university of groningen
faculty of science and engineering

Deep Reinforcement Learning for Physics-based Musculoskeletal Model of a Transfemoral Amputee with a Prosthesis Walking on Uneven Terrain

Bachelor's Project Thesis

Sarah de Boer, s3628701, s.de.boer.27@student.rug.nl,
Supervisors: Prof Dr R. Carloni, V. Raveendranathan, Msc

Abstract: This paper focuses on deep reinforcement learning for physics-based musculoskeletal model of a transfemoral amputee with a prosthesis walking on uneven terrain. A multilayer perceptron is used with as learning algorithm proximal policy optimization in combination with imitation learning. The imitation data, of a healthy subject walking on flat surface, is gathered from an open dataset and preprocessed to fit the model. The physics-based musculoskeletal model of a transfemoral amputee subject has a prosthesis on the left leg. This prosthesis contains two actuators, one at the knee joint and one at the ankle joint. The model is simulated using the opensource simulation software OpenSim. Two versions of the model are used, one with Coordinate actuators (OpenSim4.1) and one with Activation Coordinate actuators (OpenSim4.2). Using the deep reinforcement learning algorithm, the model is taught how to perform a human-like gait on an uneven ground. The robustness of the learning algorithm is being tested using the musculoskeletal model and the muscle activations are observed when walking on uneven terrain instead of level-ground walking. No gait pattern could be observed, hence, no conclusions could be made regarding muscle activation when walking on uneven terrain.

1 Introduction

Reinforcement learning is one of the three paradigms of machine learning, where the agent is reinforced with rewards in order to learn to perform a task [26]. A reward function is designed to give the wanted behavior a high reward and unwanted behavior a low reward. The RL agent then tries to maximize its reward using an optimization algorithm. Deep learning is a subclass of machine learning that uses multi-layered structures, also called neural networks, to learn from complex data [12]. Reinforcement learning can be combined with deep learning, called deep reinforcement learning (DRL), where the learning algorithm of the deep neural network is reinforcement learning.

Deep reinforcement learning (DRL) is becoming popular in combination with musculoskeletal models to study human gait [1], [5], [29], [24]. De Vree and Carloni [5] mention a few reasons for why DRL can be applied in transfemoral prosthesis research. First of all, in the NeurIPS 2018 Artificial Intelligence for Prosthetics challenge, participants were asked to build a controller for a transtibial amputee model with the goal of moving it forward, and were encouraged to use DRL. This provides a promising method for transfemoral amputee models. Secondly, DRL is specialized to deal with continuous action spaces. The motions performed by a musculoskeletal model of a transfemoral amputee are in a continuous action space, since muscles can activate also partly instead of fully activate or deactivate. Reinforcement learning would be unsuitable for such a large action space, but DRL can provide solutions for this. Thirdly, computer simulations in combination with DRL techniques do not require the agent to have knowledge about the environment. The agent tries actions, receives rewards and penalties and learns to perform the desired task. Therefore, little experimental data is needed to find efficient solutions. Lastly, DRL is flexible, hence being suitable for making and studying adaptations of a prosthetic device quickly. Different specifications of the device can be tested to find the optimal combination that leads to highest performance.

DRL for lower/upper limb prosthetics Katyal et al. [11] use a shallow neural network in combination with reinforcement learning to learn a policy for in-hand manipulation from raw images. Vasan et al. [28] use reinforcement learning to teach a powered prosthetic arm to perform tasks as an intact arm. This method, called learning-from-demonstration, requires experimental data for muscle activations. Mudigonda et al. [17] demonstrate that it is possible to learn robust grasp policies for anthropomorphic hands by means of DRL. De Vree and Carloni [5] use DRL to teach a physics based musculoskeletal model of a transfemoral amputee how to walk.

DRL for bipedal robots/humanoids Xie et al. [31] used a realistic model of a bipedal robot to demonstrate the effectiveness of DRL. They teach the walking controllers to imitate a reference motion with DRL. A feedback control problem is formulated as searching for an optimal imitation policy for a Markov Decision Process. Muzio et al. [18] use DRL with different optimization algorithms to teach a humanoid robot to perform the task of ball-dribbling in the RoboCup 3D Soccer Simulation domain.

DRL for other dynamic systems Unmanned aerial vehicles (UAVs) are taught how to perform a stable hovering task in a continuous state action environment using DRL [14]. Apuroop et al. [2] trained a convolutional neural network (CNN) with a long short term memory layer (LSTM) to simultaneously generate the tiling shapes and the trajectory with minimum overall cost for a modified hTrihex, a honeycomb-shaped tiling robot. Su et al. [25] proposed a DRL algorithm for satellite attitude control systems.

Different optimization algorithms for DRL Using DRL, different optimization algorithms can be utilized. Su et al. [25] based their proposed DRL algorithm on deep deterministic policy gradient (DDPG) [13]. DDPG is an actor-critic method based on the deterministic policy gradient (DPG) algorithm. Su et al. [25] use the policy network as actor network, which is used for parameterization and the value network is used as critic network, which is used for value function approximation. Apuroop et al. [2] used the actor-critic experience replay (ACER) reinforcement learning algorithm to train the CNN for the hTrihex tiling robot. ACER is an off-policy approach of A3C, which is an on-policy actor-critic algorithm [16].

Trust region policy optimization (TRPO) [22] is used by [14] and [17]. This method is a model-free, on-policy, actor-critic-based algorithm. Proximal policy optimization (PPO) [23] is based on TRPO but has a better sample complexity and is easier to implement. Xie et al. [31] use PPO to teach the walking controllers of a bipedal robot to imitate a reference motion. De Vree and Carloni [5] propose another optimization algorithm that combines PPO with imitation learning to teach a musculoskeletal model how to walk. Table 1.1 gives a summary of state-of-the-art DRL algorithms used in multiple dynamic optimization tasks.

Different models used in gait analysis using DRL Song et al. [24] give an overview of deep reinforcement learning used for modeling human locomotion control. They mention a hierarchical approach for modeling human locomotion control, which is closer to the way animals control locomotion. A lower layer generates basic motor patterns, and a higher layer modulates the basic patterns by sending commands to the lower layer [19]. Two control mechanisms make up the lower layer: reflexes [9] and central pattern generators (CPGs) [15]. Multiple lower layer control models have been proposed, either CPG-based, reflex-based or a combination of the two [24].

Terrain adaptation has been achieved by a reflex-based controller that was combined with a deep neural network (which operated as a higher layer controller) [29]. Wang et al. [29] use PPO to train the deep neural network. This deep neural network is taught how to modulate the actions the model needs to take, in order to adapt to changing terrain. They experimented with different types of terrain obstacles, such as: step obstacles, slope obstacles, wavy obstacles and a combination of those three.

In [5] physics based musculoskeletal models were used, which modeled the human skeleton with a simplification of the muscular system. Anand et al. [2019] compared a neuromusculoskeletal model

Article	Inputs	Parameters	Outputs	Objective function	Kind of validation	Application
Proximal policy optimization (PPO) [23]						
[23] [2017]	Model data coming from the humanoid	The neural network's weight vectors	Continuous action	Minimum of clipped and unclipped objective	Ability to perform benchmark test	RoboSchool humanoid
[31] [2018]	Joint angles and velocities of all joints and pelvis position, orientation, velocity and angular velocity	The neural network's weight vectors	Target joint angles for active joints	Closeness of joint angles and pelvis position to the reference	Increase in reward over time	Bipedal robot
Proximal policy optimization (PPO) [23] + imitation learning [10]						
[5] [2021]	Position and velocity of the joints and model position	The neural network's weight vectors	Continuous action	Maximize distance travelled, minimize loss	Increase in average reward over time	Transfemoral amputee simulations
Deep Deterministic Policy Gradient (DDPG) [13]						
[25] [2019]	Dynamic and kinematic data coming from the satellite	The neural network's weight vectors	Continuous action	Mean squared error of the value network and the policy network	Increase in average reward over time	Satellite attitude control system
Trust Region Policy Optimization (TRPO) [22]						
[14] [2019]	Dynamic and kinematic data from multicoper	The neural network's weight vectors	Continuous action	Minimize difference of the distance of the vehicles' current and goal positions	Increase in average reward over time	Continuous Control for Multicopter Systems
[17] [2018]	Internal position and velocity of the 25 joints of the hand and the object position	The neural network's weight vectors	Position of 16 actuators	Penalize the distance from the palm of the hand to the object	Ability to grasp objects	Anthropomorphic hand
Actor-Critic with Experience Relay (ACER) [30]						
[?] [?]	Map of uncleaned and cleaned tiles	The neural network's weight vectors	Maps generated to clean the tiles	Root mean square error is utilized	Increase in average reward over time	Area Coverage Path Planning for hTrihex Robot

Table 1.1: State-of-the-art of deep reinforcement learning algorithms for dynamic optimization.

Article	Training algorithm	Degrees of Freedom	Application task
Musculoskeletal model			
[5] [2021]	DRL, PPO + imiation learning	Healthy subject: 10 DoF, Transfemural amputee: 12 DoF	Level-ground walking
Torque-based model			
[1] [2019]	DRL, PPO + imiation learning	14 DoF	Level-ground walking
Hierarchical neuromuscular model			
[29] [2019]	Particle Swarm Optimization and DRL with PPO	8 DoF	Terrain adaptive walking

Table 1.2: Different models used in gait analysis using DRL.

with a torque based model and found that the neuromusculoskeletal model performed better when compared to human data. Different models have been used in gait research using DRL. Table 1.2 gives an overview of the papers using these different models.

This paper focuses using DRL for a transfemoral amputee model. The algorithm proposed by [5] is used and adapted to use for a transfemoral amputee model with prosthesis. Furthermore, it is investigated what happens to the muscle acitvation when walking on an uneven ground. The model of a transfemoral amputee subject is developed by Raveendranathan [20]. The prosthesis is osseointegrated into the femur of the left leg [21]. The prosthesis contains two actuators, one at the knee joint and one at the ankle joint. Two different versions of the model are used in this paper (due to different versions of OpenSim), one with Coordinate actuators (Opensim4.1) and one with Activation Coordinate actuators (Opensim4.2).

The interaction between the feet and the uneven terrain object is modeled using the elastic foundation force algorithm in OpenSim [8]. The contact material parameters used in the research of DeMers et al. [7] are used in the current research as well. The algorithm proposed by [5] is adapted to fit the goals of this study and new imitation data is processed and used in the DRL framework. An open data set from [3] is used, this dataset contains data from healthy subjects doing multiple activities, such as treadmill walking, walking up the stairs and walking up a ramp. For the current research level ground walking data is used. The results are validated on this dataset.

The main contributions of this study are:

- To use the new musculoskeletal model of a transfemoral amputee with a prosthesis in gait cycle research using Opensim.
- To test the robustness of the PPO + imitation learning algorithm using the new musculoskeletal model.
- To validate the DRL algorithm on imitation data coming from an open dataset of healthy subjects during level-ground walking.
- To analyze and evaluate muscle and actuator activations during walking on uneven terrain.

The remaining sections are organized as follows: Section 2 describes the materials used in this study, the imitation data preparation, modeling of the uneven terrain and the adaptation of the musculoskeletal model are explained. In Section 3 the DRL framework is explained, as well as the design of the reward function. The results are described and discussed in Section 4. Lastly, a conclusion is given in Section 5.

2 Materials

2.1 The amputee model with prosthesis

The physics-based osseointegrated transfemoral amputee model is developed by Raveendranathan

[20], in OpenSim. The model has 15 muscles in total and 2 actuators, with 14 degrees of freedom: 3 between the pelvis and the ground (2 translational and 1 rotational), 3 for each hip joint (2 translational and 1 rotational), 1 for lumbar extension, 1 for each knee joint and 1 for each ankle joint. The prosthesis, containing the two actuators, is osseointegrated into the femur of the left leg. In this research two different versions of the model were used. First, a model was used with Coordinate actuators (OpenSim4.1), which were implemented as the motors for the prosthetic joint. A Coordinate actuator is an ideal actuator (with instantaneous dynamics). A second version contains Activation Coordinate actuators (OpenSim4.2). These actuators had an advantage of including first order activation dynamics.

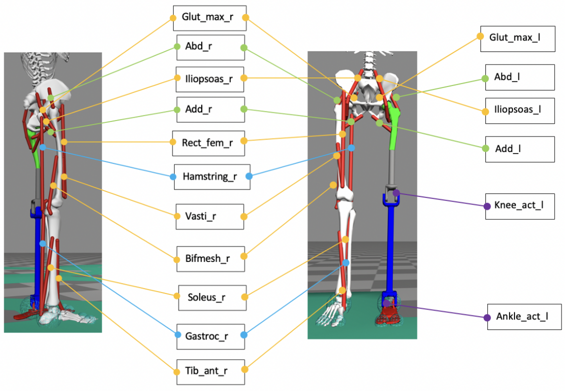

Figure 2.1: The tranfemoral amputee model with the muscle names. The yellow lines mean that the muscle is an uniarticular muscle, the blue lines mean that the muscle is a biarticular muscle, the green lines are for adduction and abduction of the hip and the purple lines are for the actuators.

The following of this subsection describes the two versions of the model.

2.1.1 Amputee model in Opensim4.1

In Figure 2.1 the different muscles and actuators the model contains can be seen. The yellow lines mean that the muscle is an uniarticular muscle, the blue lines mean that the muscle is a biarticular muscle, the green lines are for adduction and abduction of the hip and the purple lines are for the actuators. In Table 2.1 the muscles and their function can be found. Table 2.2 presents the range of motion for the joints and actuators in the model.

Muscle name	Function	Left	Right
Gluteus maximus	Hip extension	Yes	Yes
Hip abductor	Hip abduction	Yes	Yes
Iliopsoas	Hip flexion	Yes	Yes
Hip adductor	Hip addction	Yes	Yes
Rectus femur	Hip flexion, knee extension	No	Yes
Hamstring	Knee flexion, hip extension	No	Yes
Vasti	Knee extension	No	Yes
Biceps femoris	Knee flexion	No	Yes
Soleus	Ankle extension (plantarflexion)	No	Yes
Gastrocnemius	Knee flexion, ankle extension	No	Yes
Tibialis anterior	Ankle flexion (dorsiflexion)	No	Yes

Table 2.1: The 15 muscles present in the transfemoral amputee model with prosthesis and their function.

Joint	Range of motion
Pelvis tilt	[-90, 90]
Pelvis list	[-90, 90]
Pelvis rotation	[-90,90]
Hip flexion (r+l)	[-120,120]
Hip addaction (r+l)	[-45, 45]
Hip rotation (r+l)	Locked to 0
Knee angle (r+l)	[-120, 10]
Ankle angle (r+l)	[-60, 30]
Lumbar extension	Locked to -5

Table 2.2: The range of motion for the joints in the transfemoral amputee model.

The simulated biological muscles that the transfemoral amputee model contains are based on a first-order dynamic Hill-type muscle model between excitation and activation [27]. The Hill-type muscle model includes a contractile element (CE), a parallel elastic element (PE) and a series elastic element (SE), as can be seen in Figure 2.2. The generated muscle force is a function of three factors:

the fiber-length, the fiber-length-velocity, and the muscle activation level, which can range between 0% and 100%. 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.

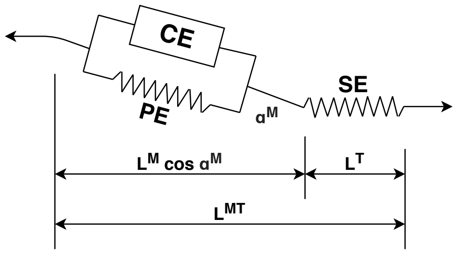

Figure 2.2: Hill-type muscle model that describes the musculo-tendon contraction mechanics in the transfemoral amputee models. It includes a contractile element (CE), and two elastic elements (one parallel and one series). The elements generate a force on the tendon [27]. Figure from [5].

The DRL algorithm outputs a vector of muscle and actuator excitations, based on the observation vector coming from the model. OpenSim calculates the muscle activations from the excitations by using first-order dynamics equations of the Hill-type muscle model. During each time-step of 5 ms, the simulation:

1. Computes the activations of the muscles based on the provided excitation vector;
2. Actuates the muscles;
3. Computes the torques based on the activations;
4. Computes the ground reaction forces;
5. Computes the positions and the velocities of the joints and the bodies segments;
6. Generates a new state based on the forces, velocities, and positions of the joints.

The Opensim4.1 version of the transfemoral amputee model contains Coordinate actuators. Namely, actuators that apply a generalized force in the direction of a generalized coordinate. This version of the model was used to test walking on the uneven terrain object.

2.1.2 Amputee model in Opensim4.2

The Opensim4.2 version of the transfemoral amputee model contains the same 15 muscles as the Opensim4.1 model (see Figure2.1). They are modeled using the Hill-type muscle model. The actuators, however, are different from the ones used in the Opensim4.1 version. Instead of Coordinate actuators, Activation Coordinate actuators are used. The Activation Coordinate actuator generates a force with first-order linear activation dynamics. This actuator has one state variable, activation, with $\dot{a} = (u - a)/\tau$, where a is activation, u is excitation, and τ is the activation time constant (set to 0.02). The default activation is set to 0.01. This version of the model was taught how to walk on flat surface.

2.2 Data preparation

The imitation data used in the learning algorithm has been retrieved from an open dataset [3]. The dataset contains data from 22 able-bodied adults, age 21 ± 3.4 yr, height 1.70 ± 0.07 m, mass 68.3 ± 10.83 kg. The subjects were instrumented unilaterally on their right side with 11 EMG (Biometrics. Ltd. Newport, UK), 3 goniometers (Biometrics. Ltd. Newport, UK), 4 six-axis inertial measurement units (Yost, Ohio, USA), and bilaterally with 32 motion capture markers following the Helen Hayes Hospital marker set (Vicon. Ltd., Oxford, UK). Ground reaction forces were recorded using force plates (Bertec, Ohio, USA). From this dataset one subject was chosen, AB06, height 1.80 m mass 74.8 kg.

The dataset contains motion capture marker data, which was used to scale the amputee model to fit to the data. The dimensions of each segment in the model are scaled so that the distances between the virtual markers (on the model) match the distances between the experimental markers (from the data). The scaling was performed using the Scale tool in the OpenSim software [6]. The motion capture markers, the pink balls in Figure 2.3, were fixed to the joints they belong to, by specifying the location and the body to which the marker is attached (i.e., which body its location is measured with respect to). Two consecutive timestamps from the data, while the subject was standing idle, were used to scale the model to fit the data.

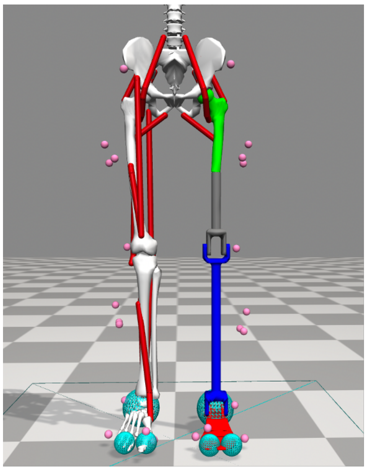

Figure 2.3: The physics-based osseointegrated transfemoral amputee model developed by Raveendranathan [20]. The red lines are the muscles, the pink balls the markers and the blue spheres on the feet are the contact meshes.

Next, a transformation of the data was needed to fit the orientation of the model. Using the marker data in the dataset and the experimental data preview tool in Opensim, the data could be rotated around the Y-axis with 270 degrees. This way the axis matched between the models orientation and the orientation of the subject in the dataset. After the rotation of the data, the joint angles were retrieved using the inverse kinematic tool in Opensim. The inputs are the scaled osim model file, experimental marker trajectories for a trial obtained from a motion capture system (obtained from the dataset), and a setup file containing the weights for the markers. The output is a motion file with the joint angles. It was ensured that the maximum marker error was below 2-4 cm and the RMS under 2 cm. Once the joint angles were gathered, the resulting motion file was converted to a comma separated values (CSV) file. Next, the velocities and accelerations were calculated, using the joint angles and added to the CSV file.

2.3 Terrain object

The amputee model with Coordinate actuators (Opensim4.1) is taught how to walk on uneven ground. Using a modeling software this terrain object was created. In Figure 2.4 the terrain object can be seen with its dimensions. The height of the peaks is 0.6 cm. The width of the object is 6 meters, the length is 8 meters and the height is 0.321 meters. The dimensions of the object are chosen to correspond to the circuit performed by the subject in the imitation data. It was chosen to let the agent start at the flat surface and first let it learn to take a step on that terrain before walking on the uneven part.

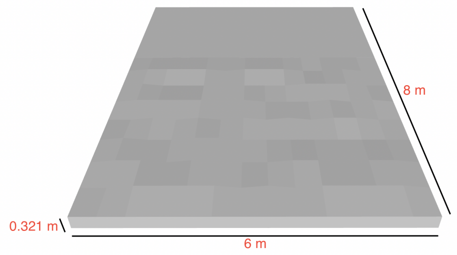

Figure 2.4: The uneven terrain object with 'peaks' of 0.6 cm and dimensions chosen to fit the circuit the subject in the imitation data is walking.

The terrain object is then added to the Opensim4.1 model. Before adding the terrain object, the interaction between the ground and the feet of the model was modeled using the Hunt Crossley force. Instead of the Hunt Crossley force, another OpenSim modeling algorithm is used to model the interaction between the feet and the new terrain object, namely the Elastic Foundation Force [8]. In order to use this, the spheres on the feet were swapped for contact meshes. The contact material parameters that were used can be seen in Table 2.3. These parameters were selected by DeMers et al. [7] to represent rubber contacting rubber, with a shoe sole thickness of 2 cm.

Stiffness	50 MPa/m
Friction coefficient	0.9
Dissipation	5 s/m

Table 2.3: Contact material parameters used for modeling the interaction between the uneven terrain object and the contact spheres on the feet of the transfemoral amputee model. These parameters were selected by DeMers et al. [7] to represent rubber contacting rubber, with a shoe sole thickness of 2 cm.

3 Methods

3.1 Deep neural network

The deep neural network used is a multi layer perceptron (MLP), which is a feedforward neural network that uses backpropagation to learn. The MLP consists of 4 layers, one input layer, one output layer and 2 hidden layers. The input to the network is the state of the model, which consists of 88 different values, described in Table 3.1. The hidden layers each consist of 312 neurons. The output is an action, muscle activation of the 15 muscles and the 2 actuators. The muscle activations are clipped to the range [0, 1] and the actuator activations are clipped to the range [-1,1]. In the policy of the multilayer perceptron, a multicategorical probability distribution is used, which results in a discrete action space. It was chosen to go with this probability distribution, because it was a main approach in existing frameworks (also the one used in [5]). Furthermore, reinforcement learning itself is quite challenging and tends to produce good results earlier with discrete action spaces. Using the multicategorical probability distribution means that in the activation vector only 0's or 1's are present. The actuators need a value in the range [-1,1]. It was chosen to increase the output vector with 2 values (resulting in 19 values). The last four values are now related to the two actuators. The negative value of the 16th element in the activation vector was added to the 17th element and given to the knee actuator. The negative value of the 18th element in the activation vector was added to the 19th element and given to the ankle actuator. This way, the first of the two elements in the output vector (at places 16 and 18), belonging to an actuator is responsible for the negative rotation of the actuator and the second element is responsible for the positive rotation of the actuator. This results in either -1's, 0's or 1's for the actuator activations.

For each neuron v_i in the neural network, the output y is calculated using a general output function, $y(v_i) = \tanh(b + \sum_{i=1}^{n} x_i w_i)$, where n is the number of $i = 1$ inputs from the previous layer, x the input to the neuron, w the weight between the current and the previous neuron, b the bias, and tanh the activation function.

Pelvis height, pitch, roll + velocities	3+6
Ground reaction forces (left + right)	3+3
Joint positions (left + right)	4+4
Joint velocities (left + right)	4+4
Muscle forces, lengths and velocities	15+15+15
Actuators force, speed, control, actuation, power and stress	6+6
Total size of observation vector	**88**

Table 3.1: The variables in the observation vector of the model.

The goal of the framework is to optimize the weights such that the gait of the model is close to the human gait. In Figure 3.1 an overview of the framework can be seen.

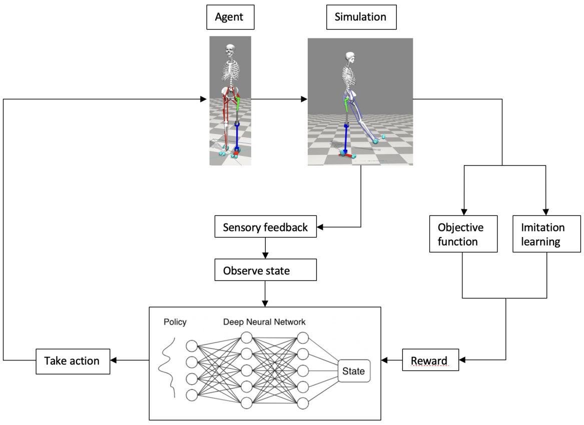

Figure 3.1: Overview of the DRL framework. The agent's state and received reward are fed into the deep neural network and a new policy with an action is outputted and given back to the agent. The reward term consists of two parts, the objective function and the imitation learning.

3.2 Learning algorithm

In order for the deep neural network to learn, a learning algorithm is needed. In the current research reinforcement learning (RL) is used. The RL agent is given rewards after an action has been taken. The task is then to maximize the reward and therefore learn the wanted behaviour (a human-like gait pattern). The optimization strategy used in this paper is proximal policy optimization (PPO) with the addition of imitation learning [5]. PPO has been proposed by Schulman et al. [2017] and has some of the benefits of trust region policy optimization (TRPO) [22], but is said to be simpler to implement, and have a better sample complexity. PPO has the following main objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{3.1}$$

where $\mathbb{E}$ is the expected value, $\hat{A}^t$ is the advantage estimation, i.e., the difference between the expected and the real reward from an action, and ϵ is the clip value, set to 0.2. $r_t(\theta)$ denotes the probability ratio: $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, where θ and θ_{old} are the new and old parameters. π_θ is the policy corresponding to the parameters θ, and a_t and s_t are the action and state vectors at timestep t respectively. The objective function of PPO differs from TRPO in such a way that it clips the probability ratio to stay within a certain region, ensuring that the policy updates are not too large. Using the objective function of PPO the weights of the deep neural network are updated to ensure that actions with a low reward have a low probability of being chosen again and actions with a high reward have a high probability of being chosen again. The clipping in the objective function ensures that the updates are not too large.

Table 3.2 summarizes the hyperparameters and specifications used in the PPO learning algorithm, these values are taken from [5]. The only hyperparameter used to experiment with in this paper was the hidden layer size, but after some trials it was chosen to stick to 312 neurons, because decreasing the hidden layer size did not result in a better learning curve. δ controls the size of the allowed policy updates (i.e., the Kullback-Leiber divergence should be smaller than δ) and γ is the discount factor ($0 \leq \gamma \leq 1$), i.e., a meta-parameter that determines to what extent the agent considers future rewards.

Parameter	Value	Parameter	Value
Parameter noise	Yes	PPO clip parameter (ϵ)	0.2
PPO batch size	512	PPO optimiz. per epoch	4
PPO entropy coefficient	0.01	PPO δ	0.9
PPO γ	0.999	Number hidden layers	2
Entropy coeff.	0.01	Hidden layer size	312
Policy network(s)	1	Activation function	tanh

Table 3.2: Summary of the hyperparameters and specifications used in the proximal policy optimization algorithm, adapted to fit the current research from [5].

3.3 Reward function

The reward received by the model after taking a certain action consists of two parts, the goal reward and the imitation reward (as can also be seen in Figure 3.1). The first part, Equation 3.2, is designed to let the agent walk for multiple steps and to keep the agent 'alive' (meaning to stay with the pelvis above a certain height).

$$Reward_{goal,t} = \sum_t Reward_{distance} + \sum_t Reward_{alive} \tag{3.2}$$

The second part of the reward is the imitation reward, Equation 3.3. This reward has two different aspects, the first one is the position reward, how close are the joint angular positions to the imitation data. The second part are the velocities, how close are the joint angular velocities to the imitation data.

$$Reward_{imitation,t} = \sum_t Reward_{position,t} + \sum_t Reward_{velocity,t} \tag{3.3}$$

The position and velocity losses are calculated using the following loss function: $L = (x_{observation} - x_{data})^2$, where x can be position or velocity of the ankle, knee or hip joints. The reward term is then determined using: $R = e^{-\alpha \cdot L}$, where α is a weight constant and L is the loss. The total reward is then calculated by combining the imitation reward and the goal reward:

$$Reward_t = 0.6 \cdot Reward_{imitation,t} + 0.4 \cdot Reward_{goal,t} \tag{3.4}$$

The 60% and 40% terms have been chosen regarding the research done by De Vree and Carloni [5]. In their work they investigated the use of these scaling terms and found that this distribution showed the best results. It was not further investigated in the current research.

4 Results & Discussion

In this section the results of the simulation will be presented. First, the DRL algorithms performance will be discussed. Next, the limitations and some possibilities for future work will be discussed.

4.1 Algorithms performance

No gait pattern has been observed with both models. The average reward per episode for both versions of the model are plotted in Figure 4.1 (red line for the version with Coordinate actuators and green line for the version with Activation Coordinate actuators). It can be seen that there is not a real increase in reward over time for both models, suggesting little to no learning. In Table 4.1 the mean total reward and standard deviation of the simulations with both models are presented.

Model	**Mean**	**Standard deviation**
Coordinate actuators	55	9,6587
Activation Coordinate actuators	56,6	9,6293

Table 4.1: Mean total reward and the corresponding standard deviation of both models, one with Coordinator actuators and one with Activation Coordinator actuators.

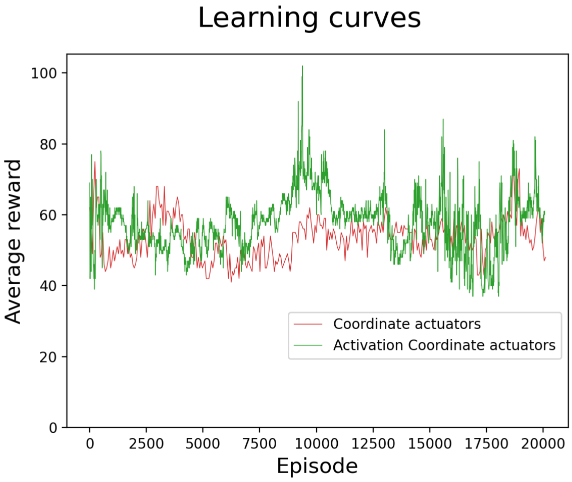

Figure 4.1: Mean rewards for the final runs of the Coordinate actuator model (red line) and the Activation Coordinate actuator model (green line).

4.2 Limitations and Future Outlook

The algorithm was not able to generate a gait pattern for both models. There could be different reasons for this, which will be discussed in this section. Future research is needed to investigate the different modifications that could be done in order to possibly generate a gait pattern.

Multicategorical probability distribution Firstly, in the proposed algorithm by De Vree and Carloni [5] a multicategorical probability distribution is used. This choice makes the action space discrete, which decreases the size of the action space. The multicategorical probability distribution generates either 1s or 0s and gives these to the action vector. The action vector, an activation value for each muscle and actuator, is then given to the model and an action is performed. The muscles have no difficulty with the binary input, since the opensim software ensures that the discrete values are transformed into continuous values. Next to that, as described in Section 2, the muscles are modeled using the Hill-type muscle model. This ensures that there is a delay between the excitation being given to the muscle and an activation coming out, which is not an on-off function. However, the Coordinate actuators in the transfemoral amputee model

(Opensim4.1) do not have this delay function and are rather modeled as a step function, meaning that an input of either 0 or 1 results directly in full activation or de-activation. The version with Activation Coordinate actuators (Opensim4.2) resolved this problem as there is an activation function involved (as described in Section 2). With the Coordinate actuator version it was investigated what happened if the multicategorical probability distribution was swapped for a Gaussian probability distribution, because values in any range can be produced by this distribution. This approach resulted in exploding gradients, which was likely caused by the enormous action space. PPO was found to not work well with large continuous action spaces, as the gradients become unstable [4]. One approach for future research could be to change the PPO algorithm to an algorithm which is more stable using large continuous action spaces in combination with the new version containing the Activation Coordinate actuators.

Actuator activations Secondly, while the muscles need an excitation value in the range of [0,1], the actuators need an excitation in the range [-1,1]. As described in the previous paragraph, a multicategorical probability distribution was used. This distribution generates either 0s or 1s and not -1. It was chosen to increase the size of the action vector by 2 values. Then, after the distribution generated 19 values, all either 0 or 1, the negative value of the 16th element was added to the 17th element and the negative value of the 18th element was added to the 19th element as well. This way the two values for the actuators could either be -1, 0 or 1. This could be another limitation of the study, as the DRL algortihm now learns 19 values instead of 17. And it might not understand that the last 4 values are for the actuators. Switching to the Gaussian probability distribution fixes this limitation as now only 17 values are needed, but the same problem of the exploding gradients arises. More research is needed in this area to identify a suitable algorithm.

Maximum muscle forces Lastly, since the left leg of the transfemoral amputee model contains less muscles and instead 2 actuators, it might be the case that the maximum muscle force needs to be increased. In this study the maximum muscle fiber forces of all muscles have been increased by 50% compared to the original model by Raveendranathan [20]. Next to that the maximum isometric force of the 2 actuators was also slightly increased. Due to time limitations, it was not further investigated if and which increase had the most effect. This could be investigated further in future research.

5 Conclusion

By using a physics-based osseointegrated transfemural amputee model one step further in the transfemural prosthetics research has been achieved. Unfortunately, no gait pattern has been observed. Nonetheless, great steps towards a walking model have been conducted. Data has been processed and made useable for DRL research with imitation data. The Elastic Foundation Force has been implemented in a model to model the interaction between the feet and the uneven terrain object. A new version of the model with Activation Coordinate actuators (Opensim4.2) has been used and has a promising theory, but should be investigated more. No conclusions can be made about what caused the inability of the models to walk, but suggestions have been made for future research.

Due to the inability of the models to perform an emerging gait, no conclusions can be made regarding the robustness of the algorithm proposed by [5]. It might be the case that modifications need to be made to the algorithm regarding the probability distribution, but there are no certainties relating this. Furthermore, no conclusions can be made regarding the muscle activations when walking on uneven terrain.

6 Acknowledgements

The author would like to thank her supervisors, Raffaella Carloni (Professor, University of Groningen) and Vishal Raveendranathan (Doctoral candidate, University of Groningen) for the supervision and help during the project. Next, the author would like to thank Aurelien J.C. Adriaenssens (BSc student, University of Groningen) for the discussions and help with the Elastic Foundation algorithm and the code and her fellow Bsc students (University of Groningen), Milan van Wouden, Shikha Surana, Ruxandra Petrescu and Robin Kock for the discussions on the models and algorithms during the project. This work was funded by the European Commission's Horizon 2020 Programme as part of project MyLeg under grant no. 780871.

References

[1] A. S. Anand, G. Zhao, H. Roth, and A. Seyfarth. A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model. *IEEE-RAS International Conference on Humanoid Robots*, pages 537–543, 2019.

[2] K. G. S. Apuroop, A. V. Le, M. R. Elara, and B. J. Sheu. Reinforcement learning-based complete area coverage path planning for modified htrihex robot. *Sensors*, 21:1067–1087, 2021.

[3] J. Camargo, A. Ramanathan, W. Flanagan, and A. 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.

[4] C. Ching-Yun Hsu, C. Mendler-Dnner, and M. Hardt. Revisiting design choices in proximal policy optimization. *pre-print*, 2009. arXiv:2009.10897.

[5] 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*, 29:607–618, 2021.

[6] S. Delp, F. Anderson, A. Arnold, P. Loan, A. Habib, C. John, E. Guendelman, and D. Thelan. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedial Engineering*, 55:1940–50, 2007.

[7] M. S. DeMers, J. L. Hicks, and S. L. Delp. Preparatory co-activation of the ankle muscles may prevent ankle inversion injuries. *Journal of Biomechanics*, 52:17–23, 2017.

[8] M. W. Hast, B. G. Hansona, and J. R. Baxter. Simulating contact using the elastic foundation algorithm in opensim. *Journal of Biomechanics*, 82:392–396, 2019.

[9] H. Hultborn. Spinal reflexes, mechanisms and concepts: From eccles to lundberg and beyond. *Progress in Neurobiology*, 78:215–232, 2006.

[10] A. Hussein, M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50, 2017.

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

[12] J. D. Kelleher. *Deep Learning.* MIT Press, 2019.

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

[14] A. Manukyan, M. A. Olivares-Mendez, M. Geist, and H. Voos. Deep reinforcement learning-based continuous control for multicopter systems. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1876–1881, 2019.

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

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016.

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

[18] A. F. V. Muzio, M. R. O. A. Maximo, and T. Yoneyama. Deep reinforcement learning for humanoid robot dribbling. *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6, 2020.

[19] G. Orlovsky, T. G. Deliagina, and S. Grillner. *Neuronal Control of Locomotion: From Mollusc to Man.* Oxford University Press, 1999.

[20] V. Raveendranathan. Simplified transfemoral amputee model for deep reinforcement learning. *Internal research*, in progress.

[21] V. Raveendranathan and r. Carloni. Musculoskeletal model of an osseointegrated transfemoral amputee in opensim. In *8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob).*

[22] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *pre-print*, 2015. arXiv:1502.05477.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization. *pre-print*, 2017. arXiv:1707.06347.

[24] S. Song, L. Kidzinski, X. Bin Peng, C. Ong, J. Hicks, S. Levine, C. G. Atkeson, and S. L. Delp. Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Preprint available at bioRxiv*, 2020.

[25] R. Su, F. Wu, and J. Zhao. Deep reinforcement learning method based on ddpg with simulated annealing for satellite attitude control system. In *2019 Chinese Automation Congress (CAC)*, pages 390–395, 2019.

[26] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[27] D. Thelen. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of Biomedical Engineering*, 125:70–77, 2003.

[28] G. Vasan and P. Pilarski. Learning from demonstratio: Teaching a myoelectric prosthesis with an intact limb via reinforcement learning. In *IEEE International Conference on Rehabilitation Robotics*, pages 1457–1464, 2017.

[29] J. Wang, W. Qin, and L. Sun. Terrain adaptive walking of biped neuromuscular virtual human using deep reinforcement learning. *IEEE Access*, 7:92465–92475, 2019.

[30] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *pre-print*, 2017. arXiv:1611.01224.

[31] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246, 2018.