Evaluating Deep Reinforcement Learning Algorithms for Physics-Based Musculoskeletal Transfemoral Model with a Prosthetic Leg Performing Ground-Level Walking

Bachelor's Project Thesis

Shikha Surana, s3701816, s.surana@student.rug.nl,
Supervisor: Prof. Dr. R. (Raffaella) Carloni & M.Sc. Vishal Raveendranathan

Abstract: This paper compares deep reinforcement learning algorithms for a physics-based musculoskeletal osseointegrated transfemoral model with a prosthetic leg performing ground-level walking. The algorithms compared are: Proximal Policy Optimization (PPO) with Imitation Learning, and PPO with Covariance Matrix Adaptation (CMA) and Imitation Learning. The imitation dataset is a public dataset consisting of the joint kinematics of a healthy subject walking in a straight line on flat terrain. Unfortunately, both DRL algorithms were unsuccessful in generating a healthy gait, however, this paper does evaluate them based on: cumulative reward, similarity of kinematics to imitation data, and muscle/actuator usage. Compared to PPO+IL, PPO-CMA+IL received a 78.5% larger mean cumulative reward, 64.7% larger mean duration of an episode, and a decrease of 12.7% in muscle and actuator usage. In contrast, both algorithms performed similar in terms of the proximity between model's joint kinematics and the imitation data. Future research will refine the reward function and reduce the erraticness of actuator forces.

I. Introduction

Deep reinforcement learning (DRL) successfully handles high-dimensional state spaces (deep learning) while maximizing utility based on past experiences (reinforcement learning) to solve problems for continuous state and action spaces. DRL has been widely used to simulate a human-like walking gait in a variety of models, such as a self-balancing exoskeleton model [1], a neuromuscular model [2], and physics-based biped characters [3, 4, 5].

The aim of this study is to simulate a physics-based musculoskeletal transfemoral (above-knee) amputated model with an osseointegrated prosthetic lower limb to perform ground-level walking [6]. Proximal Policy Optimization (PPO) is a current state-of-the-art DRL algorithm introduced to the field of robotic locomotive tasks [7]. Previous research has illustrated impressive performance using PPO [8] and PPO with Imitation Learning [2, 9] to simulate a human-like gait in models similar to the one used in this research. These studies provide both inspiration and justification of using PPO with Imitation learning with the proposed osseointegrated transfemoral model to optimize its kinematics and muscle activations, and simulate its forward dynamics on flat terrain. Furthermore, given the limited research on PPO with bipedal locomotion and its proposed limitations [10], this paper compares the PPO with Imitation Learning algorithm with one of its variations. The algorithms to be compared are: (1) PPO with Imitation Learning (PPO+IL) [2, 9], and (2) PPO with Covariance Matrix Adaptation (PPO-CMA) [10] and Imitation Learning (PPO-CMA+IL). Imitation Learning for the algorithms is implemented via an extensive and public dataset provided by Camargo et al. [11] which includes 27 able-bodied subjects performing various locomotion activities ranging from walking to ascending/descending stairs. The dataset used in this research involves subject AB027 walking forwards in a straight line on a flat terrain.

The implementation of the DRL algorithms and the computer simulations of the transfemoral model is realized through OpenSim which is an open-source software system [12]. The musculoskeletal model consists of 15 muscles and 2 actuators controlling 14 degrees of freedom. The sound leg has 11 muscles, and the amputated leg has 4 muscles with an actuator each at the knee and ankle joint. This study adds to the current field of research by making the following three contributions. First, this study compares the PPO+IL algorithm with the PPO-CMA+IL algorithm. This involves modifying the PPO-CMA implementation to include imitation learning, the functionality of the model and the OpenSim framework. Second, this study brings forth a novel osseointegrated transfemoral amputee model with a prosthesis to the analysis of gait patterns in OpenSim, as well as, evaluates the muscle and actuator activations during ground-level walking. Lastly, this study utilizes the normal-walking motion capture dataset [11] to train the model and comments upon the efficacy of using this particular motion dataset as an imitation dataset.

The novelty in this research lies within the modified PPO-CMA+IL architecture shown in Figure 1 where the osseointegrated transfemoral model is the agent and the algorithm works as follows. The agent receives the activa-

tion values for the muscles and actuators (i.e. the action) from the policy mean and variance networks and enacts it in the simulation resulting in a new observed state. This state contains the measurements of the agent's muscles and joints which is scaled and input to the neural networks for the next action. In contrast to the original PPO-CMA algorithm, the reward used to train the networks is composed of two terms; the advantage estimates and the imitation term. The advantage estimate represents the relative value of the actions in the policy and the imitation term represents the proximity of the agent's behaviour to the imitation dataset. The imitation term is the novel addition, and the approach of combining it with PPO-CMA is inspired by de Vree and Carloni [9].

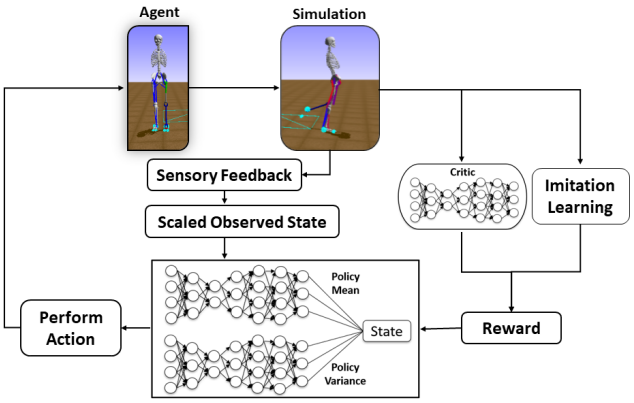

Figure 1: The proposed PPO-CMA with Imitation Learning architecture to simulate the forward dynamics of the transfemoral model with a prosthesis.

To summarize, this paper has two main objectives. First, use the two DRL algorithms, PPO+IL and PPO-CMA+IL, to control the muscles and actuators of the transfemoral model to achieve a gait pattern with forward dynamics comparable to healthy subjects. Second, compare the two DRL algorithms based on training time, cumulative reward and the proximity between the model's gait and a healthy gait.

The rest of the paper is structured in the following way. Section II presents the theoretical background on DRL and its role in bipedal locomotion, as well as, introduces motivation behind implementing and comparing the PPO and PPO-CMA algorithms. Section III introduces the methodology of this research by explaining the structure of the deep neural networks and the role of imitation learning in training the model to perform a healthy gait pattern. Section IV describes the implementation of the model in OpenSim and the processing of the imitation dataset. Section V examines the empirical results, and finally, the concluding remarks of this study are formulated in Section VI.

II. Theoretical Background

This Section describes the use of OpenSim and states the motivation behind using DRL to simulate the forward dynamics of the osseointegrated transfemoral model with a prosthesis. Furthermore, the PPO and PPO-CMA algorithms are introduced and the justification behind using these state-of-the-art algorithms is presented.

A. OpenSim

OpenSim is an open-source software system that provides the platform to develop human-like musculoskeletal structures and analyse its dynamic simulations of a wide range of motions [12]. This study uses OpenSim to visualize the osseointegrated transfemoral model's behaviour when given both the imitation dataset or actions from the DRL algorithms, and to process the imitation dataset which is further explained in Subsection IV.B.

The motivation behind using OpenSim is explained by previous studies utilizing this application to study and/or simulate bipedal locomotion. Shachykov, Shuliak and Henaff used OpenSim to generate the forward dynamics simulations of their neuro-musculoskeletal model to simulate both a healthy gait and an interrupted walking pattern with a sudden stop [13]. Furthermore, de Vree and Carloni developed a healthy and a transfemoral amputee OpenSim model and simulated its forward dynamics to generate a healthy gait using a DRL algorithm [9]. Moreover, Becker, Emmanuel, and Jean-Mare used OpenSim to simulate the forward dynamics of a musculoskeletal model of a horse forelimb to estimate the joint loading [14]. Lastly, Luengas-C, Camargo, and Garzón used OpenSim to analyse the possible instability effects of dynamic alignment during a transtibial amputee model's gait in the sagittal plane [15].

B. Deep Reinforcement Learning

Deep reinforcement learning modifies the reinforcement learning approach by outsourcing the decision-making responsibility to one or more neural network(s) which is better equipped to handle high-dimensional input states. As a result, DRL has been widely used in bipedal locomotion tasks where the observed state space is relatively large and the computation benefits proposed by DRL can be achieved. The motivation behind using DRL to simulate a healthy gait in the transfemoral amputee model with a prosthesis is justified through previous research which are described below.

This study builds upon the research conducted by de Vree and Carloni who used DRL to train both a healthy and transfemoral amputee model to perform ground-level walking where the model's gait resembles that of a healthy subject [9]. The results showed that DRL successfully simulated the forward dynamics of both mod-

els to perform normal walking with a stable and relatively healthy gait. Hence, using the DRL approach to simulate a healthy gait with the prosthesis model is a promising method as the activity performed by this study's model is also normal walking and the differences between the two models are relatively minimal. This study specifically adds to this previous research by further evaluating the PPO+IL algorithm by comparing it with PPO-CMA+IL. Furthermore, Dong et al. proposed a DRL-based training framework for their self-balancing exoskeleton which is a robot that aids paraplegic patients to walk. The training algorithm is centred around policy gradient descent and the experimental results demonstrate an adequate control policy for the exoskeleton. Lastly, various studies have successfully used DRL in combination with imitation learning to train physics-based characters in performing a wide range of motions [3, 4, 5]. All these studies demonstrate the success DRL coupled with imitation learning has achieved in the field of bipedal locomotion.

B.2. Proximal Policy Optimisation

The PPO algorithm was developed by Schulman et al. in an effort to combine the benefits of the trust-region policy optimisation (TRPO) algorithm with an algorithm that is simple to implement, easy to tune and has better sample complexity [7]. PPO is a policy gradient method which means the PPO function indicates how to update the current policy of the agent so that it converges to the optimal policy. PPO is also an on-policy method which entails the agent interacts with its environment to create experiences that it can learn from. Since these experiences are discarded after each iteration, on-policy algorithms, such as TRPO, tend to be sample inefficient. Hence, to tackle both the sample inefficiency issue and the complexity behind the TRPO algorithm, Schulman et al. developed PPO. In general, the PPO algorithm alternates between sampling experiences for the current policy and optimizing the objective function using the older experiences resulting in greater stability, reliability and sampling efficiency. The following paragraphs provide justification for using this specific learning algorithm.

As previously stated, this study takes inspiration from the work of de Vree and Leanne who brought forward the notion of utilizing PPO with imitation learning to analyse gait patterns of healthy subjects [9]. Furthermore, Anand et al. also implemented the PPO algorithm with imitation learning to train their neuromuscular model to perform human walking behaviour. This study's results showed a close resemblance between the model's gait and a healthy gait at walking speeds ranging from 0.6 m/s to 1.2 m/s [2]. In a different study conducted by Melo and Máximo [8], a novel model-free DRL framework based on PPO with no prior knowledge was proposed to create a new running policy that surpasses the state-of-the-art velocity recorded by [16] in the RoboCup 3D Soccer environment. Their results demonstrates that their proposed method exceeds the top forward speed by 50.3%, relative to the best results achieved by [16], and they report on PPO's sample efficiency as their model was able to learn the motions in only a few hours. However, one significant drawback stated in this research is PPO's high sensitivity to hyper-parameters which leads to distinct policies.

Moreover, Xie et al. work with a realistic model of a biped robot, called Cassie, and use a DRL framework to train the controllers for bipedal locomotion in a model-free manner in which the optimisation of the policy is handled by PPO. Their results demonstrate that a DRL-based approach which includes PPO as the optimizer is able to effectively train controllers on real biped model. Lastly, PPO is chosen as one of the learning algorithms because the creators of PPO [7] compared it against various policy gradient methods that are considered to be effective for continuous problems, such as TRPO, cross-entropy method (CEM), vanilla policy gradient with adaptive stepsize, advantage actor critic (A2C) [17], and A2C with trust region [18]. The algorithms were compared within several different MuJoCo environments and the results illustrated that PPO outperformed the five remaining methods in almost all continuous control environments. Furthermore, PPO was also tested on Roboschool environments where a 3D humanoid is trained on high-dimensional control problems, such as running, steering, and rising up from the ground, and the results show that PPO allowed the humanoid to successfully learn and accomplish these tasks.

B.2. PPO with Covariance Matrix Adaptation

Hämäläinen et al. proposed the PPO-CMA algorithm in response to PPO's exploration variance shrinking prematurely which leads to slower progress [10]. Inspired by the CMA evolutionary strategy (CMA-ES), PPO-CMA dynamically expands and contracts its exploration variance to achieve faster progress and is less prone to remaining stuck in local optima. Hämäläinen et al. compared their PPO-CMA algorithm with PPO in various Roboschool continuous control environments and concluded that PPO-CMA achieved better results and was less sensitive to hyper-parameter tuning. Given that their study proposed several advantages of using PPO-CMA over PPO, this paper utilizes this algorithm to also train the osseointegrated transfemoral model with a prosthetic limb and further examines whether these advantages apply in this specific bipedal locomotion task. Moreover, since PPO-CMA has not been used to simulate the forward dynamics of a musculoskeletal model with a prosthesis, this study investigates whether PPO-CMA+IL can train the model to walk.

Before discussing the methods inspired by CMA-ES

and deployed by PPO-CMA, the relation between CMA-ES and robot locomotion is addressed. The CMA-ES algorithm aims to find the optimal solution by repeatedly updating a covariance matrix and optimizing the objective function. CMA-ES has frequently been used in robotic applications [19], specifically, it is widely used to optimize policy based controllers for simulating a walking motion [20, 21]. Furthermore, Geijtenbeek, van de Panne, and van der Stappen used CMA-ES to simulate a natural walking gait in 3D bipedal characters by optimizing an objective function that penalized errors in the character's speed, head orientation and velocity, and effort [22].

The PPO-CMA algorithm avoids premature convergence and achieves stability by incorporating three techniques from CMA-ES. The first CMA technique ensures stability by fitting the sampling distribution only to positive weighted samples, i.e. samples below the median fitness are set to 0 and have no effect. Similarly, PPO-CMA's loss function uses only the positive advantage estimates to train the policy networks. Note, the loss function used by PPO-CMA is not the clipped surrogate loss function but rather a standard policy gradient loss function which is explained further in Section III. To avoid losing information when discarding negative advantages, Hämäläinen et al. proposed a mirroring technique to convert the negative advantages to positive advantages which are then also used to train the policy networks.

The second technique is called Rank-μ update which aims to elongate the exploration distribution by first updating the variance policy network and only then the mean policy network. PPO-CMA incorporates this method by using two neural networks (unlike PPO) and follows the same updating procedure.

Finally, the third technique approximates the evolution path heuristic where states with consistently increasing mean fitness values are determined and increased exploration in the same direction is performed. PPO-CMA implements this method by storing data from several previous iterations and training the variance network with this history of training data instead of only the previous iteration's data. As a result, this method is more sample efficient than various other on-policy algorithms.

III. Methods

This paper compares the DRL algorithms PPO+IL and PPO-CMA+IL to train the transfemoral amputee model with a prosthetic limb to walk on flat terrain. This section outlines the components used in the DRL algorithms.

A. Deep Neural Networks

A policy maps states with actions and in DRL algorithms is represented by deep neural networks. This study uses a multi-layer perceptron, a feed-forward artificial neural network (ANN), for both algorithms. The ANN used in PPO consists of 4 layers; an input layer with 102 neurons, two hidden layers each with 312 neurons, and an output layer with 17 neurons. The output y of each neuron v_i is calculated via the tanh activation function with the following equation: $y(v_i) = \tanh(b + \sum_{i=1}^{n} x_i w_i)$. Here, b is the bias, n is the number of neurons in the previous layer, x is the input to the neuron, and w is the weight between the current and previous neuron.

In contrast to PPO, PPO-CMA utilizes two ANNs, i.e., a policy mean network and a policy variance network. However, both ANNs include the same 4 layers: an input layer with 102 neurons, two hidden layers each with 128 neurons, and an output layer with 17 neurons. This network uses the *Leaky ReLU* activation function (*lrelu*) which solves the dying *ReLU* problem by generating small negative outputs when the input is below 0. The *Leaky ReLU* activation function is defined as follows: $y(v_i) = lrelu(b + \sum_{i=1}^{n} x_i w_i)$. In addition to the policy networks, this algorithm also makes use of a critic network which is also composed of a feed-forward ANN. The critic network consists of 4 layers: an input layer with 89 neurons (dimension of the state vector + 1), two hidden layers each with 128 neurons, and an output layer with 1 neuron. Plus, this network also uses the *Leaky ReLU* activation function. Lastly, Adaptive Moment Estimation (Adam) was selected as the optimization algorithm responsible for reducing the loss function for all three ANNs and its 'learning rate' hyper-parameter was tuned at 0.0005 (for all three networks).

The input to both neural networks is a continuous variable that represents the state of the agent and the optimal values taken from the imitation dataset. Note, the imitation dataset is used to both train the model to perform a healthy gait and validate the model's gait after training. The state variable includes the positions/rotations and linear/rotational velocities of the joints' and actuators' angles as well as measurements of other body segments of the transfemoral prosthesis model, and the pelvis, hip, knee and ankle positions from the imitation dataset at time-step *t+1*. The imitation data at time-step *t+1* is fed to the network because the output is the action to be performed at time-step *t+1* as well, and thus the network can use the information about optimal values to compute the optimal action vector. Please refer to Table I which specifically states the different elements constituted in the observation vector that is input to the policy networks.

The output of the PPO-CMA neural networks is modelled by a Gaussian policy as the action space is contin-

uous, whereas, PPO's policy network is modelled by a multicategorical probability distribution which results in a discrete actions space (i.e. the action vector is binary). It can be argued that due to the differences in the output (i.e. continuous vs. discrete action space), the comparison of the two algorithms may be considered insignificant. However, given that Hämäläinen et al. have compared the two algorithms, this research builds upon that study and verifies whether the proposed advantages of PPO-CMA over PPO can be observed when training the musculoskeletal model to perform normal walking. The ANNs of both DRL algorithms output an action vector that contains the activation values between [0,1] for the 15 muscles and [-1,1] for the 2 actuators. These ranges represent the minimal and maximal values of activation for the muscles and actuators, respectively. The neural network computes the output activation vector from the input state vector by calculating the values of the nodes in each consecutive layer via the activation functions, weights between the linked nodes, and the biases. This paper aims to optimize the weights of the deep neural networks in such a manner that for each given state being input to the network, the optimal action is output which results in the agent performing a healthy walking gait.

Table I: The elements present in the model's state vector.

Model Property	# of elements
Pelvis pitch, roll, yaw and velocities	3+6
Joint positions and velocities (hip adduction & abduction, knee & ankle) for both legs	8+8
Ground reaction forces for both legs	3+3
Force, length and velocity of each of the 15 muscles	15+15+15
Force, speed, control, actuation, power and stress for both actuators	6+6
Imitation dataset values at timestep *t+1* of pelvis, hip (add. & abd.), knee and ankle angles	6+4+4
Total size of the state vector	**102**

B. The Learning Algorithms

Several learning algorithms are used to train the neural network to achieve the desirable state-action behaviours that results in the model enacting a healthy gait. The following subsections describe how the weights of the neural networks are optimized to obtain the desired output.

1) Imitation Learning: One of the terms used to train the neural network is the imitation learning term which is realized through a reward function. The reward function computes the utility of performing an action in a given state and thus, this reward value informs the agent of the relative value of its action. After every action, the only information the agent is provided with is the reward which ranges from $[-\infty, \infty]$, and the agent's behaviour is tuned to maximize the cumulative reward.

In this paper, the reward function is created to reward the model when its actions produce a healthy gait and penalize all other scenarios. This is achieved by comparing the agent's pose and orientation with the desired pose and orientation that is recorded in the imitation dataset. This includes the measurements of the following body segments:

- Joint position of the pelvis in the x, y and z direction.
- Joint angle of the pelvis list, tilt and rotation, and left/right hip flexion and adduction, knee and ankle.
- Joint angular velocity of left/right hip flexion and adduction, knee and ankle.

Hence, for the agent to produce a healthy gait that is comparable to that in the imitation dataset, the more similar the agent's state is to the imitation data, the higher rewards it receives.

The reward function utilized in this study is shown in Equation 1 which consists of 2 elements; reward based on the agent's pose, reward based on the proximity of the agent's joint positions and angles to the imitation dataset.

$$Reward_t = reward_{pose,t} * w_p + reward_{imitation,t} * w_i \quad (1)$$

The reward function is represented by $Reward_t$ where t is the time-step. The reward that is based on pose is shown by $reward_{pose,t}$ which computes the proximity between the x, y, z position of the agent's pelvis and the desired pelvis pose at time-step t. Specifically, the root-mean-square error (RMSE) is computed between the current and desired pelvis pose denoted as $penalty_{pose,t}$ and the pose reward is computed via the following equation: $reward_{pose,t} = \exp(-8 * penalty_{pose,t})$.

The imitation reward is represented by $reward_{imitation,t}$ which is calculated by determining the proximity between the agent's joint angles' positions and velocities and the desired angles' positions and velocities. This includes the joint angles of the pelvis list, tilt and rotation, and left/right hip flexion and adduction, knee and ankle, as well as, the Joint angular velocities of left/right hip flexion and adduction, knee and ankle. The RMSE between the aforementioned measurements of the agent's joints and the desired angles and velocities is computed. This results in a penalty term for the joint angle $penalty_{angle,t}$ and a penalty term for the angular velocities $penalty_{vel,t}$. Finally, the imitation reward is computed as follows:

$reward_{imitation,t} = \exp(-2 * penalty_{angle,t}) * 0.75 + \exp(-0.1 * penalty_{vel,t}) * 0.25$.

During the experimentation phase, it was observed that the model's hips, knees, and ankles were not properly bending, and since, the reward function has a great influence on the agent's behaviour, the imitation reward was modified in the following manner by taking inspiration from [23]. If the model's left/right hip flexion and adduction, knee and ankle angles were between the interval of $[\omega - 25\,\text{deg}, \omega + 25\,\text{deg}]$ where ω is the desired joint angle, the imitation reward is increased by 20%, otherwise, the reward remains the same. Lastly, the position and the imitation reward's weight on the goal reward is determined by w_p and w_i respectively, which are hyper-parameters between [0,1] and needs to be tuned for each DRL algorithm.

The desired joint positions, angles, and velocities used in both of the reward terms is provided by the imitation dataset in [11]. Since the rewards are computed at a specific time-step t and the imitation data is time-stamped, the desired joint positions, angles, and velocities used in the reward computation are taken at the relevant time-step from the dataset. As stated, the closer the agent's state is to the desired state in the imitation dataset, the smaller the penalty which results in a larger reward. As a result, the agent is encouraged to imitate the dataset and generate a normal walking gait.

2) PPO with Imitation Learning: Drawing inspiration from [2, 9], this paper uses PPO with imitation learning to simulate the forward dynamics for the osseointegrated transfemoral amputee model with a prosthetic limb to generate a healthy gait. Before discussing how imitation learning enhances PPO to simulate a human-like gait, the inner workings of the PPO algorithm is first presented.

The PPO algorithm involves a clipped objective function that forms a pessimistic estimate of the current policy's performance to ensure that the new policy is sufficiently close to the old policy. This method of constraining the size of policy updates leads to stability in reaching the optimal policy. Equation 2 below defines the clipped surrogate objective function used by PPO to determine the new policy.

$$L^{CLIP}(\theta) = \widehat{\mathbb{E}}_t \left[min(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t) \right] \quad (2)$$

In Equation 2, the expectation operator, $\widehat{\mathbb{E}}_t$, indicates the empirical average over a final batch of samples, and the advantage function, $\widehat{A}_t$, denotes an estimate of the relative value of a selected action which is computed by subtracting the actual reward received during training by the baseline estimate. Furthermore, $r_t(\theta)$ represents the probability ratio between the new policy, $\pi_\theta(a_t|s_t)$, and the old policy, $\pi_{\theta old}(a_t|s_t)$. Here. π is the policy, θ and θ_{old} are the new and old parameters, and a_t and s_t are the action and state vectors and time-step t. By using the old policy to evaluate the new policy, PPO uses older experiences to shape the new policy which results in better sample efficiency. Notice that $r(\theta_{old}) = 1$ (i.e. dividing the old policy by itself), hence, the aim of this function is to penalize policy changes that push $r_t(\theta)$ farther from 1 and the following paragraph describes this is achieved.

As stated, the objective function takes a pessimistic estimate which is carried out by the *min* operator that takes in two arguments; the normal policy gradient objective, and the clipped version of the objective which is responsible for constraining the policy update. The *clip* operator clips the probability ratio, $r_t(\theta)$, by confining the new policy between the interval $[1-\epsilon, 1+\epsilon]$. If the advantage estimate is positive (i.e. new policy yields higher rewards than old policy), the probability ratio is clipped at $1+\epsilon$ and if the advantage estimate is negative, the probability ratio is clipped at $1-\epsilon$.

This paper combines the PPO algorithm with imitation learning to decrease training time and increase performance [9]. As a result, the policy network is trained by the advantage estimates and the rewards obtained by the agent (computed via Equation 1). The reward function specifies two hyper-parameters: weight of the position reward w_p and imitation reward w_i. Several trials with varying imitation and position reward weights were executed to determine the optimal hyper-parameter values and it was observed that the position reward weighted at 40% and the imitation reward weighted at 60% (i.e. $w_p = 0.4, w_i = 0.6$) yielded the best performance. Note, performance is defined in terms of cumulative reward received by the agent and similarity between the model's gait and a healthy gait (observed by the author). Table II summarizes the hyper-parameters and training details, originated from [9], used in the PPO algorithm. In this table, δ denotes the upper bound for the size of the policy update, and γ is the discount factor ($0 \leq \gamma \leq 1$) that determines the degree to which the agent considers future rewards.

3) PPO-CMA with Imitation Learning: The PPO-CMA+IL algorithm this paper proposes is an addition to the original PPO-CMA algorithm [10] and is the main contribution of this paper. By combining PPO-CMA with imitation learning, the agent should arrive at the global optima in a time-efficient manner while also producing human-like gait patterns.

Given the model's action space is continuous, PPO-CMA utilizes a Gaussian policy that outputs a state-dependent mean $\mu_\theta(s)$ and covariance $C_\theta(s)$ for sampling the actions. Therefore, this algorithm makes use of two policy networks; policy mean network and policy covariance network. The covariance network is responsible for the exploitation-exploration balance as it allows the ex-

Table II: The hyper-parameters and training details used in the PPO algorithm.

Parameter	Value	Parameter	Value
Iterations	X	Parameter Noise	Yes
Episodes	X	PPO Clip Parameter ϵ	0.2
Steps	X	Batch Size	512
Policy Network(s)	1	PPO Optimiz. per Epoch	4
Activation Function	tanh	PPO Entropy Coefficient	0.01
No, of Hidden Layers	2	PPO δ	0.9
Size of Hidden Layers	312	PPO γ	0.999

ploration variance to dynamically expand and contract which leads to faster progress compared to PPO. The policy parameters of the two neural networks are updated via the loss function denoted in Equation 3 which uses a diagonal covariance matrix defined by a vector $c_\theta(s) = diag(C_\theta(s))$.

$$\mathcal{L}_\theta = \frac{1}{M}\sum_{M}^{i=1} A^\pi(s_i, a_i) \sum_j \Big[\frac{(a_{i,j} - \mu_{j;\theta}(s_i))^2}{c_{j;\theta}(s_i)} + 0.5 \log c_{j;\theta}(s_i)\Big] \tag{3}$$

In Equation 3, M is the minibatch size, i indexes over the minibatch, and j indexes over the action variables. Moreover, $A^\pi(s_i, a_i)$ represents the advantage function which indicates the value of selecting action a_i in state s_i. Typically, the advantage function is multiplied with the *log* of the policy (i.e. $\log \pi_\theta(a_i|s_i)$). However, since the Gaussian policy outputs both the state-dependent mean $\mu_\theta(s)$ and covariance $c_\theta(s)$, the log of the policy is rewritten as $\sum_j \Big[\frac{(a_{i,j} - \mu_{j;\theta}(s_i))^2}{c_{j;\theta}(s_i)} + 0.5 \log c_{j;\theta}(s_i)\Big]$. In addition to PPO-CMA using a different loss function, there are three significant differences between PPO and PPO-CMA which are explained in detail in Subsection B.3. Briefly, these differences are; (1) only positive advantages and mirrored negative advantages are used to train the policy networks, (2) first the covariance network is updated and only then the mean network, and (3) the evolution path heuristic is estimated by utilizing data from several previous iterations.

This paper proposes to combine the PPO-CMA algorithm with imitation learning to encourage the model to perform a healthy gait pattern. Similarly to PPO+IL, this proposed algorithm also trains its neural networks with both the advantage estimates and the rewards received by the agent. After tuning the hyper-parameters in the reward function (position reward w_p & imitation reward w_i), the optimal reward weights were observed to be 40% for the position reward and 60% for the imitation reward (i.e. $w_p = 0.4, w_i = 0.6$). Table III summarizes the hyper-parameters and training details used in the PPO-CMA algorithm. Note, GAE λ stands for generalised advantage estimate λ.

Table III: The hyper-parameters and training details used in the PPO-CMA algorithm.

Parameter	Value	Parameter	Value
Iterations	X	PPO Clip Parameter ϵ	0.2
Episodes	X	Batch Size	512
Steps	X	History Buffer Size	12
Policy Network(s)	2	No. of Hidden Layers	2
Critic Network(s)	1	Size of Hidden Layers	128
Activation Function	*Leaky ReLU*	GAE λ	0.95
Adam Learning Rate	0.0005	PPO γ	0.99

Lastly, this Methods section is concluded by briefly presenting the general procedure of both of the DRL algorithms. During the agent's training phase, it experiences countless iterations and each iteration includes several episodes. Each episode consists of numerous time-steps and at each time-step of 0.005 seconds, the agent performs an action. An episode is terminated when the agent's action results in a falling motion which is determined by a pelvis height of smaller than 0.6m. After a number of episodes are completed in the iteration, the iteration is finished and the neural network can be trained with both the advantage estimates and the rewards the agent received during the past iteration. In the case of PPO-CMA, the negative advantage estimates are converted to positive ones, the policy variance network is first trained using both the current iteration's and the history buffer's data, and then, the policy mean network is trained using only the current data. After optimizing the neural network(s), the next iteration is started and the aforementioned process is repeated.

IV. Implementation

This section describes the implementation of the two DRL algorithms on the OpenSim musculoskeletal model, namely the osseointegrated prosthesis model. Furthermore, the imitation dataset's processing procedure and role in the validation of the model's gait is outlined.

A. OpenSim Model

The musculoskeletal model developed in OpenSim includes various hill-type muscles which connect to joints to produce different forces and motions [24]. Through the OpenSim software, one can study the model's joint kinematics, and the muscle-tendon's forces and joints' moments [9]. By utilizing the environment developed by Kidziński et al. for the 2017 NIPS Competition [23], the DRL algorithms can be implemented on the OpenSim models as the environment serves as a bridge between the OpenSim software and the Python programming language (www.python.org).

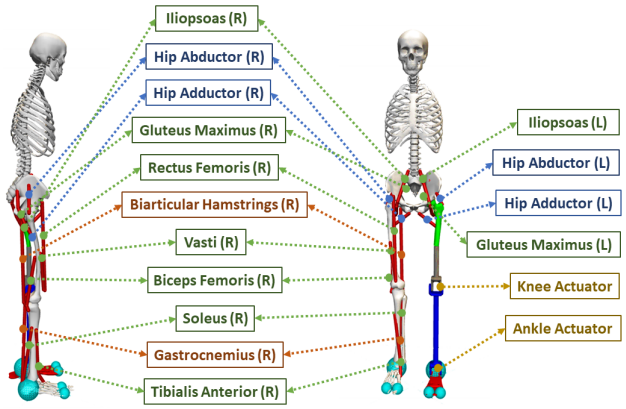

Figure 2: Shows the 15 muscles and 2 actuators possessed by the osseointegrated transfemoral prosthesis model. The uniarticular muscles are labelled in green, biarticular muscles in red, hip adduction and abduction in blue, and actuators in blue.

The physics-based osseointegrated transfemoral amputee musculoskeletal model with a prosthetic limb is developed by Raveendranathan [6]. As stated, this model contains 15 muscles and 2 actuators which can be visualized through Figure 2 which shows the uniarticular muscles labelled in green, biarticular muscles in red, hip adduction and abduction in blue, and actuators in blue. Furthermore, Table Table IV defines the action performed by each muscle and actuator as well as states the leg it resides in. These 15 muscles and 2 actuators control 14 degrees of freedom (DOF): 6 DOF at the pelvis (pelvis tilt, list, rotation, x, y, and z), 4 DOF at the hip joints (hip adduction and abduction for both legs), and 4 DOF at both knee and ankle joints.

The 15 muscles presented in the musculoskeletal model are simulated biological muscles which are based on a non-linear first-order dynamic Hill-type muscle model which relates muscle excitation to activation [25]. Even though the Hill-type muscle model does not realistically represent the human muscle architecture, it does quite precisely simulate the gross biomechanical behaviour of a muscle-tendon unit in a computationally inexpensive manner [26]. The Hill-type muscle model is illustrated

Muscle or Actuator	Primary Function	Leg
Biarticular Hamstrings	Hip extension, knee flexion	Right
Rectus Femoris	Hip flexion, knee extension	Right
Vasti	Knee extension	Right
Biceps Femoris	Knee flexion	Right
Gastrocnemius	Knee flexion, ankle extension	Right
Soleus	Ankle extension	Right
Tibialis Anterior	Ankle extension and flexion	Right
Hip Abductor	Away from body's vertical midline	Both
Hip Adductor	Towards body's vertical midline	Both
Iliopsoas	Hip flexion	Both
Gluteus Maximus	Hip extension	Both
Knee Actuator	Knee flexion and extension	Left
Ankle Actuator	Ankle flexion and extension	Left

Table IV: States the 15 muscles and 2 actuators present in the osseointegrated transfemoral prosthesis model as well as defines its actions and the leg(s) its resides in.

in Figure 3 which also shows a contractile element (CE) connected to elastic elements both in series (SE) and in parallel (PE) (note: figure taken from [9, 25]). This Figure 3 also shows various muscle properties; the muscle fibre length L^M, the tendon slack length L^T, and the pennation angle α^M, all of which determine the level of muscle activation.

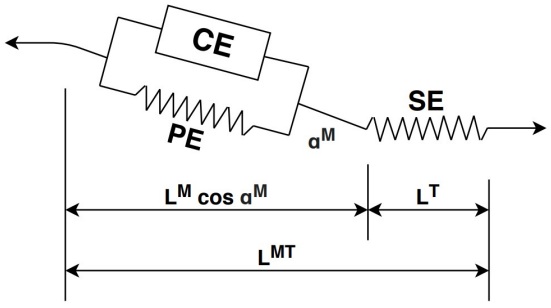

Figure 3: The Hill-type muscle model is utilized in the musculoskeletal model to perform the musculotendon contraction. This includes a contractile element (CE) connected to elastic elements both in series (SE) and in parallel (PE) which all produce a force on the tendon. Figure taken from [9, 25].

At each time-step, the neural network(s) used in the DRL algorithms outputs a vector containing 15 muscle excitations and 2 actuator excitations. OpenSim utilizes the first-order dynamics equations of the Hill-type muscle model to compute the muscle activations from the muscle excitations. Finally, the model's muscles are actuated, and its new state (i.e. torques, ground reaction forces, and joint positions and velocities) is calculated. Since this study uses OpenSim version 4.2, the actuators are Activation Coordinate actuators. The actuators' activation value is computed via a first-order linear activation dynamics equation where the resulting activation value is: $\dot{a} = \frac{u-a}{\tau}$. Here, u is the actuator excitation, a is the activation constant set to 0.01, and τ is the activation time constant.

B. Validation Dataset

The open-source motion capture dataset provided by Camargo et. al [11] is used to implement the imitation learning reward function with aim of encouraging the model to generate healthy gait patterns. Furthermore, this dataset is also used to validate and compare both of the DRL algorithms as it serves as a benchmark for a healthy gait pattern. This dataset contains human locomotion data from 22 able-bodied subjects taken during different locomotion modes (i.e. stairs, ramps, and treadmill) and different terrain conditions (i.e. speed and inclination). Specifically, the following measurements were recorded from the subjects; electromyography (EMG), inertial measurement unit (IMU), goniometer (GON), and joint kinematics, moments and powers which was computed using the inverse dynamics functionality of OpenSim.

This study utilizes the motion data from subject AB06 performing ground-level walking in a straight line, and specifically, the imitation dataset is composed of the following joint-level kinematic measurements. The joint position and angle of the pelvis in the x, y, and z axis, and the joint angle and angular velocity of the left/right hip flexion and adduction, knee and ankle. Note, the motion dataset provided by [11] involves subject AB06 walking in a circuit, however, the imitation dataset used in this study only consists of the data points where the subject is walking in a straight line. Lastly, the subject AB06 was chosen because it had the closest resemblance to the model's height and weight.

Before incorporating the motion capture dataset in the imitation learning part of the proposed DRL framework, the data went through the followed processing steps. First, the orientation of the data was rotated such that the subject's forward walking motions would be along the OpenSim model's x-axis. By using OpenSim's experimental data preview tool and the subject's marker data, the data was rotated 270 degrees about the y-axis. Second, the joint (i.e. hip, knee and ankle) angles were computed via the inverse kinematic functionality present in OpenSim. In more detail, the osim model file, marker data, and a file containing the weights of each marker were given as input to the inverse kinematics tool which outputs a motion file containing the joint angles.

Third, the motion file was reconstructed into a comma-separated values (CSV) file. This CSV file was expanded upon by computing the velocities of the joint angles and adding these measurements to the file. Lastly, the OpenSim environment used to implement the DRL algorithms follows the convention of using radians to represent angles, and thus, all angles in the CSV file were converted from degrees to radians. The resulting imitation dataset was validated in the following two ways. First, the degree-based imitation dataset was converted into a motion file and was used to simulate the forward dynamics of the model in OpenSim which resulted in a gait pattern comparable to a healthy subject. Similarly, the radian-based imitation dataset was given as input to the model in the OpenSim environment used in the DRL framework and the resulting gait pattern was also comparable to a healthy gait.

V. Results & Discussion

This Section presents the results of the two DRL algorithms on the osseointegrated transfemoral model with a prosthesis. The first Subsection shows the results of PPO+IL and the second Subsection shows the results of PPO-CMA+IL. Both algorithms are evaluated by comparing the imitation data [11] with the kinematic results of the simulation. The third Subsection compares the two DRL algorithms and finally, the last Subsection discusses the limitations of this research and outlines possible avenues for future research.

A. Performance of PPO with Imitation Learning

1) Algorithm's Performance: Figure 4 (red lines) shows the performance of PPO+IL on the osseointegrated transfemoral model with a prosthetic limb. The left plot shows the average reward received by the model per episode and the right plot shows the time-step length per episode (red curves) both recorded over a period of 50,000 episodes (one simulation). It can be seen in Figure 4 that the model's performance doesn't improve with increasing episodes as it consistently receives a low reward and only remains for less than 1 second in each episode before falling. This algorithm received a mean cumulative reward of 51.790 (standard deviation (SD) = 0.277) and a mean time-step of 0.717 seconds (SD = 0.010s), please refer to Table V. Lastly, by visualizing the model's performance at the end of the 50,000 episodes, it was observed that the model tried to take a step with its right (healthy) leg but was unable to balance on the

left (prosthetic) leg which ultimately led it to fall every episode.

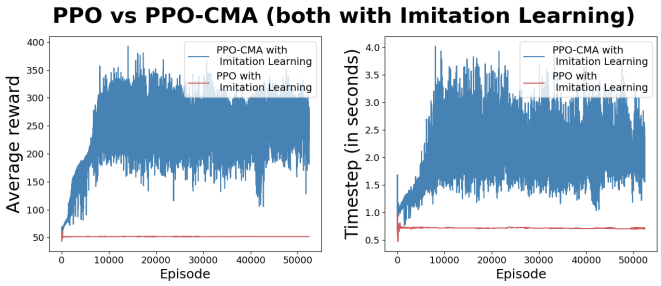

Figure 4: The learning curves of the two DRL algorithms; PPO+IL and PPO-CMA+IL, over a period of 50,000 episodes. The left plot shows the average reward received per episode, and the right plot shows the time-step length per episode. This figure demonstrates PPO-CMA performs better than PPO at maximizing rewards.

Table V: The mean total reward received and the standard deviation (SD) for the two DRL algorithms.

	Reward		Time-step (in seconds)	
Algorithm	**Mean**	**SD**	**Mean**	**SD**
PPO+IL	51.790	0.277	0.717	0.010
PPO-CMA+IL	240.672	46.814	2.030	0.387

2) Kinematics: Figure 5 (red lines) shows the kinematics for the attempted gait of the model being trained on PPO+IL. It compares the imitation data (grey region) with the horizontal/vertical ground reaction forces and the left/right hip, knee, and ankle angles of the osseointegrated transfemoral model. The plots in Figure 5 demonstrate that the model's joint angles do not closely resemble the optimal angle value and do not lie within the optimal angle interval. Furthermore, the horizontal and vertical ground reaction forces are not comparable to the optimal values stated in the imitation dataset. Table VI shows the mean difference between the model's joint angles and the optimal angles stated in the imitation dataset. It can be seen that the right knee and ankle angles are more similar to the optimal values which support the observation that the model tries to take a step with the healthy leg. Furthermore, given that the left joint angles deviate from the optimal angle values, one can hypothesize that this is the reason for the model not being able to balance on the left leg.

B. Performance of PPO-CMA with Imitation Learning

1) Algorithm's Performance: Figure 4 (blue lines) shows the performance of PPO-CMA+IL on the osseointegrated transfemoral model with a prosthetic limb. The

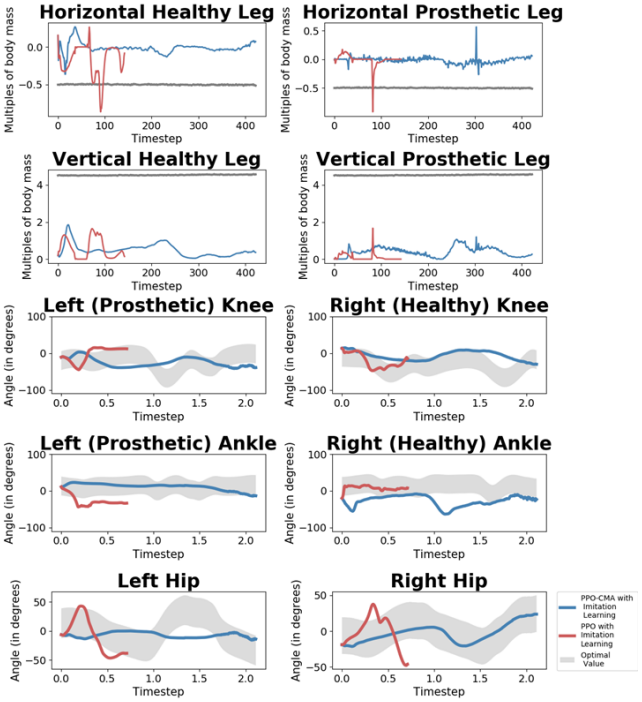

Figure 5: The horizontal and vertical ground reaction forces and hip, knee and ankle angles of the osseointegrated transfemoral prosthesis model using the two DRL algorithms; PPO+IL (red line) and PPO-CMA+IL (blue line) against the imitation data illustrated via the grey lines (the grey area is the mean value +/- 25 degrees which is the optimal interval).

Table VI: The mean difference between the hips, knees, and ankles angles and the imitation data angles (i.e. optimal values) for the two DRL algorithms.

		PPO (in deg.)	PPO-CMA (in deg.)
Left (Prosthetic)	*Hip*	23.014	22.262
	Knee	20.981	21.816
	Ankle	38.174	9.319
Right (Healthy)	*Hip*	22.709	6.485
	Knee	14.478	26.377
	Ankle	6.676	40.391
Total sum of means		**126.032**	**126.650**

left plot shows the average reward received by the model per episode and the right plot shows the time-step length per episode (red curves) both recorded over a period of 50,000 episodes (one simulation). It can be seen in Figure 4 that the average reward received increases for the first 8,000 episodes, whereafter, the average reward remains relatively constant. The steep rise in rewards seen after episode 2,000 indicates that the agent (i.e. model) has learned a policy, i.e. specific weights within the mean

and covariance networks, that allows it to maximize utility based on the reward function. This algorithm received a mean cumulative reward of 240.672 (SD = 46.814) and a mean time-step of 2.030 seconds (SD = 0.387s). Lastly, while visualizing the model's performance after 50,000 episodes, it was observed that the model would often take the first step with the left leg and fall when attempting to take the second step with its healthy leg. However, a few times the model was able to take both the first and second steps, and would ultimately fall while attempting to take the third step.

2) Kinematics: Figure 5 (blue lines) shows the kinematics for the attempted gait of the model being trained on PPO-CMA+IL. It compares the imitation data (grey region) with the horizontal/vertical ground reaction forces and the left/right hip, knee, and ankle angles of the osseointegrated transfemoral model. It can be seen through plots in Figure 5 that the model's joint angles deviate from the optimal values listed in the imitation dataset. Similarly, the horizontal and vertical ground reaction forces are also not comparable to the imitation data values. Table VI shows the mean difference between the model's joint angles and the optimal angles stated in the imitation dataset. This table shows that the left ankle angle and right hip angle have closer proximity to the imitation data values. A possible reason being, it was observed that a left ankle and right hip angle value close to the optimal value was instrumental in making the second step with the right leg as it provided both a forward push in the model's x-axis and raised the right leg to bring it forward.

C. Comparison of the Two Algorithms

1) Algorithm's Performance: The two DRL algorithms are compared below based on: cumulative reward, training time, and similarity of model's gait to a healthy gait. Figure 4 shows the average reward and time-step for both DRL algorithms: PPO+IL (red lines) and PPO-CMA+IL (blue lines). It can be seen in the left plot that while PPO-CMA+IL learns a policy based on maximizing rewards illustrated by the positive slope on the blue line, PPO+IL does not and receives the same amount of reward throughout the simulation illustrated by the horizontal red line. Furthermore, the right plot shows that PPO-CMA+IL is able to train the model to remain upright whilst trying to walk for a longer period of time compared to PPO+IL. Table V shows the mean cumulative reward received per episode by PPO-CMA+IL is over 3-fold (i.e. 78.5%) compared to PPO+IL, and the time-step (i.e. duration of an episode) is more than 2-fold (i.e. 64.7%) compared to PPO-CMA+IL. Therefore, when comparing the two algorithms based on the cumulative reward received, PPO-CMA+IL performs better. Since both algorithms were unsuccessful in simulating a healthy gait, the algorithms cannot be compared based on training time or similarity to a healthy gait pattern. Hence, the following paragraph compares the two algorithms based on kinematics, and muscle/actuator usage.

2) Kinematics and Muscle/Actuator Usage: Figure 5 shows the kinematics for PPO+IL (red lines) and PPO-CMA+IL (blue lines). In this figure, the model's horizontal/vertical ground reaction forces and hip, knee, and ankle angles (both legs) are compared with the imitation data values (grey region). It can be seen that both algorithms' kinematics deviate from the imitation data which supports the observation that neither algorithm is able to simulate a healthy gait which requires the model to generate optimal kinematics. Table VI states the difference in means between the model's kinematics and the imitation data. This table shows that the total sum of means for PPO+IL is 126.032 degrees and for PPO-CMA+IL is 126.650.778 degrees which indicates that both algorithms perform similar when considering the model's kinematics. However, given that PPO+IL's mean time-step in an episode is 63.2% smaller compared to PPO-CMA+IL and PPO+IL's total difference in kinematics means is only 13.7% better, one cannot conclusively state that the performance of both of the algorithms is similar in terms of kinematics due to the significant difference in the duration of each episode between PPO+IL and PPO-CMA+IL.

Table VII summarizes the muscle and actuator usages for both DRL algorithms. It shows the mean fibre forces for the 15 muscles and 2 actuators over a period of 50 episodes, as well as, the total sum of the mean muscle and actuator usages. It can be seen that PPO+IL results in the muscles and actuators usage being 12.7% larger compared to PPO-CMA+IL (22,090.694 vs. 19,288.317). Since lower levels of muscles and actuators usage are more energy-efficient, it indicates that PPO-CMA+IL performs better than PPO+IL. However, due to the differences in the mean duration of an episode for both algorithms, one again cannot accurately compare the muscles and actuators usage for the two algorithms without taking the time-step into consideration. Lastly, one reason for larger fibre forces generated by PPO+IL is that it was observed that the model often made extreme leg movements (i.e. complete flexion/extension of the hip and knee joints) which requires larger force compared to the medium-level bend performed by the model trained with PPO-CMA+IL.

D. Limitations and Future Outlook

The two DRL algorithms were not successful in generating a stable and healthy gait pattern for the osseointegrated transfemoral amputee model with a prosthesis. The following paragraphs discuss various avenues for future research in terms of modification to the current DRL algorithms and motivation for investigating different variations of the PPO algorithm.

Table VII: Results for the comparison of the difference in muscle and actuator usage between the two DRL algorithms.

		PPO with Imitation Learning		PPO-CMA with Imitation Learning	
	Muscle	**Right**	**Left**	**Right**	**Left**
Mean Fibre Force	Hip Abductor	1506.176	1872.624	1837.845	1859.178
	Hip Adductor	692.237	969.779	1458.571	1442.320
	Iliopsoas	1223.751	1330.222	2112.329	2212.345
	Gluteus Maximus	1542.148	1493.699	492.593	562.421
	Biarticular Hamstrings	1638.557	-	1154.800	-
	Rectus Femoris	747.641	-	1120.522	-
	Vasti	4283.897	-	939.239	-
	Biceps Femoris	256.731	-	350.625	-
	Gastrocnemius	831.361	-	1251.857	-
	Soleus	2301.862	-	947.878	-
	Tibialis Anterior	1279.616	-	1402.819	-
	Knee Actuator	-	120.394	-	142.976
	Ankle Actuator	-	116.126	-	294.519
Total sum in mean fibre forces			**19,288.317**		**22,090.694**

1) Modifications to the reward function: The reward function has a significant influence on the model's behaviour, for example, giving additional rewards to the model for properly bending its hip, knee, and ankle joints resulted in a greater similarity between the model's gait and a healthy gait. Therefore, it is important to further refine the reward function, specifically the imitation term, in the following manner. First, the optimal interval for bending the hip, knee, and ankle joint were determined to be +/- 25 degrees from the optimal joint angle value. This interval is a hyper-parameter that this research chooses after running only a few trials due to time constraints. Hence, this interval needs to be further tuned as perhaps a stricter (i.e. smaller) interval may encourage the model to move its hips, knees, and ankles in a manner that is more similar to the imitation dataset.

2) Input state vector to neural network(s): The neural network(s) at time-step t receive an input comprise of two elements, the scaled observed state of the agent at time-step t and the imitation data values at time-step $t+1$. It can be argued that it is better to input the imitation data at time-step t such that the imitation values can be compared to the scaled observed state of the agent because it is in the same time-step.

3) Increasing muscle and actuator forces: The OpenSim model consists of 15 muscles and 2 actuators that can produce a maximum generalised force. Through several trials, it was observed that the prosthetic leg was unable to carry the model's weight due to the fewer muscles it encompasses. Therefore, this study experimented with increasing the maximum isometric force of the knee and ankle actuators by 33.$\dot{3}$% and 50.0% where the original maximum force of both actuators was 300 N. These changes in maximum actuators' forces resulted in no significant change of the model's performance. However, future research will investigate different values of maximum forces on both muscles and actuators in the healthy and prosthetic leg.

4) Erraticness of fibre forces: Comparable to the study by de Vree and Carloni, this research also observed an erratic pattern in the muscles actuator forces generated by both DRL algorithms during an average of 50 episodes shown in Figure 6 and Figures 7, 8, and 9 in the Appendix. Subsequently, these forces cannot be used as direct control inputs for the muscle-like actuators within the control architecture of a prosthesis. Future research will focus on reducing the erraticness of the muscles and actuator forces, possibly through penalizing such erratic patterns within the reward function (suggested by [9]).

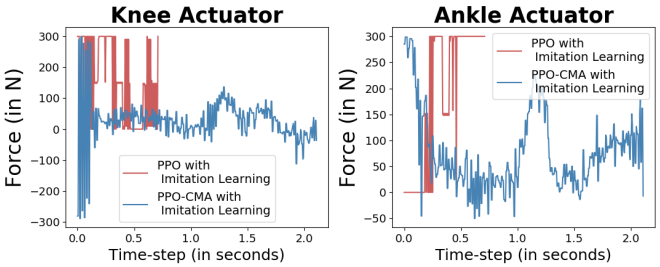

Figure 6: The actuator forces averaged over 50 episodes for the DRL algorithms: PPO+IL (red lines) and PPO-CMA+IL (blue lines).

5) Variations of PPO focused on exploration: Given all sensible modifications to the reward function and maximum muscle and actuator forces have resulted in no significant improvement of the model's performance, future research should consider a different DRL algorithm to simulate the model's forward dynamics. Specifically, different variations of the PPO algorithm with a larger focus on exploring the model's action space

should solve the issue of getting stuck in a local optimum which was experienced by the two DRL algorithms. This paper recommends investigating the 'PPO with Intrinsic Exploration Module' (IEM-PPO) algorithm proposed by Zhang et. al which was developed by applying the exploration enhancement theory to the original PPO algorithm [27]. Zhang et al. evaluated their algorithm on various environments within the MuJoCo simulator and compared it to PPO. Their results demonstrate that although IEM-PPO requires a lengthier training time, it achieves a larger cumulative reward, and has better sample efficiency, robustness, and stability.

VI. Conclusion

This research utilizes computer simulations in combination with deep reinforcement learning algorithms to study and simulate the gait patterns of a physics-based osseointegrated transfemoral musculoskeletal model with a prosthetic limb. By evaluating two DRL algorithms in their effectiveness to generate a healthy gait in the OpenSim model, this paper contributes to an expanding field of research.

The two main objectives of this study were: (1) generate a healthy gait with the two DRL algorithms, and (2) compare the two algorithms based on cumulative reward, the similarity of joint kinematics to imitation data, and muscle/actuator usage. Even though this study did not accomplish the first objective, it does examine the performance differences between the two DRL algorithms in the context of simulating a healthy gait. Compared to PPO+IL, PPO-CMA+IL received a 78.5% larger mean cumulative reward, 64.7% larger mean duration of an episode, and a decrease of 12.7% in muscle and actuator usage. In contrast, both algorithms performed similar in terms of the proximity between model's joint kinematics and the imitation data. However, given the significant differences between the mean duration of an episode for the two algorithms, the kinematics and muscles/actuator usages cannot conclusively be used as performance criteria.

Future work will focus on refining the reward function, solving the erraticness of the actuator forces and investigating the optimal maximum forces for each of the 15 muscles and 2 actuators. Given all appropriate modifications made to the current DRL algorithms do not result in the simulation of a healthy gait, different variations of PPO with a focus on deeper exploration (e.g. IEM-PPO) will be examined.

VII. Acknowledgements

The author would like to thank her supervisors, Raffaella Carloni (Professor, University of Groningen) and Vishal Raveendranathan (Doctoral Candidate, University of Groningen), for their guidance and feedback during this research. Furthermore, the author would like to thank her peers, Aurelien J.C. Adriaensses (BSc student, University of Groningen), Sarah de Boer (BSc student, University of Groningen), Robin Kock (BSc student, University of Groningen), Ruxandra Petrescu (BSc student, University of Groningen), and Milan van Wouden (BSc student, University of Groningen), for the discussions on the OpenSim model and their input with the implementation of the proposed DRL framework. Lastly, this study was funded by the European Commission's Horizon 2020 Programme as part of the MyLeg project under grant no. 780871.

References

[1] Yao Dong, Yong He, Xinyu Wu, Guangju Gao, and Wei Feng. A drl-based framework for self-balancing exoskeleton walking. In *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 469–474. IEEE, 2020.

[2] Akhil S Anand, Guoping Zhao, Hubert Roth, and Andre Seyfarth. A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 537–543. IEEE, 2019.

[3] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.

[4] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[5] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6):1–11, 2019.

[6] Vishal Raveendranathan. Simplified transfemoral amputee model for deep reinforcement learning. *Internal rewarch*, in progress.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] Luckeciano Carvalho Melo and Marcos Ricardo Omena Albuquerque Máximo. Learning humanoid

robot running skills through proximal policy optimization. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 37–42. IEEE, 2019.

[9] Leanne De Vree and Raffaella Carloni. 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*, 2021.

[10] Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. Ppo-cma: Proximal policy optimization with covariance matrix adaptation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

[11] Jonathan Camargo, Aditya Ramanathan, Will Flanagan, and Aaron Young. 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, 2021.

[12] Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.

[13] Andrii Shachykov, Oleksandr Shuliak, and Patrick Hénaff. Closed-loop central pattern generator control of human gaits in opensim simulator. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[14] Joanne Becker, Mermoz Emmanuel, and Linares Jean-Marc. Joint loading estimation method for horse forelimb high jerk locomotion: jumping. *Journal of Bionic Engineering*, 16(4):674–685, 2019.

[15] Lely Luengas-C, Esperanza Camargo, and Enrique Garzón. The effect of dynamic prosthetic alignment on the transtibial gait: Analyzing with opensim. 2020.

[16] Miguel Abreu, Luis Paulo Reis, and Nuno Lau. Learning to run faster in a humanoid robot soccer environment through reinforcement learning. In *Robot World Cup*, pages 3–15. Springer, 2019.

[17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[18] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[19] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[20] Jack M Wang, David J Fleet, and Aaron Hertzmann. Optimizing walking controllers. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–8. 2009.

[21] Seungmoon Song and Hartmut Geyer. A neural circuitry that emphasizes spinal feedback generates diverse behaviours of human locomotion. *The Journal of physiology*, 593(16):3493–3511, 2015.

[22] Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):1–11, 2013.

[23] Łukasz Kidziński, Sharada P Mohanty, Carmichael F Ong, Jennifer L Hicks, Sean F Carroll, Sergey Levine, Marcel Salathé, and Scott L Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 101–120. Springer, 2018.

[24] Archibald Vivian Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B-Biological Sciences*, 126(843):136–195, 1938.

[25] Darryl G Thelen. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *J. Biomech. Eng.*, 125(1):70–77, 2003.

[26] Yunus Ziya Arslan, Derya Karabulut, Faruk Ortes, and Marko B Popovic. Exoskeletons, exomusculatures, exosuits: dynamic modeling and simulation. *Biomechatronics*, pages 305–331, 2019.

[27] Junwei Zhang, Zhenghao Zhang, Shuai Han, and Shuai Lü. Proximal policy optimization via enhanced exploration efficiency. *arXiv preprint arXiv:2011.05525*, 2020.

Appendix

Figure 7 reports the fibre forces of the a) hip abduction, and b) hip adduction muscles of the healthy and prosthetic leg for both algorithms and is illustrated via the black lines. Figure 8 reports the fibre forces of the a) gluteus maximus, and b) iliopsoas muscles of the healthy and prosthetic leg for both algorithms and is illustrated via the black lines. The blue and red lines indicate the mean fibre force value over the time period. It can be seen that the fibre forces of the model trained with PPO-CMA+IL are closer to the mean value compared to the fibre forces of the model trained with PPO+IL. When comparing the fibre forces between the two algorithms, it is observed that the fibre forces pattern between the two deviate quite significantly. This is explained by the also significant differences in the performance between the PPO+IL model and PPO-CMA+IL model. Moreover, when comparing the fibre force pattern between the healthy and prosthetic leg, it can be seen that all muscles follow a similar pattern for both DRL algorithms.

Lastly, Figure 9 shows the fibre forces of the a) vasti, b) soleus, c) tibialis anterior, and d) biceps femoris muscles of the healthy and prosthetic leg for both DRL algorithms and is illustrated via the black lines. The blue and red lines indicate the mean fibre force value over the time period. Similarly, this figure also shows a closer proximity between the fibre forces of PPO-CMA+IL and the mean fibre force value compared to the PPO+IL algorithm.

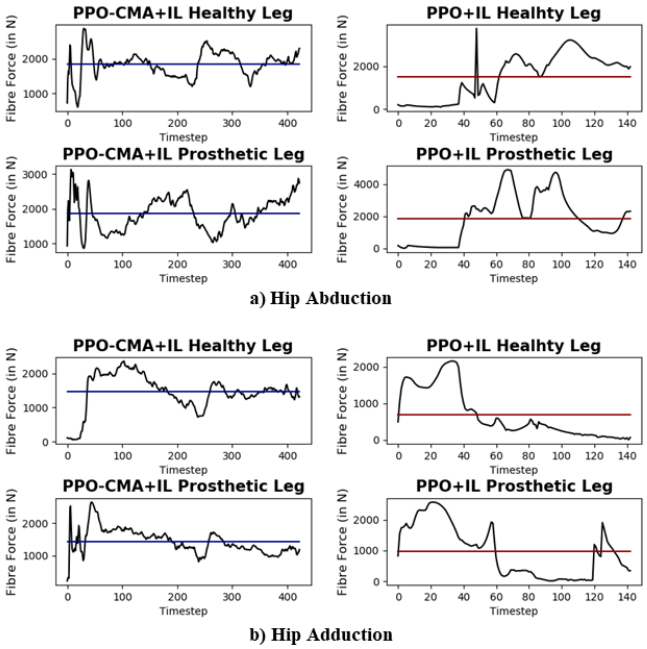

Figure 7: Fibre forces of the a) hip abduction, and b) hip adduction muscles of the healthy and prosthetic leg for both DRL algorithms is illustrated via the black lines. The blue and red lines indicate the mean fibre force value over the time period.

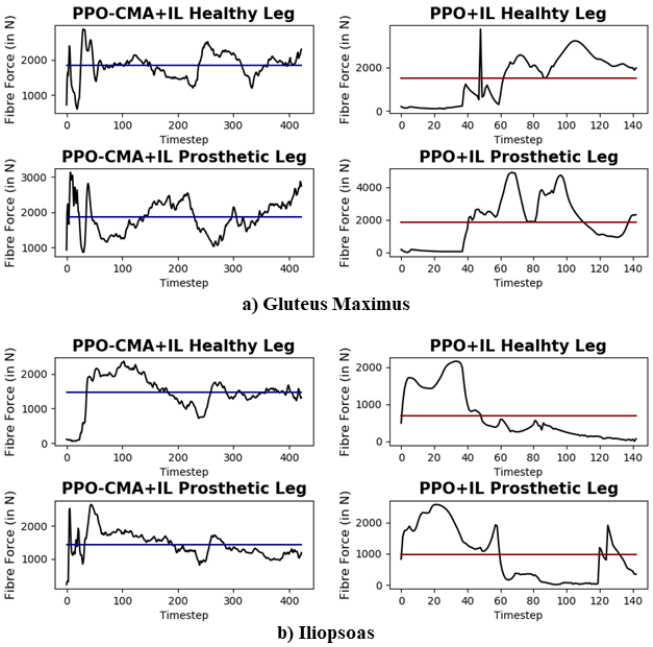

Figure 8: Fibre forces of the a) gluteus maximus, and b) iliopsoas muscles of the healthy and prosthetic leg for both DRL algorithms is illustrated via the black lines. The blue and red lines indicate the mean fibre force value over the time period.

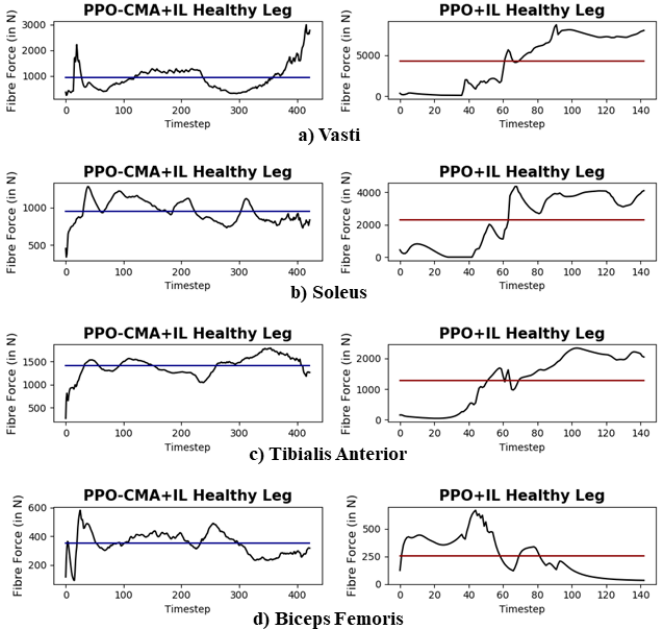

Figure 9: Fibre forces of the a) vasti, b) soleus, c) tibialis anterior, and d) biceps femoris muscles of the healthy and prosthetic leg for both DRL algorithms is illustrated via the black lines. The blue and red lines indicate the mean fibre force value over the time period.