university of groningen / faculty of science and engineering

Testing For Generality Of A Proximal Policy Optimiser For Advanced Human Locomotion Beyond Walking

Bachelor's Project Thesis

Aurélien Adriaenssens, s2946661, a.j.c.adriaenssens@student.rug.nl,
Supervisors: Prof. dr. Raffaella Carloni and Vishal Raveendranathan, MSc

Abstract: Computer simulations have become a great aid in designing and testing new prosthesis before deployment in the real world. Being able to simulate for the human walking gait has shown time and cost saving to prosthesis development. With the creativity and dynamic environments of simulations, a multitude of environments can be tested upon. This paper focuses on determining whether a currently existing Proximal Policy Optimiser with Imitation Learning is able to be generalised to a variety of human locomotion beyond just walking. The environments tested on are: fast paced walking, ascending and descending a ramp, and ascending and descending stairs. Two musculoskeletal models are used; one healthy, and one transfemoral amputee. Both models are subjected to the environments. The forces exerted, joint angle, and rewards earned of the transfemoral amputee are compared to the healthy model to validate the research. Both models are able to make progress in the environments, exhibiting greater that 50% likeliness on the training data. It is suggested that generalising the Proximal Policy Optimiser for more advanced scenarios is indeed possible.

1 Introduction

Biomechanics and artificial intelligence have seen a huge leap in progress with improving movement for humans with impaired or missing limbs (Bartlett, 2006). Computer simulations of assisted gaits with prosthesis (Ong et al., 2019) are entering the domain of cheap and efficient solutions for determining the capabilities of prostheses before real-world deployment.

Learning to walk is the first step to greater mobility, but humans live a dynamic life. Losing a limb greatly impacts future abilities to move, but with the help of a prosthesis it can be re-learned to pass obstacles such as stairs (Koganezawa et al., 1987) and to walk on inclines (Nickel, 2014). The addition of a prosthesis is shown to improve the capabilities of amputees over stairs and sloped environments. Creating these prosthesis for advanced human gaits in reality is a continuous back-and-forth process between amputee and researcher. Adding the element of simulation allows the prosthesis to be tested with in a simulated environment before being physically constructed, saving on time and money (Lambrecht et al., 2011).

Research on unilateral transtibial amputee subjects have built a substantial ground work on progressing the design of prosthesis. Simulations have been used as starting points for the development and the control of active transtibial prosthesis (LaPrè and Sup, 2011; LaPrè et al., 2014). Whole lower-limb modelling and dynamics have become more feasible with the increase in software power and material technology. Humans with a transfermoral (above knee) amputation have been getting traction with new and novel designs (Ege and Kucuk, 2019), benefiting from the quicker and cheaper testing phases before deployment. Going from theory to simulating the human gait of a transfemoral amputee with a prosthesis is advantageous before physical testing (Azimi et al., 2019).

With the state of modern technology and advanced algorithms, methods of Deep Reinforcement Learning (DRL) have shown fruitful in simulating a variety of physics-based human movements (Peng et al., 2018). The current state of DRL uses models which are simplified versions of the human body. They do not consider the set of muscles required to produce the flexion or extension torque at a joint. This research will make use of those muscle to con-

tribute a more biologically accurate representation of human locomotion. This study builds upon our previous work (De Vree and Carloni, 2021). Carrying on from the DRL and Proximal Policy Optimisation (PPO) with Imitation Learning algorithms, advanced environments will be tested in this paper to determine how generalised the algorithms are.

It is not possible to predict all environments that a human would walk about in. The advanced environments that this research explores are:

- Fast paced walking.
- ascent of a ramp.
- ascent of stairs.

Each environment will make use of two musculoskeletal models. The first musculoskeletal model is based on the model by Kidziński et al. (2018). Their model is a representation of human anatomy from the hip down, and it is used as the healthy model for this research. The model has 18 muscles controlling 10 degrees of freedom (DOF). The second model is an edited version of the first model. This model is edited in such a way to accurately represent the deficiencies present in a human with a transfemoral amputation. This transfemoral amputee model has 19 muscles controlling 14 degrees of freedom. The model use 15 muscles of the model's original 18 muscles. The complete change to the muscles is: 3 removed from the right leg, and 2 added to each side of the hip. The addition of the 4 hip muscles add 1 DOF each. The models will be loaded in the simulation software OpenSim. OpenSim helps with the the study of musculoskeletal phsyics and biology (Delp et al., 2007).

This research is carried out in two tasks. The first task is creating the environments in OpenSim that the models will be subjected to. In this task, the data from the freely available online repository CMU Graphics Lab (Carnegie Mellon university, n.d.) is used to craft the environments. In the second task, the healthy model and the transfemoral amputee model are subjected to the environment. Analysing the rate of learning and angle gaits of the musculoskeletal models of the simulations will determine the effectiveness of the PPO. Only the data for the healthy model will be reported in this research, see Section 3.5 for more detail.

The research question addressed by this paper is:

Is the current Proximal Policy Optimiser generalisible to new environments?

The paper is structured as follows. Section 2 will describe the methodology used to answer the question. The algorithms will be broken down and explained in this section. Section 3 will go through the experimental setup to carry out the simulations. The software, models, meshes/geometry, and environments will be described in detail here. Then Section 4 will show the results gathered from the experiment. The results will be analysed, presented and discussed. Lastly, section 5 will conclude this paper. Future thoughts and overall notes will be discussed in this section.

2 Method

The Deep Reinforcement Learning network in this paper is a Multi-layer Perceptron (MLP). This will be the core neural network that will have the input nodes, weights, hidden layers, and output nodes. The input for the MLP is the current state of the musculoskeletal model within the environment. The output is a vector containing the activation for each muscle within the musculoskeletal model.

Through out the experiment, the Proximal Policy Optimisation (PPO) learning algorithm is used. The PPO has been shown to work for simple walking exercises. Editing the way the PPO learns could make the simulation better adapt to a more general or advanced human walking behaviours. Changes would be made with regards to how much weight a reward is given with respect the task.

2.1 Multi-layer Perceptron

The MLP used is a feed forward neural network. The structure is shown in Figure 2.1. It has 4 distinct layers: input (214 or 218 nodes), 2x hidden (312 nodes), and output (18 or 19 nodes). The MLP's activation function is the tanh-nonlinear activation function as shown in Equation 2.1.

$$y(v_i) = tanh\left(b + \sum_{i=1}^{n} x_i w_i\right) \tag{2.1}$$

The weight of the connection between two nodes is w, the activation of the previous node is x, the

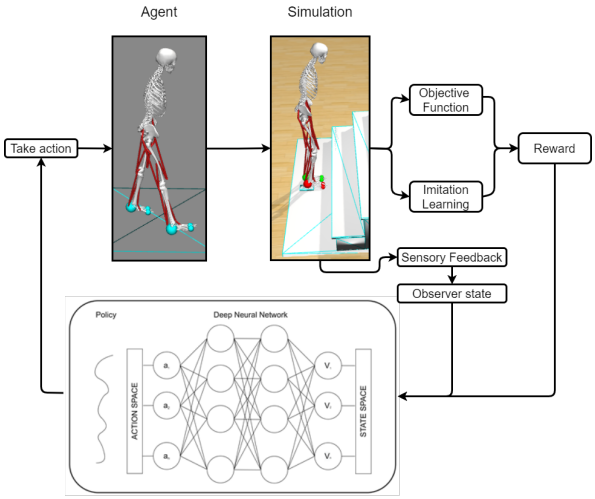

Figure 2.1: The DRL alogrithm used in the experiment. The state space feeds into the input layer with either 214 or 218 nodes, shown on the right. The output layer is the action space with either 18 or 19 nodes.

bias is b, and n is the number of nodes in the previous layer. The sum is for each node i that is connected to the current node. This calculates the output y of the node v_i. The weights of the multi-layer perceptron are optimised by the PPO.

Each layer in the MLP is fully connected, and the connections are feed forward only. The input for the MLP is the current state of the environment. The list of everything that is measured in the environment is: joint angles, muscles fibre lengths, muscle velocities, tendon forces, positions, velocities and accelerations of joint angles and body segments. This yields in input space of 214 variables for the healthy subject model, and 218 variables for the tranfemoral amputee model. The output of the MLP is a vector. The healthy model has a vector of length 18, whereas the transfemoral amputee model has a vector length of 19. The vector represents the activation of each muscle present in the model. The input values of the MLP are continuous values of the input space. The output is binary, either 1 or 0. The output is interpreted with the given mechanism inside in OpenSim called *the brain*, this activates the muscles.

2.2 The PPO policy

The PPO algorithm is derived by OpenAI (Schulman et al., 2017). The goal of the PPO is to train and optimise the weights within the network. The algorithm for how the policies get updated can been seen in Algorithm 2.1.

Algorithm 2.1 PPO overview

Require: $maxSteps >>> 1,000,000$
  $t \Leftarrow 0$
  $\epsilon \Leftarrow 0.2$
  $p_n, p_o \Leftarrow policy()$
  while $TRUE$ **do**
    if $t > maxSteps$ **then**
      break
    end if
    $p_o \Leftarrow p_n$
    $r_t\theta \Leftarrow p_n/p_o$
    $\hat{A}_1 ... \hat{A}_t\theta \Leftarrow$ compute advantage estimates
    $p_n \Leftarrow L^{clip}(r_t\theta,\ \hat{A}_1 ... \hat{A}_t,\ \epsilon)$
    $t \Leftarrow t + 1$
  end while
  return p_n

The algorithm uses the current and previous policy to update to a new policy. This will iterate for a predetermined amount of time. The probability ratio between the old and the new policy is calculated as:.

$$r_t(\theta) = \frac{\pi_\theta(\alpha_t|s_t)}{\pi_{\theta_{old}}(\alpha_t|s_t)} \tag{2.2}$$

The actions at timestep t is α_t, and the observations at timestep t is s_t. The old and new policy are represented by π_θ and $\pi_{\theta_{old}}$ respectively.

The function $L^{clip}()$ is what governs a policy. It ensure that policy updates cannot be too large. This clips the probability ratio and adds a Kullback-Leiber divergence term. This can be seen in Equation 2.3.

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.3}$$

The policy parameter is θ, $\hat{E}_t$ is the empirical expectation over timesteps, r_t denotes the ratio of the probability under the new and old policy at time t, the estimate advantage is $\hat{A}_t$ at time t, and

the hyperparameter is ϵ which is set to 0.2. Having the term, $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, in the policy will negate the incentive for the rate r_t to move outside of the clip-bound $[1-\epsilon, 1+\epsilon]$ set by ϵ.

2.3 Reward function

The action generated by the model results in it gaining a reward, this reward is mapped from a function. The reward can be either positive or negative relative to the actions the model just took. If the model does a favourable action, it gains a positive reward. If the model does an unfavourable action, it gains a negative reward (a penalty). The goal of the model is to receive the most rewards possible. Since the reward function gives the model a reward, it has a heavy impact on the model's behaviour. The actions that the reward function uses to decide how much reward the model earns are designed with respect to the task at hand.

Starting from the basis of a reward function for walking, as seen in Equation 2.4, the four main elements can be reused. The 4 elements the reward functions is reliant on are: distance covered, deviation to desired velocity, muscle cost, and time spent alive.

$$J(\pi) = \sum_t (r_{distance} - p_{velocity} - p_{costs}) + \sum_t (r_{alive}) \tag{2.4}$$

The total reward $J(\pi)$ is calculate at each time step t. The reward based on distance is $r_{distance}$, this is the distance that the model's pelvis travelled since the last timestep. A larger distance gives a larger reward. A penalty for deviating too far from the desired velocity of 1.25 m/s is given to the model, this is represented as $p_{velocity}$. A penalty is also given for the amount of energy that the model is using, denoted by p_{costs}. Lastly, the reward for the amount of timesteps spent alive is given, this is r_{alive}. The model is considered alive as long the pelvis is 0.7 meters above ground level. The reward function calculates the reward that the musculoskeletal model earns for carrying out its action at every timestep t. This reward function is formulated in a way that earning the highest positive number is the goal of the model.

Faster paced walking increases the reward for every time the model travels the same distance in a shorter amount of time. The ramp environment would require a reward for when the model travels in the vertical direction alongside horizontally. The stairs environment would also have to balance the reward for vertical to horizontal movement, but with more emphasis on the vertical movement.

It is advantageous to manipulate the reward function to better benefit the new environments due to the way a reward is determined when going from basic walking to more advanced environments. The environment for fast paced walking has a desired velocity higher than 1.25 m/s but not exceeding 2 m/s. The penalty for the velocity is adapted to this new environment. The environments for tackling the obstacles now has the model travelling in the vertical direction as well as horizontal direction. The reward for distance travelled is split into the x-axis and y-axis. The new environments are also more demanding and challenging. The penalty for the amount of energy used is reduced significantly. The reward for staying alive remains the same. The new reward function is shown in Equation 2.5.

$$\begin{aligned} J(\pi) = &\sum_t (r_{d_x} + r_{d_y} - 0.8 * p_{velocity} - 0.6 * p_{costs}) \\ &+ \sum_t (r_{alive}) \end{aligned} \tag{2.5}$$

The reward for the distance covered is changed in the new reward function. The previous $r_{distance}$ is spilt into to terms; r_{d_x} and r_{d_y} for both the horizontal and vertical distance respectively. The velocity penalty $p_{velocity}$ is now weighted by a factor of 0.8, reducing the punishment to deviate from 1.25 m/s. The penalty for energy cost p_{costs} is now weighted by a factor of 0.6. This allows the musculoskeletal model to exert more energy to pass the advanced environments without being punished.

3 Experiment

Two distinct musculoskeletal models will be used as our human biological analogy, each model represents the human's muscles from the hip down. The first model is the healthy model. This model is an accurate representation of the human muscles, with both legs a mirror of each other. The second model is the transfemoral amputee model. This model will

have muscles removed in only the right leg, simulating a transfemoral amputation.

There will be a total of 5 advanced environments that each model will be subjected to in this experiment. The environments are related to 5 activities of daily living: fast paced walking, ramp ascent, ramp descent, stairs ascent, and stairs descent. Each environment is setup that only the feet contact and exert force on the obstacle and with nothing else.

3.1 Models and software

The healthy and the transfemoral amputee model are used to carry out the experiment. The healthy model will be used as a baseline, whereas the transfermoral amputee model will be compared to it to determine the similarities or differences in performance. Further information on the parameters of the contact forces are in Table A.1.

3.1.1 OpenSim

The open source software OpenSim 3.0 was used to create, edit, and visualise the models and environment. OpenSim makes use of XML to represent its models, compromising of the sections mentioned in Table 3.1.

Section	Description
*BodySet**	Body geometry
ConstraintSet	List of constraints
*ForceSet**	Acting forces
MarkerSet	List or markers
*ContactGeometrySet**	Contact geometry
ControllerSet	Auxiliary controllers
ComponentSet	Group geometry
ProbeSet	Auxiliary probes

Table 3.1: The sections represented within the .osim file. Sections in italic and with a star have been edited for this experiment. The other sections have be left to their default state.

3.1.2 Healthy model

The healthy musculoskeletal model used in the experiment is based of the model: `gait14dof22musc_pros` (Kidziński et al., 2018). The model has been edited to be used for this experiment, the final musculoskeletal model has a total of 18 muscles (9 per leg), which control 14 degrees of freedom. The edits made to the original model is copying the left leg geometry onto the right leg, and adjusting the contact points respectively. The legs of this model can be see in the left of Figure 3.1.

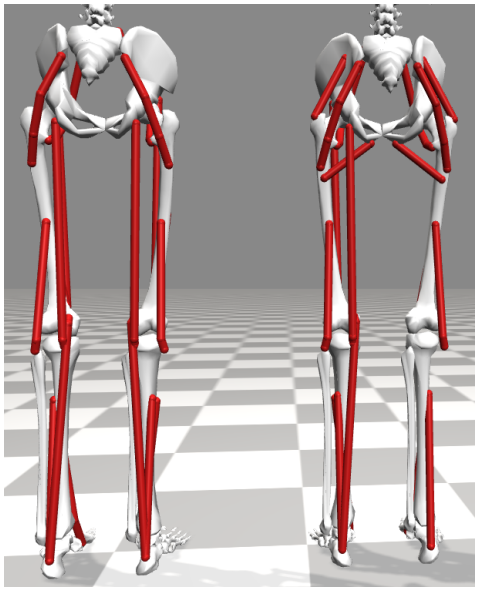

Figure 3.1: Left: the legs for the healthy model. Right: legs for the transfemoral amputee model. Right leg missing the hamstring, but has the addition of 2 muscles around the hip.

3.1.3 Transfemoral amputee model

The transfemoral amputee model is an edited version of the healthy model, where certain muscles have been removed. The removed muscles from the right leg are the three biartcular muscles: gastrocnemius, hamstring, and rectus femoris. However, to ensure the stability of this model, 2 muscles are added; the hip adductors and abductors. This model has 19 muscles which control 16 degrees of freedom. The legs of this model can be seen on the right of Figure 3.1.

3.1.4 Muscles

To be able to perform the tasks of climbing stairs the maximum isometric force for all muscles of both models were increased by 80%. The muscles for the

walking and the ramp task remained at their default values.

3.2 Environment

Using OpenSim alongside MeshLab 2020.07, SketchUp 2017, and Blender 2.83, the geometry mesh files were created and saved as `.obj` files. The meshes are triangular meshes, with the normals of the vertices and faces calculated by OpenSim. Theses meshes where imported into the `BodySet` section and the `ContactGeometry` section of the model. So that the mesh file for the contact geometry works, a strict rule is set, it is required to be a non-manifold and watertight mesh. A total of 4 unique meshes are made to be imported into a model, they are: ramp, stairs, heel, toe. The heel and toes meshes are reused in all environments. Each mesh was designed with regards to keeping the amount of vertices and faces to a minimum, as too many would have increased computational power.

3.2.1 Obstacle Meshes

Both musculoskeletal models are subjected to an environment. An environment is made up of 2 components: the musculoskeletal model, and an obstacle. The obstacles in each environment is built on a base that is 2 meters wide. Having the base be 2 meters wide gives adequate space to the left and right of the model. All environments start on a flat level surface. All 6 unique environments can be seen in Figure 3.2. An example environment is shown in Figure 3.3.

Figure 3.2 shows the starting position of each of the 6 individual testing environments. All musculoskeltal models are facing, and walking, in the positive X-direction. The left and right of the model is negative and positive Z-direction respectively. The up and down motion of the model is the positive and negative Y-direction respectively. The reference for the centre of the musculoskeltal model is its pelvis. The pelvis is placed at $(0, Y, 0)$, where Y changes between [0.94m, 1.44m, 1.94m] depending on the obstacle.

A more detailed view of the ramp and stairs can be seen in Figure 3.4 and Figure 3.5 respectively. These views show a perspective of only the obstacle section of the mesh. The surfaces are smooth, and joined at the edges with no gaps or holes. Each mesh has no overlapping faces, or duplicate vertices.

The ramp obstacle has a run of 3.25 meters and a rise of 0.45 meters. The gradient of the slope is 7.883 degrees with a length of 3.28 meters.

The set of stairs has 3 steps to it. A single step has a height of 0.20 meters and a depth of 0.25 meters. The total increase in height is 0.60 meter.

3.2.2 Foot meshes

The foot contact geometry is composed of 3 spherical meshes, 1 heel and 2 toes. The heel mesh is a sphere with a diameter of 50 millimetres, whereas a toe is a sphere with a diameter of 25 millimetres. The left and right feet are shown in Figure 3.6. This figure shows the placement of the contact mesh with respect to the bones within the foot. It is important to note that it is the geometry of the contact mesh which exerts a force, and not the bones themselves.

The coordinates for the foot geometry are mirror reflections of each other. For a foot, the coordinates (x, y, z) are: heel at (3, 2, 0) relative to the body piece `calcn`, toe 1 at (0.02 -0.005 -0.026) relative to the body piece `toes`, and toe 2 at (0.02 -0.005 0.026) relative to the body piece `toes`.

Each of the 4 meshes from Table 3.2 are imported into the `ContactGeometry` section of their .osim file. Each specific task has it own file dedicated to it, resulting in 10 individual musculoskeletal model files. The meshes for the ascent environments are identical to their descent counterparts. In those environments where the task is to go down the obstacle, the skeleton has been translated in the y-axis proportional to the height of the obstacle, and the obstacle has been rotated by 180 degrees with respect to its centre. Each environment is loaded and tested separately. In each of the eight environments which contain an obstacle, the skeleton has 2 meters of flat-level surface before reaching the obstacle.

The 4 unique meshes and their detailed level of granularity are shown in Table 3.2. The ramp mesh, stair mesh, toe mesh, and heel mesh are triangular-hollow meshes, and only have exterior faces.

To ensure that the meshes are able to contact with one another, the switch from the Hunt Crossley Force to Elastic Foundation Force was made (Hast et al., 2019). The parameters for the Elastic

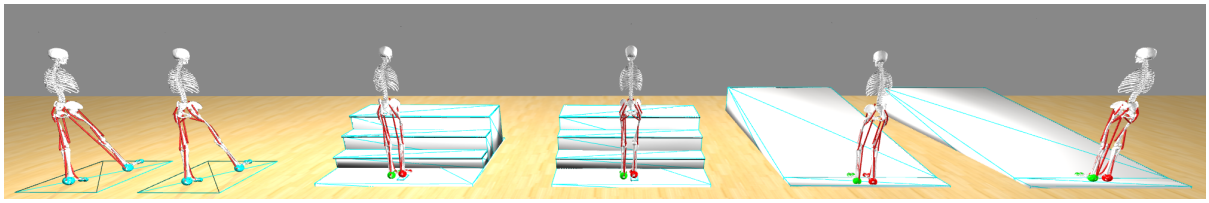

Figure 3.2: From left to right: healthy, transfemoral, healthy stairs ascend, transfemoral stairs ascend, healthy ramp ascend, transfemoral ramp ascend. The obstacles are rendered in white. The contact meshes are render in blue wire-frame.

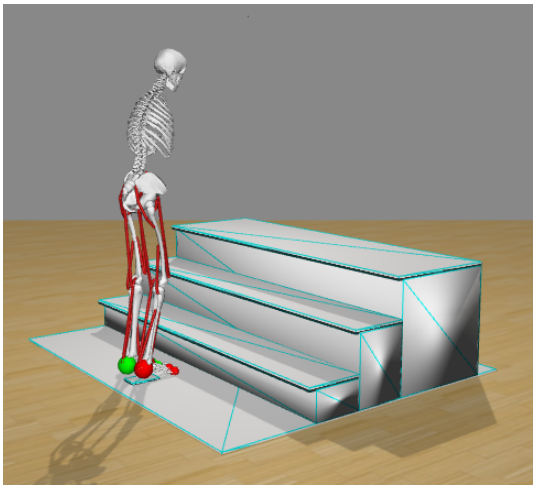

Figure 3.3: The environment of the stairs ascent experiment showing starting position as well as obstacle. The musculoskeletal model's starting postion is based on the starting position of the training data.

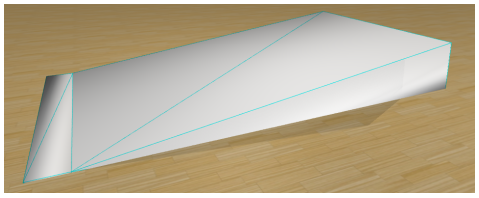

Figure 3.4: A close up of the ramp.

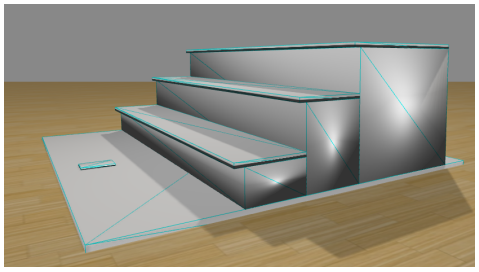

Figure 3.5: A close up of the stairs.

Foundation Force were taken from research which investigated foot-ground contact (DeMers et al., 2017). The exact values can be found in Table A.1.

3.3 Training data

For the basic task (fast walking), the data for 1.75 m/s was taken from our previous research (De Vree and Carloni, 2021). The training data for the advanced task (stairs and ramp) are obtained from the Graphics Lab - Motion Capture Library (Carnegie Mellon university, n.d.), provided in c3d motion capture format. The subject number 74 with trial 19 was used for stairs and subject number 14 with trail 22 was used for ramp.

The c3d files contained tracking balls to determine the location of the the subject in 3D space, as shown in Figure A.1. Each tracker has a label designated to it to represent its location on the body, only the trackers on the hips and legs were used to calculate the training data. The necessary trackers were exported to a trc tracking file with the use of Mokka software. Those tracking balls were mirrored on the healthy musculoskeletal model in the OpenSim software to be able to run inverse kinematics. The inverse kinematics resulted in the needed format for the OpenSim learning environment. The root-mean-squared-error of the inverse kinematics has a max value 0.05 for all trackers. The velocity of the joint at time t was then calculated using the the formula

$$v_i(t) = \frac{p_i(t) - p_i(t-1)}{t}$$

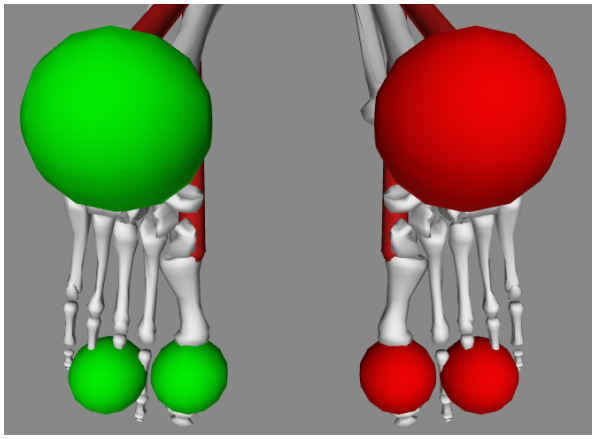

Figure 3.6: A view from below of the positioning of the contact geometry respective to the foot bones. Green: The heel and the toe mesh for the left foot. Red: The heel and the toe mesh for the right foot White: foot bones.

Mesh	Vertices	Faces
ramp	24	36
stairs	48	72
heel	107	210
toe	107	210

Table 3.2: The 4 meshes and their respective vertices and faces.

where i is the joint, $p_i(x)$ is the position of the joint at time x, and t is time. The velocities of the joint i at the start and end of the training data are $v_i(1)$ and $v_i(n-1)$ respectively, with n being the total amount of samples in the training data.

3.4 Hardware

To be able to run this experiment, different mediums of hardware are used. The most prominent is the Microsoft Azure: Cloud Computing Services. A total of three servers were allocated on Azure, each of them are the NC6 Data Science Virtual Machine for Linux (Ubuntu 18.04) with 6 cores and 56 gigabytes of ram. Other personal hardware was used as well, 4 systems running Ubuntu 18.04. System 1 has an Intel core i5 3570K and 8 gigabytes of ram, and system 2 has an Intel core i7 8700K with 16 gigabytes of ram, system 3 has an AMD Ryzen 3800x with 32 gigabytes of ram, and system 4 has an AMD Ryzen 5950x with 64 gigabytes of ram.

The python version is `Python 3.6.10`, the tensorflow version (non-gpu) is `tensorflow 1.15`. The program makes use of `mpi4py` to be able to run on 4 cores. The approximate time to run a single iteration on the Azure servers is 356 seconds, on system 1 it is 250 seconds, and on system 2 it is 170 seconds, system 3 it is 112 seconds, and system 4 it is 95 seconds.

Data can be collected from the simulation after each iteration. The rate at which the data is sampled during the experiment is every 5 iterations. The trained model is saved using tensorflow-checkpoint. The angles of the joints can be extracted from the trained model in degrees. The training reward per timestep is also recorded, it as the mean of the reward. The length of each training period is recorded as well.

3.5 Transfemoral model training

The OpenSim environment is fully established for the transfemoral model. The model loads into the OpenSim environment error free as well make contact with the geometry and activate its muscles. The training data has been setup and is ready to be used. However, due to unexplained behaviour those models were not run during the experiment, and hence their results are omitted.

4 Results and Analysis

The reward per timestep for each environment were plotted. The angle, in radians, of the joints of interest (knee and ankle) during a simulation's run-time were plotted.

4.1 Healthy stairs ascend

The reward obtained by the healthy musculoskeletal model climbing up the stairs is presented in Figure 4.1. Only the healthy stairs and ramp environments are discussed in this paper to show the ability of the PPO.

The reward increases rapidly for the first 19383 timesteps, then it levels out until 55951 where it then again increases rapidly until approximately 76099, from here onward the learning is continuous and slow. The rapid increases in reward represent

when the model has learned to correctly climb 1 step. Since steps are non-linear motions, this jumps in reward represent this well.

In the end, the model was able to climb stairs for 300 simulation-timesteps, and maxed out at 215 reward. The model was able to achieve 215/300 = 71.6% of its total reward, this suggest that the model's gait is 71.6% accurate to the training data.

Looking at the angles of the ankle in Figure 4.2 we can see the symmetry in the gaits, as well as how well the measured angles follow the real world data.

The ankles show some symmetry between them, but the right ankle seems to deflect far more that the left. The knees show very consistent symmetry throughout the gait cycle. Neither knee is experiencing large deflections. Both the knees and the ankles follow the real world data closely throughout the simulation.

The fibre forces are shown in Figure 4.3. The forces for the the bicep femoris are erratic but show some continuous cycles for both legs. The bicep fermoris averages 613 N on the left and 654 N on the right. The vasti muscles show clearly that it is either the left or the right vasti which is exerting force at any given time, not both. The vasti averages 2773 N for the left and 2472 N for the right. There are times where the vasti is not exerting a force and this is when the foot is lifted from the ground.

The ground reaction forces are shown in Figure 4.4. The each foot is in contact with the ground 2 times and is lifted 2 times. The most force is exerted on the Y-direction, then the X-direction, and lastly

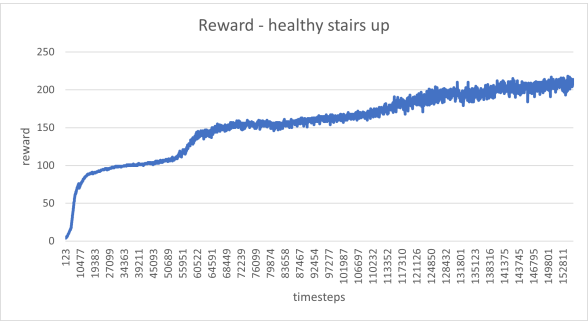

Figure 4.1: The rewards gained during the learning process of the healthy stairs ascent environment

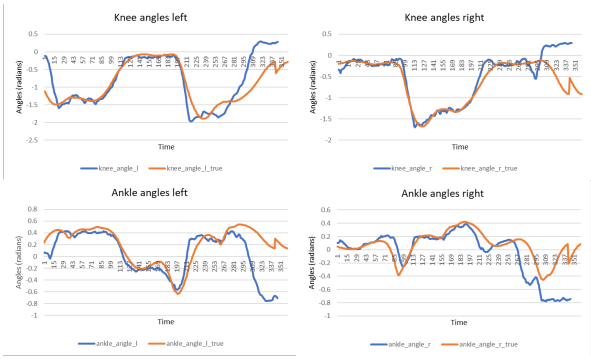

Figure 4.2: The angles of the ankles and knees during stairs ascent. blue = measured, orange = real life data

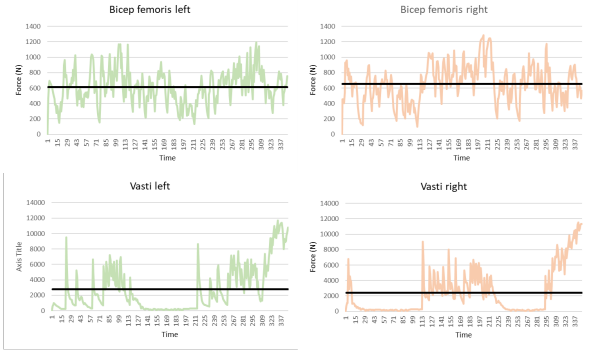

Figure 4.3: The fibre forces during stairs ascent of the bicep femoris and vasti muscle. The black line indicates the average.

the Z-direction which is rarely exerting any force. The Y-component of the force peaks at the very beginning and end of the contact. It maxes out at a force of -1602 N on the left and -1387 N on the right. It has an average of -334 N on the left and -347 N of the right.

4.2 Healthy ramp ascend

The reward obtained by the healthy musculoskeletal model walking up the ramp is presented in Figure 4.5.

The reward increases rapidly for the first 13,000 timesteps then the increase becomes linear. The initial gain is due to the model leanrning to stand and take the first step. Since the environment is continuous after this initial step, the learning exhibits continuous learning. The further the model walks

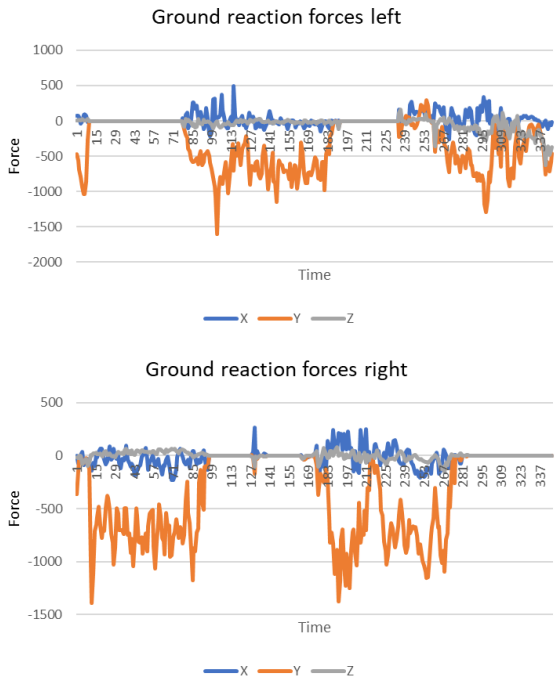

Figure 4.4: The ground reaction forces during stairs ascent for all contacts points for both feet.

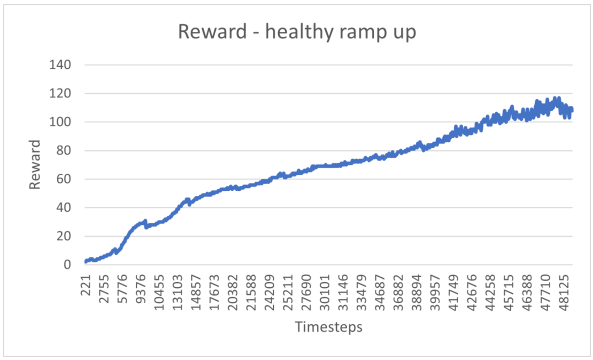

Figure 4.5: The rewards gained during the learning process of the healthy ramp ascent environment

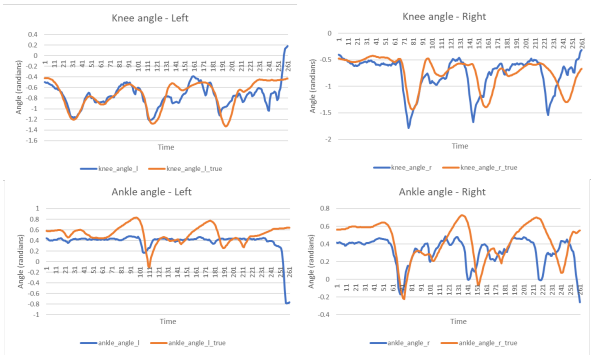

Figure 4.6: The angles of the ankles and knees during ramp ascent. blue = measured, orange = real life data

up the ramp, the more reward it achieves. The final 4000 timesteps shows the greatest flucation, this is due to it being the most recently learned behaviour and it takes time to reach the end of the ramp resulting in the final moments being iterated over less often.

This model was able to make it to the top of the ramp after 210 simulation-timesteps, achieving a max reward of 120. This gives it an accuracy of $120/210 = 57.1\%$ of its total reward. The simulation was able to mimic the real life data to an accuracy of 57.1%.

Looking at the angles of the ankle, Figure 4.6, we can see the symmetry in the gaits for the knees but not so much for the ankles.

The ankles shows greater deflection for the left leg than it does for the right leg. Both ankles seem to show periodic repetition every 75 timesteps. The knees shows similar repeating patterns every 75 timesteps. However, the left knee is rotating less than the right knee. This suggests that the right leg was used more excessively in this trial than the left leg.

The fibre forces are shown in Figure 4.7. The bicep femoris shows 3 distinct cycles which last 75 timesteps. The forces during this time are erratic. The bicep femoris averages 886 N on the left and 939 N on the right. The vasti has peaks which are consistent with the gait cycles. The vasti averages 1312 N on the left and 1334 N on the right. The foot is lifted when the vasti is exerting 0 force.

The ground reaction forces are shown in Figure 4.8. The each foot is in contact with the ground 4 times and is lifted 3 times. The largest force is exerted in the Y-direction, then in the X-direction, and finally the weakest force is experienced in the Z-direction. The Y-component of the force peaks often and is not smooth, it maxes out at a force of -2668 N on the left and - 2469 N on the right. It has an average of -299 N on the left and -262 N of the right.

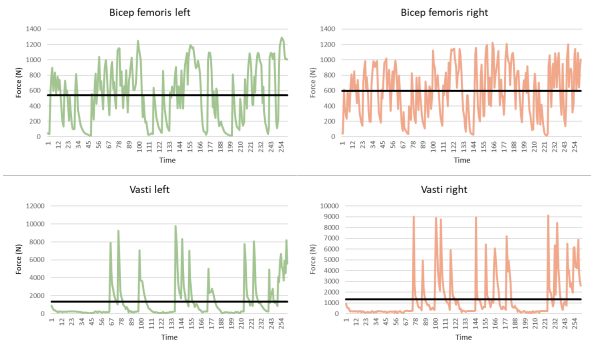

Figure 4.7: The fibre forces during ramp ascent of the bicep femoris and vasti muscle. The black line indicates the average.

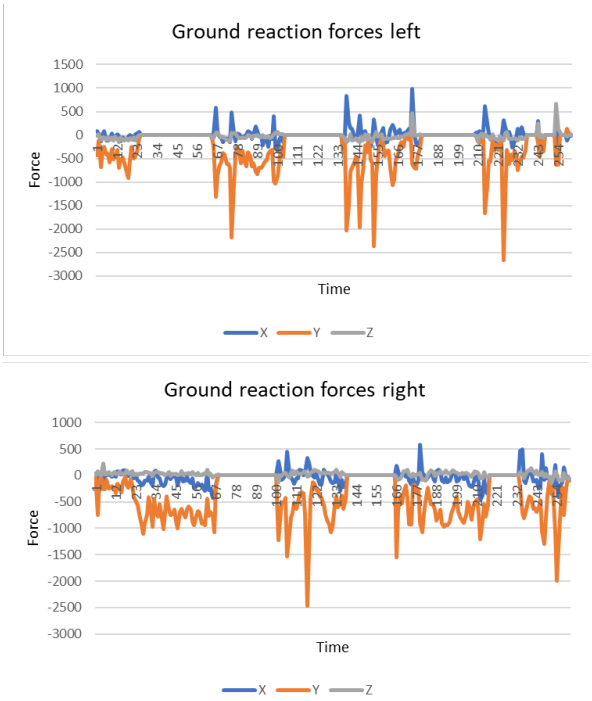

Figure 4.8: The ground reaction forces during ramp ascent for all contacts points for both feet.

5 Conclusions

What the results show is that the PPO algorithm is able to learn to cope with the advanced environments of ascending stairs and ramps.

The rewards obtained show that it is possible for the PPO to achieve a reasonable reward which results in approximate human-like locomotion. However, these rewards are only obtained after a vastly greater training period. The results from Leanne (De Vree and Carloni, 2021) shows adequate rewards obtained after 50,000 epochs, whereas the rewards obtained in this research appear after roughly 150,000 epochs which is 3x more training. This increase in training time does not linearly translate to an increase in computational time, training for stairs and ramps takes a minimum of 148 hours compared to the previous 8 hours.

The angles of the knees and ankles for the musculoskelatal models do show angles which relate to human-like walking. This shows promising applications of PPO for future artificial intelligent research methods in the field of human locomotion.

The computational cost of the running so many simultaneous simulations resulted in only ascending of the obstacles. Exploring the environments of descend the obstacles is essentially built into this papers research but avoided due to time constraints.

Overall, an understanding of the application of new contact forces, building of meshes, the alterations of a reward function, and musculoskeletal inverse-kinematics from training data shown in this paper has built a solid foundation for more research into prosthetic development.

6 Acknowledgements

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 Raffaella Carloni (Professor, University of Groningen) and Vishal Raveendranathan (Doctoral candidate, University of Groningen) for the supervision, support, and general discussion through out the project.

References

V. Azimi, T. Shu, H. Zhao, R. Gehlhar, D. Simon, and A. D. Ames. Model-based adaptive control of transfemoral prostheses: Theory, simulation, and experiments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–18, 2019.

R. Bartlett. Artificial intelligence in sports biomechanics: new dawn or false hope? *Journal of sports science & medicine*, 5(4):474–479, 2006.

Carnegie Mellon university. Cmu graphics lab - motion capture library., n.d. `http://mocap.cs.cmu.edu/`.

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*, 29:607–618, 2021.

S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. 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, Jennifer L Hicks, and Scott L Delp. Preparatory co-activation of the ankle muscles may prevent ankle inversion injuries. *Journal of biomechanics*, 52:17–23, 2017.

M. Ege and S. Kucuk. Design and dynamic model of a novel powered above knee prosthesis. In *2019 Medical Technologies Congress (TIPTEKNO)*, pages 1–4, 2019.

Michael W Hast, Brett G Hanson, and Josh R Baxter. Simulating contact using the elastic foundation algorithm in opensim. *Journal of biomechanics*, 82:392–396, 2019.

Łukasz Kidziński, Sharada P. Mohanty, Carmichael F. Ong, Jennifer L. Hicks, Sean F. Carroll, Sergey Levine, Marcela Salathé, and Scott L. Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In Sergio Escalera and Markus Weimer, editors, *The NIPS '17 Competition: Building Intelligent Systems*, pages 101–120. Springer International Publishing, Cham, 2018.

K. Koganezawa, H. Fujimoto, and I. Kato. Multifunctional above-knee prosthesis for stairs' walking. *Prosthetics and Orthotics International*, 11 (3):139–145, 1987.

J. M. Lambrecht, C. L. Pulliam, and R. F. Kirsch. Virtual reality environment for simulating tasks with a myoelectric prosthesis: an assessment and training tool. *Journal of prosthetics and orthotics : JPO*, 23(2):89–94, 2011.

A. K. LaPrè and F. Sup. Simulation of a slope adapting ankle prosthesis provided by semiactive damping. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 587–590, 2011.

A. K. LaPrè, B. R. Umberger, and F. Sup. Simulation of a powered ankle prosthesis with dynamic joint alignment. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1618–1621, 2014.

Eric Nickel. Passive prosthetic ankle-foot mechanism for automatic adaptation to sloped surfaces. *Journal of rehabilitation research and development*, 51(5):803, 2014.

Carmichael F. Ong, Thomas Geijtenbeek, Jennifer L. Hicks, 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):1–27, 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 (TOG)*, 37(4):1–14, 2018.

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

A Appendix

	Parameter	Value
Right leg	appliesForce	true
	geometry	[stair_c, ramp_c] r_heel r_toe1 r_toe2
	dissipation	5
	stiffness	50000000
	static_friction	0.9
	dynamic_friction	0.9
	viscous_friction	0.9
	transition_velocicty	0.1
Left leg	appliesForce	true
	geometry	[stair_c, ramp_c] l_heel l_toe1 l_toe2
	dissipation	5
	stiffness	50000000
	static_friction	0.9
	dynamic_friction	0.9
	viscous_friction	0.9
	transition_velocicty	0.1

Table A.1: The values of the Elastic Foundation Force respective of the right foor. The geometry value can be chosen from either stairs_c or ramp_c. The left foot is analogous to the right foot, just "l_" rather than "r_"

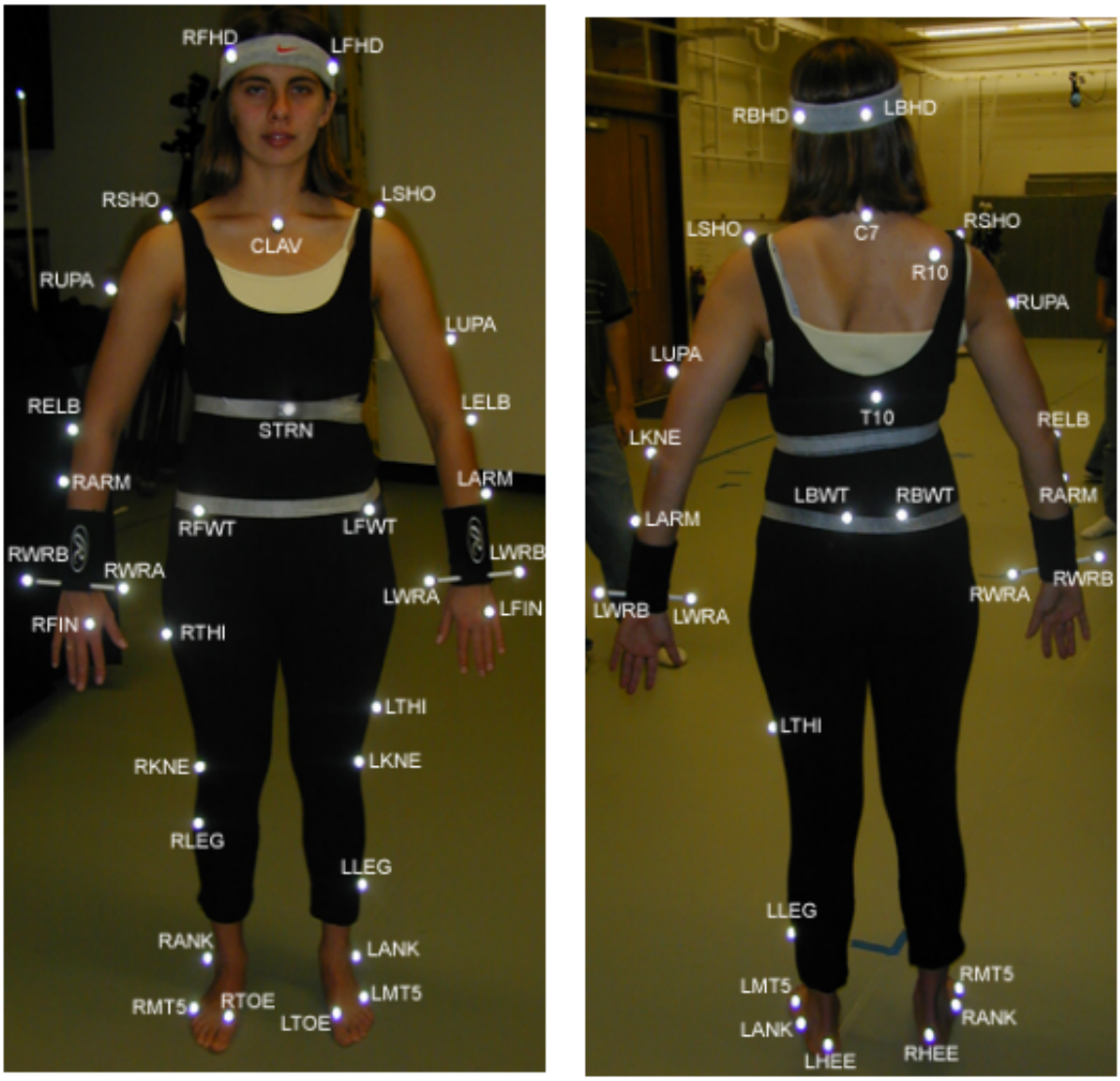

Figure A.1: The position of the trackers on the human body.