MASTER THESIS

EXOSKELETON CONTROLLER DESIGN FOR CYBATHLON BY PARALLEL REINFORCEMENT LEARNING

Sibolt Mellema

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF BIOMECHANICAL ENGINEERING

EXAMINATION COMMITTEE
prof. dr. ir. H. van der Kooij
dr. ir. A.Q.L. Keemink
dr. ir. J.J. de Jong

DOCUMENT NUMBER
BE - 897

UNIVERSITY OF TWENTE.

10-JAN-2023

Summary

Patients with Spinal Cord Injuries (SCI) experience physical disabilities due to the lost ability to walk. Wearable robotics, like an Exoskeleton can restore parts of the lost ability to walk. Challenges for an Exoskeleton is to perform different tasks such as stair ascending/descending, tilted paths or steppingstones. An exoskeleton gains the ability to perform the task by their controller. Current controller designs can be time-consuming due to complex systems, time-invariant and hard-to-measure quantities.
Controller design by Reinforcement Learning may tackle the problems of time-invariant and hard-to-measure quantities. Parallel Reinforcement Learning shows the potential to tackle the time-consuming problem of the controller design. The focus in this work is to utilize parallel Reinforcement Learning for controller design. The software Isaac Gym provides a platform for parallel Reinforcement Learning, which uses a model of the Exoskeleton to move through the modelled environment.
The design method for the controller has two stages. The first stage is to generate a model of the Exoskeleton worn by a human. The goal of this design is to prove the capabilities of locomotion for the Exoskeleton model. The second stage includes an environment with obstacles. The obstacles resemble different tasks which the Exoskeleton must complete. This stage includes four obstacles form the Cybathlon course for Exoskeletons (a race in which Exoskeletons completes different daily tasks). The design method includes an evaluation of the resulting Reinforced Learned controllers. The evaluation consists of two sections, a qualitative and a quantitative evaluation; to evaluate the gait of the exoskeleton and quantitative comparison between locomotion.
The results have two designs for the Exoskeletons' controller. The first design has a controller for locomotion on a smooth surface with human-like gait. The second design of the controller has the aim to overcome the obstacle. Unfortunately, the design did not show qualitative and quantitative results to overcome the obstacles. Both controllers have a training time under two hours, achieving a good controller within an hour.
This works shows promising result for controller design with Reinforcement Learning for an Exoskeleton worn by humans. Further research can improve the design of the Reinforcement Learned controller by adjusting the reward function, variable setpoints for the body's velocity or implementing Recurrent Neural Networks in the Reinforcement Learning method.

Table of Content

Chapter 1 Introduction

Worldwide, many people experience different physical disabilities. One of the most impactful and life-changing disability is Spinal Cord Injury (SCI), a condition where patients have little to no functions in their lower limb. Every year, around 250.000 – 500.000 people around the globe suffer from SCI, reflecting the burden of the condition (WHO, 2013).
Wearable robotics could help SCI patients by restoring their mobility and thereby improving their quality of life. An example of a wearable robot is the exoskeleton, an electromechanical device that aims to enhance the physical mobility of a patient. Exoskeletons are either commercially available or still in research and developmental stage. An example of an exoskeleton specifically developed to support SCI patients is The Symbitron exoskeleton. (Meijneke et al., 2021).
This specific exoskeleton's goal is to perform multiple tasks, including ascending and descending stairs, walking on uneven surfaces, and avoiding approaching objects and/or people. ETH Zurich started a competition for wearable robotics called the Cybathlon. In multiple disciplines, Cybathlon stimulates the research and development of wearable robotics. The exoskeleton discipline challenges the participants to face different obstacles and terrains (Jaeger & Jaeger, 2022).

Controller Options for Exoskeletons

The exoskeleton needs a controller to function and perform the tasks. The controller of the Symbitron exoskeleton has three layers. The first layer is task control, which recognizes the task the exoskeleton should perform. The second layer is human-robot interaction control, which controls the forces between the robot and the human. The third layer is joint control, which controls the torque / position of the degree of freedom (Anam & Al-Jumaily, 2012; Copaci, 2018; Meijneke et al., 2021). The design of these controllers is based on knowledge of the system to reach the desired state. When systems are complex, hard-to-measure parameters or time-variant, controller design can be challenging or time-consuming (Brysona & Ho, 1975).

Another approach for controller design can overcome the deficiencies. Reinforcement Learning (RL) could be a suitable approach. RL benefits from the increase in computing power and results in satisfactory results and high accuracy. The current application of RL in wearable robotics is task recognition (Amamcherla et al., 2018; Briouza et al., 2021; Zheng et al., 2020). RL is already applicated in other areas than wearable robotics. For animal-like and humanoid robotics, a RL application have shown successful control of a set of hardware (Rudin et al., 2021).

Despite the successful results of RL, it tends to demand a substantial amount of computing resources. RL tries to find a control policy by simulation or on a physical setup. RL uses trial and error to control the actions of one robot (either physical or simulated). The sequential behaviour of RL and complex robotic systems result in long training times or the need for a large computer cluster (Makoviychuk et al., 2021). The company NVIDIA has developed Isaac Gym for parallel RL on the graphics card (GPU) to tackle these drawbacks. Current GPU hardware can run the simulation and training of thousands of robots simultaneously (Makoviychuk et al., 2021).

Research Overview

In this work, the aim is to investigate the abilities of RL in controlling an exoskeleton. This research states two important requirements. First, the controller should perform a human-like gait and second the controller should overcome different obstacles from the Cybathlon courses in a random order.

This research has the following outline: it starts with background information in Chapter 2, focusing on RL and RL algorithms. Chapter 3 discusses the method and set-up, addressing the implementation of RL, obstacles design, exoskeleton design and evaluation method. Chapter 4 will discuss the results of the evaluation criteria, followed by the discussion of the result in Chapter 5. In chapter 6, the final conclusions and future recommendations will be stated.

Chapter 2 Background

In order to be able to understand the methods outlined in chapter 3, this chapter will give background information on reinforcement learning, the Isaac Gym software and the design requirements for the models and simulations.

2.1 Reinforcement Learning

The goal of reinforcement learning (RL) is to learn a control task where the agent controls an actor in an environment. RL is one of the three pillars of machine learning and focuses on solving sequential decision problems, also known as a Markov Decision Process (MDP). A MDP models the interaction between an agent and the environment (Sutton & Barton, 2015). An agent in RL is the part which learns and interacts with the environment. The interacting environment is unknown to the agent and the agent cannot control (parts of) the environment (Sutton & Barton, 2015).

2.1.1 Components of RL

A MDP includes the following components: the set of states (S), the set of actions (A) and the reward (R). The mapping from a state to an action is called a policy π, also known as control strategy. RL learns a control task by maximizing the reward R (Sutton & Barton, 2015; Xu et al., 2014). A property of MDP is that the future of the control process only depends on the current observation. So, the history of the control decisions and state evolutions do not influence the next decision (François-lavet et al., 2018). Appendix A.1 contains more information on MDP.
In many cases, the model of an actor in a MDP process is not known a priori. Therefore, RL learns a model by means of function approximation. The approximation is a combination of basic functions with corresponding weights. Often, the selected basic function is Deep Learning (DL) approximation. DL approximation uses an artificial neural network for the approximation (Learning, 2017)(Baheti, 2022), see A.2 for more information.

An agent can maximize its reward R based on two optimization categories. These categories are the value-based and policy-based optimizations. Where the value-based optimizations focus on a good estimation of a value-function to find a policy, the policy-based optimizations estimate the policy by itself. To maximize the reward R, the policy-based optimizations have an objective function: the expected return. An optimization method with the policy-based optimization is the gradient ascent, called policy gradient (PG) approach. The policy gradient

$$\nabla J(\theta) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s), s\sim\eta_{\pi_\theta}(\cdot)}\left[Q_{\pi_\theta}(s,a)\nabla\log\pi_\theta\,(a|s)\right] \qquad (1)$$

shows the gradient of a general objective function. $J(\theta)$ is the expected return, Q_{π_θ} is the Q-function under the parameterized policy π_θ, which 2.1.2 will discuss. $\nabla\log\pi_\theta(a|s)$ is a function that scores the performance of the policy: $\nabla\pi_\theta/\pi_\theta$ (Bhagat et al., 2019). It is preferred to use policy-based methods over value-based methods, because the function approximation in value-based algorithms can overestimate the value-function. However, function approximations are necessary to implement RL to train a controller, as both the action and state space are continuous (François-lavet et al., 2018; Schulman et al., 2017), see Appendix A3 for more information.

2.1.2 Reward Functions

The reward R is the summation of intermediate rewards r_t at time steps t. The reward function calculates the numerical r_t at t , see Appendix A1. As the reward function calculates a reward at every timestep, there is a scaling factor for the reward value: the discount factor gamma, γ. This factor sets the influence of future rewards obtained by the agent. The higher the factor, the more impact future rewards will have (Vamvoudakis et al., 2021). During training, RL estimates the future reward of the agent under policy π. The discounted accumulated reward

$$R_\pi(s) = \mathbb{E}\left[\sum\nolimits_{t\geq=0} \gamma^t R(s_t, a_t)\,|a_t \sim \pi(\cdot\,|s_t), s_0\right] \tag{2}$$

is the estimation of the obtained reward (Morales & Zaragoza, 2011; Nguyen et al., 2017). $\mathbb{E}$ is the expectation of the reward, the bar | means that the action a_t follows from policy π and current state s_t, starting in state s_0. The discounted accumulated reward can define the state-value function (V-function) and the action-value function (Q-function). The V-function

$$V_\pi(s) = \mathbb{E}\left[\sum\nolimits_{t\geq 0} \gamma^t R(s_t, a_t, s_{t+1})|a_t{\sim}\pi(\cdot\,|s_t), s_0 = s\right] \tag{3}$$

is the estimation of the discounted accumulated reward when the agent starts in state $s = s_0$ and follows policy π. The Q value

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R(s_t, a_t, s_{t+1})|a_t{\sim}\pi(\cdot\,|s_t), a_0 = a, s_0 = s\right] \tag{4}$$

estimates the discounted accumulated reward with the agent starting in state s with action a selected and following policy π afterwards. The difference between the Q- and V-function results in a relative value for action a_t. In other words, the advantage function

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) \tag{5}$$

expresses the relative contribution of action a_t to the cumulative reward. The agent will follow the policy π afterwards (Briouza et al., 2021; Byun et al., 2020; Wang et al., 2019). In RL, an algorithm learns by trial and error based on the feedback on the selected actions. Often, the cumulative reward (2) is the feedback for the agent on the performance of the current policy and the agent adapts its policy to maximize the cumulative reward.

2.1.3 Proximal Policy Optimization

Within the policy-based optimizations, there are multiple algorithms. The used algorithm in this work is the Proximal Policy Optimization (PPO). PPO is model-free, online (optimization of the current policy) and can manage stochastic policies. This algorithm uses the PG optimization to optimize an objective function. The goal in PPO is to update the policy to maximize the expected reward

$$\eta(\pi) = \mathbb{E}_{s,a}[Q_\pi(s,a)] \tag{6}$$

for a long-term policy. The expected reward (6) is a general objective function. The expected reward depends on s, the sequence of states and a, the sequence of actions following policy π (Zhu & Rosendo, 2021). Most PG methods use an estimator for the policy gradient and use this estimator as input for a stochastic gradient ascent (SGA) algorithm (Schulman et al., 2017). The importance of

using an estimator in combination with a SGA algorithm is that it leads to a reduced computational burden. This increases the iteration speed and makes it possible to only use a subset of the data set. The drawback of using a SGA is that the convergence rate is lower (Gao et al., 2021).
There are different policy gradient estimators, the most common gradient estimator in RL is

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t)\, \hat{A}_t\right] \tag{7}$$

$\hat{\mathbb{E}}_t$ is the empirical average of the expectation and $\hat{A}_t$ is an estimation of the advantage function (5). For the estimation of the PG, the SGA algorithm requires an input to calculate the gradient estimator. The PG algorithm needs an objective function which can be designed by the user itself. However, the gradient of the objective function must be a gradient estimator. For example, an objective function for (7) is: $\hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t|s_t)\, \hat{A}_t\right]$ (Melo & Maximo, 2019; Schulman et al., 2017).
The objective function in PPO is different compared to the common gradient estimator (7). Instead of using the score function in the objective function, PPO uses a probability ratio

$$r_t(\theta) = \pi_\theta(a_t, s_t)/\pi_{\theta_{old}}(a_t, s_t) \tag{8}$$

for the action. The probability ratio is the probability of action a_t under a new policy divided by the probability of the action in the old policy (assuming the agent is in the same state) (Melo & Maximo, 2019; Schulman et al., 2017). The objective function is: $L^{PPO} = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$. The gradient of this objective function has the score function $\nabla \log \pi_\theta(a|s)$ with the probability ratio as a scaling factor (Schulman et al., 2017). To increase the stability and reliability of the algorithm, PPO clips the objective function

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{9}$$

between an interval (Melo & Maximo, 2019; Schulman et al., 2017). The clipping value ϵ determines the interval of clipping and is a hyperparameter of the algorithm. The policies in PPO are stochastic functions and give a probability for action A in state S (Schulman et al., 2017; Wang et al., 2019). The optimization takes place in the parameter domain, so the algorithm uses parameters gradients.

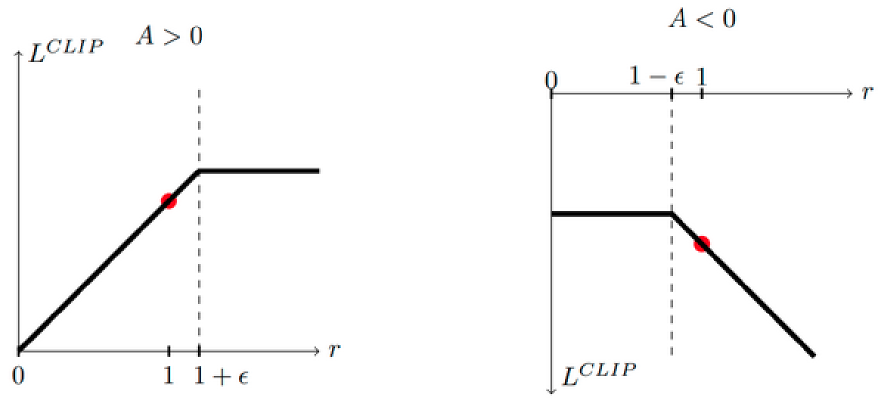


Figure 1: The clipping action of the objective function in the PPO algorithm. When the action becomes more likely (A>0), the clipping limits the objective function to prevent a too large update step (Schulman et al., 2017).

Figure 1 shows the objective function with clipping. When the advantage function (5) is positive, the PPO algorithm clips the probability function when it is higher than one plus the clipping value. A positive advantage function indicates that the new policy results in good action (increase of expected return). Clipping the objective function limits the update in the parameters. It prevents a possible lower expected return of the policy, as the parameter gradients have no information on the policy increase. In case a good action has a low probability ratio, the algorithm is free to undo the step with the calculated value.

The same holds for actions with negative advantage values (decrease of expected return), except that these actions result in mirrored clipping. When the advantage function is negative, PPO clips the probability function when it is lower than one minus the clipping value (Schulman et al., 2017; Wang et al., 2019). The author refers to the literature on TRPO and PPO for the mathematics to ensure safe update regions.
The objective function (9) depends on an estimator of the advantage function (5) and introduces variance in the estimation of the expected reward. A high variance can result in an overestimation of the estimated expected reward, which is undesirable. There are multiple options for variance reduction, the most common is the subtraction of a baseline from the expected reward (Bhagat et al., 2019; Briouza et al., 2021; Lillicrap et al., 2016). Often, the V-value is subtracted from the expected return, which is γ -just, meaning it does not introduce any bias. The estimator of the advantage function (5) becomes the temporal difference (TD) error

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{10}$$

function by subtraction of the V-value from the estimated expected return (Lillicrap et al., 2016). Lillicrap et al. state that a generalized advantage estimator (GAE)

$$\hat{A}^{GAE}{}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1}^V \tag{11}$$

depends on TD error. The estimator of the advantage function in PPO can be this GAE (Schulman et al., 2017; Wang et al., 2019). A positive TD error shows that the selected action increases the accumulated discounted reward. A negative TD error shows that the action results in a lower accumulated discounted reward.

2.1.4 Actor-Critic

The PPO algorithm optimizes the policy by approximating the objective function with parameters. The objective function includes a GAE, however, the GAE has two hyperparameters: the discount value γ and GAE estimator λ. The hyperparameter λ controls the compromise between the variance and the bias of GAE, where $\lambda = 1$ has a good accuracy but high variance and $\lambda = 0$ has low variance but introduce a significant bias when the estimation is not exactly the Value-function, see Figure 3 for a visualization of bias and variance (Lillicrap et al., 2016).

A structure to approximate the policy and the GAE is an Actor-Critic. The Actor in the structure approximate the policy objective given in Section 2.1.3 Proximal Policy Optimization. The goal of the Actor is to find a policy which maximizes the expected return. The Critic performs a second optimization step for the value-function. The value-function of the Critic will be the temporal difference, to criticize the resulting policy of the Actor (Li et al., 2018). Figure 2 shows the structure of the Actor-Critic method.

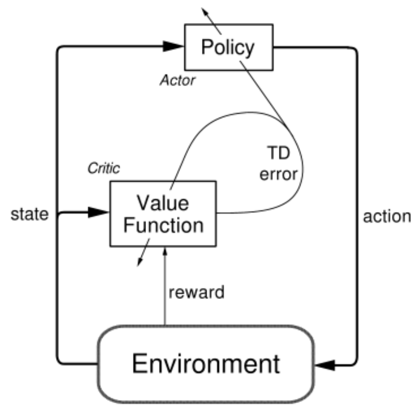


Figure 2. Structure of the Actor-Critic method. The Critic evaluates the performance of chosen action (TD error), and the agent tries to find the optimal policy. From (Lillicrap et al., 2016).

Both the actor and the critic need a function approximation to approximate the policy and a value function, respectively. With a neural network as a function approximator, two implementations are possible. The first implementation is a neural network that shares the hidden layers which split at the output layer into two (one for the actor and one for the critic). The second implementation splits the input layer into two lines resulting in two sets of hidden layers.

The benefit of a shared network between the Actor and the Critic is a quicker convergence, as only one set of hidden layers should be trained. However, the benefit of using two lines of hidden layers is that the Critic can learn other patterns which is not relevant for the Actor and vice versa (Li et al., 2018). The network can also split at any hidden layer to weaken the trade-off; however, this functionality has no implementation in Isaac Gym (Makoviychuk et al., 2021).

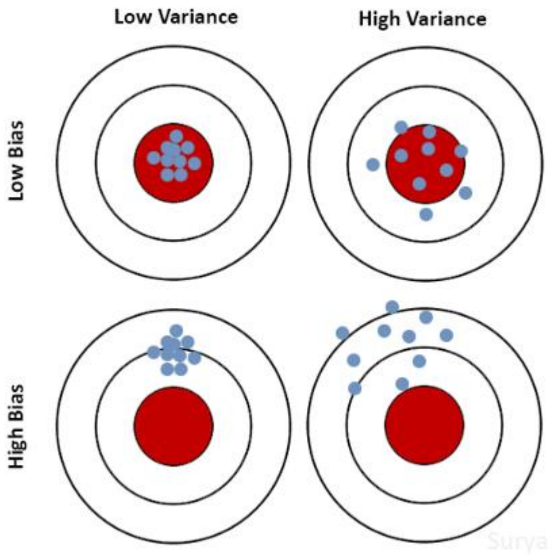
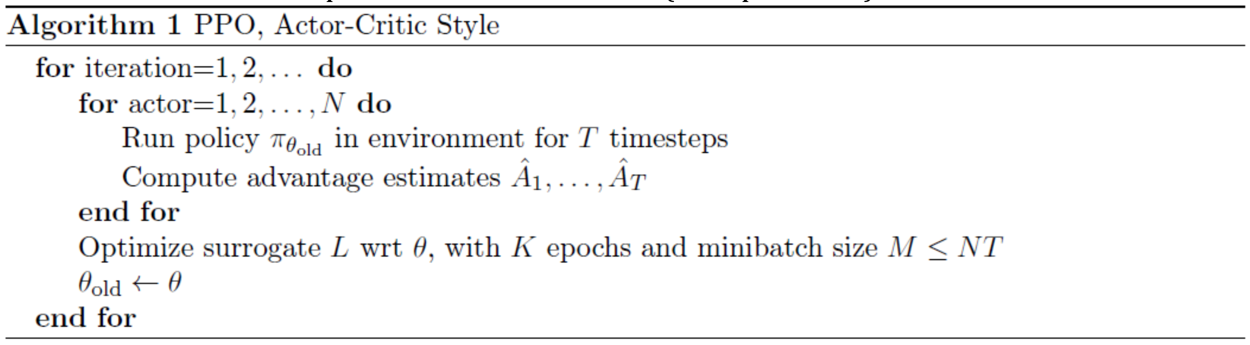


Figure 3. Visualization of bias and variance. A low bias means that the estimation is close to a real value. A low variance means that the estimations are closely spread between each (Gutta, 2020).

2.1.5 PPO, Actor-Critic Style

The implementation of the PPO algorithm has two steps. The first step is the collection of data for T timesteps and N rollouts. The collected data is the input for the calculations of the advantage estimates of each timestep. At each timestep, the current policy generates a set of actions and the algorithm observes the new states and corresponding cumulative reward (Byun et al., 2020; Melo & Maximo, 2019; Schulman et al., 2017).
The second step is the optimization of the policy. To achieve this, the algorithm calculates the estimated advantages on the sampled cumulative rewards. With the estimated advantages, the algorithm constructs the loss function (9) for the parameter update (Melo & Maximo, 2019; Schulman et al., 2017).

The algorithm updates parameters θ with the loss function in K epochs, from now on called mini epoch. In each mini epoch, the algorithm constructs the loss function based on a mini batch of samples, which is smaller than the collected batch (Byun et al., 2020; Schulman et al., 2017). An important remark is that the surrogate loss function is an empirical average over a finite (mini) batch of samples and results in a single average of the long-term reward (Schulman et al., 2017).
The clipping of the objective function prevents the multiple gradient steps to result in a policy with probability ratios larger than the clipping values (Melo & Maximo, 2019; Schulman et al., 2017; Zhu & Rosendo, 2021). The old policy $\pi_{\theta,old}$ in probability ratio (8) remains the reference policy in the equation as the algorithm has collected the batch for this policy. Figure 4shows the pseudo code for the PPO algorithm with an Actor-Critic structure (Schulman et al., 2017). The next section (2.2) will discuss the second line of pseudocode in more detail (multiple actors).



```
Algorithm 1 PPO, Actor-Critic Style
for iteration=$1, 2, \ldots$ do
    for actor=$1, 2, \ldots, N$ do
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$
    $\theta_{\text{old}} \leftarrow \theta$
end for
```

Figure 4. Pseudo code for the PPO algorithm. The algorithm alternates between sampling and optimization. During sampling, the algorithm collects data on a roll-out with the current policy. The collected data is the input for the advantage estimations. The optimization of parameters is with the objective function from (9) (Schulman et al., 2017).

2.2 Hardware Requirements

This section will discuss the implementation of the actor and the environment in Isaac Gym. Isaac Gym needs information on the actor and the environment. The agent is solely the neural network trained by RL. The actor is the combination of the agent and the hardware which the agent can control.

2.2.1 Actor

The actor in this work is the Symbitron Exoskeleton, see Figure 5. The Symbitron Exoskeleton has multiple configurations, depending on which joints the exoskeleton actuates. In this work, the exoskeleton actuates two degrees of freedom (DOF) at the hip joints (abduction/adduction and flexion/extension), one at the knee joint (flexion/extension) and one on the ankle joint (plantar and dorsal flexion). Each leg of the exoskeleton has four actuated joints and one non-actuated joint. The non-actuated joint is the inversion/eversion of the ankle (rotation around the length of the foot) (Meijneke et al., 2021).

A computer model of the Symbitron exoskeleton includes the weight, dimensions and inertia parameters. Two more parameters are necessary, namely the torque limits of the motor and the joint limit. All joint actuators have a peak torque of 102 Nm, except the knee joint. The actuator of the knee joint has a higher peak torque, around 150 Nm (Meijneke et al., 2021).

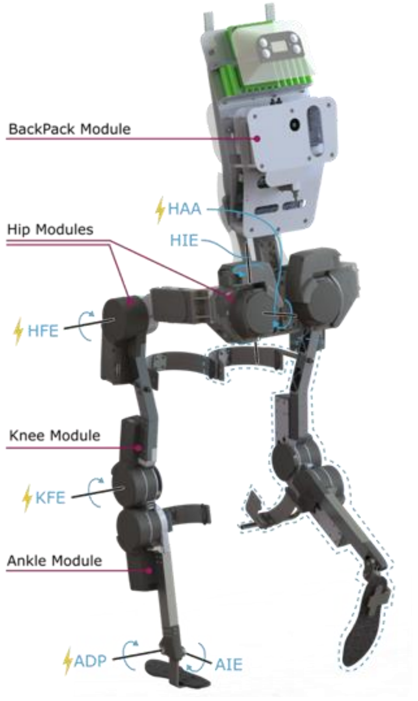


Figure 5. The exoskeleton which is the actor for the RL problem. This figure gives an overview of the structure of the hardware. The yellow lightning symbols show the powered joints (Meijneke et al., 2021).

The goal of the exoskeleton is to increase a person's mobility. There are limits on the exoskeleton's joint rotations, just like how humans experience. The limit of joint angles depends on the orientation of multiple lower limb joints. The assumption for joint limits is that the limit of a joint is reached when all other joints have zero rotation. It prevents lower limb configurations outside of the human flexibility.

The hip joint can perform multiple rotations, including abduction/adduction (hip AA) and flexion/extension (hip FE). The limits for the hip AA are 0.70 to –0.70 radian (positive abduction, negative adduction) (*Physiopedia*, 2022). The limits for the hip FE joint is 1.3 radians for flexion, and the extension limit is 0.5 radians (Grimmer et al., 2020). The flexion of the knee joint is 2.35 radians, and the extension is 0 radians (*Physiopedia*, 2022). Finally, the maximal dorsiflexion of the ankle is 0.17 radians, and the maximal plantarflexion is 0.52 radians (*Physiopedia*, 2022).

2.2.2 Environment

The goal of the policy is to control the exoskeleton and be able to overcome obstacles of the Cybathlon. This goal sets a requirement for the environment. There are ten obstacles in the Cybathlon course for exoskeletons. There are obstacles which require the exoskeleton pilot to use his arms and others require evasive manoeuvres. This work focuses on static obstacles and obstacles with forward movements only.

The environment includes four obstacles from the list: Free walking, stairs ascending/descending, steppingstones, and tilted path, see Figure 66. The next chapter (Chapter 3 Method and Setup) will discuss the implementation of the obstacles. The free walking obstacle is a flat surface of 3m. In this environment, the goal of the exoskeleton is to walk without the use of crutches. The stairs obstacle has four steps with a step height of 17cm. The step width differs for ascending and descending. For ascending, the step width is 28cm and for descending the step width is 35cm. The steppingstone obstacle has nine circular stones which all have a diameter of 25cm. The variation between the step length and the step width is either 40cm or 55cm. Either the left or right foot will begin and a steppingstone cannot be stepped on twice. The final obstacle has a tilted surface of 12.5-degree, horizontal length of 1.48m, vertical length of 0.33m, and the length is 4m. The red surface in the tilted path cannot be stepped on by the exoskeleton (Jaeger & Jaeger, 2022).

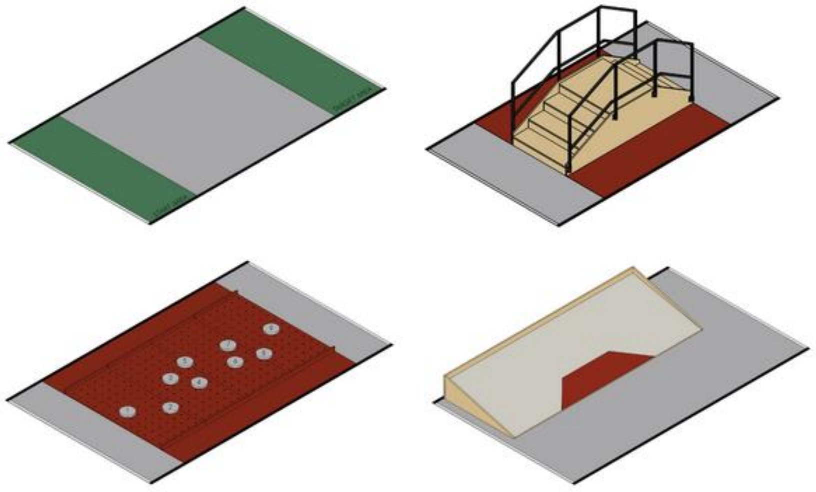


Figure 6: The obstacles from the Cybathlon race, implemented in this work. In general, the exoskeleton has to move in a forward motion to overcome the obstacles (adapted from(Jaeger & Jaeger, 2022)*).*

2.3 Isaac Gym

This section discusses the software program, Isaac Gym, which facilitates a framework to construct RL in parallel. Besides aspects of RL, Isaac gym requires information of the agent and environment, formulating the design requirements.

2.3.1 Software

Simulation software makes it possible to increase the iteration speed and the safety of the learning process. Without a simulator, reinforcement learning should learn each iteration in the real world, increasing the risk of damaging the robot, the environment or both, see Figure 7 (Makoviychuk et al., 2021; Rudin et al., 2021).

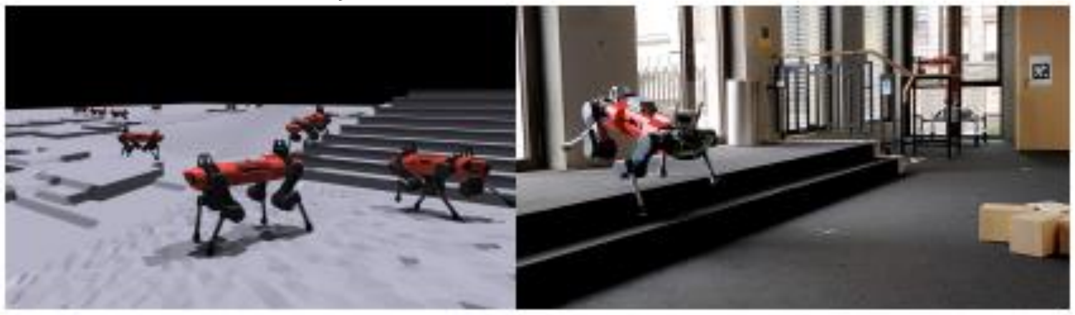


Figure 7. The difference between simulation and real-world application. The left picture shows training in a simulated environment. The left picture shows testing of the policy in the real world. Without simulation software, the training is in the real world, increasing the possibility to damage the robot, environment, or both (Makoviychuk et al., 2021).

Isaac Gym makes it possible to use reinforcement learning and simulation on a graphics card (GPU) in a computer, see Figure 7 (Makoviychuk et al., 2021; Rudin et al., 2021). A single implementation of the PPO algorithm has one environment and one actor. The drawback of a single actor-environment is that for high dimensional problems the sample complexity of exploration results in exceptional training times (Anonymous, 2020).

It means that this single actor needs to explore all the state and action spaces which requires an enormous number of iterations/explorations. Parallel RL simulation might be a solution to counter this exploration problem. Parallel RL simulations collect data from multiple actors-environments runs (with N actors) to collect a larger batch of data from the current policy (Anonymous, 2020; Makoviychuk et al., 2021). It drastically speeds up the collection of data (multiple simulation with one actor or one run of simulations with multiple actors).

At the start of RL developments, all the steps were run on Central Processing Unit (CPU) in a computer. With development in the Graphics Processing Unit (GPU), the algorithm speed could be increased to transfer the optimization step to the GPU, as the GPU architecture increases the parameter optimization of an ANN (Anonymous, 2020; Makoviychuk et al., 2021).

With the interference path (collecting data with the current policy) on the CPU and the training path (the policy optimization step) on the GPU a bottleneck raises in the RL process. This bottleneck is the data transfer between the CPU and the GPU. A possible solution is a large cluster of CPU cores to generate a large batch for the GPU, hiding the bottleneck of data transfer (Anonymous, 2020; Makoviychuk et al., 2021). Another solution is to transfer the interference path to the GPU as well, where Isaac Gym comes into play, see Figure 8. NVIDIA developed the software Isaac Gym to run the whole RL process on the GPU. It overcomes the data transfer bottleneck. It shows promising results as it achieved locomotion within 20 minutes on a single GPU (Makoviychuk et al., 2021).

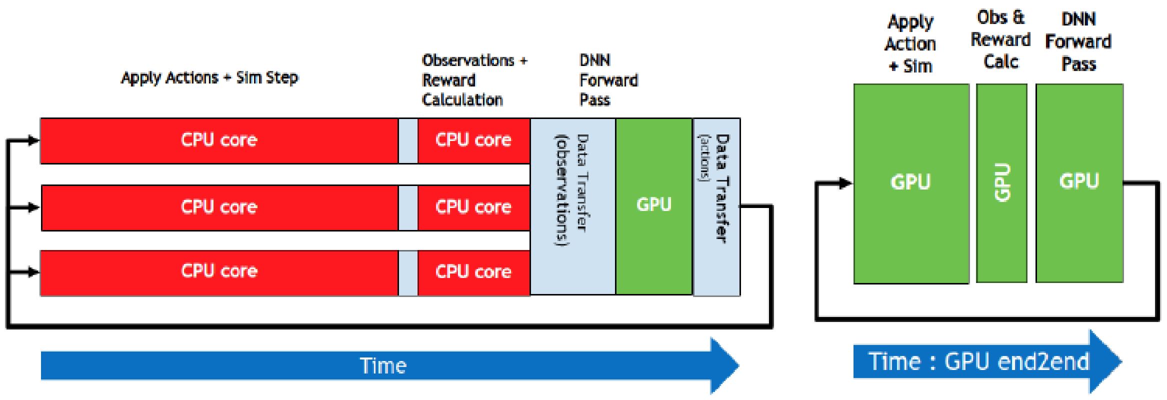


Figure 8. Difference between GPU accelerated parallel learning (left) and parallel learning all on the GPU (right). The blue blocks show the data transfer during the process. Besides the lack of data transfer, the application of actions and the simulation process reduces simulation time (Makoviychuk et al., 2021).

The implementation of Isaac Gym and the PPO algorithm is as follows:

1. Initially, the policy parameters are random, and they generate the first policy
2. Isaac Gym generates one environment with N actors (where $N \geq 1000$)
3. Simulate N rollouts of T timesteps in Isaac Gym
4. Compute advantage estimates for T timestep for each N rollouts
5. Optimization of (9) with respect to parameters for K mini epochs and minibatch $\leq$ NT
6. Update policy for all N actors in simulation
7. Repeat from step 3 until iteration $\geq$ maximum iteration or reward higher than the score to beat.

2.3.2 Definitions Isaac Gym

Up till now, the background discussed the RL, the RL algorithm PPO, the Actor-Critic structure, Isaac Gym, the hardware, and the selected obstacles. This section gives an overview of terms used in coming chapters and their definitions.

- ***Actor***: The combination of hardware and the agent. It measures the state of the actor and applies the actions from the policy to the hardware of the actor.
- ***Agent***: The controller for the exoskeleton. It determines the actions based on the observations and the current policy.
- ***Batch size:*** The batch is the data set of actions, states, and rewards for the rollout of T timesteps for all initialized actors.
- ***Clipping value*** ϵ***:*** The clipping value limits the change in the probability of an action in a certain state. It prevents substantial changes in the policy and mimics a 'trust region'.
- ***Discount factor*** γ*:* A scaling factor for future rewards. It let the designer choose the importance of future rewards to the cumulative reward. A high value of γ (~ 1) means that future rewards are important.
- ***Environment***: The environment is the world the actor moves in. The environment is passive and cannot be controlled directly by the agent. Indirect control is possible by interaction between the actor and part of the environment (robotic hand lifting a wooden block)
- ***Epoch***: The number of initializations of actors during training or testing. When 3072 actors are in parallel with a setting of 3000 epoch, the total number of epochs is 9.2 million epochs.
- ***Iteration:*** An iteration is the process in which the RL algorithms collect a batch by simulation and performed a policy parameter update step.
- ***Kullback–Leibler (KL) divergence:*** A measurement of the distance between two distributions. In case of PPO, the KL divergence is a measurement how different the new

policy is compared to the old policy. Isaac Gym can use the KL divergence to limit policy update steps.

- ***Learning rate α:*** The learning rate is measurement of the update step of the parameters in the gradient descent. The last step in the pseudocode is the update of the parameters. The new parameters will be the old parameters plus the gradient of the objective function times the learning rate.
- ***Mini epoch K:*** The amount of mini-batch SGD iterations. Mini-batch SGD updates the policy parameter to reduce the loss function. The objective function evaluates the new parameters during each mini epoch to prevent too large parameter updates. The next mini epoch updates the parameters by SGD and a new mini batch of sampled data.
- ***Mini-batch size:*** The size of the mini batch for the mini batch SGD parameter update. The mini batch size cannot be larger than the batch of sampled simulation of N actors for T time steps
- ***Number of rollouts N:*** Number of simulations which are sampled for the batch. In Isaac Gym, the number of rollouts is equal to the number of actors in parallel.
- ***Objective function L:*** A function which specifies the objective for the RL algorithm. In most RL problems, the objective is to find a policy which yields the highest cumulative reward.
- ***Observation***: An observation is the measurement of a state of the actor. The actor senses part of its states by sensors. The observation is this set of measured states.
- ***Policy π:*** The mapping of state s_t to action a_t
- ***Score to beat:*** A numerical value for the average reward. The score to beat is an average of the obtained reward and when this is reached, the training process will stop.
- ***Time horizon T:*** The horizon is the total of time steps the algorithm simulates to generate the batch.
- ***Variance GAE λ:*** The hyperparameter for the generalized advantage estimator. It controls the influence of variance in the advantage estimation. A high λ leads to a high variance of the estimation but a low bias. On the other hand, a low λ leads to a lower variance of the estimation and a higher bias, see Figure 3.

2.3.3 Previous Research

The work of (Makoviychuk et al., 2021) shows the implementation of parallel RL for multiple environments. The goal in this paper is to validate the reduction of training time for the parallel PPO algorithm. They focus on the effect of parallel actors and the horizon length on the training time. The different environments in their work show the application of Isaac Gym to a wide range of tasks.

A set of tasks discussed by (Makoviychuk et al., 2021) is for locomotion robots. One of the models for locomotion is a humanoid model with 21 DOF, including DOF in the upper body. (Makoviychuk et al., 2021) set the goal for the model to run with a high velocity towards a target in the far distance. The target is at a further distance than the humanoid can cover during the simulation time. The surface of the environment, in which the humanoid operate, is flat. The resulting policy for the humanoid achieves locomotion, with arms swinging unnatural to human-gait (Makoviychuk et al., 2021).

Another locomotion model is of a quadruple legged robot called ANYmal, presented in Figure 9. (Makoviychuk et al., 2021) first showed results for training the locomotion of the ANYmal robot on a smooth surface. The result of this simulation training is that the algorithm converges within 200 seconds of training. The difference between the humanoid and the ANYmal is the goal for the models. Where the humanoid must run as quick as possible towards a static target, the ANYmal model has different static targets with a randomized linear velocity (Makoviychuk et al., 2021).

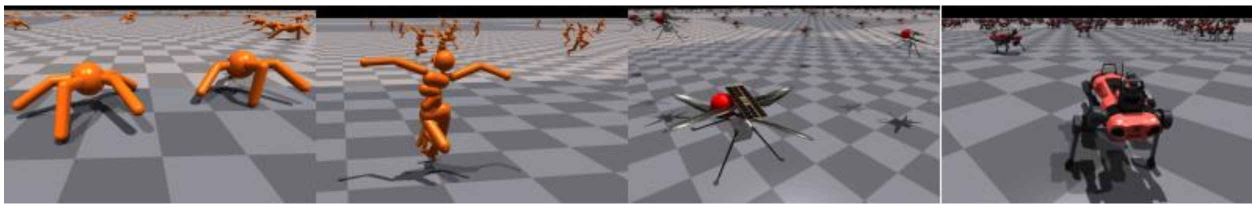


Figure 9. Locomotion simulation of four robotics models. The second model is the humanoid model, and the fourth model is the ANYmal model. The snapshot for the humanoid model shows that the policy lifts and spread the arms in the air. The snapshot of the fourth model shows that the ANYmal has setpoints for different XY points, from [BRON Isaac Gym]

(Makoviychuk et al., 2021) trained a policy for the ANYmal model to transfer this policy from the simulation software towards real world application. The goal of this policy-transfer is to show the benefit of Isaac Gym. The environment in the simulation differs from the first ANYmal implementation. Where the first implementation is on a smooth surface, the second environment has uneven terrain, simulation boxes, stairs, and tilted paths. The outcome (Makoviychuk et al., 2021) mentioned is the time of successful training of a policy.

(Rudin et al., 2021) focused on the ANYmal model and an uneven terrain as environment. Besides the focus on the training time of a policy, their work discusses the modifications of hyperparameters for the PPO algorithm and the performance of the trained policy. The paper discusses the elements of the reward function and that the policy outputs desired joint positions. An PD controller in the ANYmal convert the joint positions to torques (Rudin et al., 2021).

(Rudin et al., 2021) evaluated the trained policy on different step heights with a linear velocity of 0.75 m/s. The policy can make steps with a height of 0.2 m, where higher step heights drastically drop the success rate. The conclusion is that the software can transfer the trained policy to a real ANYmal robot and navigate in the real world. Besides the ANYmal robot, (Rudin et al., 2021) implemented two different quadruple models and one bipedal model, called Cassie.

(Rudin et al., 2021) does not specify results of training for the Cassie robot, other than with some modification to the reward function, they found a walking gait. (Siekmann et al., 2021) focused on sim to real transfer of a policy trained for stair-like terrains. The work does not mention training time in the simulation software, different than Isaac Gym. The findings in the work are that a bipedal robot can manoeuvre over stair-like terrain with proprioceptive feedback, trained in a RL simulator software.

The goal in the current research is to design a RL training implementation to train a policy for an exoskeleton bipedal model. This exoskeleton model includes weight and inertia of a human wearing the exoskeleton. The above-mentioned papers showed that the development in RL software can reduce the training time within an hour. Also, training of a quadruple robot in Isaac Gym results in a trained policy capable of navigation over uneven terrain within 20 minutes (Makoviychuk et al., 2021; Rudin et al., 2021). RL can train a policy to control a bipedal robot, also on uneven terrain (Siekmann et al., 2021). Literature includes more work on the Cassie robot to increase the robustness of the policy (Xie et al., 2019). Work on the Cassie robot showed that RL can train a policy for a bipedal model, and Isaac Gym reduces the training time for the policy.

Chapter 3 Method and Setup

This chapters continues with the implementation of the RL training. It starts with the synthesis of the RL simulator. The synthesis will discuss the different segments of the RL simulator. The implementation of the different segments will follow. The end of the chapter discusses the evaluation of the trained RL policy.

3.1 Synthesis

This section will discuss the synthesis of the design processes during the research. Figure 10 shows the design process to find a policy with RL, starting with initialization of (new) hyperparameters and reward function. The process has five steps before the final policy is reached. At the two evaluation steps, the process can restart at the initialization step. This happens when the evaluations have insufficient scores and can be improved on.

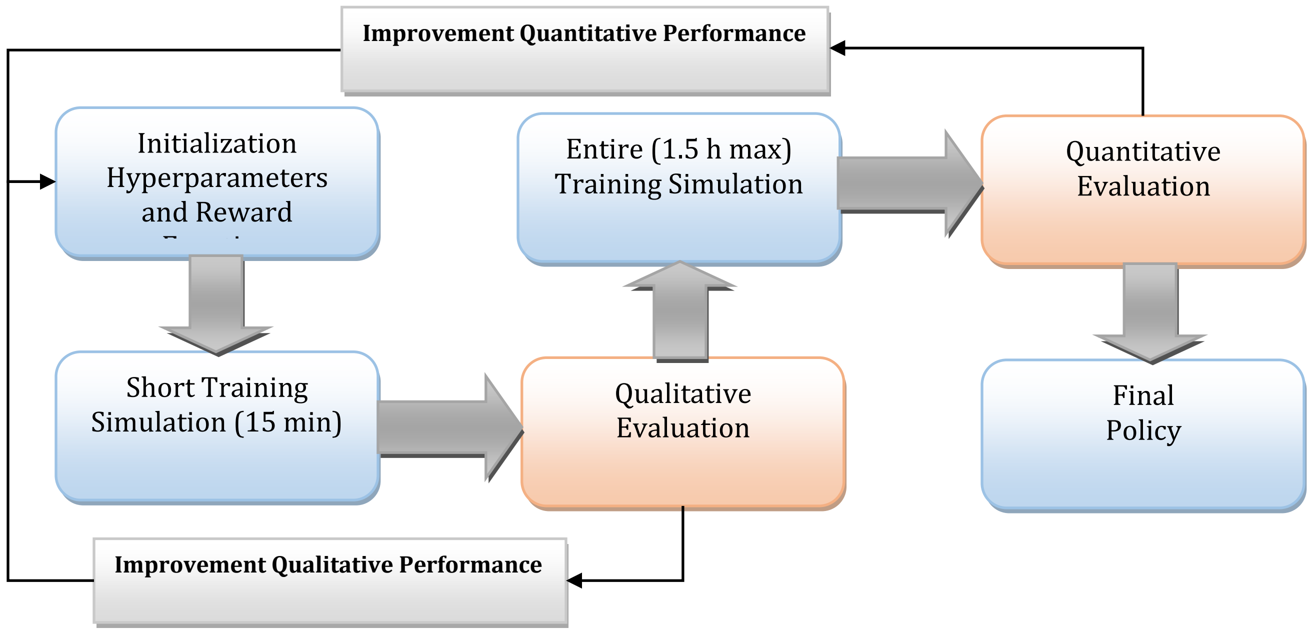

Improvement Quantitative Performance
Initialization Hyperparameters and Reward
Entire (1.5 h max) Training Simulation
Quantitative Evaluation
Short Training Simulation (15 min)
Qualitative Evaluation
Final Policy
Improvement Qualitative Performance

Figure 10. Overview of the process in the design of RL policy for an exoskeleton.

3.1.1 Evaluations and Policies

The design process (Figure 10) has two steps for evaluation, the qualitative and the quantitative evaluation. The qualitative evaluation describes the actors' movement as a result from the learned policy. The quantitative evaluation expresses the performance of the trained policy by means of measurements. The difference is that the qualitative evaluation focuses on the observed gait of the actors, whereas the quantitative evaluation has its focus on the quantitative performance of the trained policy.
This research focuses on the design of two policies. The designing of a robotic model starts with locomotion on a smooth surface, followed by modelling of an uneven terrain and adapt the design to the new environment. For this reason, the design process includes two designs for the exoskeleton model.

3.1.2 Measurements during Simulation

The aforementioned analyses require their own data collection method. The collected data of the qualitative evaluation are the weights and biases for the neural network(s) of the best policy during short training. With the simulation properties of Isaac Gym, the simulations show the gait from the best policy. When the policy shows human-like gait, the design process to the next step. Otherwise, the design restarts at initialization step (see Figure 10).
After the entire training process, the design process is at the quantitative evaluation. This evaluation consists of data collection during the entire training process and two additional test simulations. The software collects the data in an array and write the data to a comma separated values (CSV) file for further process. The test simulations load the best policy found during training. The software measures data after two hundred actor resets during training, reducing the number of data points significantly.
The first test simulation focuses on the policy behaviour of all agents. It collects the data at the resets of each individual agent, so the resulting CSV file consists of a matrix where each row represents a simulated epoch of one agent. The second test simulation focuses on one agent and collects data over time. The software collects data points at every time step in the simulation for the selected agent. The standard test simulation contains at least one single epoch of the agent in the simulation. During the data processing, an analysis of this dataset is necessary to select data for one simulated epoch.

3.1.3 Analyses of Measurements

The qualitative evaluation contains five criteria. Section 3.3 will discuss the evaluations in more detail. Appendix B describes the time evolution of the joint angles of the hip, knee and ankle during human gait (Han & Wang, 2011). A comparison between the observed policy behaviour and (Han & Wang, 2011) resulted in the scores for the criteria.
The test simulations save the quantitative data in a CSV file. The first step in the data analyses of the quantitative data is the processing of the data. The program MATLAB can import CSV file to process the data. Self-made scripts load the CSV file, process the data and make scatter plots of the quantities against time or number of simulations.
The analyses method for the training data is a regression method to determine the trend to estimate future data points of training. For the test simulations, the data analyses are either a statistical analyses or a time series analyses. The time series analysis is exploratory to discover the shape of interesting patterns.

3.2 Implementation

Isaac Gym is the software which trains and simulates the policy for the RL problem. The implementation of the RL problem in Isaac Gym includes the generation of four scripts. Three of the four scripts contain information on the exoskeleton model, parameters and hyperparameters. The last script combines the three scripts and has additional data to construct the simulation and training of the RL problem.

3.2.1 First Script: Exoskeleton Model

The first script describes the model for the actor (the hardware controlled by the agent). The actor is the exoskeleton, and the script includes a computer model of the Symbitron exoskeleton. The script has the format of an unified robot description format (URDF) from a CAD model of the exoskeleton. It includes the visual and inertia properties of the rigid links in the model. The model has five joints per leg and the same orientation and direction of rotation as depicted in Figure 5.

In addition to the provided model, collision properties are added to the URDF file for each rigid link. Otherwise, the exoskeleton model falls through the simulated ground. The CAD files of each link provide parameters for the visual properties. The parameters from the same CAD file provide the collision properties for the model.
Moreover, all joints have continuous rotation for the joints in the provide model. The continues rotation makes it impossible to define angle limits for the joints and to apply the limits of Section 2.2 to the actor model. By changing the specifications of the joints in the URDF file to revolute, the user can specify limits for the joint. The limits include limitations for the angular velocity, joint torque and angle limit.

Motors actuate the joints of the exoskeleton. Currently, a controller actuates the joints and set the torques for the motor. The implementation in Isaac Gym is that the policy sets the torques for the motor, called an effort mode. Therefore, a limit is necessary for the torque input of the actuators.
For each joint, the action output of the policy is a continuous value between -1 and 1. A PD controller scales the selected output of the policy to a torque value for the actuators. The PD controller transforms the desired joint angles (from policy) to torques because humans cannot precisely control their joint torques. The torque limit is hard coded with the limit of 102 Nm for the hips and ankle joints and 150 Nm for the knee joint. The inversion-eversion of the ankle is passive and therefore has a torque limit of 0 Nm.

The ankle in-/eversion is not actuated (zero effort control) but contains a spring-damper. The URDF model is extended with a torsional spring, which models the spring of the spring-damper. Literature studies showed that the stiffness of the ankle joint (in-/eversion) is in between 15 and 45 Nm/rad. The modelled spring has a stiffness of 15 Nm/rad by the assumption that 15 Nm/rad is the passive stiffness. During the simulation of the physics, all joints have a small damping of 0.1 Nms/rad. The damping in the simulation models the damper of the spring-damper. All joints have this damping to improve the simulation of the physics as a small amount of energy will be dissipated.

3.2.2 Second Script: Settings of The Simulator

The second script is a configuration file (YAML) for the simulation of the actors. The constructed configuration file has three sections: the physics simulator, the environment and the simulation details. The selected physics simulator is the standard physics simulator of Isaac Gym: PhysX. The environment section contains specific information of the terrain and selects the number of parallel actors, the length of the episodes and the scaling parameters for the reward function. It also includes the path to the computer model of the actor and specific friction parameters for the environmental model. The number of parallel actors in the training simulation is 3072 actors. The simulation section has more detail on simulation settings, namely: the simulation time step, sub steps between action generation, normal vector, selection GPU/CPU, gravity and the details of PhysX simulations.

The specification of the GPU memory limits the number of parallel actors. The GeForce RTX 3060Ti is the selected GPU and has 8GB of video memory. The length of the episode is 3000 simulation steps. Subsection 3.2.6 will discuss the reward function and will show the reward scaling parameters.
The simulator has a time step of 1/60 (0.016) seconds in which the simulator simulates the dynamics for a new selected action. The sub step is 2 which means that every selected action has two physics simulation steps before the selection of a new action. The physics simulation is with 2 sub steps 1/120 of a second.

The normal vector in the simulation is in the direction of Z in the standard orthogonal frame and the gravity vector is -9.81 in the Z direction. The pipeline for the simulation is the GPU pipeline to utilize to time reduction during training and simulations. The settings for the PhysX simulator are the standard settings for the PhysX simulator for Isaac Gym. Appendix D shows the YAML file for the simulator.

3.2.3 Third Script: Settings PPO Algorithm

The third script is the configuration file (YAML as well) for the RL algorithm. The configuration file consists of four sections for the parameters of the RL algorithm. The first, section algo, determines the implemented algorithm. The second section is the selection of the algorithm model. The third section determines the implementation of the network for the algorithm. Within this network section, the user can select the separation of the network, the action space properties and the construction of the hidden layers in the neural network. The last section of the configuration file is the configuration of the algorithm. This section contains the most hyperparameters for the algorithm, as well as the save frequency of the policy (process of policy development), and a termination state for the cumulative reward (score to beat).

Appendix E shows the full configuration YAML file for the RL algorithm. The implemented algorithm is an Actor-Critic algorithm for continuous state- and action-space. As Chapter 2 Background discussed, both spaces are continuous, and the implementation of an Actor-Critic increases the training process as the Critic learns a value function which reduces the variance. The model for the Actor-Critic has a logarithm standard deviation (logstd) for the continuous action space. It means that the algorithm constructs an action distribution with a mean value and a standard deviation. The settings for the mean are a default initialization and the standard deviation is fixed and has an initialization value of zero. In other words, the network trains to output one value without a distribution.

The network structure is a multi-layer perceptron (MLP) which is fully connected, see Figure A2. The MLP has three hidden layers:

- First layer: 512 nodes
- Second layer: 256 nodes
- Third layer: 128 nodes

The choice for three hidden layers comes from the trade-off between network versatility and learning speed. Network versatility means how well a network can distinguished different problems without forgetting one of the solutions. With three layers, the network seems versatile to learn a gait pattern and have a relative low trainings time.

The last section sets the hyperparameters for the algorithm. The values for the hyperparameters results from the qualitative tuning of the algorithm to find a human-like gait. The found hyperparameters are the discount factor γ (0.99), variance parameter of the Generalized Advantage Estimator λ (0.95), the learning rate α (adaptive), the clipping value (0.2), KL divergence (0.008), and the horizon length (48). Section 3.1.2 discussed the first qualitative evaluation is of 15 minutes of training. This qualitative evaluation is to tune the hyperparameters by trial and error. The range for the hyperparameters is within the advised range for the hyperparameter, and the author refers to the literature for more information on the hyperparameters and their nominal ranges (AurelianTactics, 2018).

3.2.4 Fourth Script: Settings Isaac Gym

The last script is a python file which sets the settings for Isaac Gym. This script takes information from the three previous mentioned scripts for its implementation. The three scripts give information on the model of the actor, the settings for the simulation, and the settings for the RL algorithm.

The python file starts with the initialization of the simulator and RL algorithm, as well as the initialization of the tensors for the PhysX calculations and starting positions of the actors. This is followed by the creation of the simulator with the initialized values followed by the creation of the ground plane. Normally, the ground plane is set by the normal vector and the friction coefficients. Section 3.2.5 shows different ground planes in Isaac Gym and will discuss the implementation in Isaac Gym.

The third part is the creation of the actors to the environment. This part loads the model of the exoskeleton and places it in the environment. The environment has different rows where the script can load actors. The reason for multiple rows is that the ground plane can have random disturbances, which should increase the robustness of the policy. Test simulations on newly generated ground planes evaluate the robustness of the policy. The loading of the actors also includes the starting orientations and velocities of the joints.

The last part of the initialization to the environment is the drive mode for the joints. The last two subsections will discuss the reward function for the reward calculations and the observations of the actor's states, Section 3.2.6 and 3.2.7 respectively.

3.2.5 Environment implementation

Sections 3.1.2 and 3.1.3 discussed two types of environments for the training and testing, resulting in two different policies. The first environment resembles an open grass field with a smooth curved surface (SCS) and natural height difference. The model of a grassy terrain for the open field is used to prevent the policy to adapt minimal step height. As the real world has no perfectly smooth surfaces, a policy with minimal step height may result in easy tipping over in real world applications. Figure 11 shows the smooth curved surface within the ground plane.

The ground plane is a grid of 16 by 16 of different surfaces. The total ground plane for the grassy field is 256 SCS with different randomizations in the height curvature. Due to the randomness of the ground plane, the connection between two surfaces may not line up. This will result in steps at the connections between surfaces. Figure 11 shows misalignment in the yellow box.

Isaac Gym supports one method for the generation of terrain. Isaac Gym has six functions to generate a height map for the simulation. Each function generates a different terrain type, such as a sloped path, stairs, random placed obstacles. The terrain functions result in a triangle mesh data set which is loaded in the simulation by the final script. In the combination script, the function for the creation of the ground plane includes a function which generates the triangle mesh for each of the 256 parts of the ground plane. Appendix D shows this function.

Chapter 2 Background discussed the chosen obstacles from the Cybathlon course. The free walking obstacle requires the exoskeleton to walk without crutches. The implementation of the free walking obstacle is the SCS in Figure 11. The model of the exoskeleton is without crutches which results in a policy independent of crutches. The assumption is that a trained policy on SCS also performs good on a manufactured surfaces with lower height differences. Around the uneven terrain, a padding of flat surface is present. The padding tests the performance of the policy at flat surfaces.

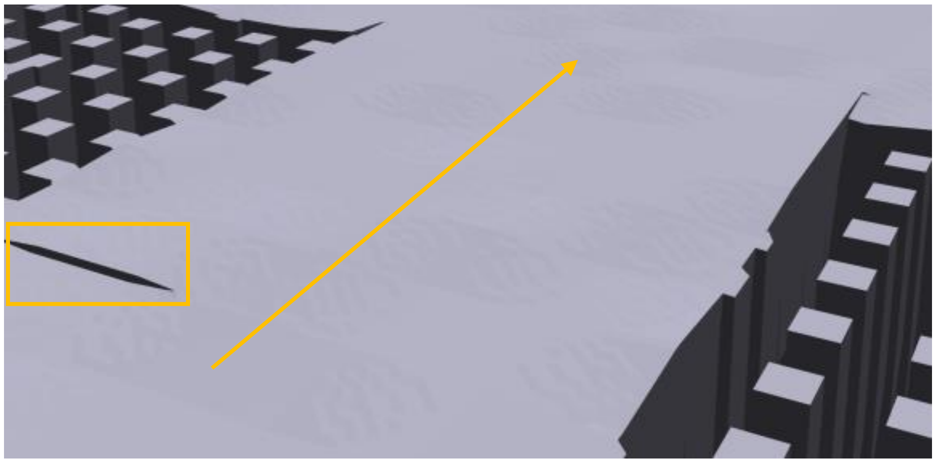


Figure 11. Smooth curved surface. Connection between two terrains may result in a threshold. A smooth curvature to model imperfection of grassy fields. The yellow arrow shows the direction of the actors over the terrain. The yellow box shows a misalignment at the connection of two surfaces.

The second obstacle is the model of stairs in the Cybathlon. The step height for the stairs is 17cm and have a step width of 28cm at one side and a step width of 35cm on the other side (Section 2.2 Hardware Requirements). The plateau between the stairs has a length of 1m. Figure 12 shows the implementation of stairs obstacles as specified by Section 2.3. A small difference between the specification of the step height and step widths might be possible. The reason for this is the limitation of the height map implementation. Isaac Gym has standard functions for surfaces, where the user should select sub steps for the width and height. The drawback is the smaller the sub step, the more video memory Isaac Gym requires. Too small sub steps can result in crashes during initializations.

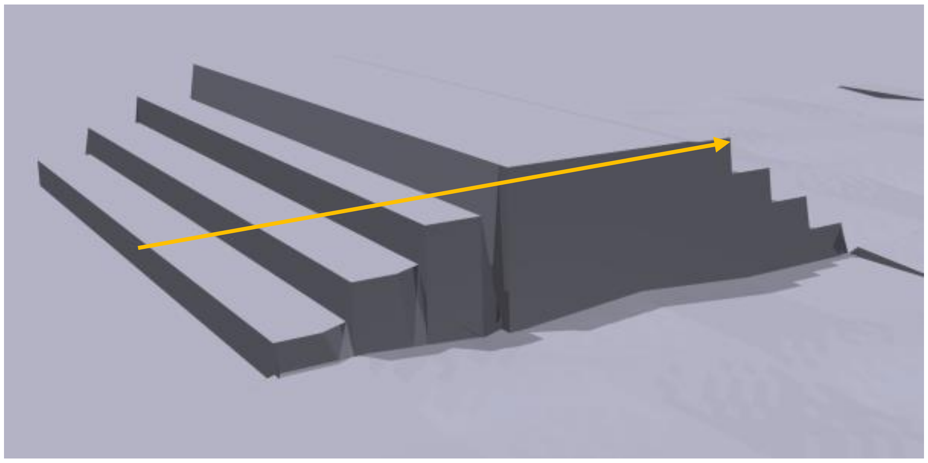


Figure 12. Implementation of stairs. The step height is 17cm for both sides. One side has a smaller step width of 28cm compared to 35cm on the other side. The yellow arrow shows the direction the actors should follow to overcome the stairs obstacle.

The tilted path is the third obstacle. On this surface, the goal of the exoskeleton is to walk perpendicular to the slope of the tilted path. Figure 13 shows the implementation of the tilted path. The yellow arrow shows the direction in which the exoskeleton should move. Section 2.3 showed the measurements of the height of the tilted path and the horizontal length of the tilted path (not in walking direction). Calculations leads to a slope of 12.5 degrees, but due to the same implementation drawback, the estimated slope in Isaac Gym is 12 degrees. The surface on the tilted path is uneven, again to prevent minimal step height policies. The unevenness models the sidewalk tiles, which have different orientations and heights compared to the tilted path.

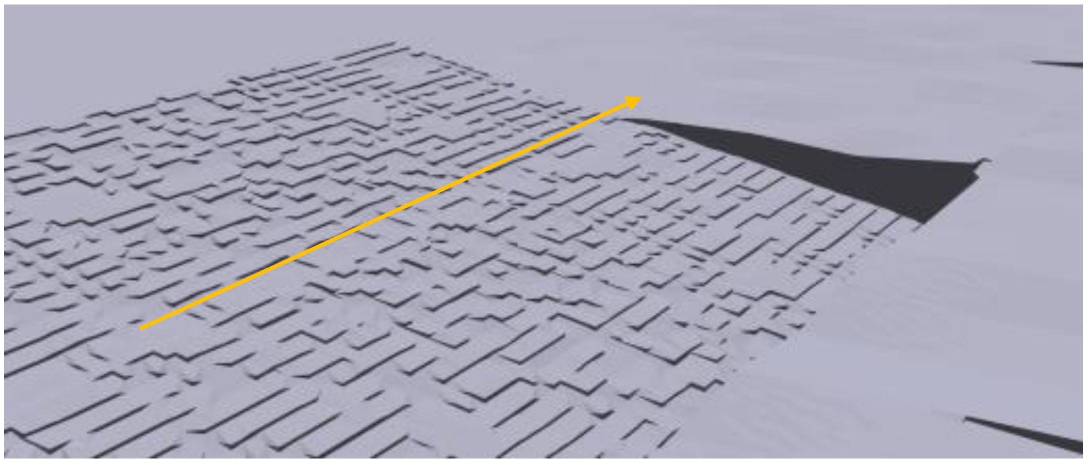


Figure 13. Implementation of the tilted path. The tilt is 12 degrees and a mirrored orientation exist (left side down and right side up). The uniform disturbance simulates tile differences of real-life tilted streets. The yellow arrow shows the movement direction of the actor over the surface.

The last obstacle from the selection is the steppingstones surface. Section 2.3 showed that the distance parallel and perpendicular to the movement has two values: either 40 or 55cm. The steppingstones have a diameter of 25cm. Figure 14 shows the implementation of the steppingstones in Isaac Gym. The figure shows differences between the obstacle in the Cybathlon course and in the simulation. The reason is that the steppingstone function in Isaac Gym only supports constant distance between the steppingstones in both directions. Also, the steppingstones in Isaac Gym are squares instead of circles. The platform size is 25cm for both stones, which gives the steppingstones in Isaac Gym more surface compared to the circular stones in the Cybathlon. The height of the steppingstones is equal to the surface height, the actor does not need to increase the step height to step on a steppingstone. The last difference is the depth between the steppingstones. Isaac Gym automatically sets a depth of -10m which cannot be set by the user. For the implementation, this will not be a problem as not stepping on a stone will result in tipping over of the actor.

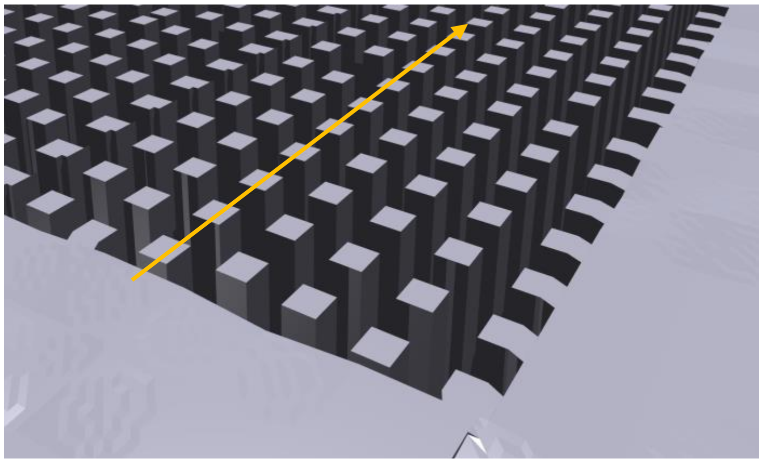


Figure 14. Implementation of the stepping-stones. The platform has a surface of 25 by 25cm and the distance between stones is 35cm. There is no height difference between the stones. The yellow arrow shows the direction of movement for the actor.

3.2.6 Reward Function

A crucial part of the algorithm is the reward function. The reward function maps the actions and states (transitions) to the numerical reward value. The reward function consists of individual parts which contribute to each simulation step or at the end of the epoch. The reward function in this research has nine components for RL of a human-like gait controller, including:

- Alive Reward (per time step) [r_{alive}]. At each iteration step the agent gets a reward for surviving the transition to the new state under the selected action. A requirement for the actor is to not tip over during the simulation. The alive reward should help training a policy which prevent tipping over of the actor. The agent receives a numerical reward of 1 for surviving a control time step.
- Heading Reward (per time step) [$r_{heading}$]. This is a reward that the agent gets when the actor faces towards the target position. The requirement for the obstacles is that the actor follows the direction given by the yellow arrows in Figures 11-14. The arrows are put in the direction of the target position. The heading reward gives a reward for movement direction of the actor towards the target position: $r_{heading} = V_{target} \cdot V_{agent}$, where V is the vector in the 3D space with length 1.
- Base posture reward (per time step) [$r_{posture}$]. The agent gets a positive reward for keeping the base posture upright. In case the posture tilts heavily towards a direction, it may become uncomfortable for the exoskeleton's user. The base posture reward is a design requirement to ensure comfort for patients. $r_{posture} = V_z \cdot V_{base}$.
- Base velocity (per time step) [$r_{velocity}$]. The set goal for the agent is to achieve an average velocity of the actor. At each time step, the agent receives a reward for the velocity of the actor's base. The goal is to train a policy with human-like gait. Human gait has an average velocity of 5km/h. $r_{velocity} = v_{goal}^2 - \left(v_{goal} - v_{base}\right)^2$. The addition of v_{goal}^2 has been chosen to generate a positive reward for the velocity. The maximal reward for the agent is 2.25 per time step for the velocity term.
- Hip flexion (per time step) [r_{hip}]. The flexion and extension of the human hip favours flexion of the hip, hence the agent gains a reward for hip flexion (Han & Wang, 2011). The design requirement of human-like gait introduces this reward term. The angle of the hip flexion joint scales the reward for the reward term. The simulation implements limits for the joints and the limit for the hip flexion is 0.9 rad. The reward is the measured joint angle, as it is positive for hip flexion and negative for hip extension.
- Power cost of joints (per time step) [r_{power}]. The exoskeleton has a battery pack which supplies power to the actuators. To increase the operation time of the exoskeleton between recharging, the power consumption ($P_{rot} = \tau \cdot \omega$) should be as low as possible. The requirement for a low cost of transport resulted in this cost term. The calculations for the power cost depends on the power consumption of the joints: $r_{power} = \sum \frac{|\tau \cdot \omega|}{|\tau_{max} \cdot \omega_{\max}|}$. The power cost is scaled to 1 as all individual rewards are between -1 and 1.
- Lateral position (per time step) [$r_{y_{lat}}$]. To stimulate forward movement of the agent, the lateral displacement of the base body adds to the cost of the agent. The requirement is the movement direction on the surfaces (yellow lines in Figures 11-14). The cost for the lateral displacement depends on lateral displacement squared.
- Death cost (per epoch) [r_{death}]. When the agent takes actions that results in a tipping over of the exoskeleton, the agent takes a penalty. The cost is a result from the requirement that the exoskeleton should not tip over. Three termination criteria for death cost are: height base body, lateral distance, and rotation of transversal rotation base body (tilting of the

base body towards a direction in the XY-plane). The last two criteria are not a result from a requirement, but an adjustment to the reward function to stimulate forward movement and upright posture. As the simulation restarts when the exoskeleton is tipped, the cost of a death is per epoch. The death cost has a constant value of 2000. The value of the death cost is high, but it scales with the number of control steps in the simulations.

- Covered distance reward (per epoch) [r_{dist}]. At the end of an epoch, the agent receives a reward for the total covered distance. The goal of the agent is that the actor moves toward a target. The distance to the target is larger than the distance the actor could cover (assuming simulation time and average velocity). The reward is the measured distance the actor covers during a simulation. The choice for a single reward at the end of an epoch is that a distance reward per time step is not linear when the actor moves with an average velocity. The further the actor goes, the higher the reward for the distance. A reward at the end of the epoch prevents dominance of the distance reward for large distances covered.

All rewards/cost per control step have a value of 1 as maximum except the velocity term (and the summation of the power costs). The tuning of the policy is not only the tuning of the hyperparameters, also the tuning of the reward function. The tuning of the reward function

$$\begin{aligned} r = {} & 0.5\, r_{alive} + 0.7\, r_{heading} + 0.5\, r_{posture} + 15\, r_{velocity} + 0.855\, r_{hip} \\ & - 0.005\, r_{power} - 0.1\, r_{y_{lat}} - 0.25\, r_{death}(torso \leq h_{term}) \\ & - r_{death(yaw \leq \theta_{term})} - r_{death}(y \leq y_{term}) + r_{dist}(reset = true) \end{aligned} \tag{12}$$

includes the calculations of the rewards as well as the scaling factors for each reward/cost. The reward function calculates the reward at each time step. The overall reward for an epoch is $R_{epoch} = \sum_{t=0}^{T} r_t$ where T is the total number of timesteps during an epoch. The scaling factors can favourite a reward/cost over the rest, steering the behaviour of policy. For example, a high scaling factor for the velocity reward steers the policy to match the base velocity to the set velocity.

3.2.7 Observation

The actor measured observations from the actor's states. The states depend on the interaction of the actor with the environment. The choice is to use realistic observations, which comes from sensors which could be present on the exoskeleton. The current hardware setup of the Symbitron exoskeleton may not have installed all sensors. The observations of the model are:

- Actuator rotational position and rotational velocity
  - From the data of the actuator, calculations give the **step height** of both feet
  - Sensor to measure **joint torques**
- Sensors on the base body
  - Gyroscope (**yaw, roll, pitch**) to measure the rotation of the base body on the X, Y, and Z axis
  - Translational accelerometer to measure the base body **acceleration**
  - Translational velocity meter to measure the base body **velocity**
  - Height measuring device (Lidar or vision system for example). To measure the **height of the surroundings**, the simulation has 80 points in a rectangle around the base body, see Figure 15. At these points, the agent measures the corresponding height of the surroundings. It gives information on the height map around the exoskeleton.
  - Gravity sensor (direction gravity towards base body)
  - **Heading direction** which may be given by a camera system which senses the target or the input from a joystick of the exoskeleton.

- **Angle to Target** as a result from the calculation between the translation velocity meter and the heading direction.

3.3 Evaluation

The qualitative evaluation evaluates the learned policy on five criteria. These criteria have a scoring system of 1 to 3 for each criterion. A score of 1 means that the policy scores minimally on the criterion, in other words, the policy does not show the expected motion. A score of 2 means that the policy shows resemblance with human-like gait, but there is room for improvement. A score of 3 shows that the policy shows human-like movement. Table 1 shows the five used criteria for the evaluation and their scoring definitions. Appendix B. Human Gait includes an overview of parameters for the evaluation of human gait. The qualitative evaluation focuses on the spatio-temporal parameter category, the movement of the exoskeleton over time. The focus of the evaluation is on three parameters in this category: the step size, the swing phases, and events during the observed gait.

Table 1. Overview of the criteria for the qualitative evaluation and the three scoring options for the criteria.

Criterion	Minimal (1)	Sufficient (2)	Optimal (3)
C1. Swing patterns of both legs	• Angle deflection Hip flexion – extension (FE) minimal • Angle deflection either flexion or extension • Hip FE near set angle limit of joint	• Angle deflection Hip FE small but observable • Angle deflection hip FE either flexion or extension (larger deflection • Moves Hip FE joint with minimal angle rotation	• Angle deflection Hip FE large and observable • Angle deflection hip FE alternates between flexion – extension • Hip FE joint zero average angle deflection and noticeable angle deflection
C2. Movement of the knee joint	• Angle deflection knee joint not observable • Angle knee joint near initial angle of zero radian	• Angle deflection knee joint observable but small • Angle knee joint near initial angle of zero radian	• Angle deflection knee joint observable and large • Knee joint angle near zero during swing phase
C3. Step size of the exoskeleton	• Minimal and not observable step size • Replacement of feet near current place of feet	• Observable step size (at least one feet size) • Feet do not pass the base body	• Observable and large step size (around 3 times foot length) • Feet pass the base body in step size
C4. Step height of feet	• No observable ground clearance • Little to no rotation of the ankle joint	• No observable height of feet or minimally observable • Ankle joint rotates toward the ground	• Observable height of the feet. Around foot height • Ankle joint rotates toward the ground

C5. Orientation and posture base body	• Orientation of base body perpendicular to movement • Posture of base body heavily leans toward a side	• Orientation of base body within 20 degrees of movement • Posture of base body leans toward a side	• Orientation of base in direction of movement • Posture of base shows minimal leaning toward a side

3.3.2 Quantitative Evaluation

The quantitative evaluation method evaluates the performance of the policy by measured quantities in order to compare it with results already found in the literature. In total, this evaluation includes six quantities: the cumulative reward (train and test), the velocity of the base body, the covered distance, the survival percentage, and the joint torques. The quantitative evaluation does not measure the angles, because criteria 1 and 2 of the qualitative evaluation are already based on the angles of the hip, knee, and ankle joints.

Both evaluation methods do not include parameters from the anthropometric category, as the simulation includes only one configuration of the exoskeleton. For the kinematic category, the evaluation excludes the parameters of acceleration and segment trajectory. The reason for this is that the goal is to learn a policy with a constant velocity, which has little to no accelerations of the base body. The exclusion of the segment trajectory is the reason that the RL tries to find a suitable policy without model or given input trajectories. For the kinetic category, the evaluation only includes the parameter torque. The exoskeleton has no force sensors at the links (yet) which makes it not possible to measure ground reaction forces for real world applications. The criterion of orientation and posture of the body aims to have zero net moment around the centre of mass. Therefore, the quantity is present in the qualitative evaluation.

The value of the cumulative reward depends on the scaling factors in the reward functions, the simulation time, and performance of the policy. It mostly depends on the design choices of the reward function; therefore, the evaluation will be the development of the cumulative reward during training. Over time, training should converge to a maximal cumulative reward. The test simulations compare the cumulative reward during testing to the training cumulative reward and the expectation of the cumulative reward.

Velocity
The goal of the exoskeleton is to achieve an average velocity of 1.5 m/s (5.4 km/h) and to mimic human gait with the assumption that human gait has a velocity around 5 km/h. A higher mean velocity goal should help the agent achieve a velocity around 5 km/h as the reward of the velocity is quadratic. The velocity of the exoskeleton (when up to speed) should be around 1.5 m/s on average.

Covered Distance
The distance, that the exoskeleton walks, depends on the velocity of the base and on the walking time as well. The simulation time is 50 seconds. When the agents start with desired velocity, the agents will travel seventy-five meters ($1.5\ \text{m/s} \cdot 50\ \text{s}$). If the learned policy is good, the walking distance will be around seventy-five meters.

Survival Rate
The survival percentage is a sign of the number of actors achieving to finish 50 seconds of moving without falling over. The simulation program resets agents that fall over. Falling over means in this research that the agent did not reach the end of the simulation time but met other termination conditions (Base under termination height, the base body vector angle larger than termination angle). The ideal policy achieves a survival percentage of 100%, meaning no agent falls over.

Cost Of Transport
A battery pack powers the exoskeleton and has a limited storage of energy. By decreasing the energy consumption of the exoskeleton, it increases the range of the exoskeleton. However, the larger the travelled distanced, the more energy the exoskeleton uses. Therefore, the quantity cost of transport calculates the energy efficiency. The cost of transport is a dimensionless quantity which makes the comparison possible between the trained policy and humans. The cost of transport for the exoskeleton is the mechanical energy the system consumed divided by the covered distance times the mass and gravitational constant ($COT = E_{mech}/m_{exo}gd_{covered}$). Multiplying the joint torques with the joint velocities results in the mechanical power of the joints. The numerical integral over the simulation of the mechanical power gives the mechanical energy during the simulation. The power cost in the reward function (12) aims to reduce the COT, therefore the evaluation of the COT during the training process.

Joint Torque
The last quantity is the joint torque of all the joints. The goal to mimic human gait means that torques of the exoskeleton show similarities to torques during human gait. The comparison between exoskeleton and human gait needs the measurements of torques during a simulation. Therefore, one simulation gathers the torques send to the motors (the scaled action vector in RL). Results from literature gives the torques of humans during gait to compare with the measured torques.

Chapter 4 Results

Chapter 3 discussed the methods to score the performance of the trained policies. Both methods (qualitative and quantitative) generate a performance score for two policies: one policy for SCS and a policy for random placed Cybathlon obstacles. The first three sections of the results will discuss the SCS policy, and the final three sections will discuss the policy for the Cybathlon obstacles.

4.1 Qualitative Results SCS Policy

Section 3.1.1 discussed the tuning of the hyperparameters and reward function. The qualitative assessments of the short trainings showed whether to change the hyperparameters. When the hyperparameters and the reward function led to a policy with good walking behaviour in the qualitative evaluation, the training is resumed.
In total, each actor could have 3,000 epochs and with 3,072 actors, the total number of epochs is 9.216 million. Figure 15 shows the exoskeleton's movement of the trained policy on the SCS. The movement shows a symmetrical walking pattern of both legs. The hip joints alternate between flexion and extension of the joint. The hip flexion/extension is below the set angle limit for the corresponding joint. The knee joints have a deflection at the toe-off phase and extent maximally during mid-swing, like human gait. The maximal extension of the knee joint is 0 radians, which the knee joint does not exceed. The ankle joints have dorsiflexion in the toe-off phase and plantar flexion during mid-swing. Again, the angles of the ankle joint do not exceed the angle limit of the joints.
Movement in the frontal plane is not clearly visible in Figure 15. However, observations in Isaac Gym showed that the base body (backpack module in Figure 5) swings around the roll axis (the axis pointing forward) of the body. The observations included rotations of the hip abduction/adduction, although the rotations are minimal.

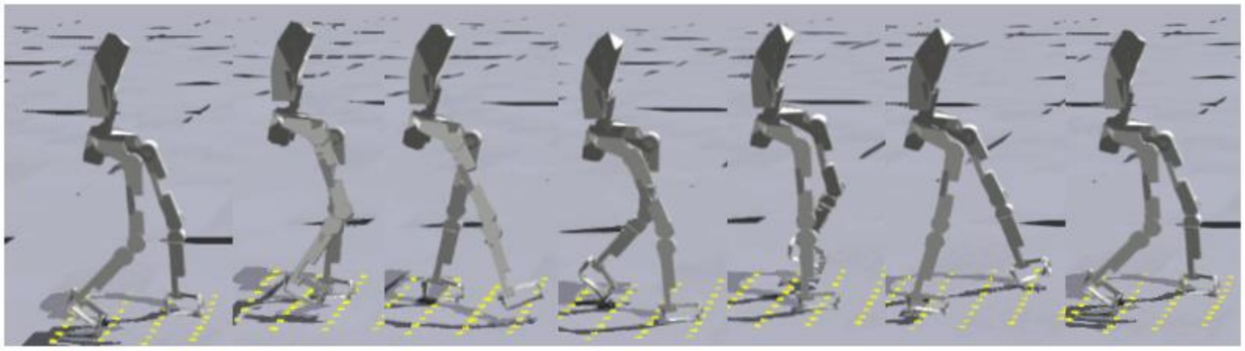


Figure 15. The exoskeleton's movement during a test simulation with the best policy found during training. Seven frames divide the movement for the qualitative evaluation of the policy.

The qualitative evaluation scores the performance of the policy. The criteria in Section 3.3 Evaluation help to evaluate exoskeleton's movement. The scores for the criteria for the SCS policy are:

• C1. Swing pattern:	3
• C2. Knee joint:	3
• C3. Step size:	3
• C4. Foot clearance	3
• C5. Posture & orientation	2

The exoskeleton's movement scored maximally on criteria 1 to 4. The exoskeleton's movement showed qualities of the criteria which correspond with the desired behaviour.

The movement scored sufficient on criterion 5. Even though the orientation of the base body is in the direction of the target, the observed body sway in the exoskeleton's movement resulted in a sufficient score. The qualitative evaluations of the short trainings all had a maximum score of 2 at criterion 5, although the results do not elaborate on policies which scores less.

4.2 Quantitative Results SCS Policy

The qualitative results evaluated the quality of the resulting movement of the exoskeleton. The quantitative evaluation will give an insight of the performance of the policy. The software observed the quantities (Section 3.1.3) during one training simulation and two policy test simulations. The training simulation measured the cumulative reward and the cost of transport over policy updates. The two test simulations measured the quantities either for multiple simulated actors (cumulative reward, covered distance, and percentage fallen agent) or a simulation of one actor over time (velocity exoskeleton's base and the joint torques).

4.2.1 Policy Training

The first result is the evolution of the reward in the training process. At every 200th epoch (reset of an actor), the software saved the reward obtained by that actor during the epoch, see Figure 16. Each data point shows the cumulative reward of an updated policy because the training updated the policy multiple times during an actor's epoch.

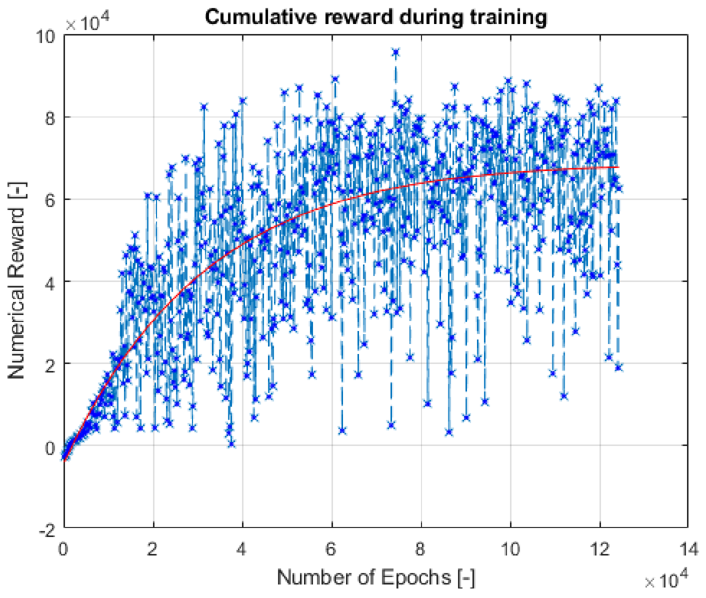


Figure 16. Cumulative reward progress during training. The first one thousand initializations have relative low reward. With later initializations, the spread between the cumulative reward is large (0-100,000 reward). The red line shows the exponential trend line for the data. The trend line shows an increase to 80,000 or more cumulative reward.

Figure 16 shows the evolution of the cumulative reward per epoch. The first thousand epochs show a strong increase in the cumulative reward towards a reward of 10,000. The evolution of the training shows an upper bound of 90,000 with one epoch achieving a higher reward. With more than 10,000 epochs, the spread in the obtained cumulative reward increases. The lower bound for the obtained reward is 0 for more than 10,000 epochs. From 10,000 to 40,000 epochs, the maximal obtained reward keeps increasing till the upper bound reward of 90,000. The number of epochs during training is 1.244 million epochs. The pattern of the reward increase looks like a logarithmic function. The red line in Figure 16 is an exponential trend line fitted to the data points. The exponential trend line has the lowest error term between the trend line and the data points. The trend line shows an increase in the cumulative reward towards 70,000. The trend line approaches the asymptotic value at 1.244 million epochs.

The second measured quantity during the policy training is the cost of transport, see Section 3.1.3. The 200th epochs show excessive costs of transport with a value more than 2,000. The cost of transport drops heavily within the first 2,000 epochs to a cost of transport between 1 and 20.

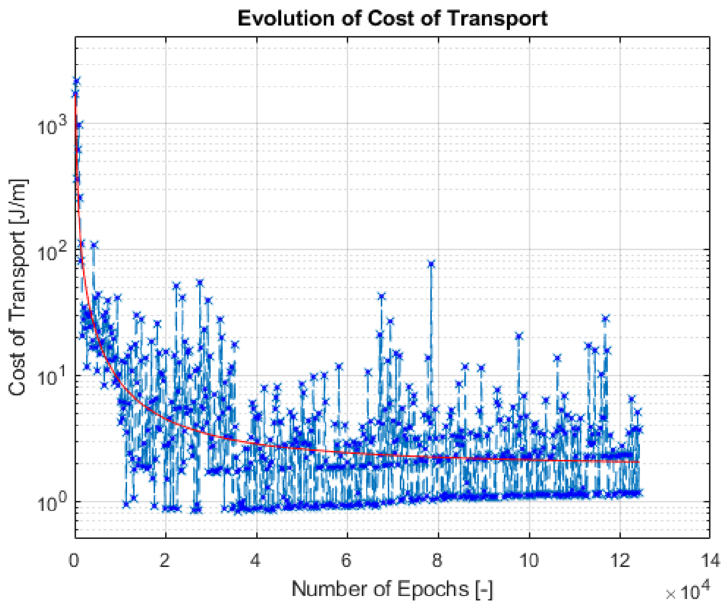


Figure 17. Evolution of the cost of transport for the training policies. The first 200th epochs show a quick decrease in the cost of energy. The cost of energy is between 0.8 and 2 with epochs having a higher cost of energy after 2,000 epoch resets. The red line shows the trendline of the data set, which is exponential trend line.

At 4,000 epochs, the policy has the lowest cost of transport with a cost of transport of 0.83. Figure 17 shows the evolution of the cost of transport during the policy training. The y-axis is a logarithm scale in order to give a good visual representation of the data. The cost of transport during training is between 0.83 and 2 after 4,000 epochs. At 1.2 million epochs, the cost of transport is between 1 and 3. Part of the epochs between 4,000 and 1.244 million epochs have a significant larger cost of transport than 3. The red line in Figure 17 is the trend line fitted to the data. On a logarithm scale, the evolution of the cost of transport is exponential decaying. The fitted data is a power function, which give the lowest error between the data and the fitted trend line. The trend line approaches the asymptotic value of 2 for the cost of transport.

4.2.2 Test Simulation SCS Policy

The training process generates a policy with the highest cumulative reward. The first test simulation evaluates the best policy in a new generated, random environment with 512 parallel actors and measures quantities at the end of the epoch of each actor. Each actor in the simulation is one epoch for the simulation, in other words, one measurement point. The policy in the test simulation is the trained policy with the highest reward.

Figure 18 shows the measured distances for 1286 epochs. Thirteen of the 1286 epochs cover less than 10 meters. The average distance the policy covers is 63.2 meters, with a standard deviation of 10.16 meters. Figure 18 shows the mean distance as the green dotted line and the standard deviations are the two purple dotted lines. The skewness of the measured distances is -2.96, showing a negative skew in the distribution.

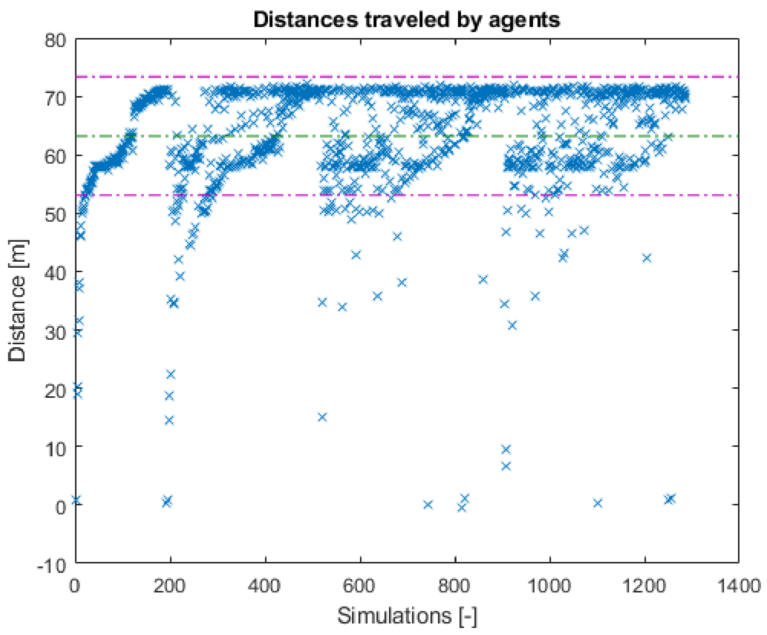


Figure 18. The covered distances per epoch in the first test simulation. The green dotted line shows the average distance covered, which is 63.2 meters The two purple lines shows the standard deviation of the covered distance of 10.16 meters. Most measurements are in the standard deviation band.

Besides the covered distances of the actors, the test simulations measure the cumulative reward. The correlation between the cumulative reward and the covered distance is 0.9988, showing a strong correlation between the cumulative reward and the covered distance. Figure C.1. (C.1Figures SCS Policy) shows the obtained rewards for 1286 epochs of the policy. This graph of the cumulative reward has the same appearance as Figure 18, except that the Y-axis has another scale. The cumulative reward of the epochs is between -2890 and 11,310. The data statistics show a mean cumulative reward of 99,440 with a standard deviation of 16,040. It shows that nearly all the obtained cumulative rewards lay between the standard deviation band, the two purple dotted lines. The skewness of the cumulative reward is -2.91, showing a negative skew in the distribution.
The measurements of the first simulation include a counter for resets other than the end of simulation. Section 3.1.3 describes causes other than end of simulation which will reset the actor. Another naming for those causes is a fallen actor. The result of the measurement is a dynamic progress of the percentage of fallen agent. After 1286 epochs, 54 percent of the epochs fails to reach the end of the simulation.

4.3 Quantitative Result Single Actor

The second test simulation measures quantities at one actor in the simulation. The simulation has a new generated surface with the same policy from the qualitative evaluation. The second simulation measures the velocity, and the joint torques of one actor during one epoch.

4.3.1 Velocity Actor

The first measurement is the velocity of the base body (backpack of exoskeleton, see Figure 5). The simulation initializes the actors with a velocity of 0m/s for the base body. At every two simulations steps the software measures the quantities (same frequency as the control input). At 100 simulations step (0.41 seconds), the base body has a velocity of 1m/s. During the rest of the epoch, the base velocity is between the 1 and 1.7m/s, see Figure 19. The data points show a pattern of an exponential decay in the increasing form. The red line in Figure 19 shows the fitted trend line of the data points. The fit for the trend line is a power trend line, as a power trend line has the lowest error between the trend line and the data points. The trend line approaches the asymptotic value of 1.4m/s. The trend line shows a quick increase of the velocity to 1.4m/s within 100 simulations step and gradually increase to 1.4m/s between simulation step 200 and 600.

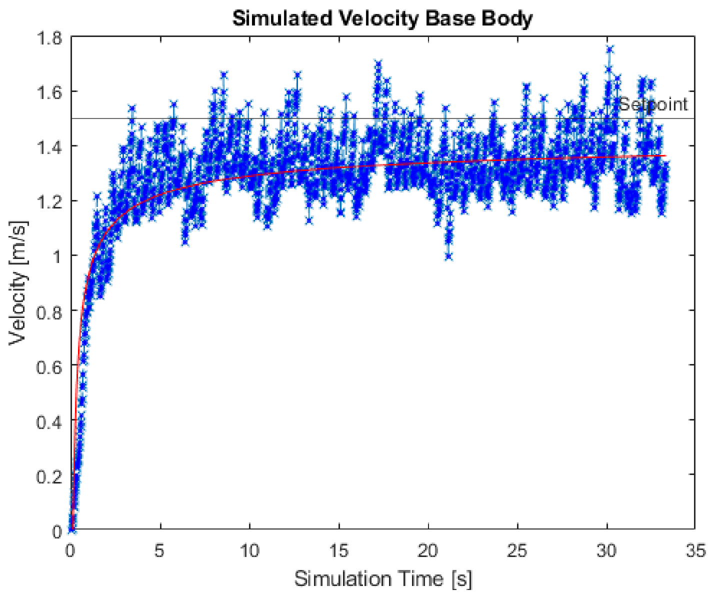


Figure 19. The velocity of the base link during one epoch. The black line shows the setpoint for the policy. The blue dots are the measured data points. The red line is the trend line (power) fitted to the measured data.

Appendix C.1 Figures SCS Policy includes the velocity of the base when the velocity has a constant pattern in Figure C.2. The data set starts at the simulation step of 50 since within a second the mean walking speed should be reached. The pattern is that the velocity alternates around the velocity setpoint. The average velocity during the epoch is 1.38 m/s and is the green dotted line in Figure C.2. The standard deviation of the velocity is 0.1191 m/s and Figure C.2. shows the standard deviation with the two purples lines.

4.3.2 Joint Torques SCS Policy

The final quantities in the measurement are the torques of the eight actuated joints (4 joints on two legs). The measurement of the eight actuated joints is for one actor as well. The agent takes multiple steps in one simulation run. Figure 20 shows the torques of the four degrees of freedom (two legs) for a simulation time of 2 seconds. In the two seconds, each leg makes about two complete leg swings. The blue crosses show the data points of the left leg joints, and the orange circles show the measured torques of the right leg. The maximal measured torques for both hip and the ankle joint are 102Nm. The absolute maximal torque of the knee joint measures 150 Nm.

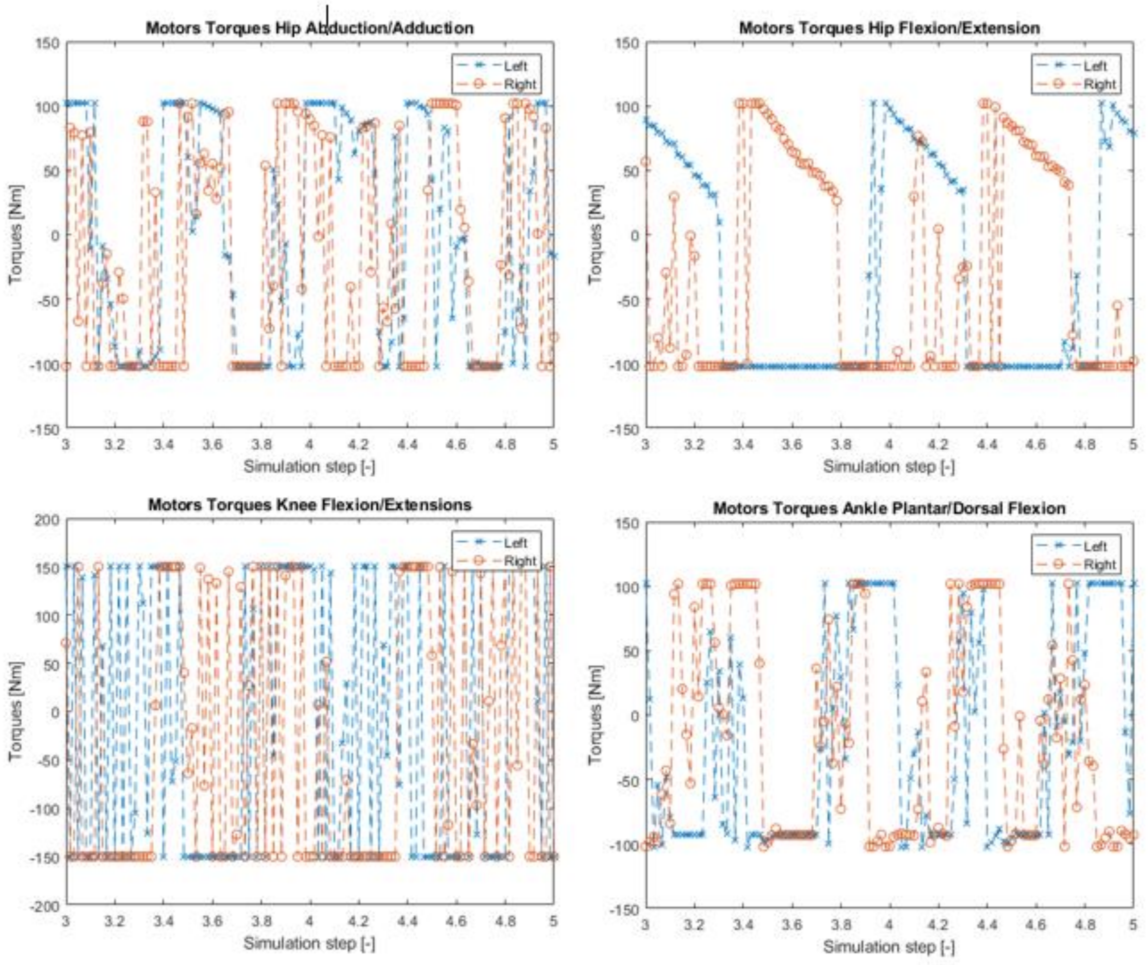


Figure 20. Actuation torque of the four DOF. The blue line indicates the left leg and the orange line the right leg. Positive torques means that the joint has abduction, flexion, or plantar flexion. Negative torques result in adduction, extension, or dorsal flexion.

The motor torques of the hip flexion/extension (FE) (top right graph) shows an alternating pattern. During the forward swing of the right leg, the left leg exerts maximal extension torque. When the left leg swings forward (torque flexion), the right left exerts an extension torque. This extension torque is not maximal as for the left leg is observed. For the hip abduction/adduction (AA) (top left graph), the graph does not show a clear pattern. The graph shows a tendency to actuate the left and right joints either with maximal extension or flexion torque. The graph for the knee joint shows a that the knee joint has a tendency for maximal extension torque. Both legs have the tendency to exert the extension torque at the same moment as the extension torque of the hip FE joint. Outside that window, the torque is between maximal extension or flexion with no clear pattern. The last joint is the ankle joint (bottom right graph). The torque alternates between dorsal and plantar flexion (positive and negative torque respectively). When the right leg shows the extension torque for the hip FE joint, the ankle joint has maximal plantar flexion torque. Vice versa is true for the left leg. In between the extension torques of the hip FE joint, the ankle joints show a tendency for dorsal flexion. The start of the dorsal flexion tendency is during the end of the extension torque of the hip FE joints. Overall, all joints show a tendency for to exert maximal joint torque in either direction. For hip FE and knee FE joint, the extension torque is favourable as the mean torque over the simulation is a torque in extension for both joints (30 and 10 Nm respectively). The ankle joint has a mean torque in the plantar flexion of 6.2Nm. However, the standard deviation is larger for the three joints with a minimal standard deviation of 80Nm.

4.4 Qualitative Result Obstacle Policy

Section 3.2.5 already discussed the two training processes. The second training process has a different environment model. Where the first environment only has smooth surfaces, the second environment includes the other three obstacles in the environment. The learning process with the obstacles has the same (hyper)parameters, reward function, and training settings as SCS. The first six meters the terrain is smooth, so the agent can develop a walking gait. The movement sequence during the first meters is comparable to Figure 15. After six meters, the agent faces the first obstacle on its path. In case the first obstacle is free walking, the agent succeeds to complete the obstacle. In this case, the agent achieves the qualitative score of Section 4.1. The trained policy achieves a different result for the other obstacles. For all other obstacles, the trained policy fails the obstacles by falling over in front or at the obstacle. The observed failure depends on the obstacle the policy faces. Table 2 shows the qualitative evaluation for the three obstacles: the stairs, the steppingstones, and the tilted path.

Table 2. Overview of the qualitative evaluation of the policy. Besides the SCS obstacle, the policy face three more obstacles: the stairs, the steppingstones, and tilted path.

	C1. Swing Pattern	C2. Knee joint	C3. Step size	C4. Step height	C5. Posture & orientation
Stairs	1	2	1	2	1
Steppingstones	3	3	2	3	1
Tilted path	3	3	3	3	1

4.4.1 Stairs

For the stairs, the policy sees the first step of the stairs and stops the actor's feet in front of the first step. Next, the policy tries to move the base body forward, falls onto the knee joints until the base hits one of the stair steps. Table 2 shows the qualitative result of the stair obstacle. The score on C1 is minimal, as the policy shows the tendency to place the feet next to each other in front of the first step. Criterion 2 scores sufficient as the exoskeleton has a deflection of the knee joint, even when it is falling over. However, the score for the knee joint is not maximal as the policy does not make any leg swing when on the obstacle. The third criterion has a minimal score. Due to the stairs in front of the feet, the policy replaces his feet at the same spot until the exoskeletons falls over. C4 has a sufficient score, as the policy has little step height. Also, the ankle joints both tend to rotate toward the ground, even when the exoskeleton is falling. The last criterion has a minimal score. The policy orients the exoskeleton towards the target; however, the posture of the exoskeleton is not sufficient. The policy fails to take the first step and tips over at the beginning of the stairs.

4.4.2 Steppingstone

The second obstacle to discuss is the stepping-stone obstacle. Criteria 1, 2, and 4 have the same score as discussed in Section 4.1, because the movement sequence from the smooth surface continues when the agent arrives at the steppingstones. By stepping between stones, the exoskeleton tends to fall forward or sideways. The tendency to fall results in a minimal score for the posture criterion. On top of that, the orientation of the base body tends to differ significantly from the target. Where criterion 3 has a maximal score for the SCS, the criterion scores sufficient for the steppingstones. The policy shows the tendency to make steps with the same step size, it is not sufficient for the steppingstones to have a constant step size. The placement of one foot in front of the obstacle can be smaller/larger than the 'normal' step size. The policy should be able to adapt the step size to the step size between the steppingstones. The lack of the adaptation result in a sufficient score, see Table 2.

4.4.3 Tilted path

The last qualitative evaluation is of the tilted path obstacle. Here, criteria 1 to 4 achieve the same score as for the first policy. The second policy shows the same movement sequence as the SCS. So, the policy results in the same swing pattern, behaviour of the knee joint, the step size, and the step height. The orientation of the exoskeleton on the tilted path is towards the target, which is optimal. However, the posture of the exoskeleton heavily leans toward the lower side of the tilted path. The base body of the exoskeleton stays parallel to the normal of the surface. On top of that, the base body sways around the roll axis during the steps. The already leaning body in combination with the alternating sway movement causes the exoskeleton to tip over easily. As both the standard posture is rotated and the body sways back and forth, criterion 5 has a minimal score. The last row of Table 2 shows the qualitative evaluation of the tilted path obstacle.

4.4.4 Other Implementations

The qualitative results in the three subsections show that training of the policy does not result in a policy capable of overcoming the obstacles. The environment in the training initialises the obstacles randomly, aimed to achieve the goal to overcome a random sequence of Cybathlon obstacles. The qualitative evaluation shows that on a random generated surface, the algorithm cannot learn a policy to successfully overcome the obstacles. Two other surface implementations have been implemented for the qualitative evaluation.

The first different surface implementation is a surface with a standard sequence of obstacles: free walking, tilted path, stairs, and finishing with steppingstones. Again, the algorithm was not capable of learning a policy which successfully completed all obstacles. On this surface, the policy could complete the free walking and tilted path obstacles. The policy failed at the beginning of the stairs at the first step.

The second different surface is an environment with only stairs on the surface (as both policies show problems starting at the stairs). Also, the trained policy on this surface was not capable of completing the obstacle. The simulations initialize the actors at the plateau of the stair obstacle. The behaviour of the policy is that the actor makes a normal step, tumbles down the stairs, but the policy manages to regain balance. When the actor reaches the new stairs, the actor stops or falls over in front of the first step of the stairs.

4.5 Quantitative Result Obstacle Policy

The quantitative evaluation of the obstacle policy has the same quantities as the SCS policy evaluation. The first test simulation also has a new generated surface with the obstacles randomly placed. The training is over 1.22 million epoch resets. The results discuss the measurements during training of the policy.

4.5.1 Policy Training

The first quantity is the cumulative reward over epoch resets. The software measures the cumulative reward of every 200th training epoch. Figure 21 shows the development of the cumulative reward during training. The first set of epochs show an increase of the cumulative reward. After 1000 epochs, the training achieves a reward of 500. Afterwards, the training achieves a cumulative reward of 2500 within the first 20,000 epochs. Then the spread in between the reward drops as well as the maximal cumulative for further epochs, with a maximal reward of 1,500. The red line is the trend line of the cumulative reward. The best fitted trend line is an exponential trend line. The trendline starts at a reward of -2500 and shows a slight increase to a reward of -250 over 1,22 million epochs after 10,000 epochs.

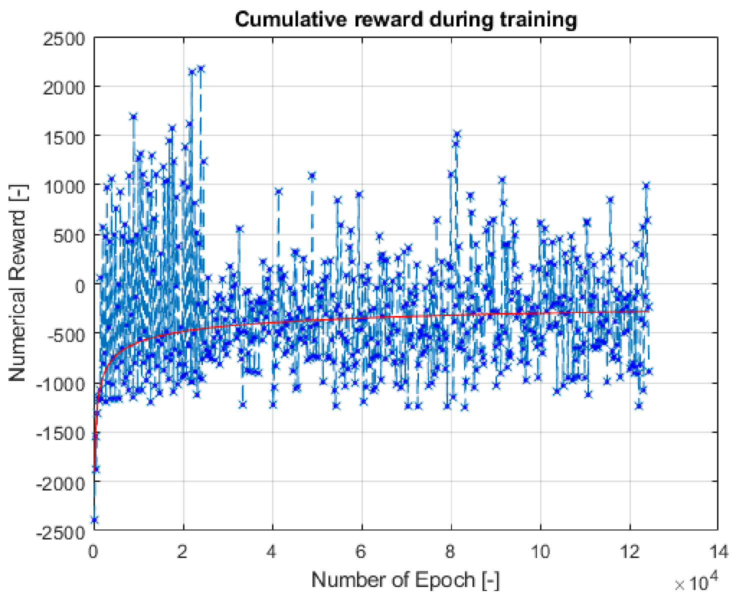


Figure 21. The cumulative reward of the policy during the training process. The blue crosses are the numerical rewards. The red line shows the trend line of the cumulative rewards. The trend line is an exponential trend line, which have the best fit with the data points.

The final measurement during training is the measurement of the cost of transport. Figure 22 shows the evolution of the cost of transport of the policy during training. The first 2000 epochs show a quick drop in the cost of transport, with a minimum cost of transport of 40. After the 2000 epochs, the training of the policy shows a stagnation of the decrease in the cost of transport. The cost of transport varies between 100 and 1000 from epoch 2000 onwards. The red line is the trend line fitted to the data, where the best fit was a linear trend line. The trend line shows a slight increase of the cost of transport from 400 to 500.

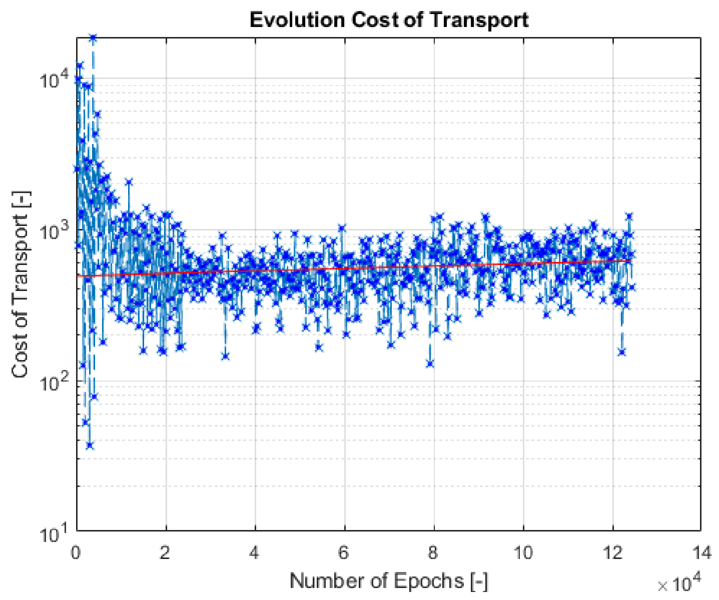


Figure 22. The evolution of the cost of transport of the policy. The blue dots/crosses are the data points of the 200th epoch. The red line is the trend line fitted to the data points. The trend shows an increase of the cost of transport.

4.5.2 Test Simulation Obstacle Policy

After training, test simulations evaluate the policy with the highest cumulative reward in testing. The first test simulation loads a random generated surface and 512 actors with the same policy. The test simulation focuses on quantities which can change at each epoch such as the cumulative reward, covered distance, and percentage fallen agent.

The first result is the covered distance by the 1286 simulations. At the end of each simulation of one actor, the software measured the cumulative reward. Figure 23 shows the resulting covered distances of the test epochs. The distribution of the covered distance has an average of 2.678 meters and a standard deviation of 1.05 meters. The mean and the standard deviation are the green and purple dotted lines in Figure 23 respectively. A small selection of the simulations achieves a distance smaller than zero. An even smaller selection of simulations achieves a distance greater than 5 meters.

The second result is the cumulative reward of the policy. Appendix C.2 Figures Obstacle Policy presents the resulting graph of the cumulative reward. As the covered distance and the cumulative have a correlation of 0.9886, Figure C.3 shows the same trend as Figure 23. The cumulative reward is between 0 and 9,000. Some simulations achieve a cumulative reward of 10,000 whereas a few achieve a reward around -2,000. The mean cumulative reward is 4868 with a standard deviation of 2296. The mean cumulative reward and the standard deviation are the green dotted line and the purple dotted lines in Figure C.3 respectively. Figure C.3 shows a data distribution which seems normal, and the skewness parameter of the dataset is -0.87323.

The final measurement in the first test simulation is the percentage of fallen agents. The definition of a fallen agent in Section 3.1.3 is applied to this measurement. The test simulation measures the current percentage of fallen agent at the end of each epoch. The result is that all the epochs ends before they reach the maximal simulation time. So, after each epoch the current percentage of fallen agent is 100%.

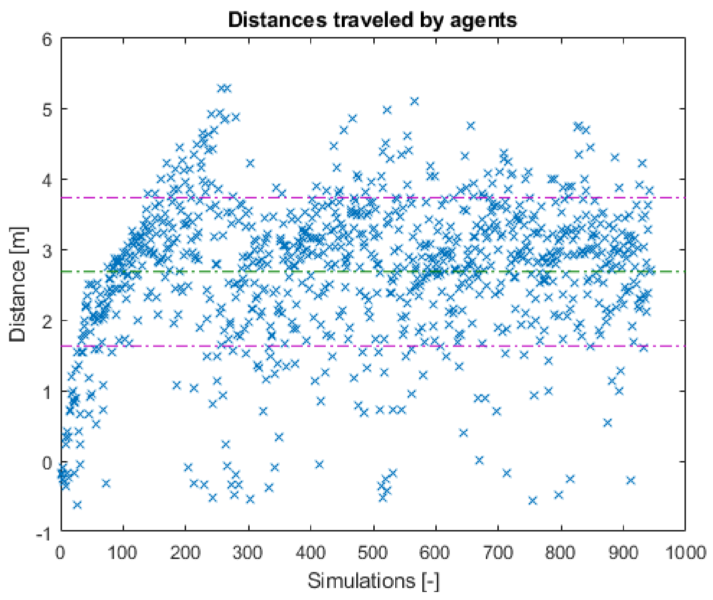


Figure 23. The covered distances of the exoskeleton during test simulations. The blue crosses show the covered distance of a simulation. The green line shows the mean covered distance of 2.7 meters.

4.6 Quantitative Result Single Actor

The final simulation is a test simulation focusing on the performance of one actor. The final simulation has a new generated surface with random placed obstacles. The quantities measured in the simulation are the velocity of the base body and the joint torques of eight actuators.

4.6.1 Velocity Exoskeleton

The velocity measurement is the velocity of the base body (backpack of the exoskeleton). The goal of the policy is still a velocity of 1.5m/s of the base body. The simulation starts with a velocity of 0m/s. The simulation has again a simulation time step of 1/60 seconds. For eight seconds, one epoch takes 478 steps in the epoch.

Figure 24 shows the velocity of the base body during one simulation of eight seconds. The figure shows the trend line fitted to the data points. The trend line ends below 1m/s with a slight increase at the end of the trend line. The trend line with the smallest error between the data set and the trend line is a third order polynomial. The dataset shows a steep increase of the velocity at the end of the epoch, exceeding the setpoint value. Appendix C.2 Figures Obstacle Policy includes the graph for the mean velocity and the standard deviation of the base velocity. The mean velocity is 0.6059 m/s, and the standard deviation in the velocity is 0.366 m/s. Figure C.4 shows the mean velocity and the standard deviation as the green dotted line and purple dotted lines, respectively. In the first second of the simulation, the velocity of the actor is negative. The following time second, the actor increases its speed to 0.8 m/s. Afterwards, the velocity has six peaks, including the steep increase at the end of the epoch. The pattern in the velocity seems to slightly increase between 2 and 7 seconds, also visible in the trend line in Figure 24 between 2 and 7 second. The black line in both Figure 24 and Figure C.4 shows the velocity setpoint, which is set during policy training.

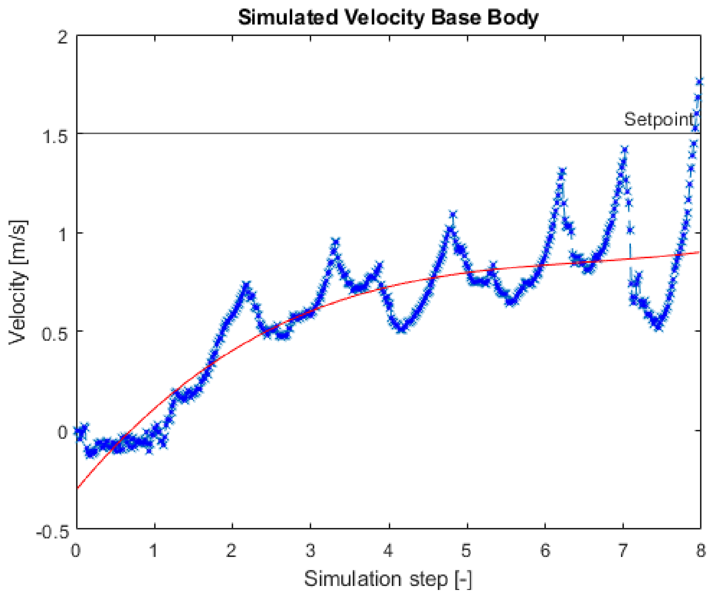


Figure 24.. The velocity profile of the base link during one epoch. The blue dots are the data points of the velocity. The red line is the fitted third order polynomial trend line. The end of shows a substantial increase of the velocity.

4.6.2 Joint Torques Obstacle Policy

The last measured quantity for the obstacle policy is the joint torque of the eight actuators of the exoskeleton. Within the same simulation as the velocity, the software measures the joint torques of each actuator. Figure 25 shows the joint torques of the four actuated joint of each leg. The blue crosses are the datapoints for the left leg and the orange circles are the datapoints of the right leg. The figure has a time window of 2 seconds of the simulation, equal to the period in Figure 20. The joint torques do not show recognizable pattern for leg swings. The absolute maximal joint torques for the hip joints and the ankle joint is 102 Nm. The absolute maximal joint torque for the knee flexion is 150 Nm.

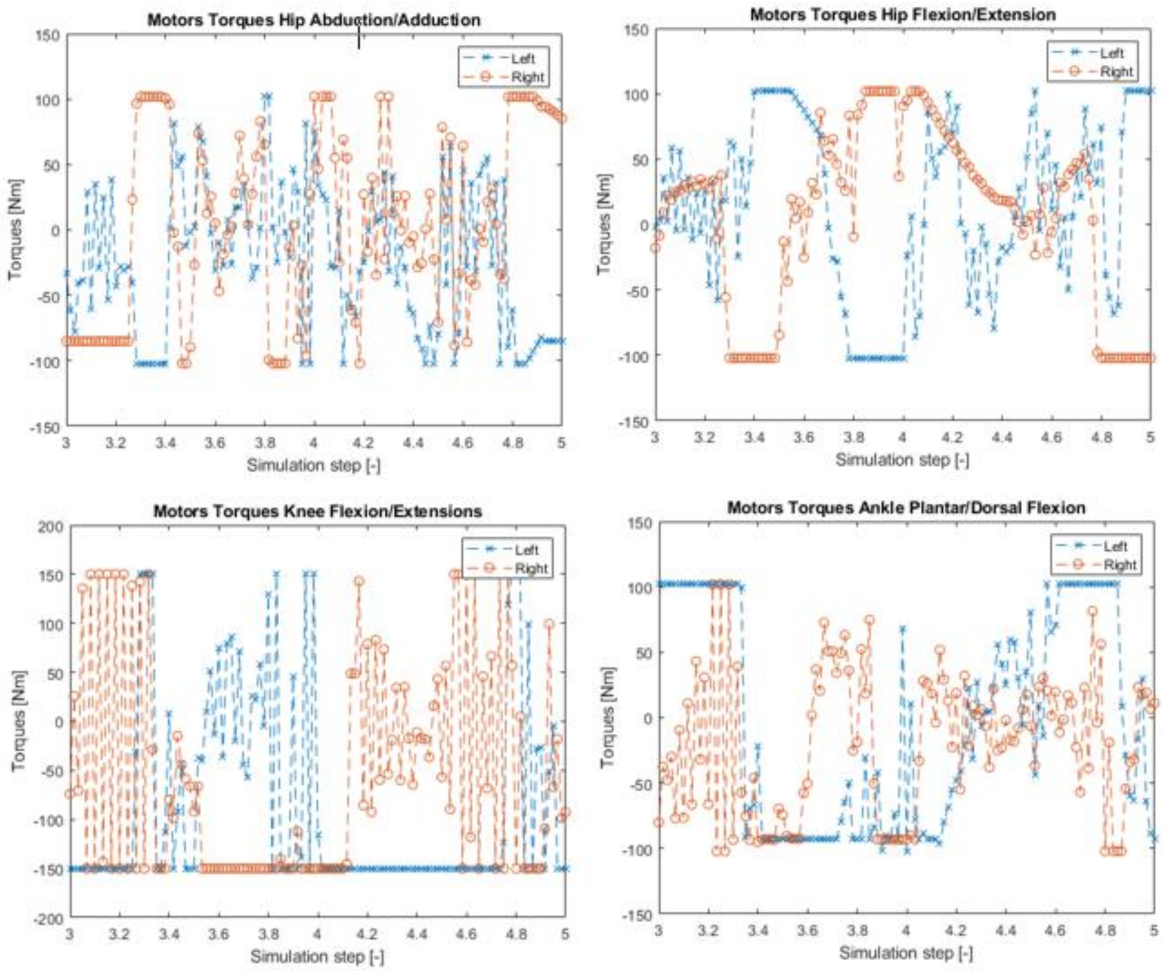


Figure 25. Overview of the torques for the four DOF. The blue crosses show the torque measurements of the left leg. The orange circles are the torque measurements of the right leg.

The top left graph shows the torques for the hip abduction/adduction (hip AA). Between 3.4 and 4.8 seconds, the torque levels alternate around 0 Nm without measurements of absolute maximal torques. This graph does not show a clear pattern in the data sets to determine eventual steps. The top right graph shows the joint torques of the hip flexion/extension (hip FE). Between 3.4 and 4.4 seconds, the graph has similar pattern as in the top right graph of Figure 20. Again, the graph does not show a clear pattern in the torque levels. Only between 3.4 and 4.0 seconds a pattern is recognizable. It seems a step where the hip FE joints alternate between maximal flexion and extension. The tendency to move around zero torque makes it harder to distinguish a torque pattern. The third graph (bottom left) shows the knee flexion/extension (knee FE). The torques levels of the knee FE tends to maximal torque extension. The graph shows a pattern for the torque level in the knee FE joint. The pattern is that one of the two legs has maximal torque extension. In the pattern, the other leg shows flexion of the knee torques. The flexion torque is not maximal or even torque in the extension direction, but overall, the torque is directed towards flexion. Some moments, the torque fluctuates around zero (3.6 to 3.8 seconds for the left leg). The pattern might indicate a single leg swing for each leg between 3.6 and 4.8 seconds. The final graph (bottom right) shows the torque activation of the ankle joint. The graph does not show any clear pattern for the torque activation and leg swings. The dorsal flexion of both ankle joints is at the same time maximal or fluctuates around zero in the selected time window. The graph does show that the left leg has higher joint torques in plantar and dorsal flexion compared to the right leg.

5. Discussion

The results discussed two trained policies for the movement of an exoskeleton. This chapter includes the discussion of the results and the interpretation of the results. The discussion has two sections, one for the results of the SCS policy and the second for the policy for the Cybathlon obstacles.

5.1 SCS Policy

The results of the SCS policy consist of different evaluations and quantities. The discussion of the SCS policy will include different subsections to discuss the results of the policy.

5.1.1 Qualitative Body Sway

The SCS policy shows a behaviour which looks like human gait. Section 4.1 shows the result of the qualitative assessment of the policy's behaviour. The last criterion of posture and orientation has a score of 2 as the base body sways noticeable around the forward axis. The sway of the body may result from the hip AA joint torques. Figure 20 shows that the hip AA joint tend to have maximal torques for both the left and right joint. The joint torques may be maximal because the shift of body weight to a leg causes an increase of moment around the joint. The measured joint torques could act opposite to the change in moment. However, it does not explain the observed sway motion of the top body.

Another explanation for body sway is that the hip FE joint is relative weak (Neumann, 2016). In the case of a weak hip FE joint, the gait of humans shows a semicircle movement in the swing phase. The semicircle movement is the result of rotation of the hip AA and the hip FE joint(Neumann, 2016). (Neumann, 2016) results apply to gaits observed by humans. Here, the system is an exoskeleton attached to the human. Therefore, the assumption is that the hip FE is too weak but simulation or training with a stronger hip FE joint should prove the assumption.

Another possibility is a difference in the dynamics of the exoskeleton attached to a human and dynamics of only a human. For the dynamics of the exoskeleton, the mass of the upper body is static where humans swing their arms and activate upper body muscles. Research shows that the arm swings improve stability during walking (Prakash et al., 2018). The exoskeleton does not include such stability and upper body forces. The lack of these forces/torques may result in compensation in the lower joints, increasing the torques in the lower joints.

To the author's knowledge, the implementation of the arm swing is not straightforward in Isaac Gym. A solution can be a model of the resulting moment of the arm swing to the base body. The resulting moment should depend on the orientation and posture of the lower body and a model to calculate the resulting force.

5.1.2 Torques Gait Exoskeleton

The learned policy for the SCS terrain shows a human-like movement. The results of Figure 20 shows the torques levels of actuated joints of the exoskeleton during movement. The joints have high torque for both rotation directions of the joint. A requirement during training is that the cost of transport negatively impact the obtained reward. The mechanical energy of a system increases with an increase of torque. The requirement of a low cost of transport led to the assumption of minimal torque of the joints, what Figure 20 does not show.

Figure A.2 shows the torque of the hip, knee, and ankle joint of humans during one gait cycle (Han & Wang, 2011). The peak of the torque for the joints are narrower during human gait. All joints show zero torque for a period during the human gait cycle, behaviour what is not observed in the results. However, the torque graphs of Figure A.2 show the nominal torque during the gait cycle. The nominal torque is the required torque divided by the weight of the human. Comparing the nominal torque cycle of the hip to the results in Section 4.3, the peak torque value at 1.3 m/s is 161 Nm. The limit for the hip joint is 102 Nm for the exoskeleton. Under the assumption that the policy can learn human-like gait, the hip FE joint is weak compared to the nominal torque found in the literature (Han & Wang, 2011).

Again, training and simulation with an increase of torque for the joints could prove the assumption that the sway and high measured torque are results of a weak lower limb joint. The knee joint can achieve the required torque within in the current limit based on the nominal torque. The increase of the knee joint may be a result of compensation in the knee joint for the weak hip and ankle joint. The ankle joint should have a torque below 167 Nm. Training and test simulations should increase the torque limit of the hip and ankle joints to 170 Nm to prove the assumption that the observed behaviour results from a too low joint torque limit.

Besides a possible too low joint torque limit, the control frequency of the policy can explain the observed result. During training and testing, the control frequency is 60 Hz. The measurements of Appendix A show continuous control of the torque level whereas the simulation has 60 control inputs per walking step (one step has a duration of 1 s). The Symbitron exoskeleton has electronics and computing processes capable of control input frequencies up to 1 kHz (Meijneke et al., 2021). An increase of the control frequency could also help to achieve a better performing policy. However, the assumption is that the control frequency of 60 Hz is sufficient. The reason is that the reaction time of humans for automatic postural adjustments is around 50 Hz (20 ms) (Nashner & Cordo, 1981).

5.1.3 Cost of Transport

During training of the SCS policy, the software saves the intermediate COT. The results show the development of the COT during training. At the beginning, training of the policy decreases the COT drastically but increases at the end of the training process. The reward function has a term which penalise policies more which have a higher COT. However, the scaling factor for the COT in the reward function is a factor 100 lower than the velocity scaling factor. This may lead to a policy update with a higher average velocity and a higher COT.

Besides the reward function, the increase of the COT could result from overfitting to the training environment. In machine learning, overfitting is the process during learning where the model learns the details and noise of the environment too well (Brownlee, 2019). This may lead to a slight increase in the obtained reward but also to an increase in the COT, as the policy may learn the noise pattern of the surface to act in advance. The advance action may result in the increasing COT at the end of the training process.

The minimal COT during training is 0.83 with a mechanical energy usage of 913.9 J/m. The calculation of the COT does not include the energy conversion losses from battery to motor, so the 913.9 J/m is the lower limit of the COT. Research of (Aoustin & Formalskii, 2018) found that the energy for a single leg swing is ~350 J for a step width of 0.5 meter and a time window of 0.4 seconds. For a complete motion of the gait, the second leg makes a swing as well, covering in total 1 meter. The energy will be for two leg swings 700 J/m. Research of (Winter et al., 1976) found an energy usage during the gait cycle of 585-840 J. The average stride of in the research was between 0.7 and 0.78 meters. They found an energy requirement between 835-1077 J/m. The lowest energy consumption during training lays in the interval found in the literature. However, the increase of the energy consumption led to an energy consumption that is higher than the found interval.

The increase of energy consumption might be due to increase of the mass of the human plus exoskeleton. The COT is a measure which scales the energy usage over the covered distance and the mass of the system. The lowest COT during training was 0.83. The literature found a COT of 0.25 for a human wearing a backpack with the same mass of the battery pack of the exoskeleton. The recorded energy usage of the study is 250 J/m, which is significantly lower than the minimal measured energy usage in the result (Winter et al., 1976).

A remark to the comparison is that the (Winter et al., 1976) measures the COT at a velocity of 1 m/s. A lower velocity may result in a lower COT as the system has less kinetic energy compared to the same system at a higher velocity. Also, energy losses (such as drag) of the system can lead to a higher COT of the system at a higher velocity. New training and test simulations for different desired velocities could shows if the exoskeleton requires more energy compared to humans at different velocities.

5.1.4 Velocity Actor

Besides the torque of the joints in Section 5.1.2, the results show the measurement of the base velocity during movement. The setpoint of for the velocity is 1.5 m/s, which is a above the average velocity of humans. The average velocity of human gait is between 1.2 and 1.4 m/s (Muro-de-la-Herran et al., 2014a). Observation of the velocity during the simulations shows that the velocity of the body swings around the 1.4 m/s. Closer inspection of the velocity pattern shows that at the moments the velocity has its maximal value in the period, one of the hip FE joints shows a flexion torque. A flexion torque means that the hip FE joints swings the leg forward to make a step. After the increase maximal flexion torque, the velocity of the base body decreases significantly.

The inverted pendulum model of the system can explain the sinusoidal pattern of the velocity. The inverted pendulum model models the systems as two legs with the centre of gravity around the height of the hip joints. The inverted pendulum model has sinusoidal pattern for the centre of gravity because the mass tends to fall to direction of movement, increasing the velocity of the centre of gravity (Zelei et al., 2021). Swinging one leg in front of the other will decelerate the centre of gravity.

The mean velocity is 1.3 m/s, which is 0.2 m/s under the setpoint despite the high scaling factor in the reward function for the base velocity. The standard deviation of 0.11 m/s, meaning that the average velocity is between within the window for the average velocity of human gait, which was a goal for the simulations. However, the training and test simulation does not include different setpoints. Currently, only the setpoint of 1.5 m/s shows desired behaviour and quantitative results. In case the actor reaches the target, currently no tests show the velocity behaviour. Further investigation could give insights to the behaviour by setting reachable target positions and different setpoints for the velocity.

Also, test simulation gives the mean velocity of one simulation. For other simulations, the velocity profile might be different leading to a different mean velocity. The test simulation for multiple simulations can include the mean velocity of the epochs. It would give more information on the mean velocity the policy achieves and the standard deviation on the achieved velocity. The mean velocity might even come closer to the velocity setpoint. As the velocity of the actor is zero at the beginning, the recommendation will be to measure the mean velocity from 1 second to the end.

5.1.5 Covered Distance

The results in Section 4.2 shows the covered distance for 1286 simulations. The maximal covered distance is 72.22 meter, which is the end of the simulated surface. The simulated surface has 16 SCS and two extra surfaces as padding, totalling to 18 surfaces with a length of 4 meters. The total length of the surface is 72 meters. The extra 0.22 meters can be the result of the forward fall of the actor at the end of the simulated surface. The simulated surface length limits the maximal covered distance possible and induce the skewness in the distance distribution. The average distance the actor covers is 63 meters.

In the simulation, the actor cannot use crutches to regain balance of the upper body. Therefore, the free walking obstacle of the Cybathlon is taken as a reference. The length of free walking in the obstacle course is 4 meters. The obtained mean value of 63 meters is high. 13 of the 1286 simulations achieves a distance less than 10 meters, 1.01 % of the simulations. For the free walking obstacle course, 98.99 percentage of the simulation complete the free walking obstacle. However, humans can walk distances of five kilometres without rebalancing with outside forces, like crutches.

The lower limit of distance is 500m argued by (Mitchell & Burton, 2006) as a maximal comfortable distance. Then, the mean distance of 63 meters is low. One of the reasons is that the length of the simulated surface. In further research, the length of the simulated surface can be doubled. The recommendation is to lower the number of lines from 16 to 8 when the length is doubled to have enough video memory on an 8GB GPU.

5.1.6 Robustness Policy

The simulations are of an exoskeleton worn by a human, where the policy solely determine the mechanical power. For the simulation to real transfer, one should look to the percentage of fallen agents. The resulting SCS policy achieves a percentage of fallen agent 54%, meaning that half of the simulations resets before reaching the maximum simulation time. Based on the found percentage, the recommendation would be to add a fall strategy to the policy in the training in case the policy transfers from simulation to real applications.

There are three remarks on the percentage of fallen agents. Subsection 5.1.4 already mentioned that the simulated surface has a maximum of 72 meters. In case the actor starts with an average base velocity of 1.5 m/s, the travelled distance during the simulation would be 75 meters in 50 seconds. Those actors get the label of fallen agent in the case the actors are successfully completing their tasks.

Second, the training simulation resets actors when the sideway displacement is larger than 2 meters. The limit on the sideway displacement remains active in the test simulations. The drawback is that agents receive a reset because they do not walk within the specified path. However, this does not mean the agent fall when outside of the specified path. The recommendation is to run the test simulations and disable the reset guideline for walking within the specified path.

The last remark is to improve randomness in the simulation to evaluate robustness. The current test simulations include robustness in the form of randomness in the generated surface. Moreover, robustness of walking depends on external disturbances. These are often external forces caused by pushes of bumping into the surroundings for example (Zelei et al., 2021). The research does not include a test with external forces. The behaviour of the policy on external forces may decrease the robustness of the policy.

The assumption is that the policy takes correction step or torque to counter the acceleration of the external force. For increases forces the assumption is that the policy cannot counter the external force and falls. For robustness test, the test simulation should not reset the simulation when the base body drops below a termination height. The policy might recover from a push which led the body to be below the termination height.

5.2 Obstacle Policy

The qualitative results shows that the policy is not able to complete the obstacles. This section includes the discussion of the results and address potential problems.

5.2.1 Different Environments

The initial goal for the obstacle policy is to move across a surface with random placed Cybathlon obstacles. The qualitative evaluation of the short-trained policy showed that tuning of the hyperparameters did not results in a desired policy. Therefore, the software implements the two different environments mentioned in subsection 4.4.4, both without a successful trained policy.

The assumption for failing the obstacles is that the reward function focuses the training too much on achieving the velocity setpoint. To validate the assumption, the test simulation should save the angle deflection and angular velocity of the joints. The data from the test simulation makes a comparison with the literature possible. (Grimmer et al., 2020) present data of the lower limb joints during daily activities, such as walking and ascending/descending the stairs.

It shows that the range of motion of the joints is lower for walking than for stair ascending/descending. (Grimmer et al., 2020)shows that all the range of motion are within the set angle joint limit in Section 2.3. The maximal angular velocity of the ankle joints according to (Grimmer et al., 2020) is above the maximal angular velocity limit of the exoskeleton.

Also, the torque for stair ascending/descending is higher in (Grimmer et al., 2020) than the torque in the exoskeleton. Also, the torque and angular velocity limits are higher for the fast walking than for slow walking. The combination of a high velocity setpoint and stair ascending/descending may increase required torques of the joints. The recommendation is to tune a new reward function with a lower scaling factor for the velocity on the surface with random Cybathlon obstacles.

5.2.2 Base Velocity

As discussed before, the velocity of the base velocity is lower than the setpoint for the velocity. The trend line for the velocity increases towards a velocity of 0.8 m/s. Again, the setpoint for the velocity might be too high for the specification of the exoskeleton, as found by (Grimmer et al., 2020) and (Han & Wang, 2011).

The end of the velocity profile shows an increase in the base velocity. The increase may be the forward fall movement of the exoskeleton. The policy may learn this behaviour to end with a velocity around the setpoint, increasing the cumulative reward significantly. It might compensate maximal for the death cost it might see coming during the data simulation.

(Zelei et al., 2021) shows that the pattern of the horizontal velocity should not change, which the result of the obstacle policy does not. The software trains a policy which is capable of a walking movement however with a lower mean velocity. Based on the trend line of the velocity, the recommendation is to retrain the policy for a velocity setpoint of 0.8 m/s.

5.2.3 Torque of Joints

The torque of the joints in subsection 4.6.2 show a tendency to set the maximal torque to the joints. Therefore, the patterns of Figure 25 are different than the observed pattern of Figure 20. The biggest change of pattern is that the time for one step is longer in Figure 25 compared to Figure 20. The explanation is the difference in the velocity of the base under the assumption that the stride does not change. As the length of each step stays equal, a lower velocity results it takes longer to cover the step length.

Besides a step takes more time, the torque pattern is different during a step. For the hip AA joint, the pattern fluctuates around zero with short moments of maximal torque. The torque of the hip FE, knee FE, and ankle joints shows more resembles to the torque profiles of (Shaari et al., 2015). The difference is the definition of the positive torque level, which is in the result opposite to (Shaari et al., 2015) for the ankle and hip FE joints. The swing phase for the knee joint is not visible in the results. The reason might be that the policy set torque on the knee joints to balance the actor.

5.2.4 Test Reward and Distance

The results of the test simulations give information on the cumulative reward and covered distance. For the obstacle policy, the cumulative reward and the covered distance is much lower than for the SCS policy. The assumption for the reason was that the obstacle policy let the actors moves until the first obstacles. However, the mean covered distance is only 2.7 meters, which is 3.3 meters before the first obstacle.

The assumption is that the algorithm observed an optimum for the cumulative reward to walk a mean distance of 2.7 meters before falling forward and to be reset. Furthermore, the skewness of the distribution of the distance is -0.8, indicating a good normal distribution. The -0.8 is much lower than the skewness of -2.9 for the SCS policy. The reduction of the skewness may be due to the large distance between the mean and the distance limit (end of surface for SCS and first obstacle for obstacle policy).

The proposed recommendation of subsection 5.1.5 will strengthen the assumption. The obstacle policy covers 2.7 meters at average with a mean reward of 4868, which is significantly lower than the SCS policy. The reward on covered distance is lower because the obstacle policy fails to complete the obstacles.

An interesting comparison will be a test simulation of the obstacle policy on the SCS surface. The assumption is that a small percentage of actors could cover a larger distance. This may reduce the skewness of the distribution more.

5.3 PPO Algorithm

The results of both policies may also depend on the implementation of the PPO algorithm. This section will discuss limitations and properties of the PPO algorithm which may influence the resulting policies.

5.3.1 Reward Function

The first subject is the formulation of the reward function. The reward function (12) shows that the velocity of the base significantly affects the obtained reward. For the SCS environment, the high scaling for the velocity seems to be beneficial to train a good policy. For the tilted path, the high scaling of the velocity reward is positive, as Section 4.4.4 shows that for a constant sequence of obstacles the policy can complete the tilted path. For the other two obstacles, the high scaling for the velocity reward is not beneficial for the found policy.

A probable reason can be the focus of the obstacles. For completing both steppingstones and stairs, the velocity of the base body may be less important. For the steppingstones it is assumed that the ground reaction forces are more important (otherwise the actor steps outside of the steppingstone). For the stairs, the assumption is that the step length is not too large while stepping on a stair step. Also, the step height should be higher than the height of the steps in the stairs.

The current reward function has no reward or cost terms for the three quantities. Adjustment of the scaling factors in the current reward function and augmentation with the three terms can result in a policy capable of completing the obstacles. In case the reward function changes, the hyperparameters may also be tuned again to the new reward function, which can be done by qualitative evaluation of the policy.

Also, the continuity of the reward function influences the performance of the PPO algorithm. (Hsu et al., 2020) found that for a one-dimensional reward with a high peak, the algorithm starts with a positive increase during training. However, when the algorithm selects actions outside of the reward peak (gaussian nature of the process), the reduction of the learned standard deviation results in a vanishing cumulative reward. The high peak of reward can leave too little information for the algorithm to recover (Hsu et al., 2020).

This problem may occur during training of the algorithm, as the velocity reward is quadratic and has a high scaling factor. On top of that, the learning rate α is adaptive during training. It means that the learning rate decreases over time resulting in a smaller steps size. When the action deviate from the action with the high reward, the decrease in step size decreases the possibility to recover from the policy update.

However, the problem of the exoskeleton is not a one-dimensional problem which makes it difficult to prove if this occurs during training. Although, a recommendation is to define terms in the reward functions that has broad peaks. It might increase the time to converge to a good policy but can increase the performance of the policy.

5.3.2 Recurrent Neural Network

In the environment with the same obstacle at same distance from initialization place, the policy was able to move across the tilted surface. The policy does not overcome the obstacle after the tilted surface, either the stairs or steppingstones. The problem may arise from the implementation of Multi-Layer Perceptron (MLP).

A MLP considers the inputs of all time steps and does not have an input for information of earlier time steps. In combination with PPO algorithm, the PPO algorithm collects timestep data (length is actor times the horizon length) for each iteration and calculate the generalized advantages (length of horizon length). At a new iteration, the algorithm collects new timestep data and discard the earlier data (Schulman et al., 2017). An MLP neural network is suitable for a smooth curved terrain as the difference in terrain heights are minimal. For the obstacles, the actors have different observation for different obstacles.

Recurrent neural networks (RNN) include a memory input in the network to store information on the time data. (Ko et al., 2017) showed that RNN performs at par with MLP for tests in the same environment. For new set points or even new terrains, RNN performs better than MLP (Ko et al., 2017). The test simulations generate a new terrain, which may cause the policy to do not overcome the obstacles. The high percentage of agent may also be reduced by implementing an RNN.

5.3.3 Deterministic and Entropy

This research made the PPO algorithm a deterministic algorithm. Original, PPO selects the action given the state stochastically. The research chooses a deterministic algorithm as the observations of the state should generate one action for each motor. The drawback is that during training, the policy aims to find one solution given the observations. It might limit the exploration of states for the agent.

With a stochastic policy, different and new states might be visited and increase the robustness of the policy. Another factor is the entropy coefficient in the PPO algorithm. The entropy coefficient enables an entropy bonus in the surrogate loss function. The entropy bonus can ensure sufficient exploration, which may be beneficial for a deterministic PPO implementation (Schulman et al., 2017).

5.4 Previous RL Actors

Section 2.3.3 Previous Research mentioned research and design of RL for robotic applications. The findings of the humanoid model in (Makoviychuk et al., 2021) provided the starting point for the design process in this work. The reward function in (Makoviychuk et al., 2021) for the humanoid model has linear reward formulations except for the action reward. The design process in this work added different rewards to the reward function and tuned the scaling factors for the given rewards.

However, the starting point for the design of the reward function had the goal to show locomotion of different models (Makoviychuk et al., 2021). Therefore, the assumption is that the reward function (12) can achieve locomotion for different models on flat surfaces, as (Makoviychuk et al., 2021) trains the models on flat surfaces. As the reward functions (12) originated for flat surfaces, the training for SCS terrain is successful and unsuccessful for uneven terrains.

On top of that, the reward functions in (Rudin et al., 2021) and (Siekmann et al., 2021) have different approaches for individual rewards. (Rudin et al., 2021) implements uneven terrain for the ANYmal, whereas (Makoviychuk et al., 2021) trained the policy on a flat surface. An example between the reward functions is the individual reward for the velocity, where the velocity reward $r_{vel} = 1 - \exp((v_{set} - v_{base})^2)$. Besides the reward function for velocity, (Rudin et al., 2021) and (Siekmann et al., 2021) have different functions for the different rewards. The positive results of the training of their models contributes to the assumption that more work can improve the reward function (12).

In (Siekmann et al., 2021), the Cassie robot can overcome the stairs obstacle, both in simulation and real world application. Besides the difference in the reward function, (Siekmann et al., 2021) uses a Recurrent Neural Network to learn a policy. The implementation of a RNN in future work can increase the robustness of the policy and may help overcoming the selected obstacles.

6. Conclusion & Recommendations

The aim of this work was to design a policy for an exoskeleton performing daily tasks as simulated by the Cybathlon course by means of Reinforcement Learning. In order to achieve a final design, the design process consisted of two policies. The first policy focused on human-like gait on a smooth surface and the second focused on overcoming randomly placed Cybathlon obstacles.

The qualitative evaluation of the first trained policy resulted in a policy which shows human-like gait on the smooth curved surface. The quantitative evaluation of this policy showed that the exoskeleton achieved a velocity of 1.32 m/s (trained for an exoskeleton's velocity of 1.5 m/s). The exoskeleton achieved an average distance of 63 meters in the simulations, with the simulated surface of 72 meter.
The joint torque pattern of the exoskeleton showed more maximal torque during a gait cycle as compared to human gait. The Cost of Transport for the exoskeleton decreased during training at the beginning but increased at the middle of training. The minimal Cost of Transport value was still four times larger than the Cost of Transport measured for humans. The Reinforcement Learning training showed increase in the cumulative reward and converged within 1.5 hours.
The qualitative evaluation of the second policy showed human-like gait for the free walking obstacle but failed to complete the other obstacles. The velocity of the exoskeleton on the free walking obstacle converged to 0.8 m/s before failure. Comparing the torque patterns during human gait, the second policy resembled more to human gait torques as compared to the first policy.
The covered distance during simulation has a mean of 2.7 meters. The training of the second policy resulted in a higher Cost of Transport as compared to the first policy, where the lowest value was 600 and 1, respectively.
The designed policies for the exoskeleton showed promising results on the smooth surface, with a mean velocity near the setpoint and covered distances. Unfortunately, the design is not yet capable of locomotion of the exoskeleton over the four Cybathlon obstacles.

Further research can improve the policy for Cybathlon obstacles. The second policy can be improved by shaping a better performing reward function, preferable a reward function without sharp peaks for actions. Also, the setpoint for the velocity can be too high for the exoskeleton to achieve properly. Further research can focus on variable velocity tackling two problems. First, it can focus on finding an optimal velocity for the exoskeleton and second, it might result in a policy which can manage variable velocity inputs. To improve the robustness of the exoskeleton, the test simulation needs a test with an external force. The external force models a disturbance of the real world, and it is insightful to observe the behaviour to external forces.
Another subject for further research is the implementation of the neural network. A Recurrent Neural Network includes a memory input to the neural network, which is beneficial for multiple tasks. The memory in a Recurrent Neural Network can recognize patterns for the different obstacles, making it more likely to perform better. The last recommendation is research to the training time to learn a policy. Depending on the hardware, the training time could be faster or slower due to more parallelisation.
Further research can focus on the dependency of the number of parallel agents on the training time, may resulting in an optimal number of parallel agents. Besides, researching the training time can give insight if the absolute training time is close to the time of adjusting the exoskeleton to a new pilot.

Bibliography

Amamcherla, N., Turlapaty, A., & Gokaraju, B. (2018). A Machine Learning System for Classification of EMG Signals to Assist Exoskeleton Performance. *Proceedings - Applied Imagery Pattern Recognition Workshop*, *2018-Octob*(1), 1–4. https://doi.org/10.1109/AIPR.2018.8707426

Anam, K., & Al-Jumaily, A. A. (2012). Active exoskeleton control systems: State of the art. *Procedia Engineering*, *41*(Iris), 988–994. https://doi.org/10.1016/j.proeng.2012.07.273

Anonymous. (2020). *ACCELERATING REINFORCEMENT THROUGH GPU ATARI EMULATION*. *2013*, 1–15.

Aoustin, Y., & Formalskii, A. M. (2018). Walking of biped with passive exoskeleton: evaluation of energy consumption. *Multibody System Dynamics*, *43*(1), 71–96. https://doi.org/10.1007/s11044-017-9602-7

AurelianTactics. (2018). *PPO hyperparameters and Ranges*. https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe

Baheti, P. (2022). *Activation Functions*. Activation Functions in Neural Networks [12 Types & Use Cases]. https://www.v7labs.com/blog/neural-networks-activation-functions

Bhagat, S., Banerjee, H., Tse, Z. T. H., & Ren, H. (2019). Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges. *Robotics*, *8*(1). https://doi.org/10.3390/robotics8010004

Briouza, S., Gritli, H., Khraief, N., Belghith, S., & Singh, D. (2021). A Brief Overview on Machine Learning in Rehabilitation of the Human Arm via an Exoskeleton Robot. *2021 International Conference on Data Analytics for Business and Industry, ICDABI 2021*, 129–134. https://doi.org/10.1109/ICDABI53623.2021.9655865

Brownlee, J. (2019). *Overfitting and Underfitting With Machine Learning Algorithms*. Machine Learning Mastery. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Brysona, A. E., & Ho, Y.-C. (1975). *Applied optimal control*. Taylor & Francis Group.

Byun, J.-S., Kim, B., & Wang, H. (2020). *Proximal Policy Gradient: PPO with Policy Gradient*. *Williams 1992*. http://arxiv.org/abs/2010.09933

Copaci, D. (2018). *A High-Level Control Algorithm Based on sEMG Signalling for an Elbow Joint SMA Exoskeleton*. https://doi.org/10.3390/s18082522

François-lavet, V., Henderson, P., Islam, R., Bellemare, M. G., François-lavet, V., Pineau, J., & Bellemare, M. G. (2018). An Introduction to Deep Reinforcement Learning. (arXiv:1811.12560v1 [cs.LG]) http://arxiv.org/abs/1811.12560. *Foundations and Trends in Machine Learning*, *II*(3–4), 1–140. https://doi.org/10.1561/2200000071.Vincent

Gao, Y., Li, J., Zhou, Y., Xiao, F., & Liu, H. (2021). Optimization Methods for Large-Scale Machine Learning. *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2021*, *60*(2), 304–308. https://doi.org/10.1109/ICCWAMTIP53232.2021.9674150

Grimmer, M., Elshamanhory, A. A., & Beckerle, P. (2020). Human Lower Limb Joint Biomechanics in Daily Life Activities: A Literature Based Requirement Analysis for Anthropomorphic Robot Design. *Frontiers in Robotics and AI*, *7*(February), 1–17. https://doi.org/10.3389/frobt.2020.00013

Han, Y., & Wang, X. (2011). The biomechanical study of lower limb during human walking. *Science China Technological Sciences*, *54*(4), 983–991. https://doi.org/10.1007/s11431-011-4318-z

Hsu, C. C.-Y., Mendler-Dünner, C., & Hardt, M. (2020). *Revisiting Design Choices in Proximal Policy Optimization*. http://arxiv.org/abs/2009.10897

Jaeger, L., & Jaeger, L. (2022). *RACES & RULES*. *September*.

Ko, B., Choi, H. J., Hong, C., Kim, J. H., Kwon, O. C., & Yoo, C. D. (2017). Neural network-based autonomous navigation for a homecare mobile robot. *2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017*, 403–406. https://doi.org/10.1109/BIGCOMP.2017.7881744

Learning, M. (2017). Machine learning Tom Micheal. In *Machine Learning* (Vol. 45, Issue 13). https://books.google.ca/books?id=EoYBngEACAAJ&dq=mitchell+machine+learning+1997&hl=en&sa=X&ved=0ahUKEwiomdqfj8TkAhWGslkKHRCbAtoQ6AEIKjAA

Li, J., Rao, R., & Shi, J. (2018). Learning to Trade with Deep Actor Critic Methods. *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*, *2*, 66–71. https://doi.org/10.1109/ISCID.2018.10116

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., & State, G. (2021). *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. http://arxiv.org/abs/2108.10470

Meijneke, C., Van Oort, G., Sluiter, V., Van Asseldonk, E., Tagliamonte, N. L., Tamburella, F., Pisotta, I., Masciullo, M., Arquilla, M., Molinari, M., Wu, A. R., Dzeladini, F., Ijspeert, A. J., & Van Der Kooij, H. (2021). Symbitron Exoskeleton: Design, Control, and Evaluation of a Modular Exoskeleton for Incomplete and Complete Spinal Cord Injured Individuals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *29*, 330–339. https://doi.org/10.1109/TNSRE.2021.3049960

Melo, L. C., & Maximo, M. R. O. A. (2019). Learning humanoid robot running skills through proximal policy optimization. *Proceedings - 2019 Latin American Robotics Symposium, 2019 Brazilian Symposium on Robotics and 2019 Workshop on Robotics in Education, LARS/SBR/WRE 2019*, 37–42. https://doi.org/10.1109/LARS-SBR-WRE48964.2019.00015

Mitchell, L., & Burton, E. (2006). Neighbourhoods for life: Designing dementia-friendly outdoor environments. *Quality in Ageing and Older Adults*, *7*(1), 26–33. https://doi.org/10.1108/14717794200600005

Morales, E. F., & Zaragoza, J. H. (2011). An introduction to reinforcement learning. *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, 63–80. https://doi.org/10.4018/978-1-60960-165-2.ch004

Muro-de-la-Herran, A., García-Zapirain, B., & Méndez-Zorrilla, A. (2014a). Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors (Switzerland)*, *14*(2), 3362–3394. https://doi.org/10.3390/s140203362

Muro-de-la-Herran, A., García-Zapirain, B., & Méndez-Zorrilla, A. (2014b). Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors (Switzerland)*, *14*(2), 3362–3394. https://doi.org/10.3390/s140203362

Nashner, L. M., & Cordo, P. J. (1981). Relation of automatic postural responses and reaction-time voluntary movements of human leg muscles. *Experimental Brain Research*, *43*(3–4), 395–405. https://doi.org/10.1007/BF00238382

Neumann, D. (2016). *Kinesiology of the Musculoskeletal System* (3rd editio). Elsevier.

Nguyen, T. T., Nguyen, N. D., Vamplew, P., Nahavandi, S., & Dazeley, R. (2017). *A Multi-Objective Deep Reinforcement Learning Framework*. *2020*, 1–21.

Physiopedia. (2022). https://www.physio-pedia.com/home/

Prakash, C., Kumar, R., & Mittal, N. (2018). Recent developments in human gait research: parameters, approaches, applications, machine learning techniques, datasets and challenges. *Artificial Intelligence Review*, *49*(1), 1–40. https://doi.org/10.1007/s10462-016-9514-6

Rudin, N., Hoeller, D., Reist, P., & Hutter, M. (2021). *Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning*. 1–14. http://arxiv.org/abs/2109.11978

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization*

Algorithms. 1–12. http://arxiv.org/abs/1707.06347

Shaari, N. A., Isa, I. S., & Jun, T. C. (2015). Torque Analysis of The Lower Limb Exoskeleton Robot Design By Using Solidwork Software. *ARPN Journal of Engineering and Applied Sciences*, *10*(19), 1–10.

Siekmann, J., Green, K., Warila, J., Fern, A., & Hurst, J. (2021). Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning. *Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2021.XVII.061

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *31st International Conference on Machine Learning, ICML 2014*, *1*, 605–619.

Sutton, R., & Barton, A. (2015). *Reinforcement Learning: An Introduction* (Second). MIT Press. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Vamvoudakis, K. G., Wan, Y., Lewis, F. L., & Cansever, D. (2021). *Studies in Systems, Decision and Control - Handbook of Reinforcement Learning and Control* (Vol. 325). http://www.springer.com/series/13304

Wang, Y., He, H., & Tan, X. (2019). Truly proximal policy optimization. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*.

WHO. (2013). *SCI patients*. https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury

Winter, D. A., Quanbury, A. O., & Reimer, G. D. (1976). Analysis of instantaneous energy of normal gait. *Journal of Biomechanics*, *9*(4), 253–257. https://doi.org/10.1016/0021-9290(76)90011-7

Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., & van de Panne, M. (2019). *Iterative Reinforcement Learning Based Design of Dynamic Locomotion Skills for Cassie*. http://arxiv.org/abs/1903.09537

Xu, X., Zuo, L., & Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, *261*, 1–31. https://doi.org/10.1016/j.ins.2013.08.037

Zelei, A., Milton, J., Stepan, G., & Insperger, T. (2021). Response to perturbation during quiet standing resembles delayed state feedback optimized for performance and robustness. *Scientific Reports*, *11*(1), 1–13. https://doi.org/10.1038/s41598-021-90305-4

Zheng, Y., Song, Q., Liu, J., Song, Q., & Yue, Q. (2020). Research on motion pattern recognition of exoskeleton robot based on multimodal machine learning model. *Neural Computing and Applications*, *32*(7), 1869–1877. https://doi.org/10.1007/s00521-019-04567-1

Zhu, W., & Rosendo, A. (2021). A Functional Clipping Approach for Policy Optimization Algorithms. *IEEE Access*, *9*, 96056–96063. https://doi.org/10.1109/ACCESS.2021.3094566

Appendix

A Reinforcement Learning Theory

A.1 Markov Decision Process

The following sequence, see Figure A1, represents an MDP. At a given time-step t, the agent observes the state s_t and reward r_t. Based on the policy (control strategy), the agent decides to perform action a_t. A policy is the mapping from the observed state to an action (Sutton & Barton, 2015). This action results in a transition from s_t to s_{t+1}. Subsequently, the reward function gives a new numerical value r_{t+1}, based on the combination of s_t, a_t and s_{t+1} (François-lavet et al., 2018; Sutton & Barton, 2015). The mapping of the measured state and reward to an action is the policy learned by RL. A policy can be deterministic or stochastic. A deterministic policy maps a state to a guaranteed action a_t, while a stochastic policy maps a probability for action a_t (Silver et al., 2014; Sutton & Barton, 2015). Another term for the sets of actions or states is space. Either the action or

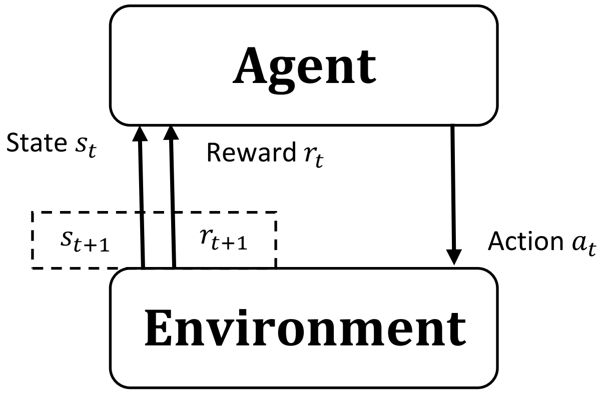


Figure A1. Overview of the sequential MDP in RL, where an agent acts in an environment by an action. The action results in a different state and reward. This process is cyclic until the process meets end states or simulation time.

state space can be discrete or continuous. A discrete space has a finite number of entries, and a continuous space has an infinite number of entries. For example, the action space is between –3 and 3. Discrete spaces can have intervals between actions of 1 (-3, -2, -1, ..., 3) while continuous spaces have all real numbers between –3 and 3 ([-3, 3]) (Lillicrap et al., 2016; Sutton & Barton, 2015).

A.2 Function approximation

In this situation, the model of an actor in a MDP process is not known a priori. Therefore, there is a need for function approximations The goal of function approximations is to match a given function as closely as possible. The approximation is a combination of basic functions with certain weights for these basic functions (Learning, 2017). There are multiple basic functions to approximate a function. The selected approximation in this thesis is the Deep Learning (DL) approximation. DL approximation uses an Artificial Neural Network (ANN) to approximate the function. The construction of an ANN consists of nodes and layers. A node has, in general, a basic function, also called the activation function. A node has multiple inputs and outputs in general, except the input and output layer. A layer in an ANN consists of parallel placed nodes. An ANN has three types of layers: an input layer, an output layer, and at least one hidden layer. The input layer is the first layer, the output layer is the last, and the hidden layers are between the input and output (at least one hidden layer in an ANN). Figure 2 gives an overview of an ANN with one hidden layer (three nodes in the input layer, four nodes in the hidden layer and one node in the output layer).

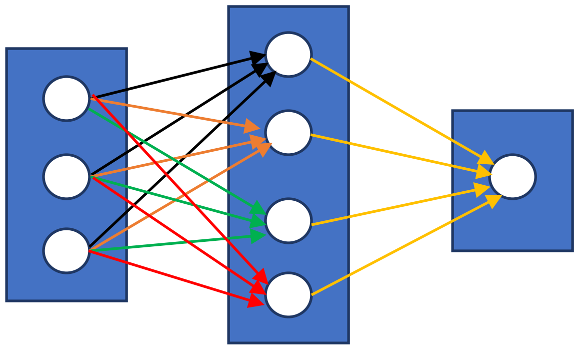


Figure A2. The structure of an Artificial Neural Network. The structure has nodes (white circles), layers (blue bars), and connection between the nodes (colored arrows).

The coloured arrows in the figures are the input values of a specific node. Each input value has a certain weight, and the total input of a node is the sum of the input values times the input weights. Inside the node, the activation function transforms the summed input values. The output of a node is the result of the activation function. There are multiple activation functions, and each activation function has a different transformation (Baheti, 2022). Three general activation functions are tanh, relu, and elu. These activation functions limit the lower bound of the output to -1 for tanh and elu and 0 for relu. The upper bounds are 1 for tanh and infinity for relu and elu (Sutton & Barton, 2015). Figure A2 approximates a function (one output) based on three inputs, for example the position based on the x, y, and z coordinate. *Figure A3* gives a graphical overview of the three activation functions.

Figure A3 shows activation functions which can be implemented in a neural network. The choice of an activation function depends on the properties of the activation function. Each of the activation functions have their own strengths and weaknesses (Baheti, 2022). ELU for examples tries to solve the weakness of ReLU of dead neurons (neurons where the gradient is zero for activation less than zero). However, the more complex activation functions lead to more computational load and possible exploding gradients (Baheti, 2022). The weakness of the Tanh activation function is the outsides of the activation function the gradient becomes small and moves to zero. This problem is called the vanishing gradient problem. It results in that in neurons with activation functions for values larger than 3 results in vanishing gradients (Baheti, 2022).

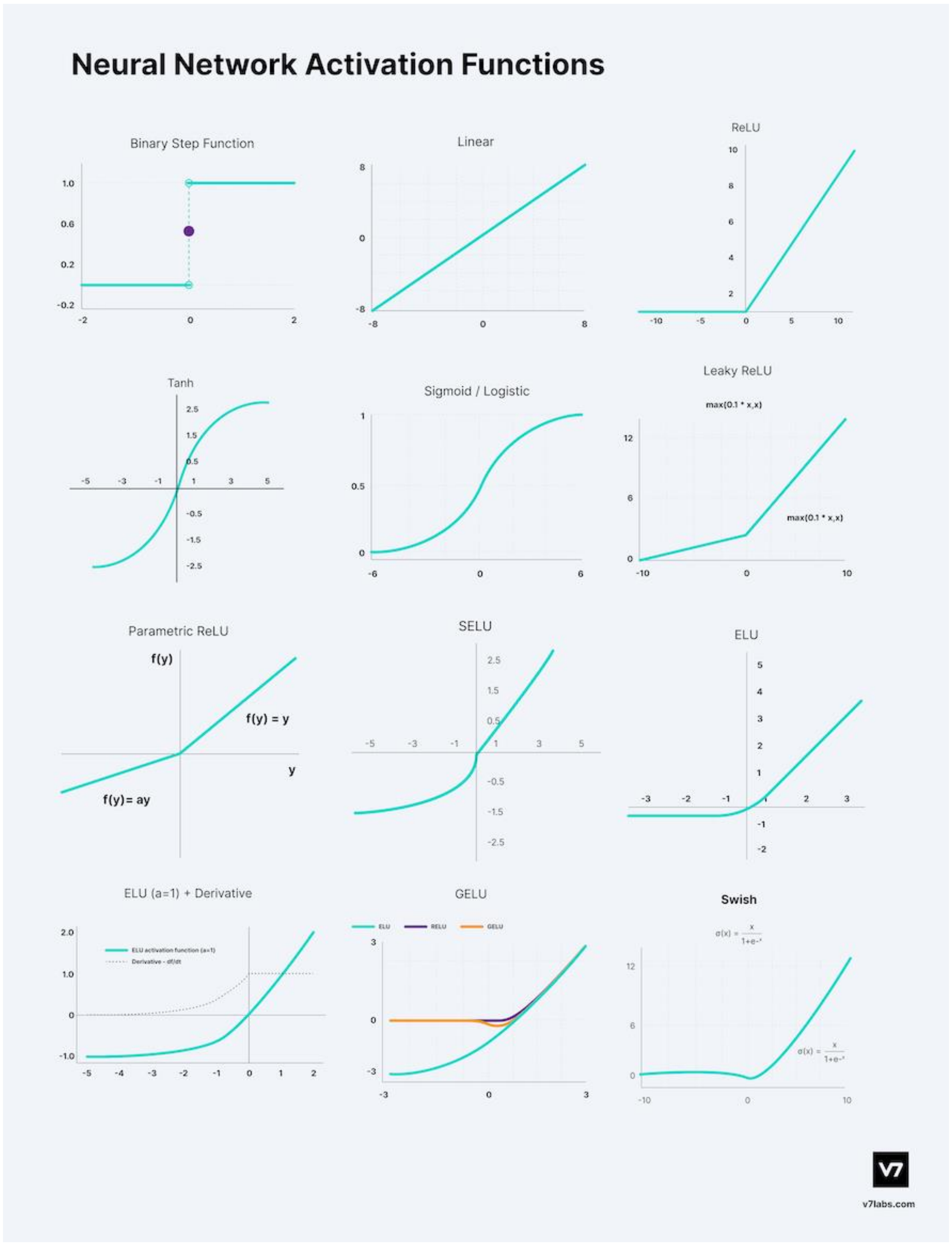


Figure A3. Overview of twelve activation functions for the nodes in a neural network. The section of activation functions discussed the functions ELU, ReLU, and Tanh. The figure shows that there are more possibilities. (Baheti, 2022) *discusses the activation function and use-cases for the activation functions. From* (Baheti, 2022).

g

A.3 Value-based versus Policy-based

An agent can maximize its cumulative reward based on two optimization categories. These categories are the value-based and policy-based optimizations. Often, value-based optimizations use the Q-function, and their aim is to find the best estimation of the Q-function, resulting in a greedy policy. At every time-step, the greedy policy selects the action with the highest state-action value (Morales & Zaragoza, 2011; Nguyen et al., 2017; Sutton & Barton, 2015). To find the best estimation of the Q-function, the algorithm requires an estimation of the state-action value for all actions in a state (Sutton & Barton, 2015). Where the value-based optimizations focus on a good estimation of a value-function, policy-based optimizations estimate the policy by itself. In general, policy-based optimizations estimates the policy with a function approximation (Morales & Zaragoza, 2011; Nguyen et al., 2017; Sutton & Barton, 2015). Appendix A.2 Function approximation explained function approximation in more detail. The estimation of the policy function results in a parameterized policy, and an ANN is the function approximation. Next, the parameters of the policy will be the weights and biases of the ANN where the inputs are the observed states, and the outputs are the actions. To maximize the cumulative reward, the policy-based optimizations have an objective function: the expected return. An optimization method with the policy-based optimization is the gradient ascent, called policy gradient (PG) approach. The policy gradient (1) shows the gradient of a general objective function. $J(\theta)$ is the expected return, Q_{π_θ} is the Q-function under the parameterized policy π_θ. $\nabla \log \pi_\theta(a|s)$ is a function that scores the performance of the policy: $\nabla\pi_\theta/\pi_\theta$ (Bhagat et al., 2019). This equation shows that the policy gradient methods rely on a value function. However, the value function only gives an estimation of the numerical reward (Morales & Zaragoza, 2011; Nguyen et al., 2017). It is preferred to use policy-based methods over value-based methods, because the function approximation in value-based algorithms can overestimate the value-function. However, function approximations are necessary to implement RL to train a controller, as both the action and state space are continuous (François-lavet et al., 2018; Schulman et al., 2017). Due to the overestimation of the value function, the agent selects actions on incorrect estimated values from the value function in case of continuous action and state spaces. In this case, there is no guarantee that these value-based optimizations will converge (Nguyen et al., 2017).

B. Human Gait

The intended use-case of the exoskeleton is to increase the mobility of SCI-patients, as discussed in Chapter 1 Introduction. The goal for the simulations of the exoskeleton controller is to obtain human-like gait. To achieve the goal, the gait parameters help in the analyses of the movement gait. There are multiple parameters for the gait, which can be divided into categories, see Figure A.1 (Muro-de-la-Herran et al., 2014b; Prakash et al., 2018).

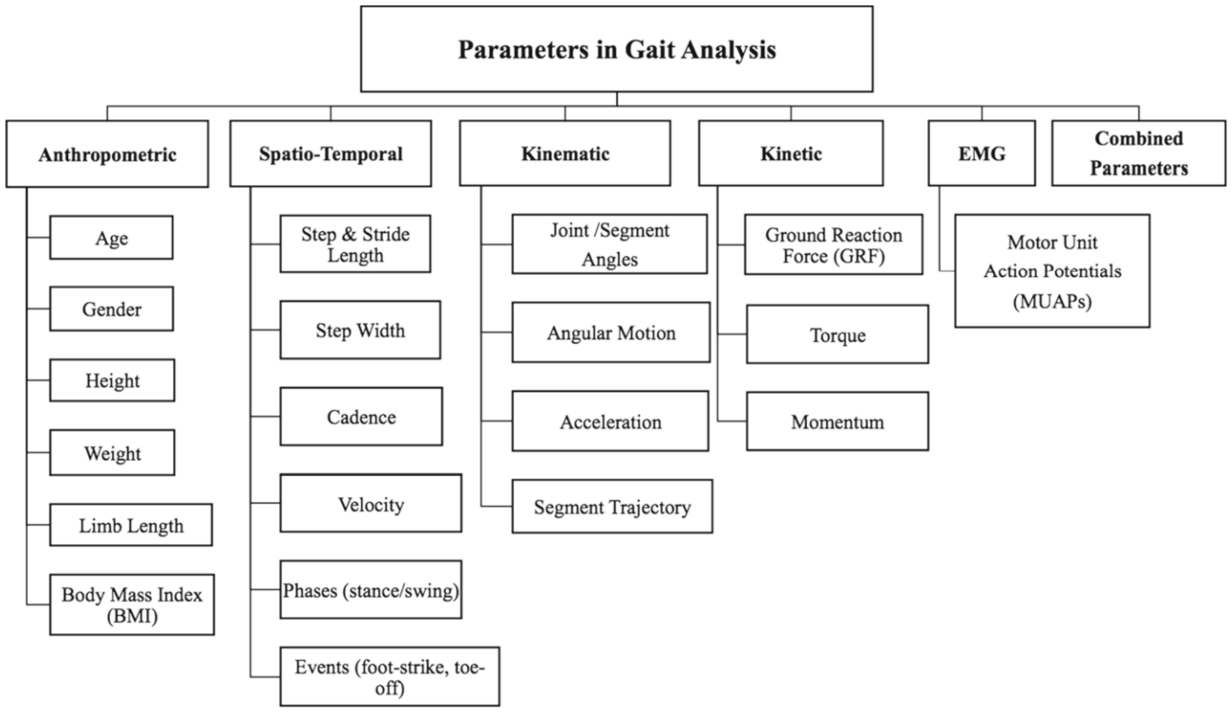


Figure B.1. Overview of the many parameters for gait analyses. The figure shows the categories for the gait parameters. The three categories of interest are the Spatio-temporal, kinematic, and kinetic categories from (Prakash et al., 2018).

The gait parameters in Figure B.1 are for the gait analyses of a human. The trained policy for the exoskeleton results the gait of the exoskeleton. As a simulation of the exoskeleton does not include a human pilot, the three categories of spatio-temporal, kinematic and kinetic are of interest. For the spatio-temporal categories, normal gait parameters of humans are velocity 1.3-1.5 m/s, stride length 1.68-1.72m, step length 0.68-0.85m, and 60.2-62.0% of stride is single support (Prakash et al., 2018). 2.2 Hardware Requirements discussed the kinematics gait parameter of joint angles applied to the exoskeleton. (Winter et al., 1976) discussed the range of motion for the hip, knee, and ankle joint during walking, among others. The last category is the kinetic gait parameters, with the focus on the joint torques. Figure B.2 shows the joint torques during a gait cycle for the hip, knee, and ankle joint (Han & Wang, 2011). (Winter et al., 1976) studies this gait parameter and found comparable results in pattern and torque levels.

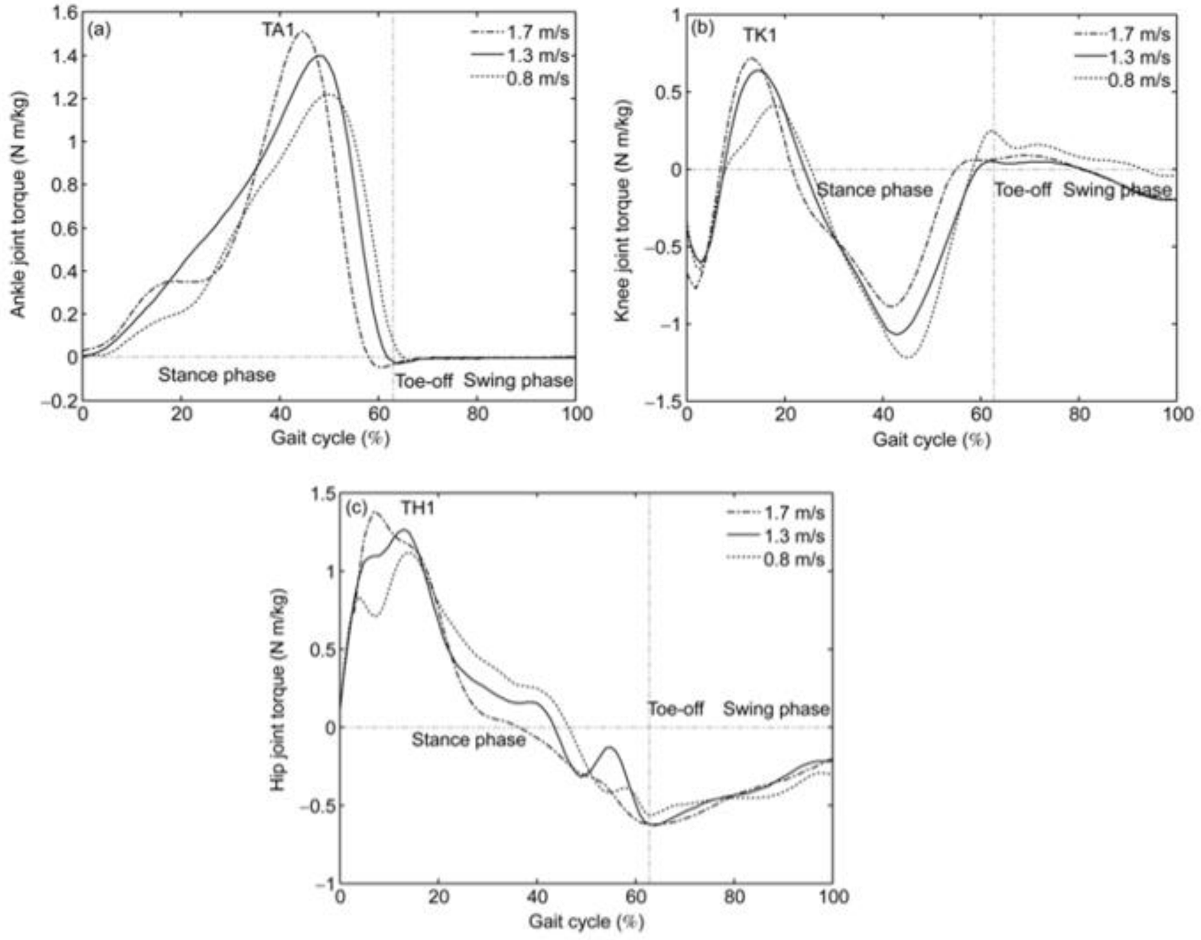


Figure B.2. Normalized torque levels of the hip FE, knee FE and angle flexion. The angle torque shows plantar flexion with a small torque for dorsal flexion from (Winter et al., 1976)*.*

C. Result Figure

C.1 Figures SCS Policy

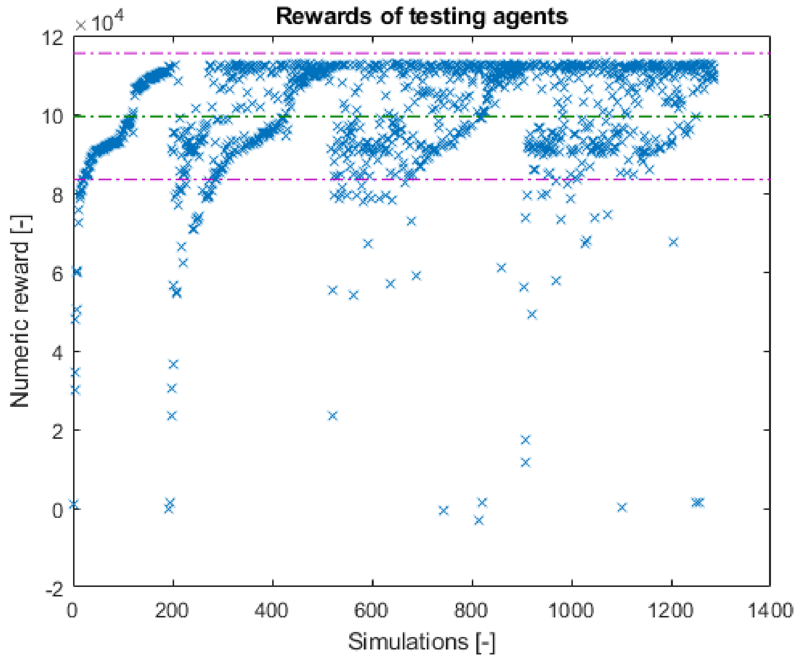


Figure C.1. Cumulative reward of actors for multiple simulations. Most simulations achieve a reward between 80.000 and 120.000. The mean reward is 99,440 (green dotted line) with a standard deviation of 16,040 (two purple dotted lines).

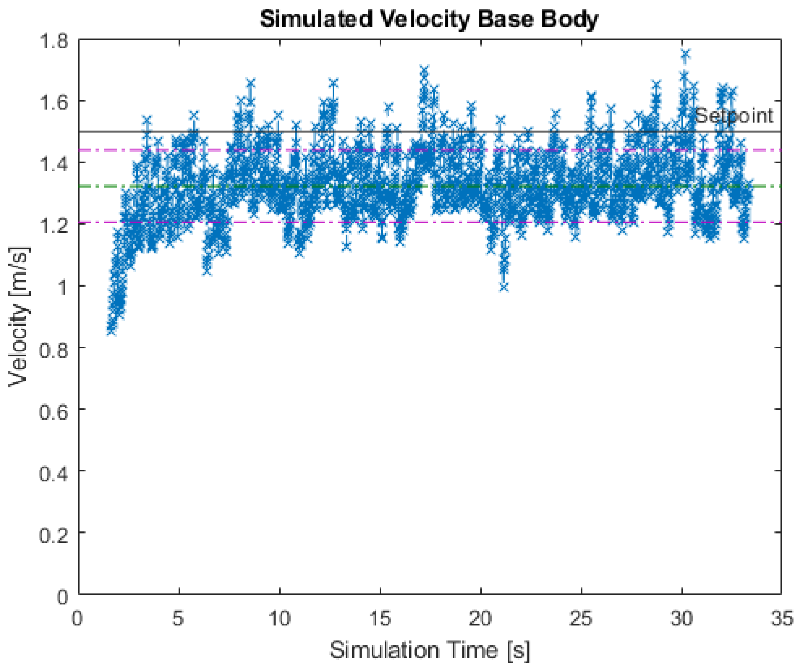


Figure C.2. The velocity of the base link with a constant pattern. The pattern alternates between mean velocity. The mean velocity is the 1.322m/s (green dotted line) with a standard deviation of 0.1167m/s (purple dotted lines).

C.2 Figures Obstacle Policy

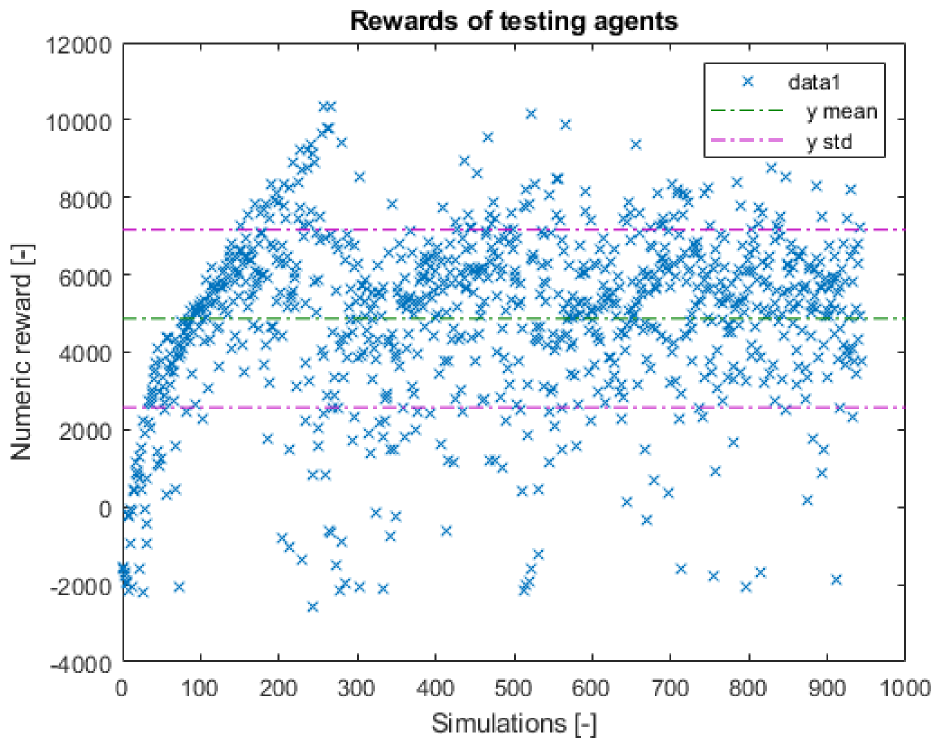


Figure C.4. Cumulative reward for test simulations. The blue crosses are the reward for a single simulation. The green dotted line shows the mean reward value (4868) and the purple lines shows the standard deviation (2296) of the reward.

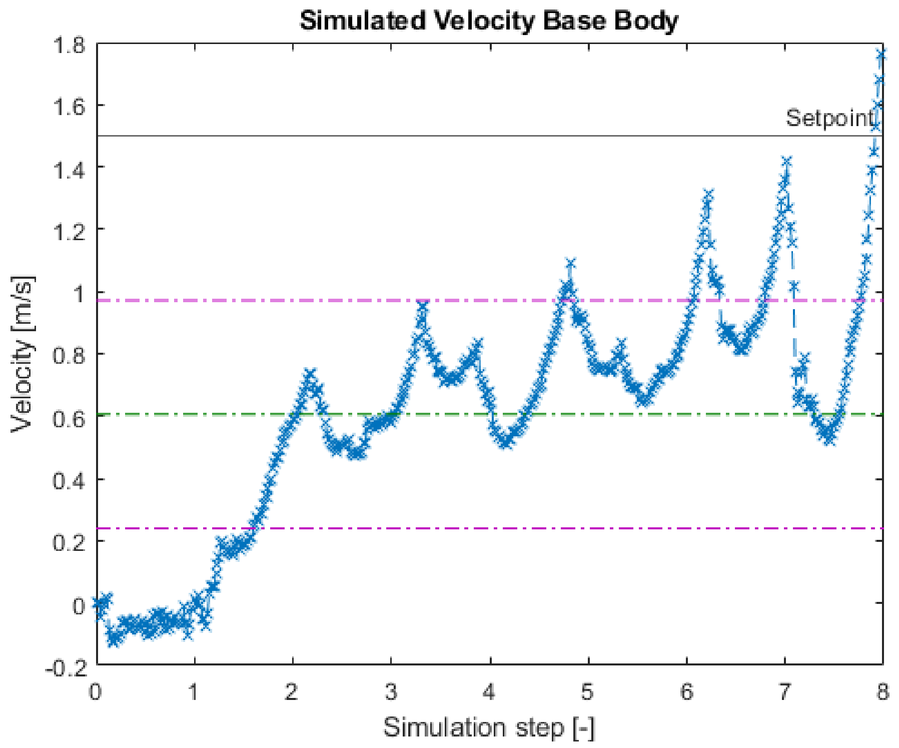


Figure C.4. Velocity of the base body. The blue crosses are the measured velocity. The black line shows the velocity set point and the green dotted line the mean velocity (0.6058) and the purple dotted line the standard deviation (0.366).

D. Simulation Configuration

Exoskeleton.yaml

```
name: Exoskeleton

physics_engine: ${..physics_engine}

env:
numEnvs: ${resolve_default:1024,${...num_envs}}
envSpacing: 2
episodeLength: 1000
enableDebugVis: False

clipActions: 1.0
powerScale: 4

# Reward parameters
headingWeight: 0.7
upWeight: 0.5

# Cost Parameters
actionsCost: 0.05
energyCost: 0.1
dofVelocityScale: 0.1
angularVelocityScale: 0.25
contactForceScale: 0.01
jointsAtLimit: 0.01
deathCost: -2000
terminationHeight: 0.5

asset:
assetRoot: "../..assets"
assetFileName: "urdf/WE2Human_Model.urdf"

plane:
staticFric: 1.0
dynamicFric: 1.0
restitution: 0.01

enableCameraSensors: False

sim:
dt: 0.0166
substeps: 2
up_axis: "z"
use_gpu_pipeline: ${eq:${...pipeline},"gpu"}
gravity: [0,0,-9.81]
physx:
num_threads: ${....num_threads}
```

```
solver_type:                                                ${....solver_type}
use_gpu:                                  ${contains:"cuda",${....sim_device}}
num_position_iterations:                                                     3
num_velocity_iterations:                                                     1
contact_offset:                                                              2
rest_offset:                                                             0.001
bounce_threshold_velocity:                                                 0.2
max_depenetration_velocity:                                               10.0
default_buffer_size_multiplier:                                            5.0
max_gpu_contact_pairs:                                                 1048576
num_subscenes:                                            ${....num_subscenes}
contact_collection:                                                          1

task:
randomize: False
```

E. RL algorithm Configuration

ExoskeletonPPO.yaml

```
params:
seed:                                                        ${...seed}

algo:
name:                                                    a2c_continuous

model:
name:                                            continuous_a2c_logstd

network:
name:                                                      actor_critic
seperate:                                                          True
space:
continuous:
mu_activation:                                                     None
sigma_activation:                                                  None

mu_init:
name:                                                           default
sigma_init:
name:                                                 const_initializer
val:                                                                  0
fixed_sigma:                                                       True
mlp:
units:                                                    [512,256,128]
activation:                                                        relu
d2rl:                                                             False

initializer:
name:                                                           default
regularizer:
name:                                                              None

load_checkpoint:                      ${if:${...checkpoint},True,False}
load_path:                                             ${...checkpoint}

config:
name:                   ${resolve_default:Exoskeleton,${....experiment}}
full_experiment_name:                                          ${.name}
env_name:                                                         rlgpu
multi_gpu:                                                        False
ppo:                                                               True
mixed_precision:                                                  False
normalize_input:                                                   True
normalize_value:                                                   True
value_bootstrap:                                                   True
```

num_actors:	${....task.env.numEnvs}
reward_shaper:	
scale_value:	0.1
normalize_advantage:	True
gamma:	0.96
tau:	0.93
learning_rate:	2e-3
lr_schedule:	adaptive
schedule_type:	legacy
kl_threshold:	0.005
score_to_win:	200000
max_epochs:	${resolve_default:2000,${....max_iterations}}
save_best_after:	100
save_frequency:	50
grad_norm:	1
entropy_coef:	0
truncate_grads:	False
e_clip:	0.2
horizon_length:	32
minibatch_size:	8192
mini_epochs:	16
critic_coef:	2
clip_value:	True
seq_len:	4
bounds_loss_coef:	0.0002
deterministic:	True