# Understanding motion kinematics using prosthetic devices after lower limb amputation through AI-engineered model

Catalina Botia
Universidad de Los Andes
201729189

Isabella Ramos
Universidad de Los Andes
201730255

Daniela Tamayo
Universidad de Los Andes
201730499

## Abstract

*hola abstract*

## 1. Introduction

Humans have a strictly bipedal locomotion, averaging 6500 daily steps at 1.3 m/s in urban environments. [19] However, lower limb amputations that affect and completely change locomotion are very common every year. For instance, in 2007 there were approximately 1.7 million people with limb loss in the United States. [14] Amputations are caused by many possible reasons; most of them are due to dysvascularity (72%), infections (8%), trauma (7%) and some other reasons. Additionally, 90% of amputations concern a lower limb. [19] In Colombia, there are no recent exact statistics, but the Colombian Association of Physical Medicine and Rehabilitation estimated that in 2006 about 200-300 persons for every 100,000 had had amputations. [5] These numbers suggest that this is a common disability and therefore it is crucial to both understand the biomechanical problem and provide solutions for the people who struggle with it daily.

Limb amputations are significantly distressful experiences that require both physical and psychological adjustment over long periods of time. In fact, physical difficulties such as pain or problems with balance and ambulation are very common in patients who have lost a limb, regardless of the cause. However, recent advances in prosthetic devices offer options that enable the patient to walk, run or simulate regular limb function so they can have a normal locomotion. [3] Still, the costs of amputation and rehabilitation in the community are considerable. A significant percentage of these costs includes the provision of the prosthesis and the process of retraining the patients in order to achieve functional mobility in the community. [4] The significant medical challenge that is not yet fully understood and must be taken into consideration is explaining how walking will change after using prosthetics. This problem is specifically challenging for lower-limb amputees, which will be the target group of our investigation.

Artificial intelligence (AI) has been a formidable tool for biomedical research. In fact, the whole healthcare system has been benefited by technological advances built through machine learning. For instance, automated medical diagnosis, image classification for pathogen identification, recognition of ailments and personalised treatment strategies, health data control, management of patient information, among others. Reinforcement Learning (RL), a sub-field of machine learning, has also proven to be particularly useful in biomedical research. The general idea of RL is to endow an agent with skills and capabilities in behavioural decision making by using evaluative feedback and interaction experience with a specific environment. In simpler words, RL aims to find the best possible behavior or path to maximize a reward in a specific situation or environment. Contrary to supervised learning, RL solves sequential decision making problems with sampled, evaluative and delayed feedback simultaneously. Moreover, RL does not seek a supervised reward signal but attempts to maximize the reward as much as possible. Such unique features make RL a distinctive strategy to tackle healthcare and biomedical problems. The lower-limb prosthetics problem is not an exception. [21] [17]

Due to the great need of new strategies to improve and assist rehabilitation with prosthetic devices, we are going to implement the NeurIPS 2018 *AI for Prosthetics* challenge. This challenge consists of developing a controller to enable a human model with a prosthetic leg to walk and run effectively. To achieve this, we will use the simulation environment *OpenSim* where we can use a human musculoskeletal model to obtain normal motion kinematics data and synthesize physically and physiologically accurate motion with a prosthesis. [1] In other words, we will be modeling how walking and running changes after getting a prosthesis. To approach this problem we will use RL in order to find the balance that maximizes the human model performance while walking. Additionally, we will explore the importance of some parameters of the human model and

their effects in motion kinematics, for example the effects of the human pose.

## 2. Related Work

The state-of-the-art on reinforcement learning with musculoskeletal models can be found in the papers of the previous NeurIPS 2017 challenge *Learning to run*. The paper *Learning to Run Challenge Solutions: Adapting Reinforcement Learning Methods for Neuromusculoskeletal Environments* (Kidziński et al.) [7] explores eight possible solutions, all of which use deep reinforcement learning approaches. The method introduced by the authors for the challenge is Actor-Critic Ensemble (ACE), which aims to improve the performance of Deep Deterministic Policy Gradient (DDPG) algorithm. This method won second place in NeurIPS 2017 challenge *Learning to run*. [7] The DDPG algorithm was proposed in 2015 by Lillicrap et al. The authors presented a method based on the general ideas of Deep Q-Learning by applying them to the continuous actor domain. The algorithm is an actor-critic, model-free method based on the deterministic policy gradient that can operate over continuous action spaces, it is to say, the algorithm is off-policy. [12]

For the *Learning to run* challenge, Kidziński et al. outperformed the DDPG algorithm by adding the ACE. Their idea was to use a critic ensemble at interference time in order to select the best action from proposals of multiple actors running in parallel. This solves one of the problems of DDPG which is dooming actions. In terms of the problem, a "dooming action" occurs when the skeleton enters an unstable state with limbs swinging and falling down. This is a state from which the skeleton cannot possibly recover from. By adding the ACE, the network can inspect the actions at interference time so the critic can anticipate low scores and recognize doom actions. [7]

The biomechanical analysis of this problem and its representation on the *OpenSim* environment for the *Learning to Run* challenge (which could be modified for the *AI for Prosthetics* challenge) is present in a more recent publication by Kidzińsky et al. [6] They explain how a muscoloskeletal simulation environment must be used to answer to two fundamental questions: how to model human systems through the generation of their corresponding equations of motion and how to synthesize motions by integrating these equations over time. Their model was similar to that of Ong et al. in 2017 [13] and included 7 bodies as subdivisions of the real human body: a set of upper leg, lower leg and foot for each of the two legs and a single body representing the pelvis, torso and head. The model also included 18 musculotendon actuators (9 on each leg) to represent the muscle groups in charge of walking. The force of these actuators was dependent on length, velocity and activation level of the muscles, which is what the investigators set as a starting point for developing the equations of motion. [6] They used TRPO (Trust Region Policy Optimization, first presented by Schulman et al. [16]) and DDPG [12] as a baseline due to their previous results and experience on the 2015 publication. [6][7]

Additionally, there has also been a recently advancing research in the use of recurrent neural networks (RNNs) in reinforcement learning (RL). One of the challenges for RL is where part of the state of the environment is hidden, so the agent not only needs to learn the mapping from the environmental states to actions, but also needs to determine in which environmental state it is. These tasks are known as non-Markovian or Partially Observable Markov Decision Processes. Long Short-Term Memory Networks (LSTMs) are a type of RNN that solve the problems in recurrent learning algorithms when learning timeseries with long-term dependencies. They solve this problem by enforcing constant error flow in a number of specialized units, called Constant Error Carrousels (CECs) which have linear activation functions that do not decay over time. They also use input gates (specialized multiplicative units) to access the CECs and prevent them from filling up with useless information. RNNs can be used in RL in many ways; one way is to let the RNN learn a model of the environment which learns to predict observations, rewards and to infer the environmental state at each point. This could be done with an LSTM because they allow the predictions to depend on information from long ago. It is also very useful in the non-Markovian tasks because the agent can learn the mapping from the predicted environmental states in a Markovian way with Q-learning. Another application is to use the RNN to directly approximate the value function of a RL algorithm. This model-free approach approximates the state of the environment by the current observation (input of network) and also with the recurrent activations that represent the agent's history. [2] [15]

## 3. Approach

For starters, the algorithm generates the Open-Sim Prosthetic environment which is set to 3D, prosthetic limb and minimum difficulty. The action space of this particular setting is a vector of length 19, which refers to the number of muscles that could be excited at some point during the simulation. As stated before, one of the promising approaches for developing RL algorithms is Deep Deterministic Policy Gradient (DDPG) [11], which even has great results for bipedal walking problems. [10] Because of this, our baseline method consists of a DDPG agent which incorporates penalties and rewards that are specific for the conext of this problem.

The agent is built up by a fully connected state action input Q-function. Along with it, the agent uses a fully connected deterministic policy, which was adapted from the

| Episodes | Batch size | Step size | Target velocity | Max memory | Max reward | Average reward |
|---|---|---|---|---|---|---|
| 15 | 512 | 2048 | 64 | 64 | 8 | 34.799 |
| 15 | 512 | 2048 | 64 | 64 | 8 | 34.799 |

Table 1. **Experimental Results:** The table is divided in four blocks, each referring to a part of the experiments made. **1.** The first block has original results without changing befault parameters. The next blocks include results for experiments with parameter toggling. **2.** Shifts in dimensionalities of the model, inner hidden dimensions, keys and values representations. **3.** Modifications of other parameters in the architecture including number of heads and layers as well as dropout. **4.** SRU implementation. **Note:** The best result in 15 epochs is highlighted in bold though there are better results for deeper experiments.

open-source code by Yoon [20]. Once the agent is set up, it starts interacting with the environment by running for a determined number of episodes. In our case, this number was initially set to 2000 in an effort to provide the agent with enough room for experimentation and possible improvements. During each episode the agent chooses an action (muscle excitation) depending on the observation of the current state (the state of muscles, joints and velocity) and the information it has learned previously. By performing this action, the reward is calculated according to the equation explained in the section above. The process is repeated iteratively and the best model is saved every time a new higher reward is calculated.

Furthermore, the baseline includes an explorer and a reward shaping function. The first one encourages the skeleton to excite different muscles and learn about them, even if it has knowledge that other muscle excitations might lead to nearly guaranteed positive results. Initially two explorers were tried, an $\epsilon$- Greedy and an Additive Ornstein-Uhlenbeck process, but the $\epsilon$- Greedy method was chosen for this baseline. Secondly, the reward shaping function is essential in the baseline since it improves the learning process by penalizing wrong actions [12]. One of them is falling, which reduces the reward in 10 points. Additional penalties are calculated with body velocity, pelvis position and body position [**?**].

### 3.1. Baseline Results

For the two methods presented previously (with and without explorer and reward shaping function), the agent was trained for a total of 2000 episodes and the reward was evaluated after each episode. Table 2 shows the maximum and final reward for both experiments and Figures 1-2 show this behavior and the changes in reward over time. Note that different strategies result in peaks on each of the graphs, representing episodes where the individual reward was outstanding, but the average trendlines have similar shapes in all of the experiments. We can also see that the Figure 2 has more peaks in the reward value which means that the agent is probably learning better with the reward penalties that were implemented. agregar nuevos exps viejos? MISMOS EJES

The table also shows that the maximum reward achieved

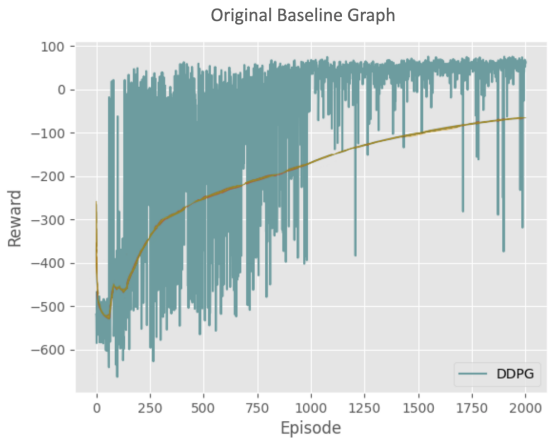


Figure 1. Reward over time for baseline DDPG algorithm with no modifications for $\epsilon$-greedy exploration or reward penalties.

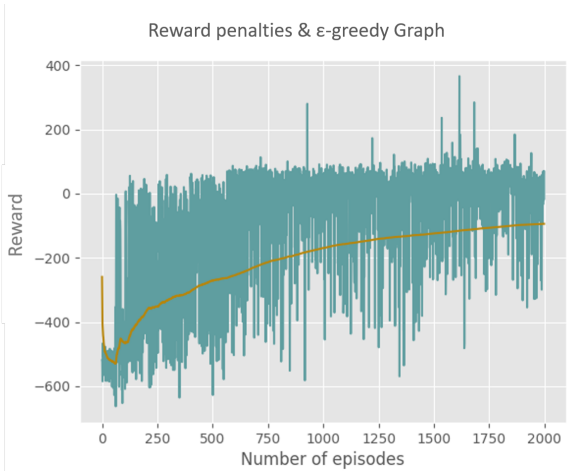


Figure 2. Reward over time for baseline DDPG algorithm with $\epsilon$-greedy exploration and reward penalty helper functions.

by $\epsilon$-greedy exploration and penalizing is way better compared to the initial method. This is a very important aspect that some competitors mentioned because they showed that using different penalties to shape the reward actually allowed the agent to learn the actions that it was supposed

to take in order to get better results [8]. The previous point makes sense because if the agent falls, the pelvis is very low, or the body position is not adequate then the reward will be affected significantly and therefore the agent will learn not to repeat those actions that have negative results.

As for $\epsilon$-greedy exploration, the contrast is clear in Figures 1 and 2, especially after episode 1000. What can be seen is that the baseline without extra exploration capacities mostly stays within the safe zone of positive reward once it manages to learn an approach that generates a decent reward, and therefore positive or negative peaks are scarce. In the other hand, the results with $\epsilon$-greedy exploration show a bigger fluctuation in rewards after epoch 1000, when the agent has already managed to get positive results. These are both very convenient (for example the maximum positive peaks) and inconvenient (nearly up to -600), showing that exploring can lead to excellent results as well as remarkably bad ones.

The results were also evaluated qualitatively since it is important to understand how the skeleton is moving. For the second baseline, which included reward shaping and exploration, the first 100 episodes showed the skeleton mostly falling in similar ways. An example of this is shown in Figure 4. Between 100 and 200 episodes, variations in performance were more evident. In fact, Figure 3 and 5 show the skeleton trying to perform a step in different ways. Some episodes with high reward showed the skeleton performing the step completely while in others the skeleton fell during the step. It is worth highlighting that rendering results in the environment is memory and time consuming, so it was used conservatively. Based on these results, our goal is to further improve the baseline method of using DDPG along with $\epsilon$-greedy exploration and reward penalties in order to fully understand the motion kinematics of this model. cambiar e greedy por el otro porque converge mas rapido

Table 2. Baseline results with DDPG using different types of learning strategies. For the first one, traditional DDPG was used with no helper functions. For the second experiment, $\epsilon$-greedy exploration and reward penalties were implemented.

| **Baseline Results** | **Max reward** | **Last reward** |
|---|---|---|
| Original | 77.2695 | 65.2503 |
| Explorer with penalties | 365.8154 | 68.7983 |

## 4. Experiments

### 4.1. Dataset

Since we are implementing NeurIPS 2018 challenge and using the *OpenSim ProstheticsEnv* simulation environment, we decided to use the same evaluation specified in the contest. A sample of the environment and its parameters can be

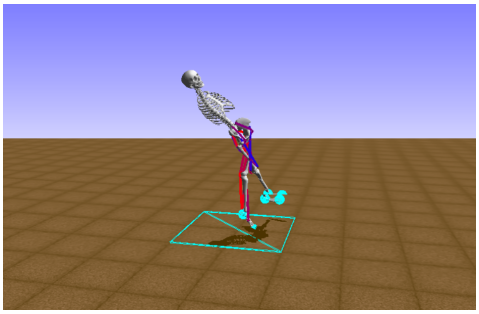

Figure 3. Visualization of prosthetic environment with baseline training for less than 100 episodes. Shows the skeleton falling.

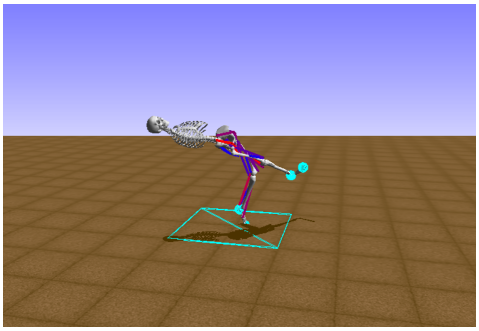

Figure 4. Visualization of prosthetic environment with baseline training between 100 and 200 episodes. Shows the skeleton leaping its leg forward to take a step, while also having a heavy inclination on the back.

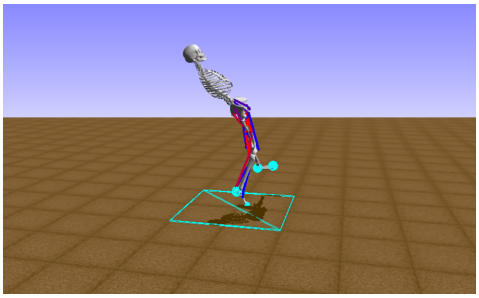

Figure 5. Visualization of prosthetic environment with baseline training between 100 and 200 episodes. Shows the skeleton taking a step.

seen on Figure 6. The general idea is to build a function that takes the dictionary describing the current state observation and returns a 19-dimensional vector of the muscle excitation actions. The objective is to follow a velocity vector and try to maximize the reward, defined by $R$ in the following equation:

$$R = 9 * s - p * p$$

This equation can be interpreted as the request to run at a constant speed of 3 meters per second. The absolute differ-

ence between horizontal velocity and 3 $\frac{m}{s}$ is represented by $p$ while $s$ stands for the number of steps reached before one of the stop criteria. There are two stop criteria that end the trial. First, if the pelvis of the model falls below 0.6 meters, and second, if 1000 iterations are reached (which is to say, 10 seconds in the virtual environment). In the *AI for Prosthetics* challenge all competitors were ranked according to their maximum reward.

| | |
|---|---|
| **# of muscles** | 19 |
| **# degrees of freedom** | 14 |
| **reward** | negative distance from requested velocity |

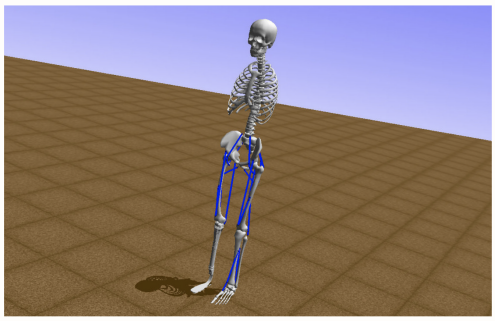

Figure 6. OpenSim's Prosthetics Environment showing a sample of a person with a lower limb prosthetic device along with the 19 main muscles that will be studied. [7]

### 4.2. Experiments

We performed some experiments using the baseline code and the *chainerrl* library. chainer exps

The original baseline using the chainerrl library was very time consuming because it was impossible to use the GPU, specifically, 2000 episodes took 16 hours. Also, we could not experiment with the actor and critic networks due to the fact that the library already had them implemented. Because of these reasons, we decided to write another code implementing the DDPG algorithm using pytorch. We expected this new algorithm to be faster and to let us experiment and understand the networks. In this new baseline we used the experience replay buffer, the actor (policy) & critic (value) networks, the target networks and the Ornstein-Uhlenbeck Process exploration. In continuous action spaces the exploration is done by adding noise to the action itself. All of the hyperparameters used were from the original DDPG paper "Continuous Control With Deep Reinforcement Learning" [12] and experiments with some of them were performed.

DDPG experiments focused mostly on following a target velocity, shaping the reward and changing hyperparameters such as number of steps, batch sizes, etc. The experiments tried were:

- **Reward shaping A:** The reward obtained in each episode was modified by adding penalties for falling and not following an expected velocity [18].
- **Reward shaping B:** The reward obtained in each episode was modified by adding penalties for crossing legs, deviation from walking forward, and taking steps to the sides. Furthermore, a bonus for bending knees was added. [9].
- **Velocity Loss** As well as the critic loss and actor loss, a velocity loss was calculated comparing the XYZ velocity relevant to the pelvis center point against a target velocity which was set to 3.
- **Velocity penalty/reward** The target velocity was set to 3 and the reward was shaped according this target velocity. If the calculated velocity exceeded the target velocity the reward increased, otherwise it was penalized.
- **Environment Wrapper + Reward Shaping B** The opensim environment wrapper was adapted from [9] and the **Reward Shaping B** was used in this experiment.
- **Environment step + Reward shaping** Some changes were made to the OpenSim environment and some reward penalties were implemented adapted from [18]. A longer time limit was used to reduce likelihood of diving strategy, the reward was penalized proportional to the velocity of the pelvis in z and x, and also if the body position in the pelvis fell. The step function was changed to include the reward with the penalties.

### 4.3. Evaluation and visualization

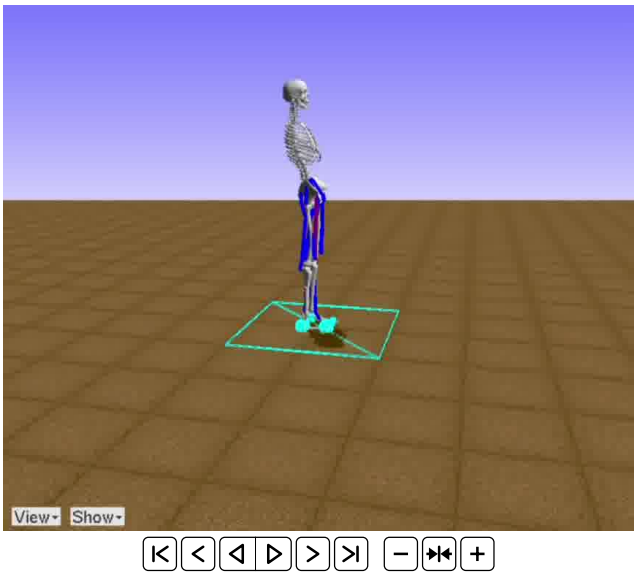


Figure 7. Reward Shaping A

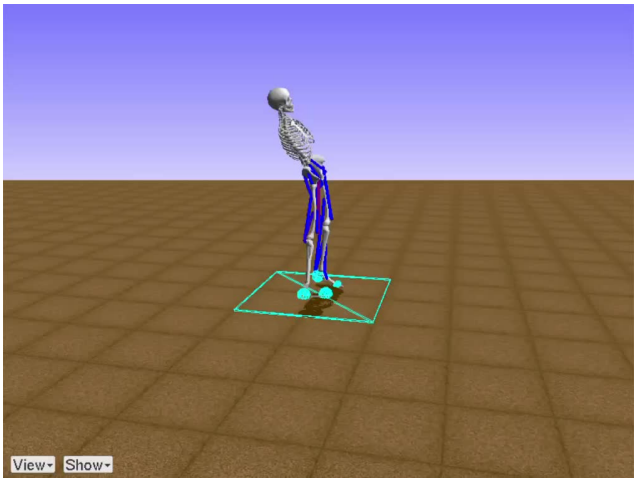


Figure 8. Reward Shaping B

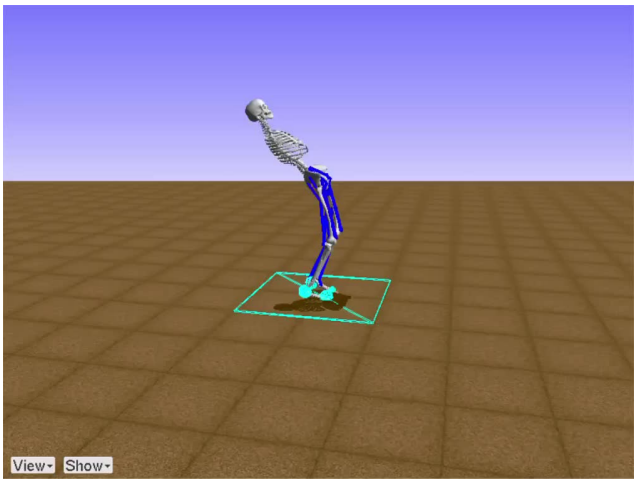


Figure 9. Reward Shaping B

## 5. Discussion

## 6. Credits

## References

[1] Neurips 2018: Ai for prosthetics challenge. https://www.crowdai.org/challenges/neurips-2018-ai-for-prosthetics-challenge, 2018. (Accessed on 08/24/2020).

[2] B. Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14:1475–1482, 2001.

[3] C. Crerand and L. Magee. Amputations and prosthetic devices. In *Encyclopedia of body image and human appearance*, pages 1–7. Elsevier, 2012.

[4] B. Davies and D. Datta. Mobility outcome following unilateral lower limb amputation. *Prosthetics and orthotics international*, 27(3):186–190, 2003.

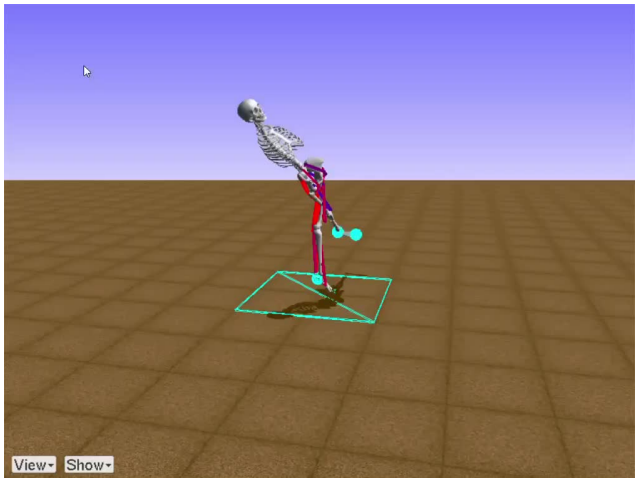


Figure 10. Reward Shaping B

[5] El Tiempo. Sí hay salida para los amputados - eltiempo.com. https://www.eltiempo.com/archivo/documento/MAM-1992341, April 2006. (Accessed on 11/26/2020).

[6] Ł. Kidziński, S. P. Mohanty, C. F. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 101–120. Springer, 2018.

[7] Ł. Kidziński, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, et al. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 121–153. Springer, 2018.

[8] Ł. Kidzinski, C. Ong, S. P. Mohanty, J. Hicks, S. Carroll, B. Zhou, H. Zeng, F. Wang, R. Lian, H. Tian, et al. Artificial intelligence for prosthetics: Challenge solutions. *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, page 69, 2019.

[9] S. Kolesnikov and O. Hrinchuk. Catalyst.rl: A distributed framework for reproducible rl research. *arXiv preprint arXiv:1903.00027*, 2019.

[10] A. Kumar, N. Paul, and S. Omkar. Bipedal walking robot using deep deterministic policy gradient. *arXiv preprint arXiv:1807.05924*, 2018.

[11] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

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

[13] C. F. Ong, T. Geijtenbeek, J. L. Hicks, and S. L. Delp. Predictive simulations of human walking produce realistic cost

of transport at a range of speeds. In *Proceedings of the 16th International Symposium on Computer Simulation in Biomechanics*, pages 19–20, 2017.

[14] M. R. Pitkin. *Biomechanics of lower limb prosthetics*. Springer, 2009.

[15] A. M. Schäfer. Reinforcement learning with recurrent neural networks. 2008.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[17] G. L. Team. Use of reinforcement learning in healthcare. *Great Learning Blog, Power ahead*, 2020.

[18] R. Wightman. Pytorch reinforcement learning for opensim environments. `https://github.com/rwightman/pytorch-opensim-rl`, 2018.

[19] M. Windrich, M. Grimmer, O. Christ, S. Rinderknecht, and P. Beckerle. Active lower limb prosthetics: a systematic review of design issues and solutions. *Biomedical engineering online*, 15(3):140, 2016.

[20] C. Yoon. Rlcycle. `https://github.com/cyoon1729/RLcycle`, 2020.

[21] C. Yu, J. Liu, and S. Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.