueda, S., Kaufman, A., and Pai, D. K. (2008). Musculotendon simulation for hand animation. *ACM Trans. Graph.*, 27(3):1–8. 4

Thelen, D. G. (2003). Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of Biomechanical Engineering*, 125(1):70–77. 16, 39

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. (2023). Gymnasium. 16

Tsang, W., Singh, K., and Fiume, E. (2005). Helping hand: An anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, page 319–328, New York, NY, USA. Association for Computing Machinery. 4

Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 1079–1086. Morgan Kaufmann. 9

Wang, J. M., Hamner, S. R., Delp, S. L., and Koltun, V. (2012). Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph.*, 31(4). 4

Zhou, T. (2019). *Core Training: Learning Deep Neuromuscular Control of the Torso for Anthropomimetic Animation.* PhD thesis, University of California, Los Angeles, USA. 4, 5

Zordan, V. B., Celly, B., Chiu, B., and DiLorenzo, P. C. (2004). Model and control of simulated respiration for animation. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, page 140, New York, NY, USA. Association for Computing Machinery. 4