university of groningen / faculty of science and engineering

Scalable Distributed Learning with PPG-IMPALA for Physics-based Musculoskeletal Models

Bachelor's Project Thesis

Andrei Voinea, s3754243, c.a.voinea@student.rug.nl
Supervisors: Prof. Dr. R. (Raffaella) Carloni

Abstract: The use of Deep Reinforcement Learning (DRL) algorithms has surged in previous years due to their ability to adapt to a variety of tasks. However, training these algorithms sometimes requires long periods of time, and the resource efficiency can be low. To this end, the current study evaluates the use of an off-policy DRL algorithm, which can efficiently use a large number of resources, during a bipedal motion control task. The proposed algorithm is used to generate a gait pattern during the simulation of a physics-based musculoskeletal model of a healthy subject. As the goal was to achieve normal speed level-ground walking, the training process was assisted using imitation data provided by a public dataset where participants performed such movements. Although the trained policy could not generate a stable gait pattern, the results indicate that the proposed architecture can increase the speed of training, when compared with an on-policy architecture.

Keywords— Deep Reinforcement Learning, Bipedal Motion Control, Scalable Machine Learning

1 Introduction

Human motion is influenced by a variety of elements, such as injury, bone architecture, muscle conditions, habits, and fatigue. Altering one such factor can lead to large differences in the overall motion, which creates a unique movement pattern for an individual. To achieve this, numerous muscles actuate the bones and joints found in the human body through contraction and relaxation, triggered by signals received from the central nervous system (Lee et al., 2019). However, the neural circuitry involved in generating the signals required to achieve locomotion behaviour is not clear, making it difficult to correctly simulate the kinematic control of walking (Song and Geyer, 2015).

To better understand the biomechanics of gait patterns and to achieve motion control, previous studies have attempted to use computer simulations. As described by Park et al. (2022), the design of such controllers ranged from using feedback control laws, data-driven control, non-linear/stochastic optimization, and reinforcement learning. Although the latter form of control was more erratic compared to the others, advances in the field of Deep Reinforcement Learning (DRL) have made its use more attractive for bipedal control. More specifically, combining the use of DRL controllers with imitation data improved the quality of learned behaviours, while also reducing the training time required (Peng et al., 2018).

De Vree and Carloni (2021) successfully applied DRL methods alongside imitation data to study the control of healthy and impaired physics-based musculoskeletal models. The results of this work showed that policy optimization methods are successful in obtaining a gait pattern when prosthetic attachments are used, which increase the complexity of a model. Although the optimization method used by De Vree and Carloni provided good results for the given task, the learning curve of the DRL models indicate that Proximal Policy Optimization (PPO) tends not to use samples efficiently for complex bipedal motion control. Similarly, the

solutions proposed by Kidziński et al. (2019) for a more simplistic musculoskeletal model showed similar behaviour in the later stages of training, when using DRL methods. More specifically, the exploration variation is limited in the initial phases of the algorithm, and appears to increase irregularly near the end of the training period.

Previous work attempted to improve upon this point by changing how the policy variance is updated. As such, Proximal Policy Optimization with Covariance Matrix Adaptation (PPO-CMA) uses an evolutionary optimization method to increase the robustness of the DRL model during training (Hämäläinen et al., 2020). Although the results of the study suggested that PPO-CMA slightly improves the efficiency of training in a bipedal motion control task, the algorithm still behaves in an on-policy manner. This can lead to resource efficiency issues when trying to scale up a parallelized version of PPO-CMA.

On-policy algorithms learn by using actions taken from the current policy, while off-policy algorithms can learn by using any set of actions. PPO and PPO-CMA are on-policy algorithms that can run multiple environments in parallel, but have to wait for a set of episodes to finish before performing a step of gradient descent for the Multilayer Perceptron (MLP). This approach can be favourable, as updates are more stable, and the policy can parse a larger set of actions at once during an update. However, the training process is as slow as the slowest parallelized worker, which wastes valuable resources that can be used for further exploration.

On the other hand, off-policy algorithms can perform a policy update with any actions that are available, without requiring synchronization between workers. However, this method can lead to numerical instabilities during training, as the actions used for training can be very different from the current policy. Strategies similar to the clipping of PPO can be used to minimize this issue, allowing gradual updates to a policy over time. Therefore, when compared with on-policy methods, off-policy algorithms can potentially manage resources more effectively, allowing for better scalability at a very low performance cost.

Therefore, in this study, the use of an off-policy method is evaluated to allow many workers to perform rollouts in a resource demanding task. The proposed DRL algorithm is based on Phasic Policy Gradient (PPG; Cobbe et al., 2020), a variant of PPO, and Importance Weighted Actor-Learner Architectures (IMPALA; Espeholt et al., 2018), both of which are described in Section 2. By combining these two architectures, an off-policy algorithm is created, PPG-IMPALA, which incorporates the benefits of policy optimization methods and the scalability of IMPALA.

A brief summary of the algorithm can be observed in Figure 1.1, which shows that PPG-IMPALA works by having a pool of workers interact with an environment to generate trajectories. At the start, each worker loads the current policy from the local storage and begins exploring the environment. At each step, the environment returns an observation and a reward, which is aggregated by each agent for a number of steps. After a number of steps is reached, each worker submits its buffer of trajectories to the centralized learner, and then restarts the exploration process with the next available policy. After receiving the buffer of trajectories, the centralized learner performs a policy update similarly to PPG. Once the policy update is complete, the centralized learner saves the new set of policy parameters to local storage, and then waits for another buffer of trajectories.

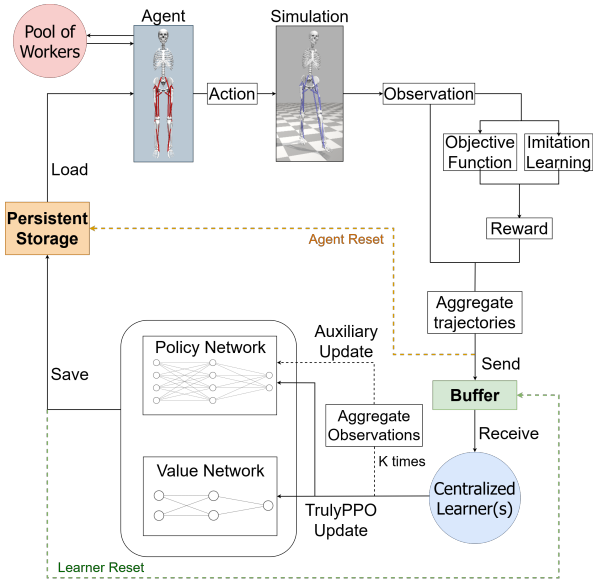

Figure 1.1: The proposed DRL algorithm for optimizing a policy trained to perform bipedal motion control on a normal, level-ground setting.

The remainder of the paper will be dedicated to elaborating the use of PPG-IMPALA alongside Imitation Learning (PPG-IMPALA + IL) to achieve a healthy gait pattern in a bipedal motion control task. Section 2 describes the theoretical background on which PPG-IMPALA is developed, as well as the software used to perform the musculoskeletal simulations. Section 3 elaborates on the design choices regarding the simulation, as well as providing specific details on the implementation of PPG-IMPALA+IL. Finally, Section 4 shows and discusses the results obtained on the bipedal motion control task, followed by concluding remarks in Section 5.

2 Theoretical Background

As described earlier, PPG-IMPALA is an off-policy, actor-critic optimization algorithm that is capable of scaling to a large number of machines without sacrificing resource utilization. To better understand the proposed architecture, this section will begin by explaining the principles behind PPO and how it was improved by the TrulyPPO variation. This information is then used to describe how PPG works, followed by an explanation of IMPALA and its proposed V-trace, an algorithm used to calculate advantages. Finally, the use of OpenSim is justified, and the mechanism behind Imitation Learning is detailed.

2.1 Proximal Policy Optimization

In Reinforcement Learning, the objective of an agent is to maximize a given reward while following a parametrized policy π_θ. To achieve this, policy gradient methods estimate a policy gradient in the form of $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right]$, and update the policy π_θ using a gradient ascent algorithm. In this case, the expectation $\hat{\mathbb{E}}_t$ represents an empirical average over a set of samples at time t, a_t and s_t are the action and state obtained at that time step, and $\hat{A}_t$ is an estimator of the advantage function (Schulman et al., 2017).

To facilitate this process while using automatic differentiation software, Schulman et al. use an objective function in the form of $L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t)\hat{A}_t\right]$, which can be used to estimate $\hat{g}$. However, since optimizing using this objective is unstable due to potential large differences between the old and new policy, PPO introduces a new objective function based on clipping (see Equation 2.1). This objective incorporates the likelihood ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, which denotes the difference between the current and the previous policy, and can be used to restrict a large policy update. Therefore, whenever the likelihood ratio surpasses the clip range, the gradient of $L^{CLIP}(\theta)$ with regard to θ will be 0, thus restricting the policy update.

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

2.2 Truly Proximal Policy Optimization

Although PPO improves training by restricting destructive policy updates, Wang et al. (2020) argue that it cannot strictly restrict the likelihood ratio, nor enforce a trust-region constraint. Firstly, the failure to restrict the likelihood ratio is due to the ill-defined clipping method, which could push the ratios even further outside the clipping range in special cases. Secondly, PPO does not explicitly attempt to enforce a trust region constraint, but the scale of the clipping mechanism indirectly affects the scale of the Kullback-Leibler (KL) divergence[1]. Since the likelihood ratio $r_t(\theta)$ is not bounded, its corresponding KL divergence cannot be bounded either.

Therefore, to address these issues, Wang et al. propose the use of a trust-region based clipping mechanism, which uses a rollback operation to lower the ratio once it exceeds the trust-region range. A new objective function is proposed in Equation 2.2 to incorporate these changes, where α is a parameter that controls the force of the rollback, δ decides the range of the trust-region, and $D_{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta)$ represents the KL divergence between the old and the new policies. More precisely, this new objective discourages the gradient from

[1] KL divergence is a type of statistical distance function, which measures how one probability distribution differs from another.

making a change that exceeds the new trust region by penalizing it using KL divergence.

$$\mathcal{F}_{KL} = \begin{cases} \alpha D_{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta) & D_{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta) \geq \delta \text{ and} \\ & r_t(\pi)\hat{A}_t \geq r_t(\pi_{\theta_{\text{old}}})\hat{A}_t \\ \delta & \text{otherwise} \end{cases}$$

$$L^{\text{truly}}(\theta) = r_t(\theta)\hat{A}_t - \mathcal{F}_{KL} \tag{2.2}$$

2.3 Phasic Policy Gradient

Architectures similar to PPO found that sharing parameters between the value and policy MLPs can be used interchangeably to optimize both objectives. However, Cobbe et al. (2020) suggest that this can be detrimental for learning, since the objectives of the policy and value function can interfere in more complex scenarios. Furthermore, sharing parameters also requires fine-tuning of hyperparameters in order to balance the two objectives, which could restrict the efficiency during training.

To increase the robustness of policy gradient algorithms and to improve sample efficiency, Cobbe et al. propose separating the MLPs of the policy and value functions. More specifically, PPG trains a policy network that contains a policy head π_θ and a value head V_{θ_π}, and a value network with a value head V_{θ_V}. The training is performed in two phases, where the *policy phase* updates π_θ and V_{θ_V}, while the *auxiliary phase* updates V_{θ_π} (and indirectly π_θ).

To achieve this, PPG uses the same objective as PPO in the policy phase to update the policy network. The value head V_{θ_V} is also updated during this phase using the objective $L^{\text{value}} = \hat{\mathbb{E}}_t \left[\frac{1}{2}(V_{\theta_V}(s_t) - \hat{V}_t^{\text{targ}})^2\right]$, where $\hat{V}_t^{\text{targ}}$ are value function targets calculated using the Generalized Advantage Estimation algorithm (Schulman et al., 2018).

During the auxiliary phase, the policy network is optimized using the joint objective described in Equation 2.3, where $L^{\text{aux}} = \frac{1}{2}\hat{\mathbb{E}}_t \left[(V_{\theta_\pi}(s_t) - \hat{V}_t^{\text{targ}})^2\right]$ is similar to the previous value function loss. Furthermore, to restrict the update of the original policy, Cobbe et al. implement the hyperparameter β_{clone} to control the difference between the new and the old policy using KL divergence.

$$L^{\text{joint}} = L^{\text{aux}} + \beta_{\text{clone}}\hat{\mathbb{E}}_t \left[D_{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta)\right] \tag{2.3}$$

2.4 Importance Weighted Actor-Learner Architectures

As described in Section 1, one challenge in exploring with a large number of agents is scalability. IMPALA proposes a way to decouple exploration from training to improve communication efficiency between workers and learners, which is achieved by transmitting trajectories instead of gradients with respect to the parameters of the policy. However, since the policy used to generate these trajectories can be outdated by several updates compared to the latest one, Espeholt et al. (2018) propose the V-trace off-policy actor-critic algorithm to correct the difference between the two policies.

Given a trajectory $(s_t, a_t, R_t)_{t=b}^{t=b+n}$ generated by a policy μ, where R_t is the reward at time t, the V-trace algorithm calculates a value approximation at state s_b using Equation 2.4. In this case, γ is a discount factor that reduces the value of future states, and $\delta_t V = \rho_t(R_t + \gamma V(s_{t+1}) - V(s_t))$ is a temporal difference for V. Furthermore, Espeholt et al. use the truncated importance sampling weights $\rho_t = \min(\bar{\rho}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})$ and $c_i = \min(\bar{c}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})$, where ρ_t defines a fixed point of the update rule and c_i measures the impact of $\delta_t V$ on the update of the value function at a previous time b. Additionally, $\bar{c}$ and $\bar{\rho}$ are truncation levels which limit the difference between the two policies.

$$v_b = V(s_b) + \sum_{t=b}^{b+n-1} \gamma^{t-b} \left(\prod_{i=b}^{t-1} c_i\right) \delta_t V \tag{2.4}$$

Therefore, given a policy π_ω and a value function V_θ, IMPALA performs a step of policy update using a set of trajectories generated by a policy μ. At training time b, IMPALA performs gradient descent on the L2 loss to the target v_b, which updates the value function V_θ using Equation 2.5, and the policy π_ω using Equation 2.6.

$$(v_b - V_\theta(s_b))\Delta_\theta V_\theta(s_b) \tag{2.5}$$

$$\rho_b \Delta_\omega \log \pi_\omega(a_b|s_b)(R_s + \gamma v_{b+1} - V_\theta(s_b)) \tag{2.6}$$

2.5 OpenSim and Imitation Learning

OpenSim is an open-source software capable of simulating the dynamics of musculoskeletal model structures (Delp et al., 2007). De Vree and Carloni have previously used OpenSim to study the performance of policy gradient algorithms on bipedal motion control tasks, thus achieving a gait pattern for healthy and impaired musculoskeletal models. This was aided by applying Imitation Learning during the training process, which sped up learning by providing an example gait pattern.

Imitation Learning encourages a character to match a reference motion by incorporating kinematic information in the goal function. For example, a policy will receive a higher reward if the motion resulting at the current time step matches certain characteristics of the reference, such as joint orientation and velocities (Peng et al., 2018). At the start of training, the model is initialized to a pre-determined state of the training data and acts according to the policy until the simulation is terminated. For locomotion tasks, Peng et al. suggest that early termination can occur when the character's torso makes contact with the ground, or when certain body parts fall below a given height threshold. In the case of De Vree and Carloni, the termination condition was met when the pelvis height fell below 0.7 m in height.

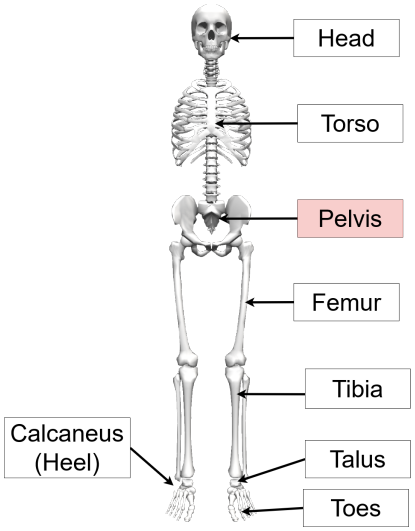

Figure 3.1: Overview of the significant body parts found in the observation space. For the lower extremity, only one of the two sides is indicated to improve clarity. The coloured box indicates that the coordinate observations related to that body part are in the global space.

3 Methods

Given the theoretical background, this section elaborates on the technicalities of PPG-IMPALA and of the bipedal motion task. In the following subsections, the environment used in the simulations is presented along with the characteristics of the musculoskeletal model. Following this, an in-depth description of the PPG-IMPALA algorithm is presented.

3.1 Environment

3.1.1 Musculoskeletal Model

OpenSim provides a realistic simulation of a given musculoskeletal model, which is built using rigid body segments that represent the bones. In the scope of this study, these body segments are connected by joints, which are actuated by Hill-type muscles (Thelen, 2003) realistically. The chosen musculoskeletal model is a reduced 3-dimensional gait model with 22 muscles, 14 Degrees of Freedom (DoF) and a weight of approximately 74.78 kg, based on the model proposed by Pandy and Anderson (2000).

For the purpose of this study, the observation space contains 104 values that represent the location of certain body parts of the musculoskeletal model. The global space, which is relative to the ground of the simulation, contains the pelvis position and orientation on the X, Y and Z axes. The local space, which is relative to the pelvis, contains the linear and angular velocities of the pelvis, the position, and velocity of the head, torso, and of the femur, tibia, talus, heel, and toes of each leg. Finally, a simplified view of the observation space can be seen in Figure 3.1.

Regarding the action space, each leg contains 11 muscles that are activated using values between 0 and 1. When provided with a value of 0, a muscle is completely relaxed, while a value of 1 will completely contract the muscle. The simulated Hill-

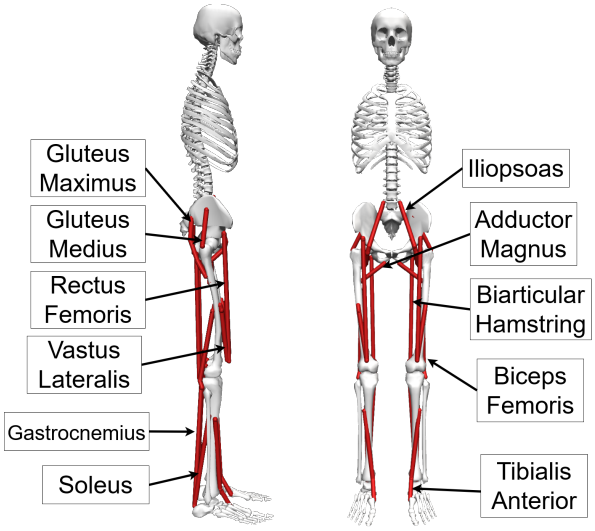

Figure 3.2: Overview of the 22 muscles used in the healthy musculoskeletal model. Only one of the two sides is indicated to improve clarity.

type muscles achieve a force by incorporating three factors: the length of the muscle, the velocity and the previously described activation level (De Vree and Carloni, 2021). As such, when a model receives a vector of muscle excitations, OpenSim calculates the changes in muscle positions and updates the current observation to match the new position of the model. A simplified view of the muscles used in the experiment can be seen in Figure 3.2.

3.1.2 Imitation Data

To implement the imitation reward function, data of a person walking normally on a level-ground surface is used. This was obtained from participant AB06 in the open-source motion capture dataset provided by Camargo et al. (2021), where participants with various body types had to perform a range of motion tasks. Participant AB06 is a 20 years old male with a height of 1.80 m and a mass of 74.8 kg, which closely matches the characteristics of the musculoskeletal model. This is supported by a low root-mean-square marker (RMS) error of approximately 0.024 ± 0.003 metres, which was obtained during the built-in OpenSim inverse kinematics conversion process. This process was used to convert the marker data of Camargo et al. to coordinate values that can be used by the musculoskeletal model. In this case, the marker error represents the distance between an experimental marker, which was placed alongside others on body parts of participants in the study, and the corresponding marker on the model. As such, the inverse kinematics process computes a pose that *best matches* the experimental markers at each frame of the movement, by attempting to minimize the RMS error of all markers. The resulting data represents a series of best match poses of a person walking in a circuit for approximately 20 seconds, at a pace of approximately 1.17 ± 0.21 m/s.

The participant data was further transformed such that the model is in the middle of a gait pattern at the origin, and that the model continues walking in the positive X direction. This was achieved by translating the original data by -575 mm on the X-axis and 1400 mm on the Z-axis, and rotating it by $-90°$ around the Y-axis. The resulting data places the model in the origin at time 14.2s of the original imitation data, which serves as a starting pose for the musculoskeletal model.

Finally, the imitation data was reconstructed as a comma-separated values (CSV) file, and all angles were converted to radians to allow the data to be used by the OpenSim module in Python. The resulting file contains the time of each pose frame, along with 17 kinematic measurements: pelvis angle and position in the X, Y and Z axes, joint angle and angular velocity of both hip flexion and adduction, and joint angle and angular velocity of the knees and ankles.

3.1.3 Reward

To facilitate the process of learning a healthy gait pattern, this study uses a shaped reward that incorporates the imitation data and information from the musculoskeletal body. As such, the environment provides a greater reward if the policy produces actions that make the musculoskeletal model follow the imitation data (joint positions and velocity), as well as keeping the pelvis above a certain height and moving forward at a desired pace.

This is achieved by first calculating a penalty for deviating from the desired behaviour, and converting it to a reward using a logarithmic function (see Equation 3.1). In this case, the reward increases inversely proportional to the penalty, where a indicates how much the reward should scale, b indicates

the threshold after which the reward is significantly reduced, and c represents the size of the transition period, after which the reward approaches 0.

$$R_{t_{\log}}(\rho_t, a, b, c) = \frac{c}{b}\log(1 + e^{\frac{a}{c}(b-\rho_t)}) \tag{3.1}$$

To reward the policy for following the imitation data, $\rho_{t_{\text{imi, pos}}}$ is obtained by summing all Mean Squared Errors (MSE) of the positions Θ and Θ_{imi}, for each component of the DoF of the model. Similarly, $\rho_{t_{\text{imi, vel}}}$ is obtained by summing all MSE of the velocities v and v_{imi} of all DoF components. As it can be observed in Equations 3.2 and 3.3, the imitation penalties are converted to a reward using the previously described logarithmic function, where the a, b and c parameters were chosen to promote imitating the positions of the imitation data. The velocity imitation reward $R_{t_{\text{imi, vel}}}$ has a smaller impact on the total reward, as it is desirable to allow the policy to adapt to different environmental conditions.

$$\begin{aligned} \rho_{t_{\text{imi, pos}}} &= \sum_{j \in \text{DoF}} (\Theta_{t_j} - \Theta_{t_{\text{imi},j}})^2 \\ R_{t_{\text{imi, pos}}} &= R_{\log}(\rho_{t_{\text{imi, pos}}}, 0.6, 3, 0.4) \end{aligned} \tag{3.2}$$

$$\begin{aligned} \rho_{t_{\text{imi, vel}}} &= \sum_{j \in \text{DoF}} (v_{t_j} - v_{t_{\text{imi},j}})^2 \\ R_{t_{\text{imi, vel}}} &= R_{\log}(\rho_{t_{\text{imi, vel}}}, 0.25, 10, 2) \end{aligned} \tag{3.3}$$

As a healthy gait pattern allows the body to move forward, another part of the total reward is represented by a positional penalty of the pelvis. As it can be observed in Equation 3.4, this is achieved by calculating the difference between the current and previous X-axis coordinate of the pelvis. To convert this penalty into a reward, the obtained value is multiplied by 10 to increase the influence of this metric. Since the value can be negative, $R_{t_{\text{pos}}}$ can penalize the policy for not moving forward.

$$\begin{aligned} \rho_{t_{\text{pos}}} &= (x_{t_{\text{pelvis}}} - x_{t-1_{\text{pelvis}}}) \\ R_{t_{\text{pos}}} &= 10 \cdot \rho_{t_{\text{pos}}} \end{aligned} \tag{3.4}$$

Since the previous penalty only incentivizes the policy to produce actions that move the pelvis forward, a secondary penalty is introduced to ensure that this movement is gradual. The policy can learn to jump forward to increase the received reward, which destabilizes a healthy gait. Therefore, $\rho_{t_{\text{vel}}}$ penalizes large changes of the pelvis X and Z coordinates by comparing the resulting velocity against a pace of 1.17 m/s. This value was chosen to match the pace of the imitation data.

$$\begin{aligned} \rho_{t_{\text{vel}}} &= |(v_{t_{\text{pelvis}}} - v_{t_{\text{imi, pelvis}}})^2 - 1.17| \\ R_{t_{\text{vel}}} &= R_{\log}(\rho_{t_{\text{vel}}}, 0.3, 0.15, 0.02) \end{aligned} \tag{3.5}$$

Furthermore, to discourage the policy from generating actions that lead to the musculoskeletal model falling, the penalty ρ_{t_h} increases when the Y coordinate of the pelvis falls below a height of 0.8 meters (see Equation 3.6).

$$\rho_{t_h} = \begin{cases} 1 & y_{t_{\text{pelvis}}} > 0.8 \\ 1 + (10y_{t_{\text{pelvis}}} - 8)^2 & \text{otherwise} \end{cases} \tag{3.6}$$

Finally, the previously described rewards and penalties are used to obtain a total reward, given by Equation 3.7. This sum of rewards ensures that the policy is rewarded for following the constraints described above, and penalizes it for falling. Furthermore, the simulation stops if the pelvis height $y_{t_{\text{pelvis}}}$ drops below 0.6 meters.

$$R_t = \frac{R_{t_{\text{imi, pos}}} + R_{t_{\text{imi, vel}}} + R_{t_{\text{pos}}} + R_{t_{\text{vel}}}}{\rho_{t_h}} \tag{3.7}$$

3.2 PPG-IMPALA with TrulyPPO

Given that the action space is continuous, the PPG-IMPALA algorithm uses an actor-critic setup to learn a policy π_θ, which represents the mean action taken when presented with state s_t, and two value functions, V_{θ_π} and V_{θ_V}. Following the methodology of PPG, the MLP used to represent the policy shares parameters between π_θ and V_{θ_π}, while the value function V_{θ_V} uses a different MLP (see Figure 3.3). Since the action space of the musculoskeletal model is between 0 and 1, the policy head π_θ uses a Sigmoid activation at the end to constrain the result in this range. Additionally, the logarithmic standard deviation of each action is learned during training as a set of 22 parameters, which are initialized to a desired value σ_{init}.

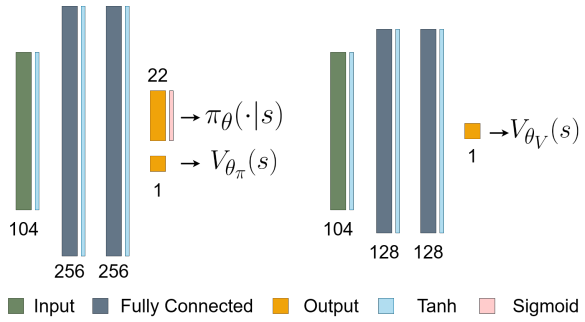

Figure 3.3: The two MLPs used to learn the policy and the value functions.

Multiple actors are initialized at the beginning of the simulation, which collect sets of trajectories that are sent to a centralized learner. While the workers continue exploring, the learner uses these trajectories to perform training similarly to PPG, in two alternating phases. During the policy phase, the objective of TrulyPPO is optimized alongside an entropy bonus. As it can be observed in Equation 3.8, this is achieved by adding the entropy of the current policy π_θ to the current objective, which is controlled by the entropy coefficient β_S. Following this, the value function network is updated according to PPG, using the L^{value} objective that returns the mean square error between the current value $V_{\theta_V}(s_t)$ and the value targets $\hat{V}_t^{\text{targ}}$. A value function coefficient β_{VF} and a value clip range ϵ_V are further applied to better control the importance of the objective L^{value}, as seen in Equation 3.9. In this case, $\Delta_\epsilon V_{\text{old}} = V_{\theta_{\text{old}}V}(s_t) + \text{clip}(V_{\theta_V}(s_t) - V_{\theta_{\text{old}}V}(s_t), -\epsilon_V, \epsilon_V)$ represents a clipped difference between the old value function and the new one.

$$L^{\text{truly}+S}(\theta) = \hat{\mathbb{E}}_t \left[L^{\text{truly}} + \beta_S S[\pi_\theta] \right] \tag{3.8}$$

$$L^{\text{VF}} = \beta_{VF} \hat{\mathbb{E}}_t \left[\frac{1}{2} \max \left((V_{\theta_V}(s_t) - \hat{V}_t^{\text{targ}})^2, \Delta_\epsilon V_{\text{old}} \right) \right] \tag{3.9}$$

Given a worker policy π_{θ_w}, the V-trace targets are obtained by first calculating $\delta_t V$ using $\rho_t = \min(\bar{\rho}, \frac{\pi_{\theta_w}(a_t|s_t)}{\pi_\theta(a_t|s_t)})$. Following this, the advantages are recursively calculated using the method proposed by Espeholt et al., as seen in Equation 3.10, where $c_i = \lambda \min(\bar{c}, \frac{\pi_{\theta_w}(a_t|s_t)}{\pi_\theta(a_t|s_t)})$ contains an additional discount parameter.

$$v_b = V(s_b) + \delta_b V + \gamma c_b \left(v_{b+1} - V(s_{b+1}) \right) \tag{3.10}$$

After finishing the policy phase, the learner stores the states s_t and the V-trace targets $\hat{V}_t^{\text{targ}}$ (which are equal to v_b) for a number of epochs. During the auxiliary phase, the joint objective L^{joint}, seen in Equation 2.3, is used to update the policy network for a number of iterations. Finally, a summarized version of the algorithm is presented in Algorithms 3.1 and 3.2, and a list of hyperparameters used during the experiments is shown in Table 3.1

Algorithm 3.1 PPG-IMPALA Worker

Load policy π_θ from disk
Initialize empty trajectory buffer T
for step $= 1, 2, ..., N_S$ **do**
  Perform rollouts under current policy π_θ
  Append set $\langle s_t, a_t, R_t, s_{t+1} \rangle$ to T
Send trajectory buffer T to Learner

Algorithm 3.2 PPG-IMPALA Learner

Initialize empty buffer B
for epoch $= 1, 2, ...$ **do**
  Receive set of trajectories T from Worker
  for iteration $= 1, 2, ..., N_\pi$ **do**
    Perform rollouts under current policy π_θ
    Compute V-trace targets $\hat{V}_t^{\text{targ}}$ for V_{θ_V} and advantages $\hat{A}_t$, given each state s_t
    for iteration $= 1, 2, ..., E_\pi$ **do**
      Optimize $L^{\text{truly}+S}(\theta)$ w.r.t. θ_π
    for iteration $= 1, 2, ..., E_V$ **do**
      Optimize $L^{VF}(\theta)$ w.r.t. θ_V
    Append $\left\langle s_t, \hat{V}_t^{\text{targ}} \right\rangle$ to buffer B
  if epoch mod $N_{\text{aux}} == 0$ **then**
    for iteration $= 1, 2, ..., E_{\text{aux}}$ **do**
      Optimize $L^{\text{joint}}(\theta)$ w.r.t. θ_π, given B
    Empty buffer B
  Store new policy π_θ on disk

Name	Symbol	Value
Number of steps	N_S	1024
Number of workers	-	50
TrulyPPO epochs	N_π	4
TrulyPPO batch size	-	256
TrulyPPO KL range	δ	0.05
TrulyPPO rollback force	α	5
Initial log stdev	σ_{init}	-0.5
Value clip range	ϵ_V	0.2
Value function coefficient	β_{VF}	1.0
Entropy coefficient	β_S	0.0
Auxiliary cycle size	N_{aux}	32
Auxiliary epochs	E_{aux}	4
Auxiliary batch size	-	256
Clone factor	β_{clone}	1.0
$\delta_t V$ discount factor	λ	0.999
Advantage discount factor	γ	0.9
Policy learning rate	-	3e-5

Table 3.1: A list of hyperparameters used during training.

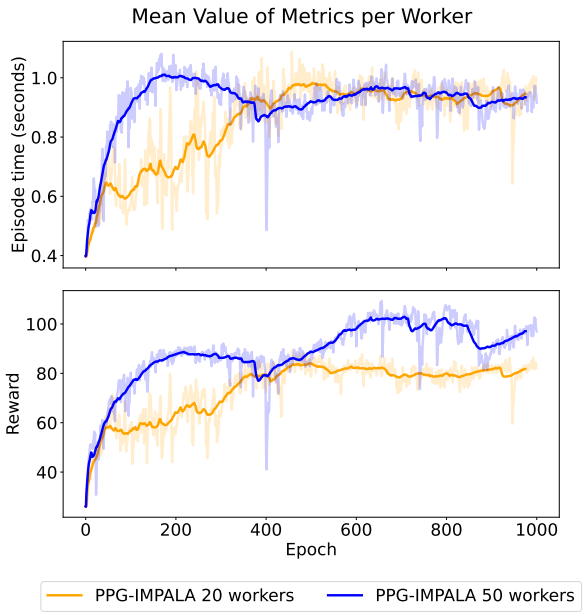

Figure 4.1: The mean episode time and mean total episode rewards of two different runs. The data was smoothed using an exponentially weighted mean, given a span of 50.

4 Results and Discussion

This section presents the results of PPG-IMPALA+IL on the bipedal motion task, and begins with an analysis of the algorithm's performance. Following this, the kinematic results of the best episode are compared against the imitation data provided by Camargo et al., and a comparison to PPO-CMA+IL is made. Finally, the limitations of the study are discussed, and suggestions for improvements are provided.

4.1 Performance of PPG-IMPALA

4.1.1 Algorithm Performance

To assess the increase in performance given a variable number of workers, the simulations report the metrics by averaging the metrics of all agents per epoch. More specifically, after the main node has received the mean total reward and mean episode time for all episodes explored by the workers, it sums each metric and divides it by the number of workers. In this case, the episodes represent trajectories that have been completed, and their number varies on each epoch. The number of episodes can be determined by dividing the number of steps N_S by the average episode time for each epoch.

Therefore, Figure 4.1 (top) shows the mean episode time per epoch of PPG-IMPALA trained using 20 workers, along with a simulation using 50 workers. As it can be observed, episode time surges earlier when training with more workers. Similarly, Figure 4.1 (bottom) shows the mean reward per epoch of the two runs. The increase in reward is more stable when using a larger number of workers, and converges to a higher reward value for the 50 workers experiment. In this case, the large difference between both metrics occurs due to the greater number of workers that create approximately 3 times more trajectories. Furthermore, almost no performance reduction could be observed during training, suggesting that the centralized learner can efficiently use the resources provided.

4.1.2 Kinematics

To identify whether PPG-IMPALA can be used to obtain a healthy gait pattern, a comparison is made between the best episode obtained during training (given by total reward and episode time) and the

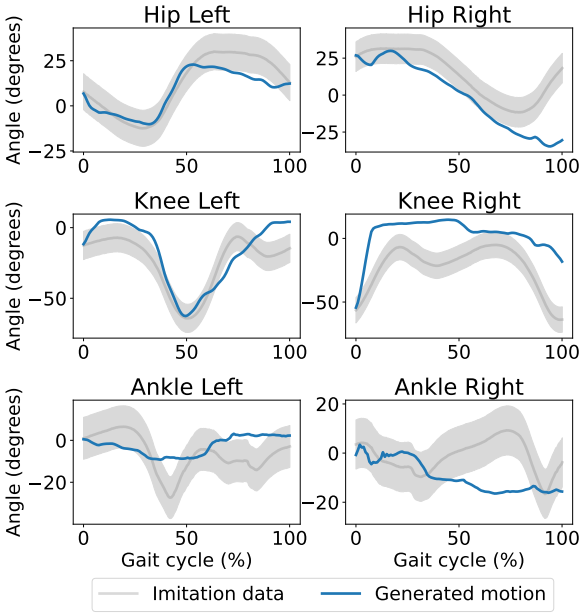

Figure 4.2: Angle difference of hip, knee, and ankle between the imitation data and the best generated motion, during a gait cycle. The imitation data is plotted with a standard deviation of 10°.

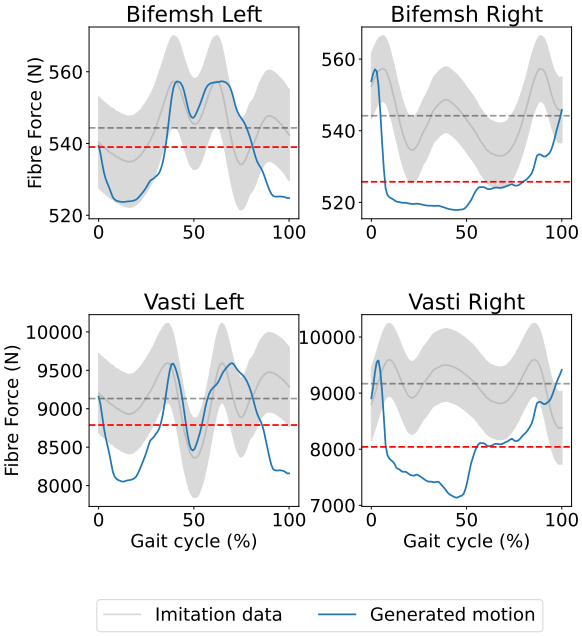

Figure 4.3: Fibre forces of the Biceps Femoris (Bifemsh) and the Vastus Lateralis (Vasti) muscles for both legs, during the gait cycle. The gray area represents the imitation data, given the standard deviation of that gait segment for the respective muscle. The dotted lines represent the mean fibre force of the generated motion (red) and the imitation data (gray).

imitation data. As such, Figure 4.2 shows the difference between the angles of the hip, knee, and ankle generated by PPG-IMPALA and the ones provided by the imitation data.

As it can be observed, the generated motion shows some similarities with the provided healthy gait. The emerging gait seems to have a similar shape early for both hips and the left knee, but deviates later in the cycle from the imitation values. For the right knee and both ankles, the resulting shapes do not match the imitation data. This discrepancy results in the model failing to generate a healthy gait pattern, which leads to falling after 2 steps.

Figure 4.3 shows the fibre forces of the Vastus Lateralis and the Biceps Femoris muscles, exerted by both legs of the musculoskeletal model during the gait cycle (approximately 0.6 seconds). As it can be observed, only the forces generated by the left leg follow the shape of the imitation data, while the right leg values diverge soon after the start of the gait pattern. For both legs, the mean values of the fibre forces are lower for the generated motion, with a larger difference from the imitation data in the case of the right leg. However, the forces generated for the left leg appear stable, indicating that a policy trained by PPG-IMPALA+IL could be used to generate forces that can potentially drive other types of actuators.

4.2 Comparison to PPO-CMA

To identify whether PPG-IMPALA performs better than other DRL architectures, a comparison was made with the PPO-CMA algorithm described in Section 1 after it was trained in similar conditions. The PPO-CMA model contained two hidden layers of 256 units each for the policy, used 2048 batches during the policy update, and performed 20000 steps for each worker before an update. Furthermore, the λ discount factor was set to 0.95, and the γ discount factor was set to 0.99, while the value clip range ϵ was set to 0.2. Finally, the pol-

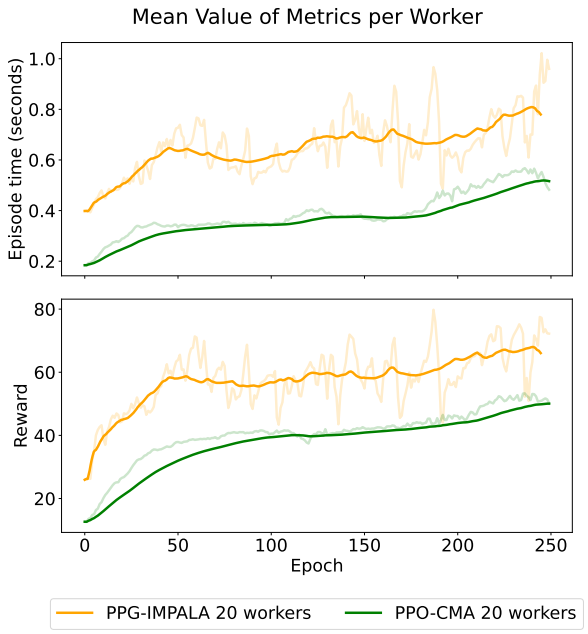

Figure 4.4: The mean episode time and reward of PPG-IMPALA and PPO-CMA, smoothed using an exponential moving average, given a span of 50. The two sets of results were aligned to match the total number of trajectories achieved.

icy used Leaky ReLU as an activation function for each layer, and was trained using a learning rate of 0.001.

As it can be observed in Figure 4.4, PPG-IMPALA has greater values earlier in the training process for both the reward and episode time. Furthermore, PPO-CMA was trained for approximately 67 hours to achieve approximately 60000 trajectories, while PPG-IMPALA was trained for approximately 5 hours. This suggests that PPG-IMPALA is approximately 14.6 times faster than PPO-CMA, and is capable of faster training by increasing the number of resources available. More specifically, PPG-IMPALA using 50 workers achieved the same number of steps performed approximately 2.5 times faster than PPG-IMPALA with 20 workers, indicating limited loss of resources during scaling. As such, greater improvements can be observed during training when increasing the number of workers, both in terms of training time and early performance of the policy on the given task.

Finally, PPO-CMA did not generate a walking pattern, but its training resulted in an episode where three steps were achieved before falling. The PPG-IMPALA implementation with 20 workers achieved a similar policy after the same number of trajectories, but the best episode only performed two steps before falling. This difference could occur due to more exploration being performed by the workers in the case of PPO-CMA, where each agent performs 20000 steps before each update instead of 1024. Moreover, the on-policy behaviour of PPO-CMA can further benefit the training process when more samples are available, which leads to more stable reward values across trajectories. Therefore, compared to PPG-IMPALA, PPO-CMA explored a desirable policy for much longer during training, leading to an episode where the musculoskeletal model travelled farther. However, such an episode was not consistent, which suggests that no clear comparison can be made between the two algorithms in terms of number of steps achieved.

4.3 Limitations and Future Outlook

As discussed previously, PPG-IMPALA was not successful in generating a stable gait pattern in the bipedal motion task. However, the results show that this DRL algorithm can be used to increase the speed of training a policy, when compared with previous approaches.

During training, many experiments led to the policy learning to stand still after the first step, or falling forward immediately after. The previously described configuration for the reward provides an incentive to continue moving forward and follow the imitation data, but it has no stopping mechanism for preventing the policy from standing still. This leads to the policy receiving a large reward by moving the pelvis back and forth without falling, which stops it from continuing the gait cycle. Therefore, future studies could attempt to implement early stopping if the model has not moved its pelvis forward a certain distance, after a given period of time. This could promote further exploration of the gait cycle, while limiting the resources being used to continue training in the standing still/falling forward scenario.

A second problem that could be observed during training was the lack of contact forces between muscles and bones. The policy could generate motions where the leg bones intersect one another dur-

ing the gait cycle, which is not possible in a realistic scenario. Therefore, future work can add another early stopping mechanism when the bones phase through any body part to promote realistic behaviour and reduce training time.

Notably, an early increase in total reward could be observed in the 50 workers experiment due to a larger number of trajectories being used for training. Since each worker submits its results to the centralized worker and continues training without waiting for the policy to be updated, this can lead to more exploration being performed when there is a larger number of workers available. However, numerous workers can overburden the centralized learner, leading to a loss of efficiency due to data being left for too long in the queue. Therefore, future studies could improve on the parallelization aspect of the proposed PPG-IMPALA architecture to allow multiple learners to communicate information for the policy update. This change can allow a great number of CPUs to be used during training, which can lead to speeds similar to the ones proposed by Espeholt et al.

Finally, to increase the performance when training a policy to solve bipedal motion tasks, future implementations of PPG-IMPALA can attempt to improve the current MLP. For instance, as the gait cycle alternates between the two legs, a neural network architecture that splits the parameters of each leg could improve performance during training. More specifically, connecting the input nodes of the body parts closest to one another for each leg, and limiting the interaction between distant ones could improve sample efficiency during training. Furthermore, a second hidden layer can be used to share information between the two legs, which can be used to generate a phasic behaviour during the gait cycle. A variation of this type of limitation to the search space can increase the speed of training, and could potentially be used to alternate the behaviour of the legs during walking.

5 Conclusion

This study explored the use of a highly scalable and parallelized DRL algorithm on a bipedal motion control task. The goal was to generate a healthy gait pattern, given a musculoskeletal model simulated using the OpenSim software. Although the trained policy was not capable of achieving a stable and healthy gait pattern, several improvements could be observed over previous approaches.

Firstly, PPG-IMPALA showed a considerable increase in performance when compared with PPO-CMA, both in terms of rewards obtained and training time. These results provide further evidence that off-policy algorithms can be more efficient than their on-policy counterparts, as theorized in Section 1. Secondly, further improvements could be observed when training with a greater number of workers. The algorithm was capable of producing a policy that could consistently receive higher rewards early in the training process in the 50 workers experiment. This could be attributed to enhanced exploratory capabilities due to obtaining more samples from slightly outdated policies during the training process.

Finally, the proposed PPG-IMPALA architecture is limited in its ability to scale up indefinitely due to it only using a single centralized learner. Future studies can attempt to implement communication between multiple learners, and evaluate different reward configurations to achieve a healthy gait pattern. Furthermore, since the resulting forces appeared to be stable in the 50 workers experiment, future work can explore the impact of the number of workers available during training on the overall stability of fibre forces generated.

6 Acknowledgments

This work was founded by the European Commission's Horizon 2020 Programme as part of the project MyLeg under grant no. 780871. Furthermore, we thank the Center for Information Technology of the University of Groningen for their support, and for providing access to the Peregrine high performance computing cluster.

Finally, the author would like to thank their supervisor, Raffaella Carloni (Professor, University of Groningen) for their guidance and supervision, and Robin Kock (MSc student, University of Groningen) for the discussions regarding the OpenSim environment and parallelization techniques.

References

Camargo, J., Ramanathan, A., Flanagan, W., and Young, A. (2021). A comprehensive, open-source dataset of lower limb biomechanics in multiple conditions of stairs, ramps, and level-ground ambulation and transitions. *Journal of Biomechanics*, 119:110320.

Cobbe, K., Hilton, J., Klimov, O., and Schulman, J. (2020). Phasic Policy Gradient.

De Vree, L. and Carloni, R. (2021). Deep Reinforcement Learning for Physics-Based Musculoskeletal Simulations of Healthy Subjects and Transfemoral Prostheses' Users During Normal Walking. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:607–618.

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., and Thelen, D. G. (2007). OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.

Hämäläinen, P., Babadi, A., Ma, X., and Lehtinen, J. (2020). PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation.

Kidziński, Ł., Ong, C., Mohanty, S. P., Hicks, J., Carroll, S. F., Zhou, B., Zeng, H., Wang, F., Lian, R., Tian, H., Jaśkowski, W., Andersen, G., Lykkebø, O. R., Toklu, N. E., Shyam, P., Srivastava, R. K., Kolesnikov, S., Hrinchuk, O., Pechenko, A., Ljungström, M., Wang, Z., Hu, X., Hu, Z., Qiu, M., Huang, J., Shpilman, A., Sosin, I., Svidchenko, O., Malysheva, A., Kudenko, D., Rane, L., Bhatt, A., Wang, Z., Qi, P., Yu, Z., Peng, P., Yuan, Q., Li, W., Tian, Y., Yang, R., Ma, P., Khadka, S., Majumdar, S., Dwiel, Z., Liu, Y., Tumer, E., Watson, J., Salathé, M., Levine, S., and Delp, S. (2019). Artificial Intelligence for Prosthetics - challenge solutions.

Lee, S., Park, M., Lee, K., and Lee, J. (2019). Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics*, 38(4):73:1–73:13.

Pandy, M. and Anderson, F. (2000). Dynamic simulation of human movement using large-scale models of the body. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 676–681 vol.1.

Park, H., Yu, R., Lee, Y., Lee, K., and Lee, J. (2022). Understanding the stability of deep control policies for biped locomotion. *The Visual Computer*.

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 Transactions on Graphics*, 37(4):1–14.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2018). High-Dimensional Continuous Control Using Generalized Advantage Estimation.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms.

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.

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.

Wang, Y., He, H., Wen, C., and Tan, X. (2020). Truly Proximal Policy Optimization.