BALNEO RESEARCH Balneo and PRM Research Journal English Edition
e ISSN 2734 - 8458 p ISSN 2734 - 8466 ROMANIAN ASSOCIATION OF BALNEOLOGY

Case report

Simulated Bipedal Exoskeleton Gait Learning Using Reinforcement Learning Techniques

Diana Marusic [1], Gelu Onose [2,+], Galina Marusic [1,+], Andrei Bragarenco [1] and Mihaela Rusanovschi [1,*]

Citation: Marusic D., Onose G., Marusic G., Bragarenco A., Rusanovschi M. - Simulated Bipedal Exoskeleton Gait Learning Using Reinfor-cement Learning Techniques *Balneo and PRM Research Journal* ***2025**, 16(4): 942*

Academic Editor(s):
Mariana Rotariu

Reviewer Officer:
Viorela Bembea

Production Officer:
Camil Filimon

Received: 14.09.2025
Published: 30.12.2025

Reviewers:
Mihai Hoteteu
Mariana Rotariu

Publisher's Note: Balneo and PRM Research Journal stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Technical University of Moldova;
2. University of Medicine and Pharmacy "Carol Davila", Bucharest, Romania

* Correspondence: mihaela.rusanovschi@iis.utm.md,
+ These authors contributed equally to this work.

Abstract: This paper introduces a Reinforcement Learning (RL) approach as a method for training a computer simulation of a bipedal lower limb exoskeleton to walk, without explicit supervision. A simulation of a bipedal exoskeleton with six joints (left and right hip, knee, and ankle) was developed, including an implementation of the Proximal Policy Optimization algorithm for training the RL agent. The system was implemented using Python and OpenAI Gym library, which provided an environment to simulate interactions and learn locomotion dynamics. The RL-trained agent is capable of learning stable locomotion by interacting with the simulated environment and a complex reward system. The results demonstrate the potential of RL for adaptive control of exoskeletons and serve as a foundation for further research in exoskeleton control and training.

Keywords: Reinforcement Learning, Exoskeleton, Policy, Agent, State, Action, Reward

1. Introduction

Reinforcement Learning (RL) is an area of Machine Learning (ML) where an agent is placed in an environment, receiving positive or negative rewards, with the purpose of learning a policy that can be used later for the agent to interact effectively with the environment. In RL, compared to other ML algorithms, the focus is on the agent learning through the processes of exploration and exploitation, without the need of explicit supervision [1, 2].

This paper introduces a method for simulating the behavior of a bipedal lower limb exoskeleton, acting as a simulated computer agent in a simulated environment, with the purpose of training the agent to walk on two simulated legs of the exoskeleton. The Proximal Policy Optimization (PPO) algorithm [3], which is a type of RL algorithm, is used in order to find the most optimal policy for a simulated bipedal exoskeleton with 6 joints: left hip, left knee, left ankle, right hip, right knee, right ankle. The learned policy can be used for research purposes as well as adapted a real-world model of a lower limb bipedal exoskeleton.

Bipedal lower limb exoskeletons have been introduced in various researches [4, 5]. A bipedal lower limb exoskeleton is a wearable mechatronic system designed to assist and augment leg motion. In the context of mobility, it can be used either to support individuals with movement impairments or to extend physical performance, enabling enhanced interaction with the environment through externally applied mechanical assistance.

Most of the bipedal lower limb exoskeletons represent complex systems, with hardware including sensors, actuators and safety mechanisms [4-7]. The advantages of RL algorithms applied to the lower bipedal exoskeletons training include flexibility in adapting to diverse problem domains and learning from experience. However,

for safety purposes, the training should be initially done only on a computer simulation. Later, the obtained results from the simulation can be applied to real-world systems. As making just a small change in the mechanics or motors of a lower limb bipedal exoskeleton (to adapt it for another person for example) changes the problem domain, the flexibility and the ease of re-training the computer simulation with RL becomes essential. In this way, RL enables efficient adaptation to new configurations without requiring a complete redesign of the control strategy, thereby reducing development time and minimizing risks when transitioning to physical prototypes [8, 9].

2. Problem formulation

The problem to be solved, formulated in a RL framework, can be represented as a tuple (S, A, P, R, γ) [10], where:

- S - the set of all possible states, which represent different configurations that the RL agent can find itself in during the interaction with the environment;
- A - the set of possible actions. At each step, the hip, knee and foot can be moved with combinations of foot_torque, knee_torque or hip_torque actions;
- P - the transition probability function. Describes how the RL agent transitions from one state to another based on the action it takes;
- R - the reward function;
- γ - the discount factor, in our case γ=1 since the agent is optimizing for the total sum of all future rewards, rather than prioritizing short-term gains.

In the formulated framework, the agent is considered the simulated bipedal exoskeleton. The gait strategy it tries to learn is defined by the policy.

2.1 States

A state represents a configuration of the RL agent (the simulated bipedal exoskeleton) at a given time step within the learning process.

In this framework, each state encodes the positions of the hip, knee, and foot joints for both legs. Assuming on the real exoskeleton are sensors on all the mentioned joints, they would serve as input data to determine the current system's states. A state *s* contains the x and y positions of each of the joints (hip, knee, foot), measured by sensors, as defined below in (1):

s= [hip1_x, knee1_x, foot1_x, hip2_x, knee2_x, foot2_x] (1)

Depending on the application and the mechanics of the lower limb bipedal exoskeleton, the state definition could be extended to include additional features, such as joint velocities, angular orientations, or ground contact indicators, which can possibly improve the results obtained by applying the RL algorithm.

2.2. Actions

An action is represented by the power given to each actuator (6 in total - hip, foot, knee on each leg). On the real exoskeleton, these actuators would be placed in the same positions (hip, foot, knee).

At each iteration of the algorithm, an action is represented by a 6-dimensional array *a*, as described in (2):

a = [l_hip_torque, l_knee_torque, l_foot_torque, r_hip_torque, r_knee_torque, r_foot_torque] (2)

where:

- *l_hip_torque* - the torque to be applied to the left hip joint actuator
- *l_knee_torque* - the torque to be applied to the left knee joint actuator
- *l_foot_torque* - the torque to be applied to the left foot joint actuator
- *r_hip_torque* - the torque to be applied to the right hip joint actuator
- *r_knee_torque* - the torque to be applied to the right knee joint actuator
- *r_foot_torque* - the torque to be applied to the right foot joint actuator

As the movement is done by alternating legs, at each step, only the actions corresponding to the moving leg are applied.

2.3. Rewards

In the RL algorithm, the reward function specifies rewards for being in a state, or doing some action in a state. The agent's objective is to learn a policy that maximizes the expected cumulative reward [1, 10]. In this paper, positive rewards are introduced for the exoskeleton moving forward and with correct joints position, while negative rewards are introduced for penalizing positions of the joints that are impossible in the real life.

3. Materials and Methods

3.1. Computer simulation of the bipedal lower limb exoskeleton

In this paper, a computer simulation of the bipedal lower limb exoskeleton was created, by using the Python library *matplotlib*. The simulation assumes that the bipedal lower limb exoskeleton has at least 3 sensors and 3 actuators on each leg: for the ankle (marked with red color), knee (marked with green color) and hip (marked with blue color) joints, as represented in Figure 1.s

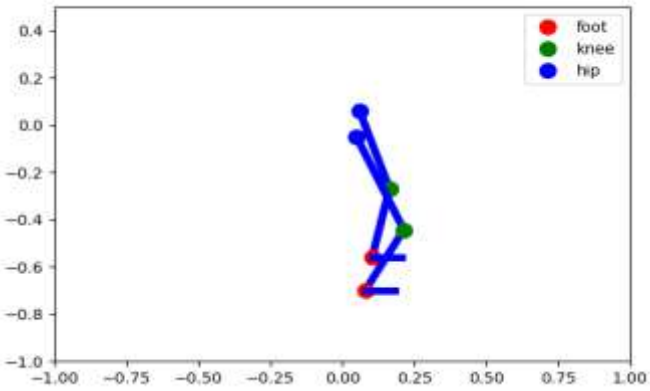


Figure 1. Representation of the simulation lower-limb bipedal exoskeleton joints: ankle (red), knee (green), and hip (blue).

3.2. Reward system

For training the simulated bipedal exoskeleton walking with RL, rewards based on real-world constraints were defined. A complex reward system was implemented, with positive rewards for the simulated exoskeleton "learning to walk" correctly in the forward direction, and negative rewards for positions that could destabilize it or for joint configurations that are not biologically correct. Real bipedal exoskeletons typically include mechanical safety mechanisms that restrict their range of motion to non-dangerous limits [4, 7]. In contrast, in an RL algorithm, potential dangers resulting from destabilization or wrong joints placement can be addressed by introducing negative rewards during the learning process.

In order to encourage the stable and correct forward walking learning, the agent (represented by the bipedal exoskeleton) receives positive rewards when the joints are in the correct position, one foot is on the ground, and the exoskeleton remains balanced. Additional rewards are provided if forward movement is achieved. Negative rewards are given whenever the exoskeleton adopts a position that would cause it to fall, which can occur for various reasons (e.g., hips misaligned, one or both knees lifted excessively high, etc.).

To further ensure correct joint positioning, a negative reward is assigned whenever a joint is inverted or misaligned. At each step, one foot should remain in contact with the ground, which also influences the reward function. For the training of the RL algorithm specifically, the reward system is additionally informed by principles of real-world biomechanics. Accordingly, rewards are structured so that biologically plausible states are reinforced, while anatomically implausible or unstable states are penalized through negative or zero rewards.

Mathematically, the reward function introduced in this paper can be expressed as a combination of rewards for various joints position, as well as the rewards for the posture and the inclination of the exoskeleton, as described in (3).

$$R = R_{hip} + R_{knee} + R_{foot} + R_{forward} + R_{posture} + R_{inclination} \quad (3)$$

where:

R_{hip} - the reward for keeping the hips balanced;
R_{knee} - the reward for keeping the knees balanced;
R_{foot} - the reward for keeping the feet close to the ground;
$R_{forward}$ - the reward for moving forward;
$R_{posture}$ - the penalties for unrealistic body configurations (knee above hip, foot above knee, foot ahead of knee, etc.);
$R_{inclination}$ - the reward for keeping the hip inclination in a stable range.

We define the following variables and constants:

h_y: hip y-coordinate;
h_x^{prev}, h_x^{curr}: hip x-coordinate at previous and current timestep;
k_y, k_x: knee coordinates;
f_y, f_x: foot coordinates.
$h^* = 0.0$ (hip target);
$k^* = -0.4$ (knee target);
f* = GROUND_POS (foot target height);
$c_{hip} = 2$, $c_{knee} = 1.5$, $c_{foot} = 1.0$, $c_{fwd} = 20.0$.

With the variables defined above, the components of the reward are defined as following:

Reward for the hips balance: $R_{hip} = 2 \cdot \max(0{,}1 - |h_y - h^*|)$

Reward for the knees balance: $R_{knee} = 1{,}5 \cdot \max(0{,}1 - |k_y - k^*|)$

Reward for keeping foot close to the ground: $R_{foot} = \max(0{,}1 - \frac{|h_y - h^*|}{0{,}2})$

Forward movement reward: $R_{forward} = \begin{cases} c \cdot (h_x^{curr} - h_x^{prev}), & h_x^{curr} > h_x^{prev} \\ 0, & otherwise \end{cases}$

Reward for the posture: $R_{posture} = \begin{cases} -2, k_y > h_y \\ -2, f_y > k_y \end{cases}$

Additionally: $R_{posture} += \begin{cases} -100, f_x > k_x \wedge k_x < h_x^{curr} \\ 0, \quad otherwise \end{cases}$

$$R_{posture} += \begin{cases} -10, f_x > k_x \\ 0, \quad otherwise \end{cases}$$

Reward for the hips inclination:

$$R_{inclination} = \begin{cases} 2 \cdot \max\left(0,1 - \frac{|\theta|}{45}\right), h_{2,x} \neq h_{1,x} \\ -1, \quad otherwise \end{cases}$$

where

$$\theta = \arctan\left(\frac{h_{2,y} - h_{1,y}}{h_{2,x} - h_{1,x}}\right) \cdot \frac{180}{\pi}$$

3.3. Policy optimization

The problem formulated above is solved through training a Proximal Policy Optimization (PPO) algorithm to maximize the reward. PPO is a type of RL algorithm that uses the stochastic gradient ascent method to perform each policy update. Its advantages include stability and reliability, as well as the ease of implementation.

In the field of exoskeleton research, recent studies have applied PPO to train controllers for lower-limb exoskeletons, highlighting its effectiveness in producing adaptive and stable gaits [8].

Experiments performed on simulated robotic locomotion show that PPO outperforms other policy optimization algorithms. Compared to other online policy optimization RL methods, PPO prevents overly large updates that can destabilize learning which is beneficial in the context of a bipedal exoskeleton that learns to walk and keep a stable posture [3, 12].

Given the complexity of the bipedal locomotion problem, including the need for balance, coordination across multiple joints, and robustness, the ability of the PPO algorithm to achieve stable learning with relatively simple hyperparameter tuning makes it a strong candidate for simulated exoskeleton gait learning.

4. Results and discussions

This section presents the results of the RL algorithm applied on the simulation of the lower limb bipedal exoskeleton learning to walk. The learning process was evaluated by analyzing the states and rewards generated during training with the PPO algorithm.

4.1 Code implementation

The Reinforcement Learning model was implemented in Python using OpenAI's Gym library [6], which provides a standardized interface for defining environments and interacting with agents. In this environment, the *step()* function models the transition dynamics of the bipedal exoskeleton by returning the next state, the calculated reward, and a termination signal. The *calculate_reward()* method computes a positive or negative reward for the agent, encouraging stable walking behavior while penalizing instability or inefficient movements. Additionally, a *reset()* method and a *get_initial_positions()* method are defined to initialize the environment at the beginning of each training episode, ensuring reproducibility and consistency. The *_render_frame()* method is called after each step to update the visualisation of the simulated bipedal exoskeleton.

The state space consists of joint positions, velocities, and sensor readings, while the action space corresponds to control signals for the actuators. By structuring the environment in this way, the PPO algorithm can efficiently learn and optimize a control policy, leveraging its stable and reliable training characteristics in the context of continuous control tasks such as locomotion.

Using Python and the Gym library simplifies implementation, allows rapid prototyping, and provides access to a wide range of existing RL tools and environments. Additionally, it also facilitates reproducibility, integration with other scientific libraries, and easy modification of custom environments for experimentation [11].

4.2 Visualisation of the results

A visualisation for the simulation of the bipedal exoskeleton was also implemented using the matplotlib library. The advantages of having a human-friendly visualisation include better understanding of the bipedal learning process, the strengths and weaknesses of the chosen RL algorithm, as well as the explainability of the learning process with RL. Although training the RL algorithm does not necessarily require such a visualisation, it is much easier to understand how the simulated bipedal exoskeleton learns and where are the possible problems or drawbacks in the learning process. This is particularly important when considering the transfer of results to real-world applications [9].

In the figures below, taken from the visualisation of the RL learning process with PPO, can be seen that the reward function effectively encourages biologically plausible joint configurations and stability of the exoskeleton. As a result, the distribution of positive and negative rewards provides direct insight into how well the agent learns human-like gait patterns.

Figure 2 presents two examples of states with positive rewards. In the first representation, the simulated left leg of the exoskeleton is on the ground, while the right leg is in the air, but the overall position of the exoskeleton seems stable, vertical, and joints are in positions that are biologically possible. In the second figure, the joints are also in a state that is biologically possible for the humans, while the general position seems very stable. This demonstrates that positive rewards are effective in maintaining the stability of the exoskeleton in the computer simulation, while encouraging positions of the joints that are biologically possible for the humans.

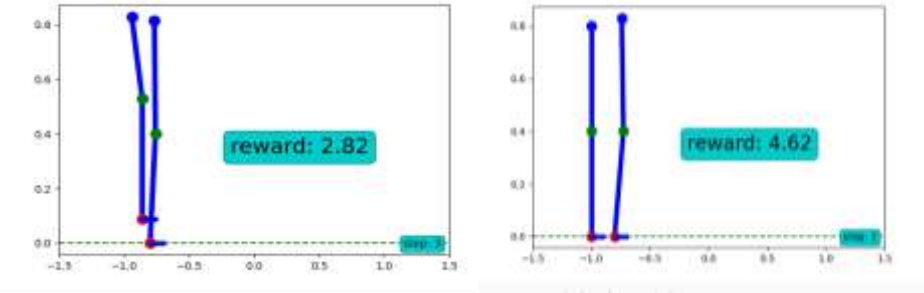
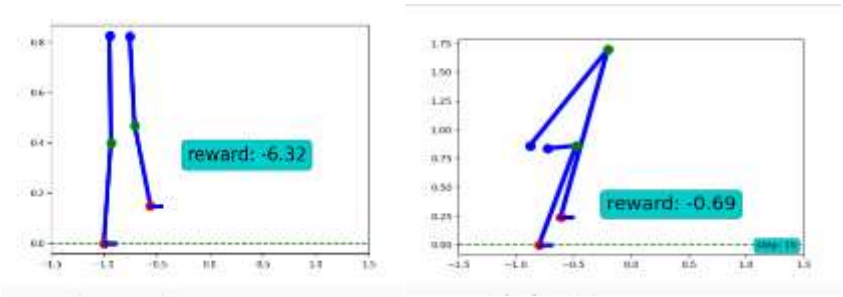


Figure 2. Example of states with positive rewards.

Figure 3 illustrates representative examples of states that were assigned negative rewards during training of the bipedal exoskeleton with the Proximal Policy Optimization (PPO) algorithm. These states correspond to configurations that are either impossible for the humans or unstable, such as unnatural joint alignments or incorrect knee positioning. The reward function penalizes such configurations to guide the agent away from non-physiological movements and towards locomotion patterns that are both feasible and stable. As shown in the figure, large negative reward values are associated with extreme deviations from biologically possible postures.



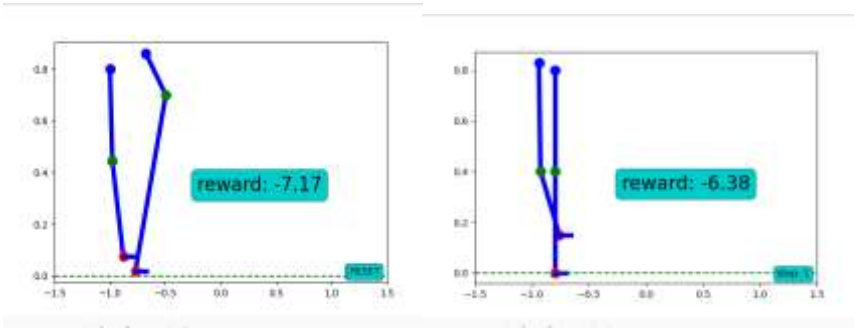


Figure 3. Example of states with negative rewards.

The results presented above demonstrate that the positive and negative rewards presented in this implementation mostly encourage the exoskeleton to be stable, to avoid falls, and to keep the joints in biologically possible states.

5. Conclusions

This paper presented an implementation of a Reinforcement Learning (RL) algorithm for the simulation of training a lower limb bipedal exoskeleton to walk. In this framework, the agent, represented by the exoskeleton, interacts with the environment, receives rewards based on biologically plausible or implausible states, and gradually learns a policy that promotes stable locomotion. The design of the reward system plays a central role in the learning process, as it encodes not only the mechanical constraints of the exoskeleton but also the principles of human gait. A complex reward system was proposed in this sense and described in the paper.

The advantages of using RL for the learning of human gait include flexibility and improvements with time. RL algorithms provide flexibility in adapting to complex problem domains, where explicit modeling of all possible states and transitions would be infeasible. By using RL, the system improves directly from experience, through exploration and exploitation. This allows the system to identify movement patterns that maintain stability and encourage forward progression. Compared to rule-based or predefined control strategies, this approach enables adaptive behavior that can adjust to changing conditions.

The implemented simulation serves as a foundation for developing control strategies in exoskeletons by validating concepts in a virtual environment before transferring them to real life systems. This work highlights the potential of RL as a robust approach for a simulated exoskeleton training and control, opening opportunities for future research in the transfer of the results from the simulation to real life.

This research focused on encouraging the simulated bipedal exoskeleton to remain stable, prevent falls, and maintain joint movements within biologically possible ranges. Future work will aim to improve the forward progression, as well as the exoskeleton's ability to support more natural and efficient locomotion.

Author Contributions: Conceptualization, G.O. and D.M.; methodology, D.M and G.M.; software, D.M. and M.R; validation, A.B.; resources, A.B.; writing — original draft preparation, D.M.; writing—review and editing, M.R.; supervision, G.O. and G.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018; pp. 1–552.
2. Murphy, K. Reinforcement Learning: An Overview. arXiv 2024, arXiv:2412.05265, submitted.
3. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. CoRR 2017, abs/1707.06347.
4. Onose, G.; Cârdei, V.; Crăciunoiu, Ş.T.; Avramescu, V.; Opriş, I.; Lebedev, M.A.; Constantinescu, M.V. Mechatronic Wearable Exoskeletons for Bionic Bipedal Standing and Walking: A New Synthetic Approach. Front. Neurosci. 2016, 10, 343. https://doi.org/10.3389/fnins.2016.00343
5. Rodríguez-Fernández, A.; Lobo-Prat, J.; Font-Llagunes, J.M. Systematic Review on Wearable Lower-Limb Exoskeletons for Gait Training in Neuromuscular Impairments. J. NeuroEngineering Rehabil. 2021, 18, 22. https://doi.org/10.1186/s12984-021-00815-5
6. Awad, L.N.; Bae, J.; O'Donnell, K.; De Rossi, S.M.M.; Hendron, K.; Sloot, L.H.; Kudzia, P.; Allen, S.; Holt, K.G.; Ellis, T.D.; Walsh, C.J. A Soft Robotic Exosuit Improves Walking in Patients After Stroke. Sci. Transl. Med. 2017, 9(400), eaai9084. https://doi.org/10.1126/scitranslmed.aai9084
7. Bengler, K.; Harbauer, C.M.; Fleischer, M. Exoskeletons: A Challenge for Development. Wearable Technol. 2023, 4, e1. https://doi.org/10.1017/wtc.2022.28
8. Luo, S.; Androwis, G.; Adamovich, S.; Nunez, E.; Su, H.; Zhou, X. Robust Walking Control of a Lower Limb Rehabilitation Exoskeleton Coupled with a Musculoskeletal Model via Deep Reinforcement Learning. J. Neuroeng. Rehabil. 2023, 20(1), 34. https://doi.org/10.1186/s12984-023-01147-2
9. Ostraich, B.; Riemer, R. Rethinking Exoskeleton Simulation-Based Design: The Effect of Using Different Cost Functions. IEEE Trans. Neural Syst. Rehabil. Eng. 2024, 32, 2153–2164. https://doi.org/10.1109/TNSRE.2024.3409633
10. Wiering, M. Introduction to Reinforcement Learning. In Reinforcement Learning: State-of-the-Art; Wiering, M., van Otter-lo, M., Eds.; Springer: Berlin, Germany, 2012; pp. 1–84.
11. OpenAI. Gym: A Toolkit for Developing and Comparing Reinforcement Learning Algorithms. Available online: https://github.com/openai/gym.
12. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms; arXiv:1707.06347, 2017.