university of groningen / faculty of science and engineering

Deep Reinforcement Learning for Physics-based Musculoskeletal Simulations of Transfemoral Prosthesis' Users during the Transition between Normal Walking and Stairs Ascending

Bachelor's Project Thesis

Ruxandra Petrescu, s3610187, r.petrescu@student.rug.nl,
Supervisors: Prof. Dr. Raffaella Carloni and Vishal Raveendranathan, MSc

Abstract: This paper proposes to use deep reinforcement learning for the simulation of a physics-based musculoskeletal model of transfemoral prostheses' users during the transition between level-ground walking and stairs ascend. The deep reinforcement learning algorithm uses the proximal policy optimization with covariance matrix adaptation and imitation learning to guarantee the level ground walking and the ascension of stairs. The optimization algorithm is implemented for the OpenSim model of transfemoral prosthesis' users. The transfemoral prosthesis has two actuators for the knee joint and the ankle joint. This study shows that the model can only take a step while normal walking, starting with the prosthesis, and it has not been able to take the second step, the transition to ascending the stairs, even with the increase of 50% of the maximum muscles' force.

1 Introduction

Computer simulations are used to study the dynamic behavior of objects or systems in response to conditions that cannot be easily or safely applied in real life (Britannica, 2021): analyzing the biomechanics of both healthy and impaired gait patterns (Geijtenbeek, van de Panne, and van der Stappen, 2013), (Ong, Geijtenbeek, Hicksand, and Delp, 2019), and to understand how assistive devices and prostheses can provide valuable support to compensate for abnormalities (Ranz, Wilken, Gajewski, and Neptune, 2017), (Harandi, Ackland, Haddara, Lizama, Graf, Galea, and Lee, 2020).

This study aims to apply deep reinforcement learning (DRL) for physics-based musculoskeletal simulations of transfemoral (above-knee) prostheses' users during the transition between level-ground walking and stairs ascent. We use a DRL Algorithm, the Proximal Policy Optimization (PPO) (Dhariwal, Hesse, Klimov, Nichol, et al., 2017), together with Covariance Matrix Adaptation (Hämäläinen, Babadi, Ma, and Lehtinen, 2020) (PPO-CMA), and imitation learning (Peng, Abbeel, Levine, and van de Panne, 2018). The imitation data used is from a public data set (Camargo, Ramanathan, Flanagan, and Young, 2021). After training, the resulting gait patterns of the prostheses' model are compared to the imitation data set to study the effects of a transfemoral prosthesis on the gait patterns and muscle's forces. Also, the comparison is used to analyze the required actuators' forces of the prosthesis.

This study is based on the work of a previous Bachelor's student, (de Vree and Carloni, 2021), where reinforcement learning has been used for physics-based musculoskeletal simulations of healthy subjects and transfemoral prostheses' users during normal walking. The previous work introduced a generic musculoskeletal model of a transfemoral amputee, without using actuators (but with a generic prosthetic device with two agonists/antagonist muscle-like actuators at the knee joint and two at the ankle joint), and the PPO algorithm with imitation learning for the transfemoral amputee and the healthy model to achieve level-ground walking.

This study proposes to use a DRL algorithm (PPO-CMA with imitation learning) to simulate the forward dynamics of an agent during the transition between normal walking and stairs ascend. The model is trained on a DRL algorithm which is based on a critic network that estimates the value of the importance of being in an action-state pair (the advantage function), and two deep neural networks (one for the policy mean and one for the policy variance) that receives an advantage function and the observed muscles' and joints' states of the agent as inputs, and outputs the mean and variance for sampling/exploring actions.

Figure 1.1 shows the diagram of the proposed DRL algorithm. The main loop describes the order of actions at each time step. The agent receives the input action and simulates it. The simulation returns the new observation state of the agent which is used in the policy training. In addition, the simulation leads to computing the reward. The joint angles and velocities from the simulation are compared to the ones in the imitation data. The reward and the observe state help train the policy.

This study aims for three main additions to the paper of de Vree and Carloni (2021).

Firstly, this paper uses a new musculoskeletal model of a transfemoral amputee with a prosthesis in the open-source software OpenSim (Delp, Anderson, Arnold, Loan, et al., 2007). The model consists of 15 muscles and two actuators to control 14 degrees of freedom. There are 11 muscles in the healthy leg (the right leg) and four muscles at the hip joint of the amputated leg (the left leg). The two actuators replace the knee and ankle joints (Internal-Research, 2021).

Secondly, this study focuses on a different activity for the model to train on. A new object is introduced in the environment, a set of stairs (each stair has a height of 11 cm). The goal is for the model to train on the transition between normal walking and ascending the stairs. To achieve this activity, an imitation data set is used and scaled for the model (Camargo et al., 2021).

Thirdly, an improvement is proposed by using PPO with Covariance Matrix Adaptation (Hämäläinen et al., 2020), a hybrid method between on-policy and off-policy Reinforcement Learning methods. PPO-CMA outputs a continuous action space, and it also performed better than PPO in eight out of nine tests made by Hämäläinen et al. (2020).

To summarize, the contributions of this paper are the following:

(1) To find and process the data used for the imitation learning

(2) To implement PPO-CMA with imitation-learning on the new activity: transition between normal-walking and stairs ascend

(3) To observe whether the model can start ascending the stairs with the healthy leg or with the prosthetic one.

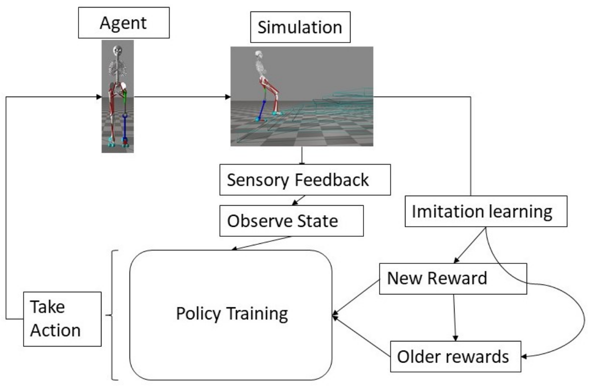

Figure 1.1: The proposed DRL algorithm for the dynamic optimization of the forward dynamic of the agent during the transition between normal level-ground walking and stairs ascend

The remainder of the paper is organized as follows. Section 2 describes the theoretical background on optimization strategies and DRL. Section 3 describes the materials used in this study. Section 4 presents the methodology and implementation of this paper, i.e., the neural network and the optimizer for training the model. The empirical results are presented and discussed in Section 5. Finally, concluding remarks are drawn in Section 6.

2 Theoretical Background

This section discusses the motivation behind the choice of using DRL with PPO-CMA and imitation learning on a model built in OpenSim in this study.

A. OpenSim

Opensim is an open-source software for modeling, simulating, controlling, and analyzing the human neuromusculoskeletal system. It performs inverse and forward dynamics, which allows for simulations of human locomotion (Delp et al., 2007). In the previous work, (de Vree and Carloni, 2021), two algorithms, PPO and PPO with imitation learning, were used on a musculoskeletal model built-in OpenSim. This study builds on the above-mentioned paper, consolidating the reason to choose the same software. The NeuroIPS 2018 AI for Prosthetics challenge's task (Łukasz Kidziński, Ong, Mohanty, Hicks, et al., 2019) was to build a controller for a musculoskeletal model to match a given time-varying velocity vector. The model used in the challenge is built in OpenSim, and from the 10 top algorithms used in the competition, four are PPO. This study has a similar goal, building a controller for a musculoskeletal model to match a data set, which motivates the choice for using OpenSim. The open-source software allows for calculating forces generated by muscles, building, manipulating, and interrogating biomechanical models, and also the changes in musculoskeletal dynamic due to human-device interaction can also be simulated (Seth, Hicks, Thomas K. Uchida, and andothers, 2018). OpenSim was also used to predict gait adaptation when weakening or contracting the plantar flexor muscle (Ong et al., 2019), a similar task to observing the gait pattern of a model using a prosthesis.

B. Transfemoral Prostheses and DRL

Previous research has shown that DRL may provide an effective tool in studying different types of prostheses, which is one of the main reasons for choosing DRL in this study. First of all, in the NeurIPS 2018 Artificial Intelligence for Prosthetics challenge, results have shown that DRL can find solutions in which the model (a transtibial amputee model with a prosthesis) learns a policy to efficiently move forward (Łukasz Kidziński et al., 2019). In addition, most of the current contributions to prosthetics research that use DRL are applications to arm prostheses. Katyal, Staley, Johannes, I., Reiter, and Burline (2016) shows the use of a neural network in combination with reinforcement to learn a policy for in-hand manipulation from raw images. Mudigonda, Agrawal, Deweese, and Malik (2018) argues that it is possible to learn robust grasp policies for anthropomorphic hands using DRL.

Secondly, the model's muscles' activations need continuous values between 0 and 1, i.e. an infinity of values, not only two. The reason for the continuous actions space comes from the fact that the muscles can be partially activated, and if the value 0 would mean inactivated, and the value 1 would mean fully activated, then our model's muscles' can receive values anywhere in between the two limits. Moreover, the same applies to the two actuators, with the exception that the values can be between -1 and 1. DRL is specialized to deal with continuous action spaces, which makes the choice of using it in this study advantageous.

Thirdly, de Vree and Carloni (2021) argues that DRL can be effectively applied to transfemoral prostheses. The results showed that DRL can generate a stable gait with a forward dynamic comparable to the healthy subjects, for the physics-based musculoskeletal model of the transfemoral prostheses' users.

C. PPO-CMA

The change from PPO to PPO-CMA from the previous work, (de Vree and Carloni, 2021), is backed up by the fact that PPO-CMA performs better than PPO in eight out of the nine RoboSchool environments tested (Hämäläinen et al., 2020).
PPO-CMA is a model-free reinforcement learning approach. It is also a hybrid method between on-policy and off-policy algorithms. The policy's mean is updated using on-policy experience, but variance update includes older off-policy experience, (Hämäläinen et al., 2020). PPO-CMA was tested on continuous control problems, training a 3D humanoid, (Roboschool, Brockman, Cheung, Pettersson, Schneider, Schulman, Tang, and Zaremba (2016)) tasks similar to the ones in this study, motivating the reason for choosing PPO-CMA (Hämäläinen et al., 2020). In addition, PPO-CMA uses an adaptation of the evolutionary algorithm Covariance Matrix Adaptation Evolution Strategy (CMA-ES) that is often used as an optimizer in robotic applications (Hansen, 2006). CMA-ES was also used in two other similar circumstances to our training goal, one for the use of torque-control of a powered ankle-foot prosthesis (Yin, Pang, Xiang, and Jing, 2018), and as an optimizer to create gait patterns for two simulation models in OpenSim,

one healthy subject and a subject with ankle weaknesses.

D. PPO-CMA with imitation learning

In this study, PPO-CMA with imitation learning is used, because the results from (de Vree and Carloni, 2021) paper show a better performance of the PPO with imitation on normal walking than PPO. Controllers trained with DRL exhibit artifacts, such as peculiar gaits, unrealistic posture, or erratic patterns in the muscles' forces which can be avoided by using an imitation data term (Peng et al., 2018). The imitation learning part in the algorithm helps when creating a reward function for natural movement. The choice of adding the imitation learning part also comes from the chosen activity.

3 Materials

A. The model

In OpenSim, a musculoskeletal model consists of rigid body segments connected by joints. Hill-type muscles connect these joints and produce forces and motion (Hill, 1938). OpenSim allows the user to observe and analyze a wide range of topics: the effects of geometry, the joint kinematics, and the properties on the muscle-tendon's forces and joint's moments. For the implementation of the DRL algorithm, this paper makes use of an adaptation of the environment in (Kidzinski, Mohanty, Ong, Hicks, Carroll, Levine, Salathé, and Delp, 2018) which provides a link between the OpenSim software and the Python programming language (www.python.org).
This study uses a new musculoskeletal model of a transfemoral amputee with a prosthesis (Internal-Research, 2021). The model has 15 muscles and two actuators to control 14 degrees of freedom. The muscles are divided into 11 for the healthy leg and four at the hip joint of the amputated leg. The two actuators replace the knee and ankle joints (concerning Figure 4.1).

The model presented contains simulated biological muscles based on a first-order dynamic Hill-type muscle model between excitation and activation (Thelen, 2003). Figure 4.2 shows the Hill-type muscle model including a contractile element (CE),

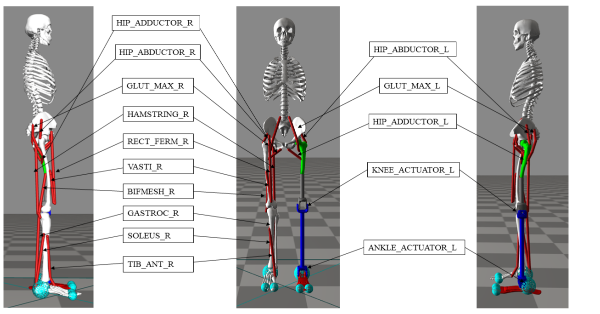

Figure 3.1: Overview of the muscles used in the models, the ones present in the healthy leg and the ones in the prosthetic leg

a parallel elastic element (PE), and a series elastic element (SE). The generated muscles force is a function of three factors: the length, the velocity, and the activation level, which can range between 0% and 100 % (in the neural network, this is translated into values ranging from 0 to 1), On the other hand, the two actuators need activations between -1 and 1. The muscle activations generate a movement as a function of muscle properties, such as the maximum isometric force, the muscle fiber length L^M, the tendon slack length L^T, the maximum contraction velocity, and the pennation angle α^M

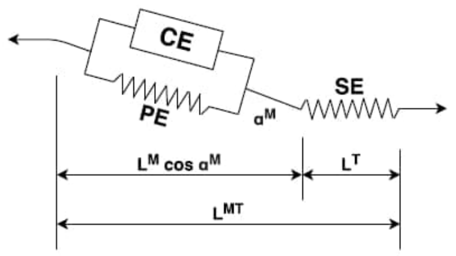

Figure 3.2: Hill-type muscle model that describes the musculo-tendon contraction mechanics in the model (Hill, 1938). It includes a contractile element (CE), a parallel elastic element (PE), and a series elastic element (SE). The elements generate a force on the tendon (Thelen, 2003)

Based on the observation of the state vector, the DRL algorithm outputs a vector of 15 muscle excitations and 2 actuator excitations. OpenSim computes the muscle activations from the excitations by using first-order dynamics equations of a Hill-type muscle model. During each time-step of 5ms the simulation: (i) computes the activations of the muscles based on the provided excitation vector; (ii) actuates the muscles; (iii) computes the torques based on the activations; (iv) computes the ground reaction forces; (v) computes the positions and the velocities of the joints and the bodies' segments; (vi) generates a new state based on the forces, velocities, and positions of the joints.

B. The stairs

The environment around the model created by Internal-Research (2021) was modified to match the new activity. A new contact geometry was built and added: the stairs. The object was constructed in blender (www.blender.org) with some final additions in MeshLab (www.meshlab.net). Each step of the stairs has a height of 11 cm and a depth of 30 cm. The top of the stairs has a length of 1 m. The stairs, the platform (the ground), the two toes (per leg) spheres and the hill (per leg) sphere are contact geometries, i.e. they can interact with each other. As a contact geometry, the stairs have specific parameters used for their elastic foundation force, presented in Table II.

Parameter	Value
Stiffness	50000000
Dissipation	5
Static friction	0.9
Dynamic friction	0.9
Viscous friction	0.9

Table II: The stairs' parameters chosen on the base of DeMers et al. (2017)

C. Data Preparation

The data used for the imitation learning part of the algorithms is from a public data set Camargo et al. (2021). The data set provides the marker data for each subject performing various activities together with the subject's OpenSim model. From the 22 subjects, the participant closest in height and weight to the model was chosen. The data was collected using markers. The stairs activities from the data set include four types of stairs of different dimensions. The data that contains the stairs with the smallest stair height is used, i.e., one data set for starting the activity with the left leg and one for starting the activity with the right leg. The marker data for both situations was uploaded to OpenSim to rotate it to match the direction our model is facing. After the rotation of the data, the model of the subject from the data set was also uploaded to OpenSim, together with the weights of each marker to perform inverse kinematics. To the inverse kinematics data, the velocities were added, and the angles measured in degrees were changed to radians. The above resulted in a CSV file to be used in imitation learning. To check if the modifications resulted in the correct data for imitation learning, the CSV file was transformed into an MOT file which was uploaded to OpenSim. An MOT file simulates a motion using an OpenSim model. The motion is identical to the one in the original data set, with the exception that the motion is rotated 90 degrees.

4 Methods and Implementation

This paper proposes to use a DRL algorithm (PPO-CMA with imitation learning) so that the musculoskeletal model of a transfemoral amputee with a prosthesis learns how to transition between walking on a flat surface at a normal speed and stairs ascending. The remainder of this Section details the components of the DRL algorithm and its implementation of the OpenSim musculoskeletal model.

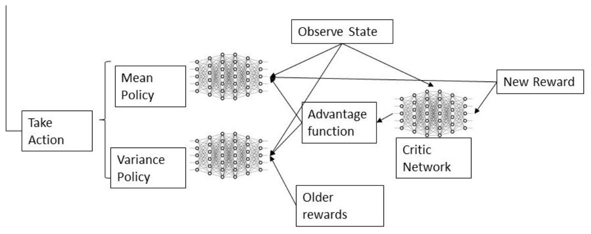

Figure 4.1: The proposed DRL algorithm (in detail) for the dynamic optimization of the forward dynamic of the agent during the transition between normal level-ground walking and stairs ascend

A. Deep Neural Networks

DRL algorithms use deep neural networks. This paper proposes the use of three multi-layer perceptrons, feed-forward artificial neural networks, the same structure as de Vree and Carloni (2021). Two neural networks are for training the policy mean and variance, and the third neural network is a critic network. This structure is used to implement the PPO-CMA algorithm Hämäläinen et al. (2020). The mean policy represents the mean of the Gaussian distribution for each output neuron. The variance policy represents the standard deviation of the Gaussian distribution for each output neuron. All three of the networks have two hidden layers with 128 neurons each. The critic network has an input layer of 89 neurons (the state dimension + 1) and an output layer of 1 neuron. The critic network computes the value-function prediction. The policy mean and variance networks have an input layer of 88 neurons and an output layer with 17 neurons. The neurons in the input layer correspond to the state of the model: 9 neurons for the pelvis position, rotation, and velocities, for each leg three neurons for the ground reaction forces, four neurons for the positions of the joints, and four neurons for the velocities of the joints, three neurons for the normalized force, length and velocities of 15 muscles, and six neurons for the force, speed, control, actuation, power and stress of each ankle and knee actuator. Table I summarizes the state variables of the agent. The output of the policy deep neural networks represents the mean and variance of the action, i.e., a 17-dimensional vector, where 15 variables represent the activation of the muscles in the model and two variables represent the activation for the knee and ankle actuators. Out of the 15 variables for the muscles activation, 11 correspond to the muscles of the healthy leg (the right leg) and four correspond to the muscles of the amputee's leg (the left leg).

	Model
Pos+Rot+Vel of the pelvis	9
Pos+Rot+Ground Reaction Forces for each leg	(4+4+3) * 2
Normalized force, length and velocities for 15 muscles	3*15
Force, speed, control, actuation, power, stress for the knee and ankle actuators	6*2
Total size of the state vector	88

Table I: The state variables of the agent

Both the state vector and action vector are continuous variables, which entails that the values they contain are neither binary nor binned to a certain distribution. The activations of the muscles have values between 0 and 1, whereas the actuators have activation values between -1 and 1. The variance network represents the balance between exploration and exploitation.

For each neuron v_i, the output y is calculated using a general output function, Leaky ReLU, i.e.:

$$y(v_i) = \underset{x \in \mathcal{X}}{\operatorname{argmax}}(\alpha * x, x) \tag{4.1}$$

where X is the set of inputs of the neurons from the previous layer. The above activation function returns x for positive input, and for a negative one, it returns a small portion of x (usually, α is 0.01). Our goal is to optimize the weights in the neural networks such that, for each given state, the deep neural network outputs the optimal mean and variance for the action to allow the agent to start walking and transition to stairs ascending.

B. The Learning Algorithms

The following subsections discuss the learning algorithms used to train the neural networks (optimizing the weights in the neural network such that the networks outputs desirable actions based on the state input).

1) Proximal Policy Optimization - Covariance Matrix Adaptation

At time t, the agent observes a state vector s_t and takes an action $a_t \rightarrow \pi_\Theta(a_t|s_t)$, where π_Θ represents the policy parameterized by Θ. As presented earlier, PPO-CMA trains two separate networks for the policy, one for the mean policy and one for the variance policy. This method is called the rank-μ update, and it is specific to the CMA-ES optimization. The rank-μ update first updates the covariance and only then updates the mean (Hansen, 2016). The effect of this method is extending the exploration distribution along with the best search directions. Executing the sampled action from the respective mean and variance results in a new state s'_t and a scalar reward r_t. The goal is to find Θ that maximizes the expected future-discounted sum of rewards $E[\Sigma_{t=0}^{\infty}\gamma^t r_t]$, where γ is a discount factor that shows the importance of long-term gains.

PPO-CMA collects experience tuples $[s_i, a_i, r_i, s'_i]$ by simulating a number of episodes in each optimization iteration.
The muscles' activations are represented as a continuous action space described by a Gaussian policy. PPO-CMA builds a policy network that outputs state-dependent mean $\mu_\Theta(s)$ and covariance $C_\Theta(s)$ for sampling the actions. The loss function, which needs to be minimized, uses a diagonal covariance matrix parameterized by a vector $c_\Theta(s) = diag(C_\Theta(s))$, (Hämäläinen et al., 2020):

$$L_\Theta = \frac{1}{M}\Sigma_{i=1}^{M} A^\pi(s_i, a_i)\Sigma_j[\frac{(a_{i.j} - \mu_{j;\Theta}(s_i))^2}{c_{j;\Theta}(s_i)} + \\ +0.5 * \log c_{j;\Theta}(s_i)] \tag{4.2}$$

where i indexes over a minibatch, j indexes over action variables, and M is the minibatch size. $A^\pi(s_i, a_i)$ denotes the advantage function, which measures the benefit of taking action a_i in state s_i. PPO-CMA maintains a history of experience over H iterations which is used in training the variance network. The mean policy is trained with the experience received from the last iteration. The critic network is also trained using the experience from the last iteration. The critic network helps compute the advantage function. The advantage function $A^\pi(s_i, a_i)$ measures the benefit of taking action a_i in state s_i. Positive advantage means that the action was better than average and minimizing the loss function will increase the probability of sampling the same action again.
Table 3.1 summarizes the hyperparameters and specifications used in the PPO-CMA learning algorithm as suggested in Hämäläinen et al. (2020).

Parameter	Value	Parameter	Value
Iterations	200-250	Adam learning	0.0003
Episodes	3200-4900	Hidden layers per neural network	2
Steps	1.500.000-1.700.000	Neurons per hidden layer	128
Training time	36-48 (hours)	Neurons in policy networks' input layer	88
Policy network(s)	2	Neurons in critic network's input layer	89
Critic network(s)	1	Neurons in policy networks' output layer	17
Activation function	Leaky ReLU	Neurons in critic network's output layer	1
Batch size	4040	History buffer size	12

Table 4.1: The hyperparamteres and specifications used in PPO-CMA learning algorithm during training

2) Proximal Policy Optimization - Covariance Matrix Adaptation with imitation learning

The reward function provides the agent with information about the value of its actions. The only information the model receives is a reward after each action. The agent's behaviors are based upon maximizing the reward. In this paper, the reward function only uses an imitation data term. The added imitation term uses experimental data (Camargo et al., 2021) to ensure that the algorithm converges to a solution and that the agent develops a natural walking pattern. For each time step, both the position and the velocity loss of the pelvis (Equation 4.5), knee, hip, and ankle joints are calculated. This is done by taking the sum of the squared error of the difference between current angles of the agent's joint the joint's angles in the data in Camargo et al. (2021) at a specific time-step (Equation 4.3). The same is done for the velocities, where their losses are calculated by the difference between the current velocities of the agent's joints and the joint's velocities in the data in Camargo et al. (2021) at specific time-steps (Equation 4.4). These losses represent a penalty, as the reward function doesn't add a positive reward, only a negative. The higher the losses, the lower the reward. This encourages the agent to keep its states as close to the ones in the data as possible. The simulation stops every time the pelvis of the model drops below 0.6 m or if the legs cross (one of the legs goes through the other). This method is called early stopping, and it is used based on Peng et al. (2018). In Peng et al. (2018) every time the actions of the model deviate too much from the imitation data, the simulation restarts. By using this method, the reward for the respective episode is cut short and the action's probability to be chosen again minimizes.

$$position\_reward_t = (pelvis\_position_t - \\ pelvis\_position\_imitation\_data_t)^2 + (ankle\_position_t - \\ ankle\_position\_imitation\_data_t)^2 + (knee\_position_t - \\ knee\_position\_imitation\_data_t)^2 + (hip\_position_t - \\ hip\_position\_imitation\_data_t)^2 \tag{4.3}$$

$$velocity\_reward_t = (pelvis\_velocity_t - pelvis\_velocity\_imitation\_data_t)^2 + (ankle\_velocity_t - ankle\_velocity\_imitation\_data_t)^2 + (knee\_velocity_t - knee\_velocity\_imitation\_data_t)^2 + (hip\_velocity_t - hip\_velocity\_imitation\_data_t)^2 \tag{4.4}$$

$$total\_reward_t = e^{-(position\_reward_t + velocity\_reward_t)} \tag{4.5}$$

C. Validation data-set

For validating the proposed DRL algorithms and, specifically, for implementing the imitation learning reward term, we use the experimental data taken from a public data-set (Camargo et al., 2021). The subjects in Camargo et al. (2021) perform each activity twice. In the imitation data, the first trial is used, and for validating purposes, the second trial is used. The data-set contains full-body motions, EMG, and ground reaction forces. This paper uses the pelvis, hip, knee, and ankle joints' angles, and velocities.

5 Results and Discussion

This Section presents the results of the DRL algorithm (PPO-CMA with imitation learning). The first part will show the results of the comparison between the simulation where the model starts the activity with the healthy leg and the simulation where the model starts the activity with the prosthetic leg. The second part will discuss the limitations and future outlook.

A. First step with the prosthetic leg

1) Normal muscles' force

Figure 5.1 shows the performance of the DRL algorithm on the trial where the model should start the activity with the prosthetic leg. The blue curve shows the reward received by the model in each episode, and the red curve shows the number of timesteps the model was "alive" (did not fall or make an illegal action) in each episode. Given that at each timestep the maximum reward the model can receive is one, the red curve also shows the maximum reward the model could have gotten at each episode. It can be noted that after 3500 episodes, the model does not learn anymore. In the first 3500 episodes, the model learns to initiate a step with the prosthetic leg however, it does not learn to lean forward and lean on the prosthesis.

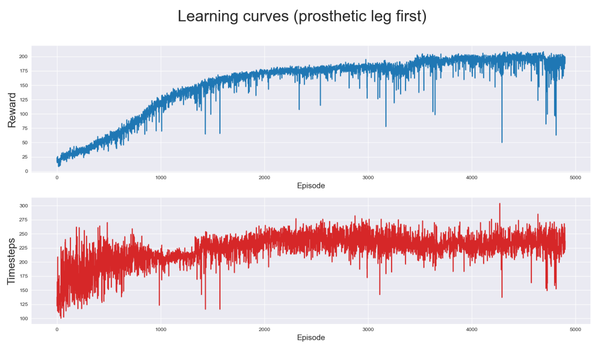

Figure 5.1: The learning curve of the DRL algorithm (PPO-CMA with imitation learning) on the task of transitioning between normal walking and stairs ascend, starting with the prosthetic leg. The graph on the top shows on the y-axis the reward and on the x-axis the number of episodes. The graph on the bottom shows on the y-axis the number of timesteps and on the x-axis the number of episodes

Because of the above-mentioned observations, we tried to increase the maximum muscles' force and the maximum actuator's force by 50%.

2) Increased muscles' force

Figure 5.2 shows the performance of the DRL algorithm on the trial where the model should start the activity with the prosthetic leg, with the muscles' maximum force increased by 50%. The blue curve shows the reward received by the model in each episode, and the red curve shows the number of timesteps the model was "alive" (did not fall or make an illegal action) in each episode. By comparing the learning curves for this trial with the trial where the muscles' forces are not increased, we can see that they follow the same trend. After 2000-2500 episodes, the model stops learning and the rewards also become unstable. The model learns to initiate a step, although more chaotically than in the trial without the increase of the muscles' force, but it does not learn how to lean forward or how to lean on the prosthesis.

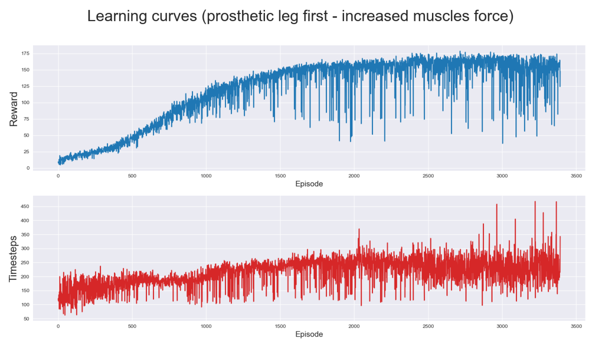

Figure 5.2: The learning curve of the DRL algorithm (PPO-CMA with imitation learning) on the task of transitioning between normal walking and stairs ascend, starting with the prosthetic leg with increased muscles force. The graph on the top shows on the y-axis the reward and on the x-axis the number of episodes. The graph on the bottom shows on the y-axis the number of timesteps and on the x-axis the number of episodes

B. First step with the healthy leg

1) Normal muscles' force

Figure 5.3 shows the performance of the DRL algorithm on the trial where the model should start the activity with the healthy leg. The blue curve shows the reward received by the model in each episode, and the red curve shows the number of timesteps the model was "alive" (did not fall or make an illegal action) in each episode. Compared to Figure 5.1 where the model should start the activity with the prosthetic leg in this trial, the learning does not stop. On the other hand, although the reward per episode is increasing, the movement of the model does not follow the imitation data. The model tries to bend the healthy knee in an attempt to lift the leg and take the first step, but it cannot move the leg in a forward direction because it cannot lean on the prosthetic leg. The reward keeps increasing because the model learns not to fall for an increasing number of timesteps, but it does not learn to imitate the data. The number of timesteps the model is alive increases so much that the model starts learning to take the second step, although it did not take the first step yet.

As mentioned for the trial where the model should start the activity with the prosthetic leg, we tried to increase the maximum muscles' force and the maximum actuator' force by 50%.

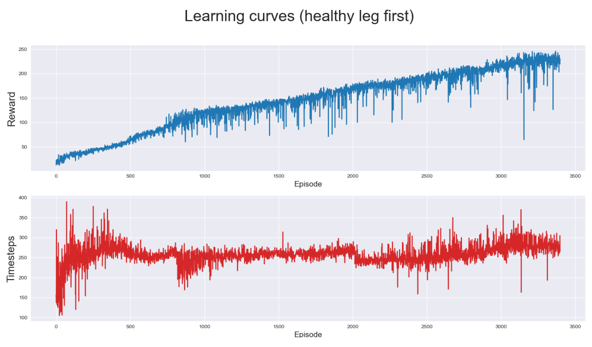

Figure 5.3: The learning curve of the DRL algorithm (PPO-CMA with imitation learning) on the task of transitioning between normal walking and stairs ascend, starting with the healthy leg. The graph on the top shows on the y-axis the reward and on the x-axis the number of episodes. The graph on the bottom shows on the y-axis the number of timesteps and on the x-axis the number of episodes

2) Increased muscles' force

Figure 5.4 shows the performance of the PPO-CMA algorithm on the trial where the model should start the activity with the healthy leg, with an increase of the muscles' maximum force by 50%. The blue curve shows the reward received by the model in each episode, and the red curve shows the number of timesteps the model was "alive" (did not fall or make an illegal action) in each episode. By comparing the learning curves for this trial with the trial where the muscles' forces are not increased, we can see that the reward per episode keeps increasing, although the model cannot take a step. The model bends its healthy knee in an attempt to lift the leg to move it forward, but because it cannot lean on the prosthetic leg, it cannot lift the leg. The reward keeps increasing because the model learns not to fall for an increasing number of timesteps, but it does not learn to imitate the data. The number of timesteps the model is alive increases so much that the model starts learning to take a second step, although it did not take the first step yet.

Compared to the trial where the muscles' forces are not increased, in this trial, the learning starts much later. For 1500 episodes, the model does not learn anything, followed by a steep increase in the reward per episode. The number of timesteps the model is alive per episode also stabilizes after 1500 episodes, but it does not increase afterward.

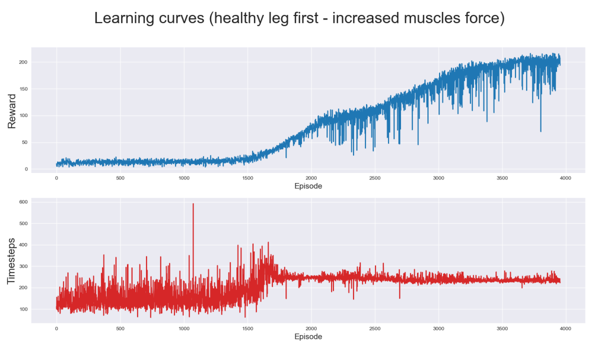

Figure 5.4: The learning curve of the DRL algorithm (PPO-CMA with imitation learning) on the task of transitioning between normal walking and stairs ascend, starting with the healthy leg with increased muscles force. The graph on the top shows on the y-axis the reward and on the x-axis the number of episodes. The graph on the bottom shows on the y-axis the number of timesteps and on the x-axis the number of episodes

C. Limitations and Future Outlook

As shown in the previous sections, deep reinforcement learning, i.e. PPO-CMA with imitation learning, is not able to generate a stable gait with a forward dynamic comparable to the healthy subjects for the physics-based musculoskeletal model of the transfemoral prosthesis users. Future research should focus on investigating how the two actuators influence the learning of the model. In de Vree and Carloni (2021), both the healthy model and the amputee model converge to a stable gait pattern, however, the model with a prosthesis used in this study cannot learn to take a step. A possible solution could be found in the activation of the muscles and actuators. When receiving an input, the muscles gradually activate towards the received input. On the other hand, the actuators jump to the received signal. The difference between how the muscles and actuators activate can lead to the reason why the model cannot take a step. Another possible solution can be found in the range of activations for the two actuators. Muscles receive activation values ranging from 0 to 1, and actuators receive activation values ranging from -1 to 1. The difference in range causes two out of the 17 output neurons of the policy networks to need twice as much training as the rest of the neurons.

Another idea for future research could focus on creating a reward function that does not use only the imitation data. This could be achieved by dividing the activity into steps. It could start by creating a reward function similar to the one in de Vree and Carloni (2021) for the first step on the ground, and then modeling a reward function for each step ascending the stairs.

6 Conclusions

By examining the usage of computer simulations to study transfemoral prostheses and gait patterns, this paper contributes to an expanding research field. Being based on a paper on deep reinforcement learning for physics-based musculoskeletal simulations of healthy subjects and transfemoral prostheses' users during normal walking (de Vree and Carloni, 2021), this paper hypothesized that similar algorithm and methods could be used for studying actuators as prostheses and also transitioning between normal walking and stairs ascending.

Testing these predictions, we observed that deep reinforcement learning, mainly PPO-CMA with imitation data, does not converge to a result for the new transfemoral model in the task of transitioning between normal walking and stairs ascending. The model using the proposed method starts learning the transitioning, but it stops at the first step, not being able to continue. Future work will focus on analyzing the actuators of the prosthetic leg and the way the activation should happen in the actuators compared to the muscles.

7 Acknowledgments

This work was founded by the European Commission's Horizon 2020 Programme as part of the project MyLeg under grant no. 780871.

The author would like to thank Milan van Wouden (BSc student, University of Groningen), Sarah de Boer (BSc student, University of Groningen), Robin Kock (BSc student, University of Groningen), Shikha Surana (BSc student, University of Groningen), and Aurelien J.C. Adriaensses (BSc student, University of Groningen) for the discussions on the OpenSim model and for the advices in the implementation of the proposed method.

References

T. Editors of Encyclopaedia Britannica. Computer simulation. encyclopedia britannica. 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, 2016.

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.

L. de Vree and R. 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 : a publication of the IEEE Engineering in Medicine and Biology Society*, 29:607–618, 2021.

Scott L. Delp, Frank C. Anderson, Allison S. Arnold, Peter Loan, et al. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.

Matthew S. DeMers, J. Hicks, and S. Delp. Preparatory co-activation of the ankle muscles may prevent ankle inversion injuries. *Journal of biomechanics*, 52:17–23, 2017.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, et al. Openai baselines. `https://github.com/openai/baselines`, 2017.

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. 32(6), 2013.

N. Hansen. The cma evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 192, pages 175–179, 2006.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016.

Vahidreza Jafari Harandi, David Charles Ackland, Raneem Haddara, L Eduardo Cofré Lizama, Mark Graf, Mary Pauline Galea, and Peter Vee Sin Lee. Gait compensatory mechanisms in unilateral transfemoral amputees. *Medical engineering amp; physics*, 77:95—106, March 2020. ISSN 1350-4533.

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.

Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. Ppo-cma: Proximal policy optimization with covariance matrix adaptation, 2020.

Internal-Research. Simplified transfemoral amputee model for deep reinforcement learning. Unpublished paper, 2021.

K. Katyal, E. Staley, M. Johannes, W. I., A. Reiter, and P. Burline. In-hand robotic manipulation via deep reinforcement learning. *Conference on Neural Information Processing Systems*, 1:1–5, 2016.

Lukasz Kidzinski, 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. *NIPS 2017 Competition Vook*, Springer, 2018.

M. Mudigonda, P. Agrawal, M. Deweese, and J. Malik. Investigating deep reinforcement learning for grasping objects with an anthropomorphic hand. *International Conference on Learning Representations*, pages 1–5, 2018.

Carmichael F. Ong, Thomas Geijtenbeek, Jennifer L. Hicksand, and Scott L. Delp. Predicting gait adaptations due to ankle plantarflexor muscle weakness and contracture using physics-based musculoskeletal simulations. *PLoS computational biology*, 15(10), 2019.

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*, 37(4), July 2018.

E. Ranz, J. Wilken, Donald A. Gajewski, and R. Neptune. The influence of limb alignment and transfemoral amputation technique on muscle capacity during gait. *Computer Methods in Biomechanics and Biomedical Engineering*, 20:1167 – 1174, 2017.

Ajay Seth, Jennifer L. Hicks, Ayman Habib Thomas K. Uchida, and Christopher L. Dembia andothers. Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Comput Biol*, 14(7), 2018.

Darryl G. Thelen. Adjustment of Muscle Mechanics Model Parameters to Simulate Dynamic Contractions in Older Adults . *Journal of Biomechanical Engineering*, 125(1):70–77, 2003.

Kaiyang Yin, Muye Pang, K. Xiang, and Chen Jing. Optimization parameters of pid controller for powered ankle-foot prosthesis based on cma evolution strategy. *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 175–179, 2018.

Łukasz Kidziński, Carmichael Ong, Sharada Prasanna Mohanty, Jennifer Hicks, et al. Artificial intelligence for prosthetics - challenge solutions, 2019.