8-2023

Simulating Motion Success With Muscle Deficiency in a Musculoskeletal Model Using Reinforcement Learning

Daniel Castillo
The University of Texas Rio Grande Valley

SIMULATING MOTION SUCCESS WITH MUSCLE DEFICIENCY IN A MUSCULOSKELETAL MODEL USING REINFORCEMENT LEARNING

A Thesis

by

DANIEL CASTILLO

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Computer Science

The University of Texas Rio Grande Valley

August 2023

SIMULATING MOTION SUCCESS WITH MUSCLE DEFICIENCY IN A MUSCULOSKELETAL MODEL USING REINFORCEMENT LEARNING

A Thesis
by
DANIEL CASTILLO

COMMITTEE MEMBERS

Dr. Dong-Chul Kim
Chair of Committee

Dr. Erik Enriquez
Committee Member

Dr. Emmett Tomai
Committee Member

Dr. Zhixiang Chen
Committee Member

August 2023

ABSTRACT

Castillo, Daniel, Simulating Motion Success with Muscle Deficiency In a MusculoSkeletal Model Using Reinforcement Learning. Master of Science (MS), August, 2023, 41 pp., 1 table, 22 figures, references, 19 titles.

Humans possess an extraordinary ability to execute complex movements, captivating the attention of researchers who strive to develop methods for simulating these actions within a physics-based environment. Motion Capture data stands out as a crucial tool among the proven approaches to tackle this challenge. In this research, we explore the effects of decreased muscle force on the body's capacity to perform various tasks, ranging from simple walking to executing complex jumping jacks. Through a systematic reduction of the allowed force applied to individual muscles or muscle groups, we aim to identify the threshold at which the body's muscles tolerate the deficiency before the motion becomes unattainable. Additionally, we seek to analyze how the model adapts its muscle activation levels, in scenarios where the motion is still achievable despite the applied deficiencies.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

The human body is composed of 206 bones and about 600 muscles. The physical condition of these systems is critical to the person's ability to execute various motions. Through muscle contraction and flexion, the body is able to perform actions such as walking, jumping, or even highly stylistic movements like flips. Exercise and training of muscle groups allow the person to increase their ability to perform these motions beyond their previous limitations. However, in cases of injury or disuse, the ability of that person to complete these actions, if they were otherwise healthy, will be reduced.

Simulating human motion has become a very desirable field in computer vision and biomechanics due to their ability to perform these complex motions. Deep Reinforcement Learning (DRL) has shown the ability to capture the complex interactions within the bodyâĂŹs systems and replicate that within a complex Deep Neural Network. *Deep Mimic* [14] takes a humanoid model whose joints are torque actuated and was able to provide a result that was able to stylistically mimic varying complex motions such as running, jumping, and flips. When it comes to muscle-actuated models it is actually more difficult to learn a control policy in which one needs to learn to set activation levels to muscle actuators. Projects such as *Learn to Run 2018* [19] show a path in which a simple 18-muscle actuated lower body model was able to learn how to walk successfully.

OpenSim [4] is a free-to-use software that allows users to develop musculoskeletal models and create dynamic simulations of movement. Their website allows for the download of varying musculoskeletal models that have been used in many studies to determine how the body uses its muscles to perform the daily tasks we do. Models such as Rajagopal provide a solid baseline as an 80-muscle actuated lower body model, into the way that our muscles act during gait. However, the

availability of a comprehensive muscle-actuated full-body model is missing. The work done by *Lee* [11] provides this model in a fantastic way. Their work has developed a 346-muscle actuated model that includes most of the skeletal muscles that are crucial to joint movement.

When muscles become weaker our ability to perform motion becomes limited. Studies have shown the strength allowed to certain muscle groups directly correlates to the person's ability to perform these actions. In this work, we will show how the decreasing allowed weakness in certain muscle groups affects the model's ability to perform a reference motion. In addition, what levels of allowed force certain upper and lower body muscle groups can tolerate till the reference motion is no longer possible? With this, we can also see which muscle groups compensate for the lack of force provided by the weakened muscle(s). To do this we use DRL to train a complex full body muscle actuated model using original motion capture data. After training, we will decrease the allowed force to muscles in the lower body to see the extent that motion is affected. Using the resulting data we will see which muscles can tolerate the decrease in allowed force and still be able to complete the targeted motion, and which muscles if any are compensatory to allow the motion.

With this method, users will be able to see the exact muscle activation at the different allowed levels of force for any muscle they choose to test. As a result, physical therapists or medical professionals can target the effects of the decrease in allowed muscle force and correctly develop a plan for the rehabilitation of a subject. In addition, the pipeline will allow for the visualization of motions with certain muscle force deficiencies, this allows persons to be able to know exactly what muscles are necessary in order to complete this motion and what is the minimum force needed.

CHAPTER II

RELATED WORK

2.1 Human Musculoskeletal Modeling

A musculoskeletal model usually represents a human body with rigid segments and muscle-tendon actuators. Hill-type muscle actuators are the most commonly used muscle actuators for these joints. This is because they can accurately simulate the dynamics of biological muscles, like the dynamic active and passable contraction. Hill-type muscles in models have been used to simulate models with muscle fatigue, like in the work done by *Konmura*, which simulates the effects of lower body muscle fatigue and how the body re-targets its motions [9]. Musculoskeletal parameters such as height, muscle force, or weight are based on measurements taken from human cadavers or large groups of people.

OpenSim [4] is open-source software that is used in Biomechanics to simulate these musculoskeletal dynamics. One of these projects is the work done by Rajagopal [15]. In this work, they created a 3D musculoskeletal model with a highly accurate representation of a healthy male's lower body. This model has 37 degrees of freedom with 80 muscle-tendon units actuating the lower body and 17 torque actuators controlling the upper body.

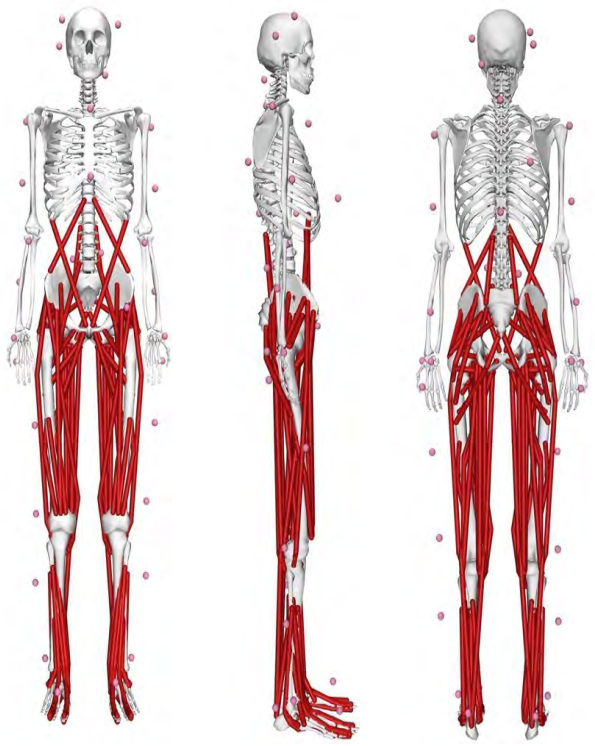


Figure 2.1: Rajagopal Model

[15]

This model was used to be a non-invasive way to study human movement and predict the effects of certain interventions on gait.

The work done by *Lee* [11] provides a tree structure of rigid bones composed of 8 revolute joints such as knees and elbows and 14 ball-and-socket type joints (hips, ankles, and shoulders). The model contains 284 muscles that correlate to the joint motions of these joints. The muscles are attached to each bone end-to-end by tendons. The site at which the muscles are attached is called origin (proximal side) and insertion (distal side). Some muscles such as the Gluteus Maximus and Deltoid are split into 3-5 sections to fit around bones due to their curved nature. For example, lower numbers such as Gluteus Maximus 1 being on the proximal side and Gluteus Maximus 4 being on the distal side. Contraction of these muscles effectively pulls the joints to a position between them.

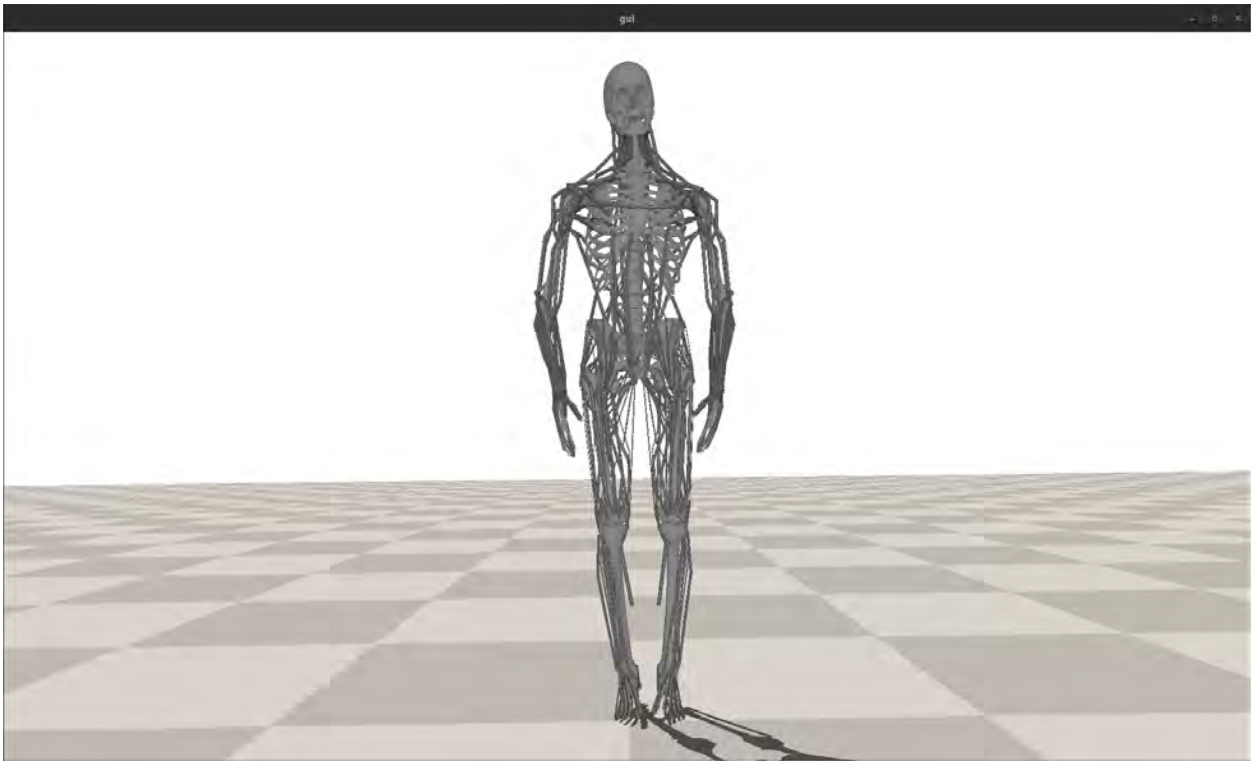


Figure 2.2: Lee Musculoskeletal Model

[11]

In this work, we refer to the level of muscle contraction [0,1], as muscle activation. Where a muscle activation level of 0 represents no muscle contraction and 1, full contraction.

2.2 Deep Reinforcement Learning

Deep Reinforcement Learning is machine learning designed to help solve decision-making problems. The objective of the algorithm is to learn a policy that maximizes a reward for an agent that is interacting with its environment. For a general RL problem at a timestep t the agent will get an observation o_t and with that observation(maybe the velocity and joint positions of a model) the policy will give the agent some action a_t, using this action the agent will then apply it to the environment and a reward will be given to the agent. A larger reward will be given if the agent successfully completes the task or a penalty will be given if it was not successful. So, the agent will use the observation and actions given by the policy to maximize the reward at the end of each time step [17].

2.2.1 Learn to Move

The *Learn to Move* competition series held through 2017-2019, helped develop control models using DRL to control a 3D human musculoskeletal model to follow target velocities through the model itself changing its walking speed and direction. The main objective of this challenge was to have participants build a controller for the human musculoskeletal model, that optimizes muscle activity so that the model walks/runs forwards as long and as fast as possible within a 10-second time frame while avoiding obstacles.

This competition's environment was on Open Sim-RL which included the 3d musculoskeletal model, reward system, and a visualization tool for the simulation. The model used for this competition was a simplified human model, with muscle actuated only at the lower body. Composed of eight rotational joined actuated by 22 muscles.

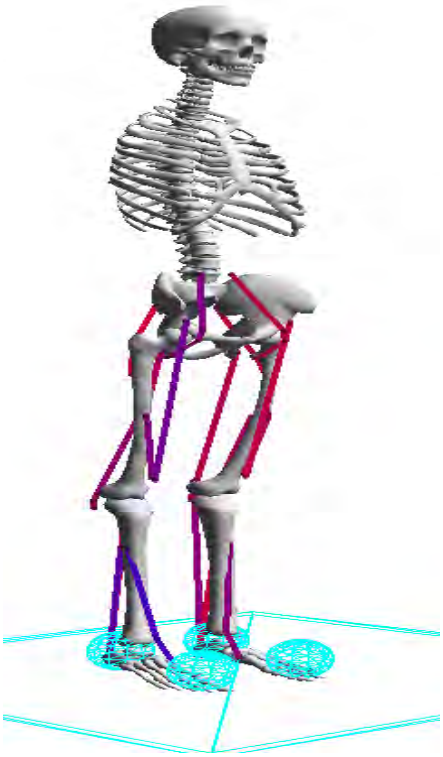


Figure 2.3: Learn to Run Human Model

[17]

The reward function for this competition was designed off previous results that produced

human-like walking The total reward J, was based of 3 components:

$$J(\pi) = Ralive + Rstep + Rtarget$$

$$= \sum_{isim} balive + \sum_{istep} (wsbs - wvcv - wece) + \sum_{itarget} btarget$$

Where R*alive* was the reward given to the agent for not falling down, R*step* was the reward given to the agent for making footsteps at the desired velocity with minimal muscle effort, and R*w*as the reward given to the agent when it reached specified target locations.

A top solution [18], utilized a technique called *Curriculum Learning* [2], where the developer designs the curriculum that is a series of smaller tasks which eventually when put together results in the original task. *Zhou*[18], utilized this method by first training the model to run at a high speed and then gradually training at a decreased speed, which eventually led to the model learning how to walk at the desired normal speed. This resulted in the model learning a very natural gait since the number of high-speed gaits is limited compared to walking at a normal speed. Starting the training at a high-speed run resulted in the model achieving a natural gait since training at a slower running speed opens the model to a larger pool of gaits that follow that slower speed. *Zhou* [18] achieved their final desired solution by training the slow normal walking policy to follow the desired walking parameters and restrict movement to allow minimal muscle activation.

Zhou [18] used an algorithm Deep Deterministic Policy Gradient (DDPG) [12], which is when an agent in a DRL environment updates the actor and critic properties at each time step during learning, stores its past results using a circular experience buffer, and perturbs the action chosen by the policy by using stochastic noise at each training step.

The second place entry in the Learn to Run competition, Kolesnikov and Hrinchuk [8] utilized TD3[5], which is similar to DDPG in that it utilizes Q-functions, but this addresses issues that stem from DDPG in that it TD3 learns two Q-functions instead of one and uses the smaller of the two resultant values to form targets for the Bellman error loss functions, it also updates the policy less frequently and adds noise to the target action adding difficulty to the policies ability to

exploit errors in the Q-function.

Third place submission Akimov [1] utilized Soft Actor-Critic [6] (SAC), which is a policy designed to maximize the trade-off between the expected return of the policy and its entropy. This increasing entropy results in high exploration, which leads to faster learning later in the policy and prevents the policy from converging too soon.

The top solution by Zhou [18] was able to learn a policy that produced human locomotion, despite the model's simplicity. For example, the upper half portion of the model does not move at all, meaning it has no allowed degrees of freedom. No submissions in this competition utilized motion capture data which could have led the teams to use imitation learning to train this simplified model to perform other human motions besides walking.

2.2.2 Deep Mimic

Deep Mimic [14] is a project in which RL methods are shown to be adaptable to learn control policies that are capable of imitating motion capture data while accomplishing user-specified tasks. The reward *rt* in this paper at each timestep t, is broken into two terms that influence the model to match the inputted reference motion while completing addition tasks. [14]

$$r_t = w^I r_t^I + w^G r_t^G$$

In this equation, r_t^I is the Imitation reward and r_t^G is the reward allowed to the model if it completes the set task objectives, with w^I and w^G being their respective weights. Here the task reward can be edited to fit different tasks as in the paper there is a *Strike* motion in which the model's goal is to strike a randomly placed target with its feet. So the task reward in this scenario is given by

$$r_t^G = \begin{cases} 1 & \text{Target has been hit} \\ exp[-4||p_t^p tar - p_t^e||^2] & \text{otherwise} \end{cases}$$

In this equation, the variable p_t^tar is the location of the target and p_t^e is the location of the model's foot. The target is then marked as a "hit" if the foot is within 0.2 meters of the target.

The imitation reward rewards the character for following the given reference motion. This reward is broken down into four terms representing the joint orientation, joint velocity, end-effector position, and center of mass. w^P, w^v, w^e, w^c are the weights given to each.

$$r_t^I = w^P r_t^P + w^v r_t^v + w^e r_t^e + w^c r_t^c$$

$$w^P = 0.65, w^v = 0.1, w^e = 0.15, w^c = 0.1$$

The pose reward r_t^P allows the model reward when it closely matches the joint orientation of the reference motion. This is calculated by taking the difference between the quaternions of the reference motion and the model.

$$r_t^P = exp[-2(\sum_j ||\hat{q}_t^j \ominus q_t^j||^2)]$$

Here, $\hat{q}_t^j$ and q_t^j are the orientations of the jth joint in the reference motion and the model, $\hat{q}_t^j \ominus q_t^j$ represents the quaternion difference and $||q||$ is the scalar rotation of the quaternion about its axis in radians.

The velocity reward r_t^v is the reward that checks the difference between the velocities of the model and reference motion's joints.

$$r_t^v = exp[-0.1(\sum_j ||\hat{\dot{q}}_t^j - \dot{q}_t^j||^2)]$$

Here the finite difference between the target angular velocity $\hat{\dot{q}}_t^j$ from the reference motion and the model's velocity $\dot{q}_t^j$ of the jth joint is taken

For the end-effector reward the hands and feet must match the position of the reference motions'. Here, p_t^e is the 3D position in meters of the end-effectors (Right/Left Hand and Right/Left

foot)

$$r_t^e = exp[-40(\sum_e ||\hat{q}_t^e - q_t^e||^2)]$$

The center of mass reward, r_t^c penalizes the model's deviation from the center of mass given by the reference motion.

$$r_t^c = exp[-10(||\hat{q}_t^c - q_t^c||^2)]$$

This work uses a PPO to train its policies. PPO [16]. There are two networks one that maintains the policy itself $\pi_\theta(a|s,g)$ and the value function $V_\psi(s,g)$. The initial state p(s_0) determines which state the agent will be in once an episode starts. A common choice to start the agent in is to have it start at the beginning of the motion and have it progress over the motion throughout the duration of the current episode. With this approach, the agent learns the motion in a sequential manner learning the beginning of the motion first and the later phases of the motion afterward. This could lead to issues in motions such as back flips, where the model must learn to first land from a jump before it can receive a high reward for the jump itself. Using the reference motion data the model can be guided during training and be used to find an optimal initialization state for each episode. The work refers to this strategy as *Reference State Initialization (RSI)*[14]. Using this method the model is able to learn motions later on into the reference motion that yield high rewards that it may not have been able to reach if it was constantly initialized to the beginning of the motion at each episode.

This work also utilizes a technique called *Early termination* [14]. Each episode always has a finite amount of time it could be active before it resets and a new one begins. However, there may be cases where the model is not yet able to imitate the reference motion and has fallen over, thus the data being collected from that episode becomes irrelevant and harmful to the overall task of imitating the motion. Thus, they incorporate a system where if certain parts of the model come into contact with the ground the remained reward the agent can receive for this episode is zero. This system of early termination shapes the way the reward function and allows it to discourage episodes

in which the model is providing irrelevant data.

CHAPTER III

METHODS

3.1 Collecting Motion Capture Data

3.1.1 Qualisys

In order to record the original motion capture data we utilized software names *Qualisys*. With *Qualisys* we were able to utilize their cameras and software to record, process, and visualize our motion capture data. The videos obtained were captured at 1080p 60 frames per second.

Each *Miqus Video Camera* has the capability to stream full HD 1080p video at 85 frames per second.

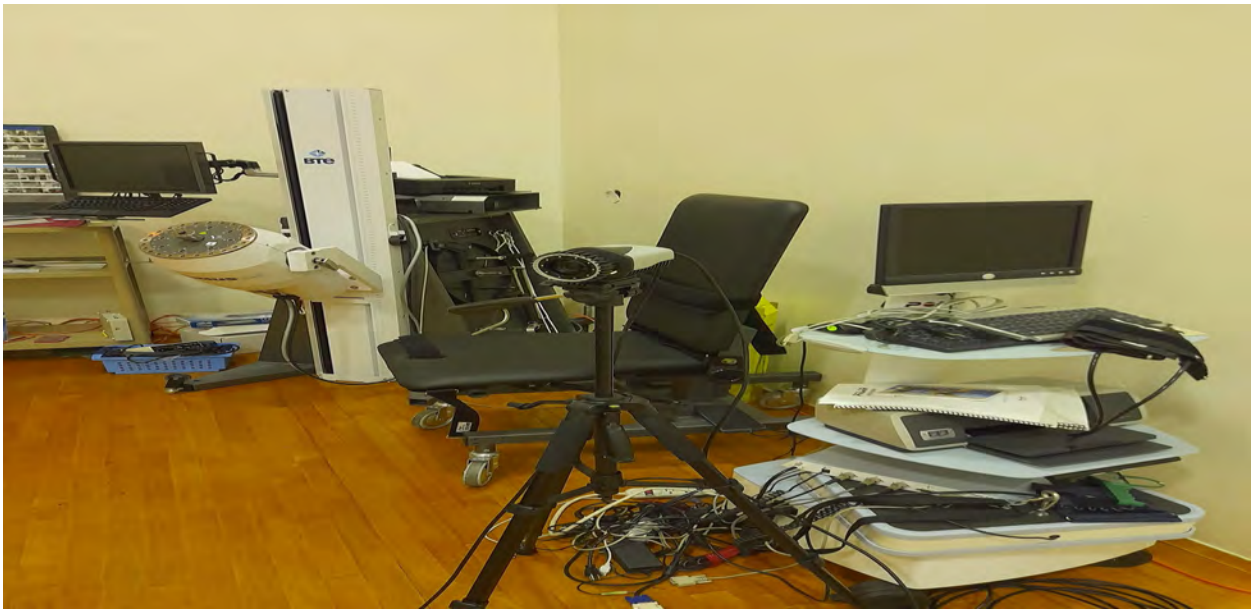


Figure 3.1: *Miqus* Video Camera

Each video camera is calibrated and synced to the other video cameras through daisy-chaining. Power and data are carried through a single cable which all leads to a single power

adapter.

The layout of the nine cameras was arranged such that the space allowed was maximized for any motion we wished to capture. The cameras were daisy-chained between themselves into a single box which carries the data to the PC on which we worked on. The PC is composed of an 11th Gen *Intel* i9-11900k CPU, 32 GB of ram, and a *NVIDIA* GeForce RTX 3090.

Figure 3.2: Motion Capture Lab

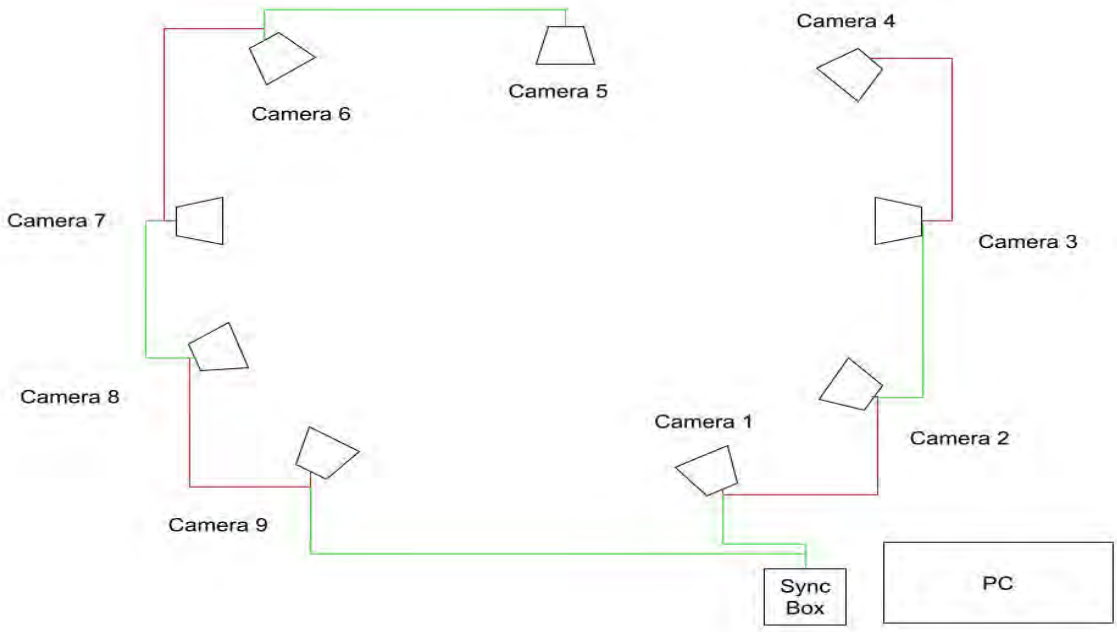


Figure 3.3: Motion Capture Lab Layout

In this figure, the red lines indicate the outgoing cables from one camera to another and the green lines indicate the incoming cables. Which all lead back to a *Sync Box*, which then transfers all data to the PC.

In the *Qualisys* software you must first define an ID of the subject of which is being recorded. This ID carries information such as height(cm) and weight(kg). For this experiment, the subject is 176.78 centimeters in height and 63.5 kilograms in weight. After the measurements of the subject are inserted you must then calibrate the cameras. This process is completed by utilizing an axis marker representing the X and Y axis and a calibration stick. Once the axis marker is set down to the desired direction of the axis the subject must take the *T-shaped* calibration stick and walk around the space that they think will be utilized in the motion and wave the stick in a *figure 8* motion from their head to feet covering the desired space for their motion. If the user needs to cover more space for their motion they will need to set the calibration recording length to a longer time frame. For our experiments, we found 15 seconds to be sufficient.

Once the video cameras are calibrated you can now record a motion. For these tests, we recorded motions of 5-second length at 1080p and 30 frames per second. If the motion is longer then the user could set the record length to a greater time frame. The resulting output shows the motion from the angle of each camera, giving a clear view of the motion from nine different angles.

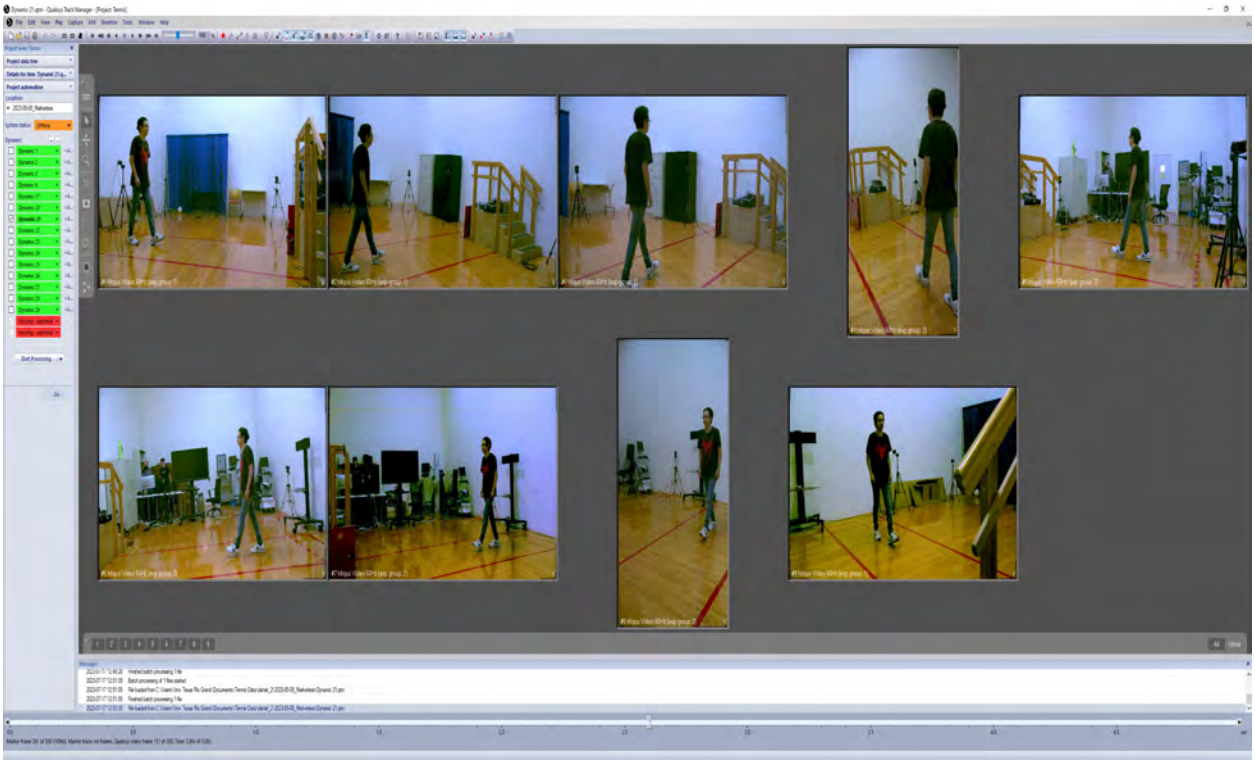


Figure 3.4: Qualisys UI

3.1.2 Theia3D

Once the videos are recorded they are then exported to a new software *Theia3D*. Using the exported video from *Qualisys*, *Theia3D* tracks the subject in the video and is able to compute the 3D position and orientation of each segment of the tracked person. With this tracking, a model is created with each detected joint of the subject being labeled. Once the video is then processed and labeled, the output is then saved as the *.fbx* file format.

Figure 3.5: Theia3d UI

Figure 3.6: Motions within the *Theia3D* software
Walk (Top)
Balance (Middle)
Jumping Jack (Bottom)

3.1.3 Blender

Once the *.fbx* file is obtained we must then re-target the motion to the provided model provided by *Lee* [11]. To do this we utilized the open source 3D computer graphics development tool, *Blender*. Within *Blender*, we can visualize the motion capture data obtained from the previous two software.

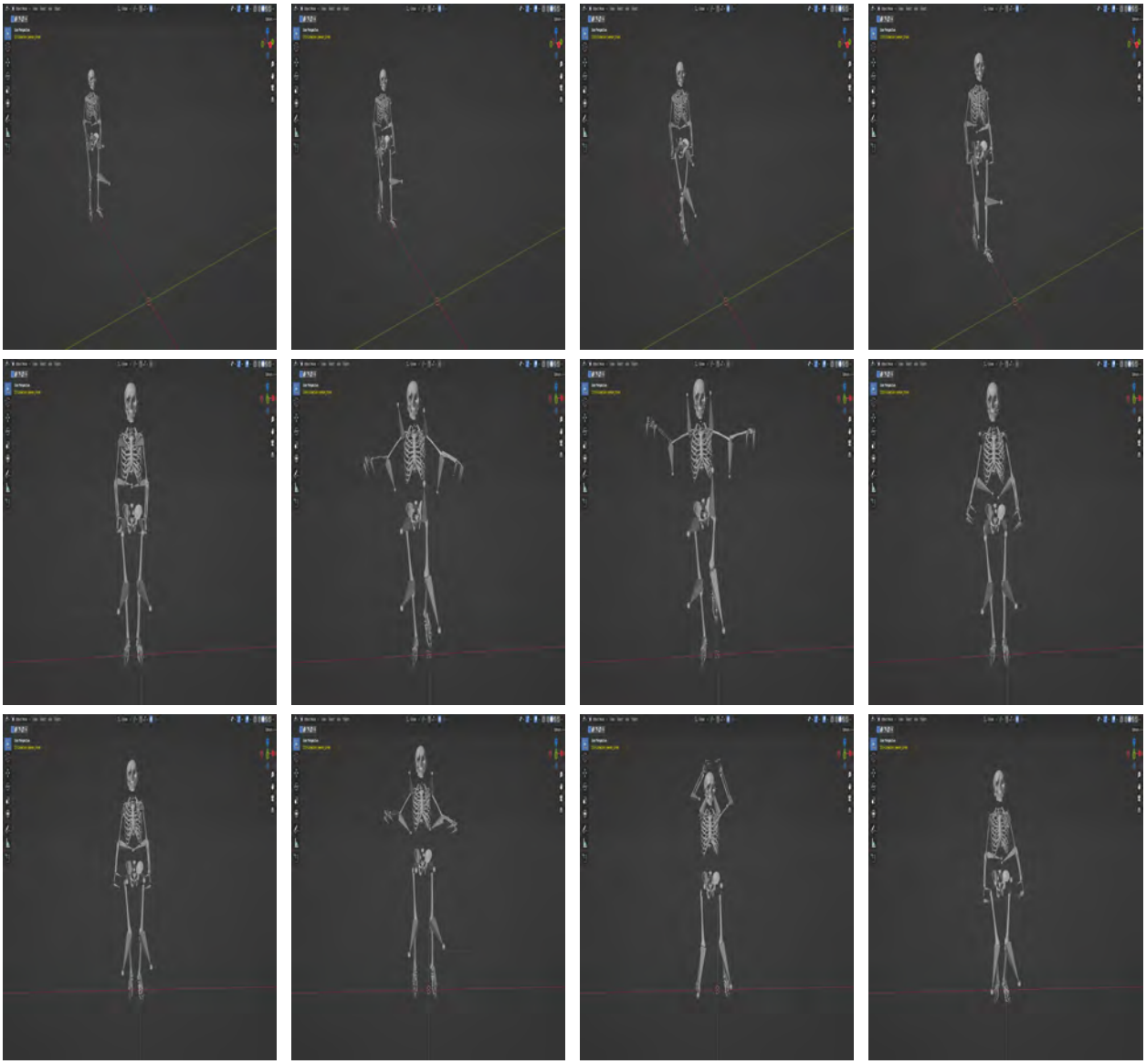


Figure 3.7: *Theia3D* motions in Blender
Walk (Top)
Balance (Middle)
Jumping Jack (Bottom)

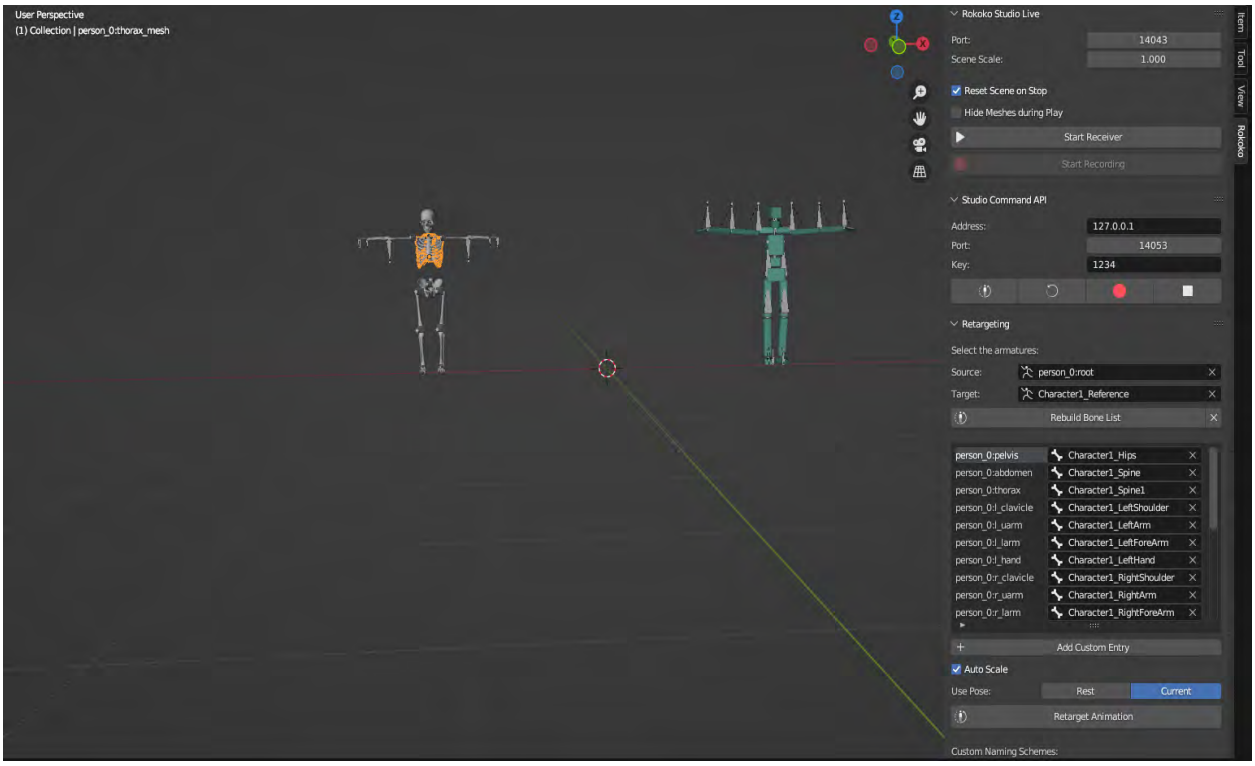


Figure 3.8: Retarget UI

We must re-target our original motion capture data to the model provided by *Lee* [11], due to naming conventions between the two models. After the motion is re-targeted onto the model from *Lee* [11]. We are now able to export that to a *.bvh* motion capture file to use in our experiments.

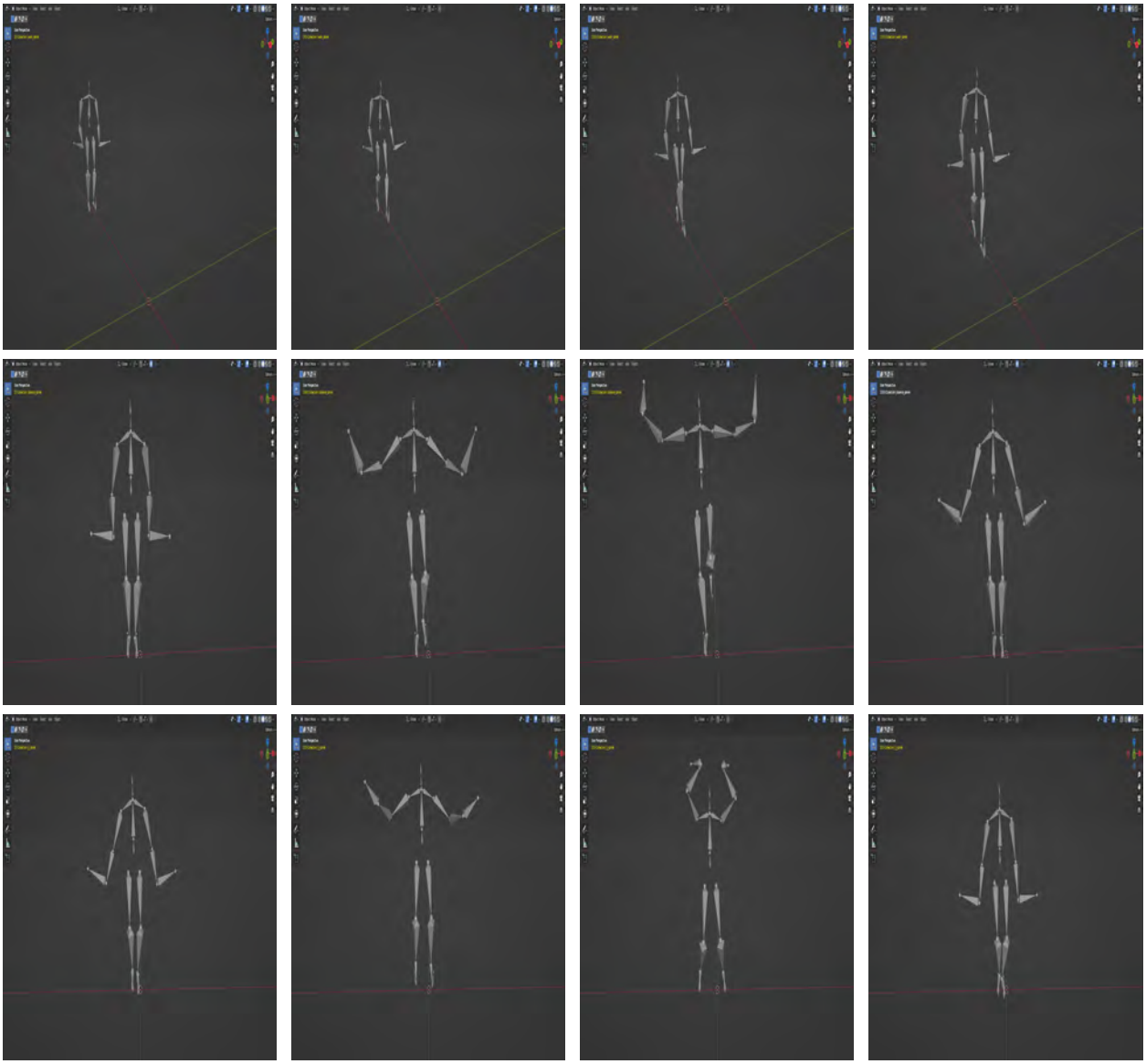


Figure 3.9: Re-targeted motions in Blender
Walk (Top)
Balance (Middle)
Jumping Jack (Bottom)

3.2 Scalable Muscle-Actuated Human Simulation and Control DRL Structure

This work follows the structure of *Deep Mimic* [14] in that the user provides motion capture data, and the goal is to learn a controller that produces motions that imitate that of the reference motion while achieving additional tasks. Here, the control policy $\pi(a|s$ controls the activation levels

of all muscles a at every time step, which leads the muscle-actuated model to imitate the reference motion. The states in this work are $s = s_{skeleton}, s_{muscle}$, which describe the kinematic, dynamic, and anatomic states of the skeleton and musculotendon units within the model.

Their work provides a two-level architecture composed of a DRL method for trajectory mimicking and a controller for learning muscle coordination. The trajectory mimicker is a random policy $\pi_\theta(u|s_{skelteon}$ that produces target poses u as output given a skeleton state $s_skeleton$, and the network parameters θ are optimized to be used in the DRL. PD servos $\ddot{q}_d = PD(u)$ determine the optimal acceleration, which is then passed to the second policy. The muscle coordinator $a = \pi_\psi(\ddot{q}_d, s_{muscle}$ is a deterministic policy that activates the muscles to produce desired muscles, ϕ is the network parameters determined by regression.

Learning a muscle-actuated control policy is three times slower compared to a torque-actuated control policy. The trajectory mimicker learns and operates at the pace of the reference motion data, which is typically 30 frames per second. While the muscle coordinator learns at the rate of the forward dynamics simulation, which could range from 900-1500 frames per second. Since the trajectory mimicker takes the state of the skeleton as input, the state of the muscles also affects the state of the skeleton. In addition, the muscle coordination policy depends on the states given by the trajectory mimicker. So, these two policies work together in order to achieve the maximum reward in the DRL.

Trajectory mimicking can be represented as the tuple $T = (S, U, P, R, \gamma, P_0)$, where the states $s \in$ S are the kinematic and dynamic states of the skeleton, which explores the environment described by properties $P : SXU \mapsto S$, skeletal states $u \in$ U is an action that the agent could take, and $r \in$ R is a reward, and p_0 is the initial state distribution. The agent explores its environment and its goal is to maximize its cumulative reward, which is represented by θ

$$\theta^* = argmax\mathbb{E}_{s_0,u_0,s_1,...}[\sum_{t=0}^{\infty} \gamma^t r(s_t)]$$

The state is defined as $s = (p, v, \phi)$, where p and v are the aggregate of the 3D position and

linear velocity of the model's bones, and $\phi \in [0,1]$ is the variable that represents the time elapsed during the reference motion. The actions by PD targets, map to the desired acceleration of the joints

$$\ddot{q}_d = k_p(u-q) - k_v\dot{q}$$

where $k_v and k_p$ are gains of the PD control.

The reward encourages the agent to imitate the reference motion while optionally completing a task simultaneously, which is dependent on the chosen reference motion. The reward is defined by the equation:

$$r = w_q r_q r_e + w_g r_g$$

Where, $r_q, r_e, and r_g$ represent pose estimation, end-effector imitation, and the task. $w_q and w_g$ are their respective weights.

The imitation and end-effector rewards encourage the model to match that of the reference motion, in terms of joint angles and the position of the end-effectors. q is the general position of the skeleton and p_e are the positions of the end-effectors (Right/Left Hand and Feet and Head).

$$r_q = exp(-\sigma_q \sum_j ||\hat{q}_j(\phi) \ominus q_j||^2)$$

$$r_e = exp(-\sigma_e \sum_j ||\hat{p}_e(\phi) - p_e||^2)$$

Here, the hat symbols represent the values given from the reference motion, j is the index of the joints, and e is the index of the end-effectors. The joint configurations are represented as quaternions and the difference is calculated by $q_1 \ominus q_2 = ln(q_2^- 1 q_1)$.

In the reward, the two imitation rewards are multiplied while the task reward is then added. This is because the two imitation rewards are closely coordinated, when the joints of the model are closely aligned to that of the reference motion then end-effectors should be as well, and vice versa. A high joint angle imitation reward leads to a high-end-effector imitation reward. However, the relation between imitation and task rewards is conflicting, so the weighted sum of the rewards

allows them to each be achieved while a compromise is found to compensate for the priority of the two.

Muscle coordination is the problem of deciding the activation levels of all muscle actuators so that the model can imitate the desired reference motion through joint accelerations. Let $a = \pi_{\psi}(\ddot{q}_d(u), s_{muscle})$ be a policy that is able to map the desired muscle activation to its acceleration. Here the muscle state $s_{muscle} = (vec(A), p)$ holds information that converts the muscle activation into forces in the joints such that $Aa + p$. The matrix A is vectorized to feed into the network policy, since A is fixed it is packed with non-zero values in the vector conversion.

Since the DRL for trajectory mimicking generates numerous episodes during training, sampling of tuples from the regression happens as well. During learning the algorithm switches between the trajectory mimicker and the muscle coordinator to collect tuples, which jointly updates the policy. This formulation allows the regression to take place without some ground truth values as input. Since the activation levels are the desired output of the regression model, some ground truth activation is required. Even though there is no ground truth, the loss function can determine how good the output is with its objective of muscle coordination. The objective is to find the terms of PD target u and the geometric alignments and physiological parameters of the musculotendon units (A and p).

CHAPTER IV

EXPERIMENTS

4.1 Training and Hardware

For this project, we utilized a multi-GPU machine running 4 1080ti *NVIDIA* GPUs. The operating system is UBUNTU 20.04. During training only one GPU was utilized, however, the network architecture from *Lee github* is written in *PyTorch*, which is a Python package for machine learning. This package allows for the utilization of multi-GPU training, which will be used in future experiments. Training time for each experiment varies based on the stylistic difficulty of the reference motion.

- Walk: 24hrs
- Balance: 28hrs
- Jumping Jack: 32hrs

Each test was trained for 15,000 total time steps with the same learning parameters:

- Learning Rate: 1e-4
- Batch Size: 128
- Buffer Size: 2048
- Clip ratio: 0.2

4.2 Baselines

In order to test how muscle deficiencies affected the model's ability to recreate the reference motion we first trained the model with its full allowed force. After training was complete we selected a muscle or muscle group and simulated the trained data with a deficiency of that group by a decrement of 20 percent at each test, while each other muscle group stayed at their full allowed force. The muscles tested for these experiments are some of the most major muscles in the upper/ lower body.

Table 4.1: Muscles and their Total Allowed Force

Muscle	Location	Original Allowed Force (N)
Gluteus Maximus	Buttock / Hip	370.520
Gluteus Medius	Buttock / Hip	549.90
Iliopsoas	Inner Hip	
Psoas Major		239.850
Psoas Major		239.850
Iliacus		207.30
Hamstrings	Back of Thigh	
Semitendinosus		301.90
Semimembranosus		581.350
Biceps Femoris		705.20
Rectus Femoris	Thigh	424.40
Vasti	Thigh	
Vastus Medialis		721.850
Vastus Lateralis		1127.70
Vastus Intermedius		512.10
Tibialis Anterior	Calf	673.70
Gastrocnemius	Calf	
Gastrocnemius Lateralis		606.40
Gastrocnemius Medialis		1308.00
Supraspinatus	Shoulder	1000.00
Deltoid	Shoulder	1000.00
Trapezius	Upper Shoulder and Neck	1000.00
Serratus Anterior	Side of Rib Cage	1000.00

4.3 Walk

For this example, we will determine if the model was successfully able to complete the reference motion and at what point the muscle deficiencies if any hinder the gait cycle.

4.3.1 Gait Cycle

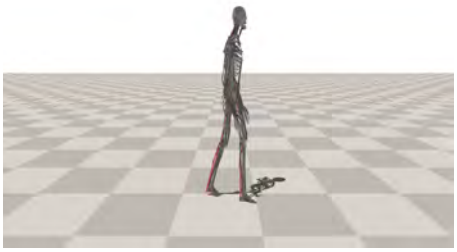
The gait cycle is the time from when the heel of one foot touches the ground to the time the same heel touches the ground again or a repetitive pattern involving steps and stride.

The cycle includes these steps:

(a) Heel Strike

(b) Loading Response

(c) Mid Stance

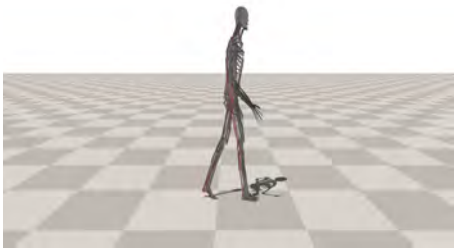


(d) Terminal Stance

(e) Pre Swing

(f) Initial Swing

(g) Mid Swing



(h) Terminal Swing

Figure 4.1: Gait Cycle Phases

- Gluteus Maximus
- Gluteus Medius
- Hamstrings
- Tibialis Anterior
- Quadriceps Femoris
- Plantar Flexors (Gastrocnemius/Soleus)

When muscles, had their allowed forces reduced simultaneously the model could tolerate the deficiencies up to 60% of its allowed force. When deficiencies of 40% or more were applied gait was no longer possible. When 60% of allowed force was allowed gait was possible however the model struggled in the Mid Stance phase of the gait cycle. When allowed 40% of its total weakness we can see that the model is no longer able to complete the reference motion, where the flexion of the hips to move forward was not sufficient and the power that is usually allowed to the Plantar Flexors was not allowed so balance was not possible.

When testing the individual muscles/ muscle groups, the results varied. Some muscles could tolerate the decrease in muscle force and were able to complete the gait at 20% of its allowed force but at a cost of higher muscle activation. In our tests, the performance of the model in replicating the reference motion was most sensitive in the Gluteus Medius and the Plantar Flexors. In these muscle groups gait was no longer when they were allowed only 40% of their original force.

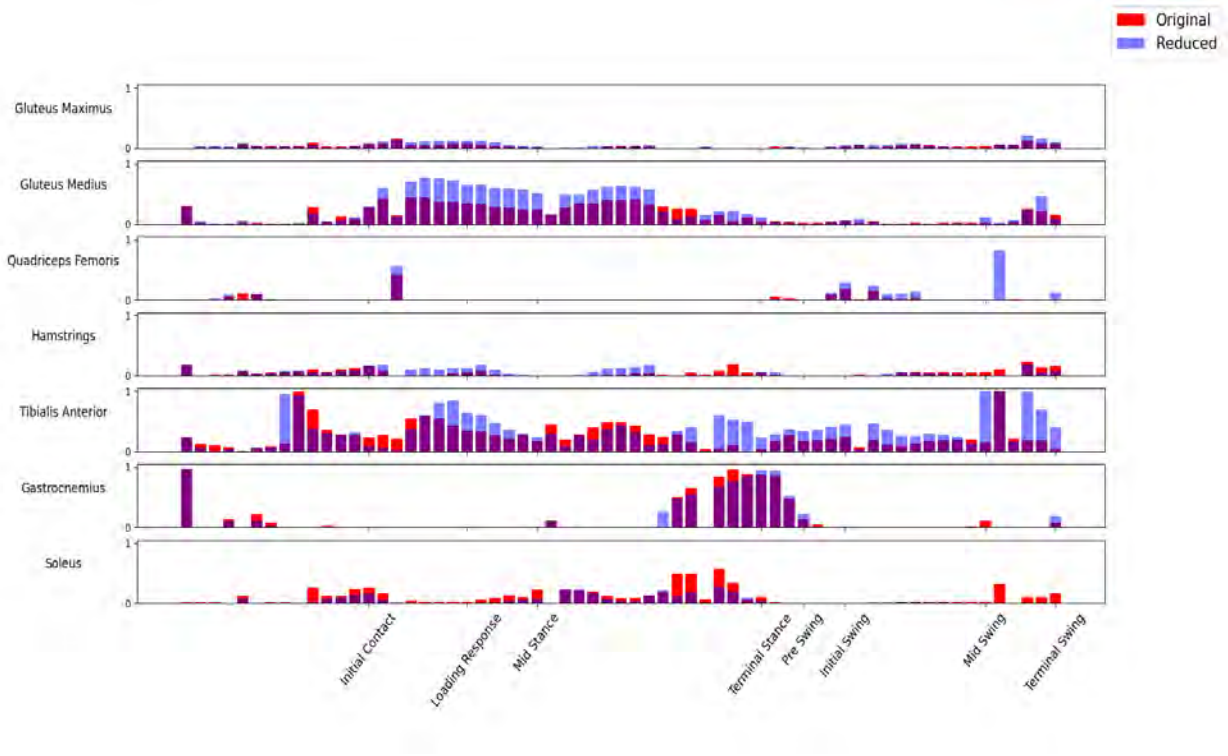


Figure 4.2: Activation levels when Gluteus Medius is allowed 60% original force.

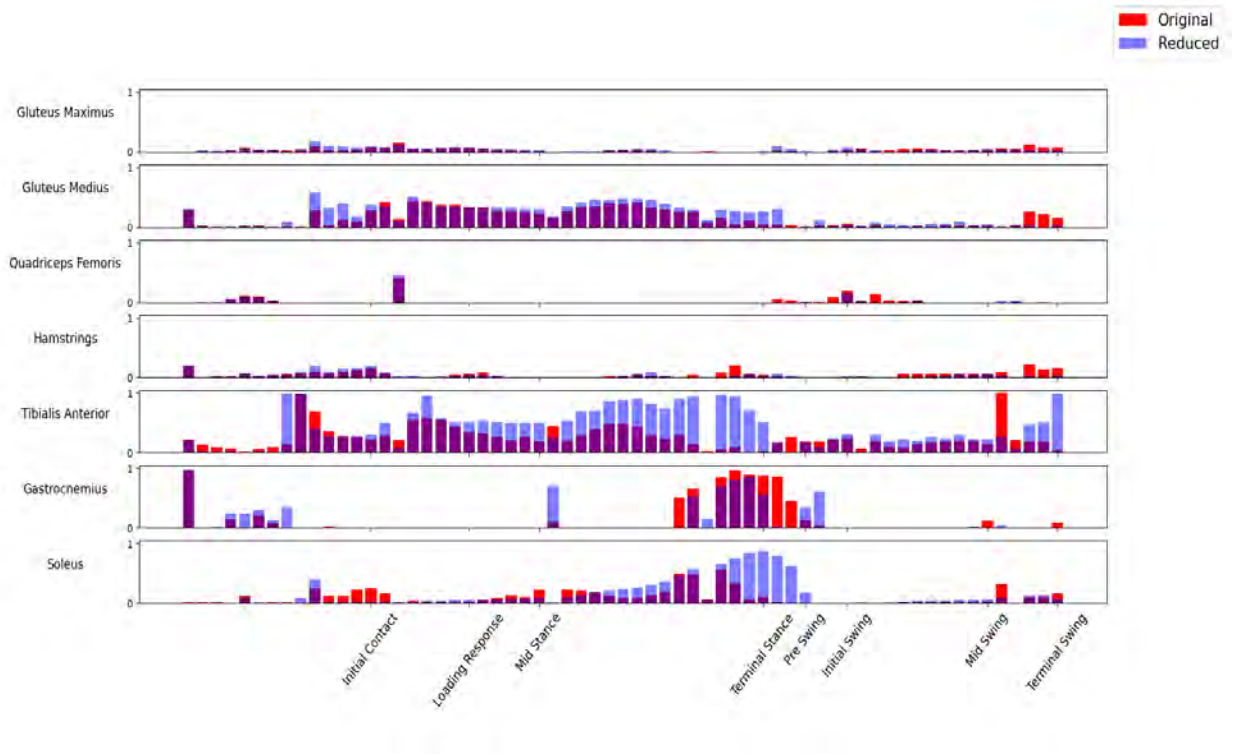


Figure 4.3: Activation levels when Plantar Flexors is allowed 60% original force.

When a muscle was weakened compensation for that weakness occurred both in the muscle that had its original force depleted and the other muscles tested. In the test where all muscles had their total allowed force decreased the activation levels of each muscle increased. This is a similar result to when each muscle had its force decreased individually, in the cases of the Gluteus Medius and the Plantar Flexors, the other muscles increased their activation the most.

4.4 Balance

For testing the Balance motion the model would have to follow the guidelines above and "survive" for 65 frames during simulation, "survive" in this case indicates that the model is balancing one leg and has not fallen over.

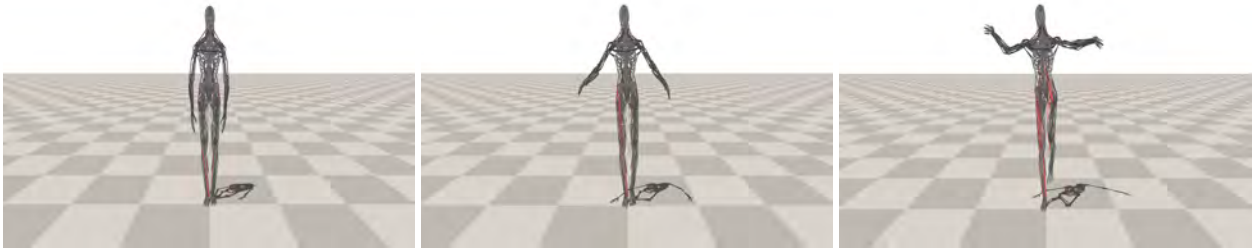


Figure 4.4: Balance

In order for the model to complete the leg lift part of the motion these muscles must be activated: Hamstrings, Psoas Major, Iliacus, and the Gastrocnemius, [3]. If these muscles are activated more in the left leg in this case compared to the right, that will mean that the model has lifted the left leg, and the model is currently balancing on one leg.

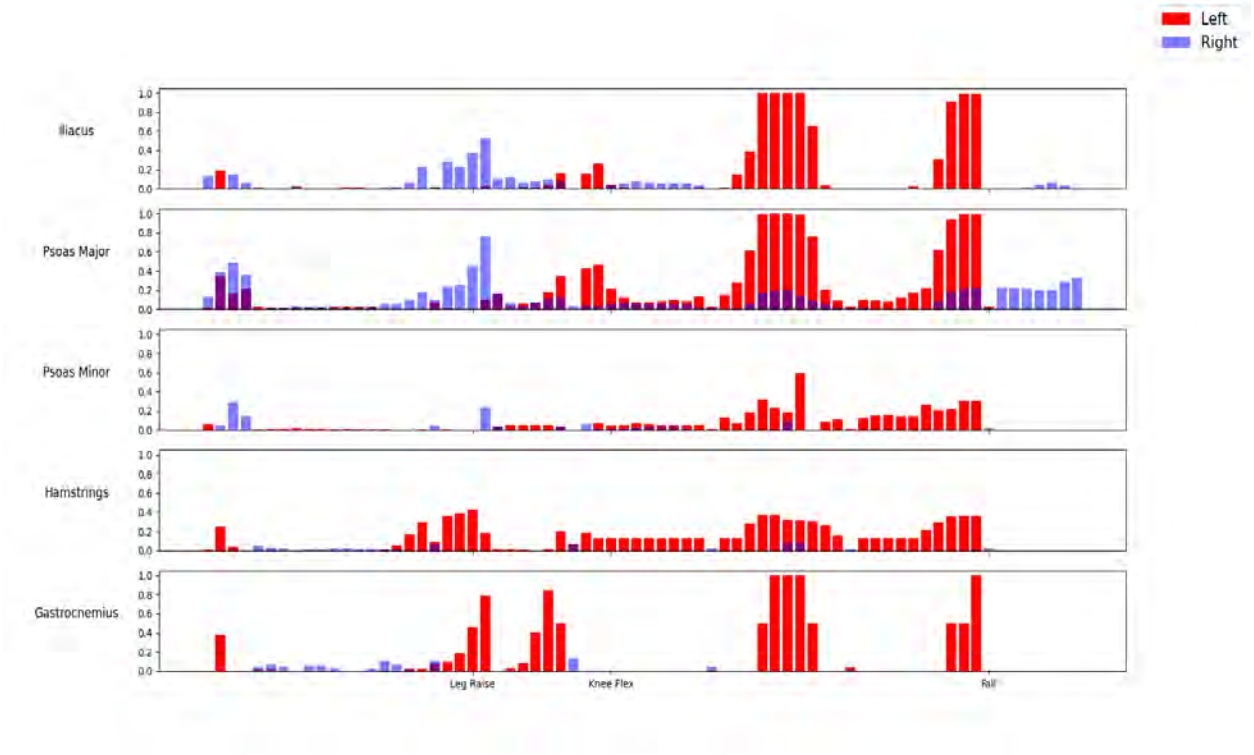


Figure 4.5: Activation levels between
right and left leg
when left knee is flexed.

4.4.1 Left leg

The model was only able to tolerate a 40% decrease in these muscles before it was unable to replicate the reference motion. in all muscles the model saw an increase in activation levels during the frames in which the left leg was being raised and when the knee was flexed.

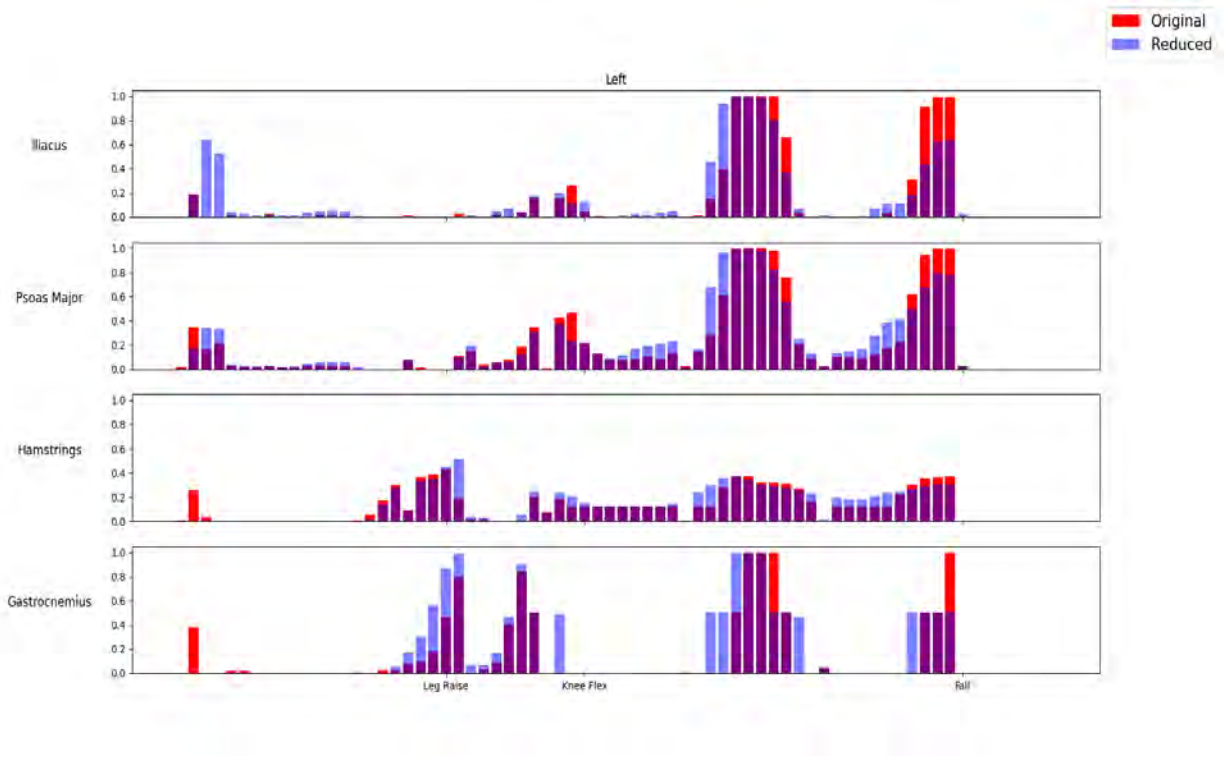


Figure 4.6: Left leg activation levels when
is allowed 60% original force

4.4.2 Right leg

When testing the right leg, the muscles chosen were the Plantar Flexors, Tibialis Anterior [10], Gluteus Medius, Gluteus Maximus, and the Vastus Medialis [13]. These muscles were much less tolerant to the decrease in total allowed force, failing to complete the reference motion when allowed 80% of the total allowed force, only being "alive" for 40 frames.

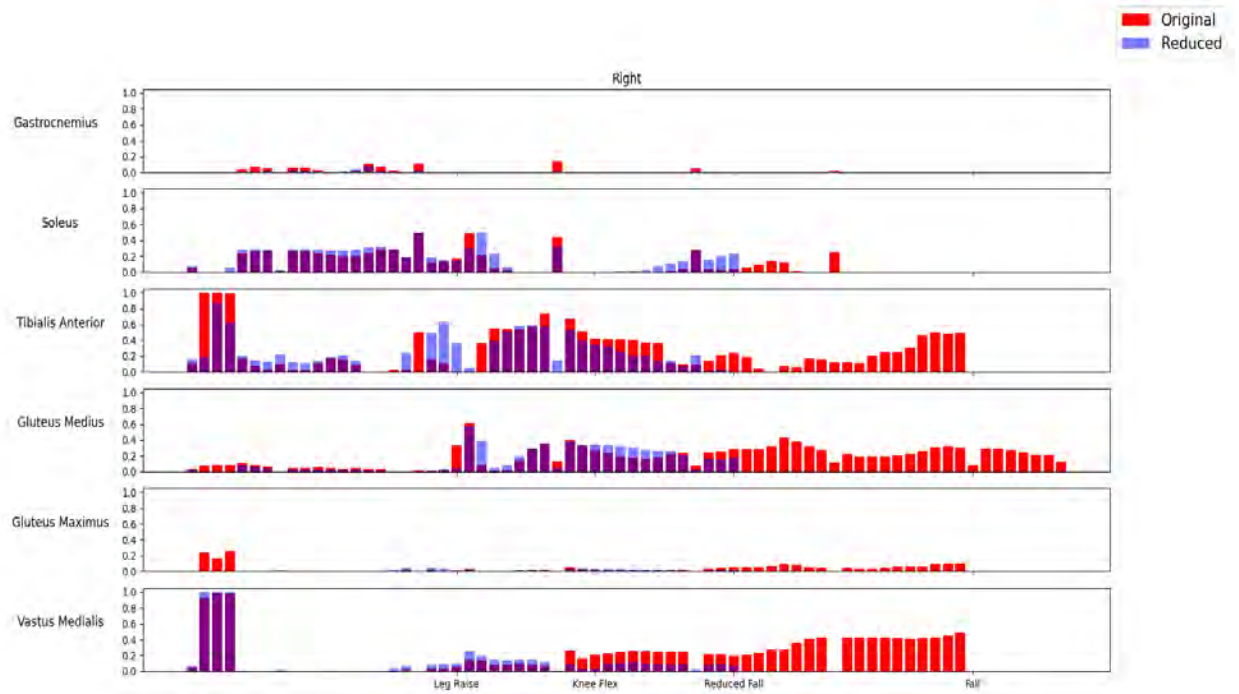


Figure 4.7: Right leg activation levels
when is allowed 80% original force

4.5 Jumping Jacks

For this experiment we will determine if the model is able to successfully reproduce a Jumping Jack reference motion and within that motion are three individual motions: hip abduction, jumping, and shoulder abduction. Abduction occurs when a body part moves away from the midline of the body and adduction is the opposite, bringing the body part closer to the body.

Figure 4.8: Jumping Jack

4.5.1 Shouder Abduction

To test how muscle deficiencies affect this motion in this experiment we decided to focus on the muscles that are responsible for shoulder abduction [7]. These muscles are the Supraspinatus, Deltoid, Trapezius, and Serratus Anterior.

These muscles were tolerant to the decrease in allowed force, completing the reference motion even when allowed only 20% of their original forces. However, the cost of replicating this motion increased their overall activation levels in all muscles.

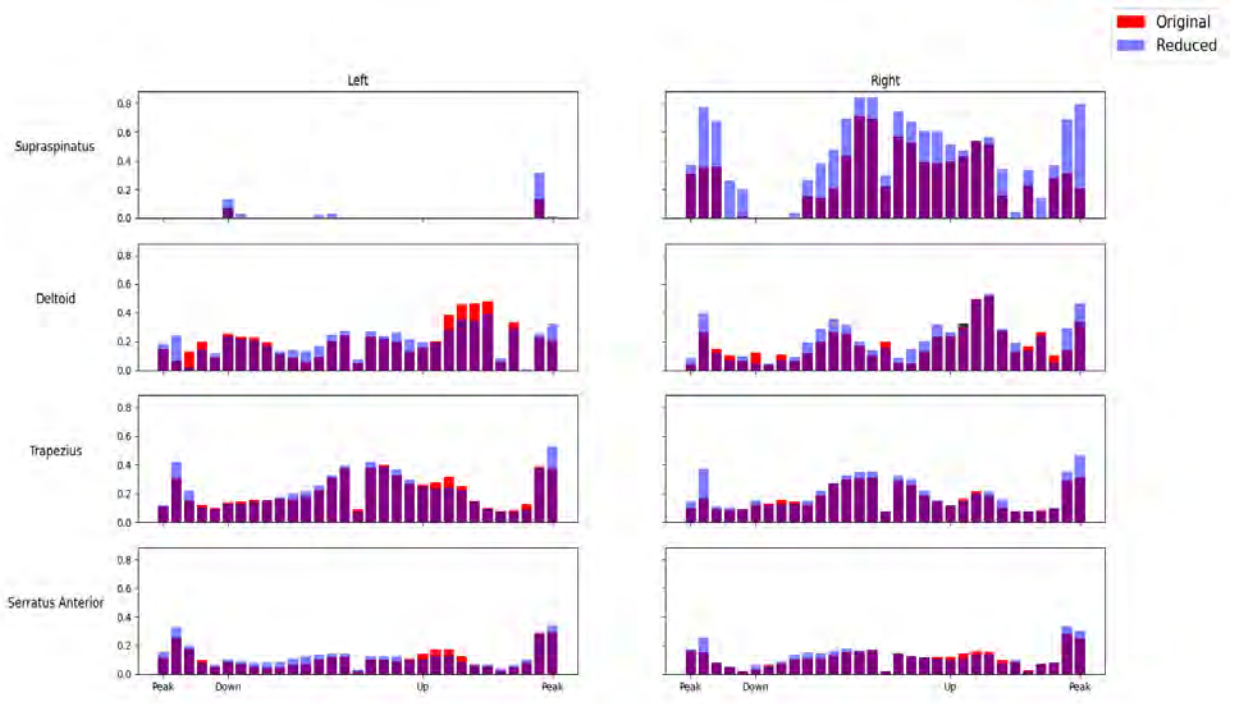


Figure 4.9: Arm activation levels when
is allowed 20% original force

4.5.2 Hip abduction / Jumping

The muscles tested for this experiment were the Plantar Flexors, Gluteus Medius, and Tibialis Anterior. These muscles failed to complete the hip abduction/adduction motion of bringing the legs together and then apart again, as well as jump when allowed 60% of their original force.

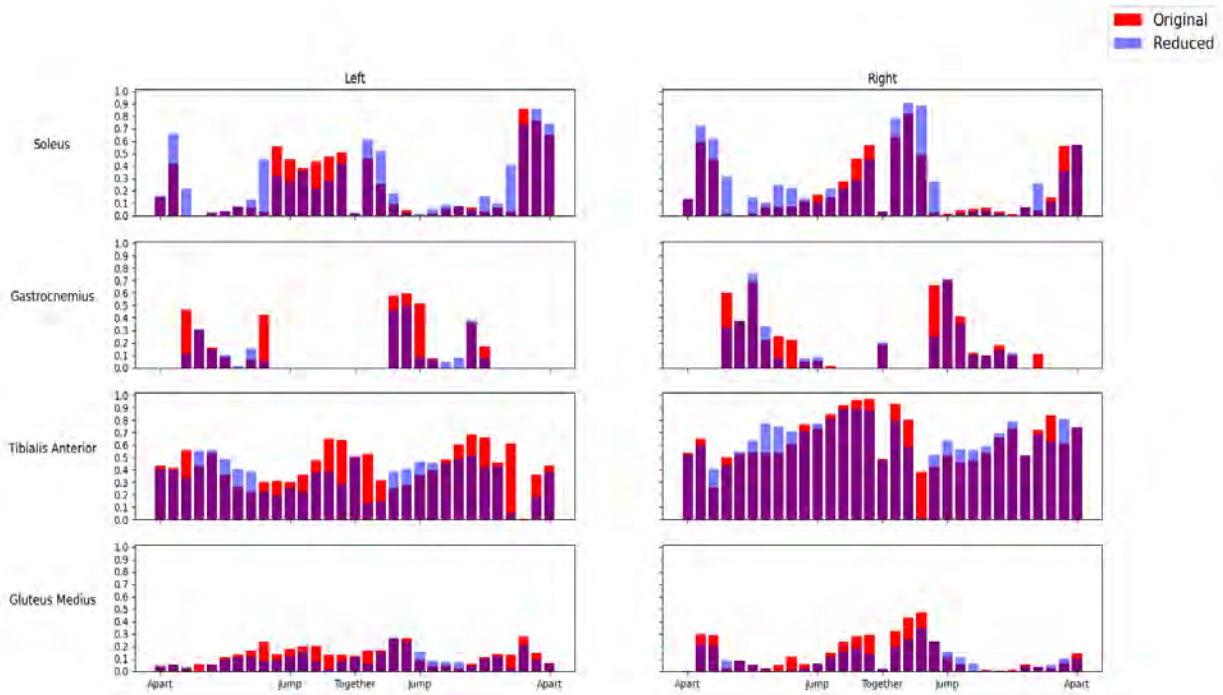


Figure 4.10: Leg activation levels when
is allowed 60% original force

4.6 Results

When the model was tasked with replicating the walking motion, in most cases was it able to do so. When allowed 60% of the original force most muscles were able to complete the reference motion. However in the experiments where 40% or lower of the original allowed force was allowed the model was either partially able to complete the reference motion or was unsuccessful completely. The muscles that seemed to affect the model's ability to complete the motion were the Gluteus Medius and the Soleus, these muscles saw increases in their activation levels, and the others in which compensated for their weakness as well.

The balance motion was much more prone to failure when the muscles in the right leg saw decreases in their allowed force. The model was not able to complete the motion when the force was decreased by 20%, again the Plantar Flexors and Gluteus Medius seeing increased levels of activation, throughout the duration in which it was able to be considered "alive".

The arms in the jumping jack motion seemed to be the least affected motion that was tested. The arms were able to complete the motion of abduction and adduction at every decrease in activation levels. However, every muscle that was tested saw increases in activation levels. The jumping and hip abduction sub motion was extremely affected by the lack of allowed force, this decrease did not allow the model to jump or move its legs.

CHAPTER V

CONCLUSION

In this work, a pipeline was developed that allows users to evaluate the tolerance that a human may have to reduced muscle force in order to complete a certain motion, through reinforcement learning. In a series of experiments we incrementally decreased the allowed muscle force to specific muscle groups and analyzed at what threshold was the model not able to complete the reference motion. The amount of weakness applied to the muscles and its effect on the reference motion differed between muscle groups and the motion. The experiments showed that when walking the model was not able to complete the motion in most cases when a 40% percent decrease to the originally allowed force was given to the muscles. In other cases, the model was able to complete the reference motion, despite a 60-80% decrease in allowed muscle forces.

When the model had muscles weakened the muscles compensated for this by increasing their activation levels and by increasing the activation levels of the other muscles that were not affected by the depletion as well. So, the model was able to complete the motion in these cases, but it was at the cost of an increase in muscle activation, which could lead to increased fatigue or in extreme cases damage to the muscle itself.

This pipeline gives insight into the minimum threshold at which three different motions are feasible. This pipeline could be used to test other motions, which the user can record through motion capture data as well. These results could be used to design programs that allow the person that is experiencing muscle strength deficiencies an opportunity to target those weakened muscles and work to improve the strength of the muscle to a point that the motion is then able to be replicated.

REFERENCES

[1] D. AKIMOV, *Distributed soft actor-critic with multivariate reward representation and knowledge distillation*, 2020.

[2] Y. BENGIO, J. LOURADOUR, R. COLLOBERT, AND J. WESTON, *Curriculum learning*, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, Association for Computing Machinery, p. 41âĂŞ48.

[3] K. CHAKRAVARTY, D. CHATTERJEE, R. K. DAS, S. R. TRIPATHY, AND A. SINHA, *Analysis of muscle activation in lower extremity for static balance*, 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), (2017).

[4] 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 (2007), pp. 1940–1950.

[5] S. FUJIMOTO, H. VAN HOOF, AND D. MEGER, *Addressing function approximation error in actor-critic methods*, 2018.

[6] T. HAARNOJA, A. ZHOU, P. ABBEEL, AND S. LEVINE, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, 2018.

[7] B. B. JOSHUA H. LAM.

[8] S. KOLESNIKOV AND V. KHRULKOV, *Sample efficient ensemble learning with catalyst.rl*, 2020.

[9] T. KOMURA, Y. SHINAGAWA, AND T. L. KUNII, *Creating and retargetting motion by the musculoskeletal human body model*, The Visual Computer, 16 (2000), p. 254âĂŞ270.

[10] K. G. LAUDNER AND M. M. KOSCHNITZKY, *Ankle muscle activation when using the both sides utilized (bosu) balance trainer*, Journal of Strength and Conditioning Research, 24 (2010), p. 218âĂŞ222.

[11] S. LEE, M. PARK, K. LEE, AND J. LEE, *Scalable muscle-actuated human simulation and control*, ACM Trans. Graph., 38 (2019).

[12] 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*, 2019.

[13] B. NORRIS AND E. TRUDELLE-JACKSON, *Hip- and thigh-muscle activation during the star excursion balance test*, Journal of Sport Rehabilitation, 20 (2011), p. 428âĂŞ441.

[14] X. B. PENG, P. ABBEEL, S. LEVINE, AND M. VAN DE PANNE, *Deepmimic*, ACM Transactions on Graphics, 37 (2018), pp. 1–14.

[15] A. RAJAGOPAL, C. L. DEMBIA, M. S. DEMERS, D. D. DELP, J. L. HICKS, AND S. L. DELP, *Full-body musculoskeletal model for muscle-driven simulation of human gait*, IEEE Transactions on Biomedical Engineering, 63 (2016), pp. 2068–2079.

[16] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, 2017.

[17] S. SONG, Å. KIDZIÅĎSKI, X. B. 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*, Journal of NeuroEngineering and Rehabilitation, 18 (2021).

[18] B. ZHOU, H. ZENG, F. WANG, Y. LI, AND H. TIAN, *Efficient and robust reinforcement learning with uncertainty-based value expansion*, 2019.

[19] ÅĄUKASZ KIDZIÅĎSKI, S. P. MOHANTY, C. ONG, Z. HUANG, S. ZHOU, A. PECHENKO, A. STELMASZCZYK, P. JAROSIK, M. PAVLOV, S. KOLESNIKOV, S. PLIS, Z. CHEN, Z. ZHANG, J. CHEN, J. SHI, Z. ZHENG, C. YUAN, Z. LIN, H. MICHALEWSKI, P. MIÅĆOÅŻ, B. OSIÅĎSKI, A. MELNIK, M. SCHILLING, H. RITTER, S. CARROLL, J. HICKS, S. LEVINE, M. SALATHÃL', AND S. DELP, *Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments*, 2018.

BIOGRAPHICAL SKETCH

Daniel Castillo earned his Bachelor's degree in Computer Science from the University of Texas Rio Grande Valley in August 2021. Additionally, he earned a Master's degree in Computer Science from the University of Texas Rio Grande Valley as well in August 2023. Daniel can be contacted via email.

- Email: danielcastillo129@gmail.com