Gyminy: Teaching a Humanoid Robot to Walk in a Reinforcement Learning Framework

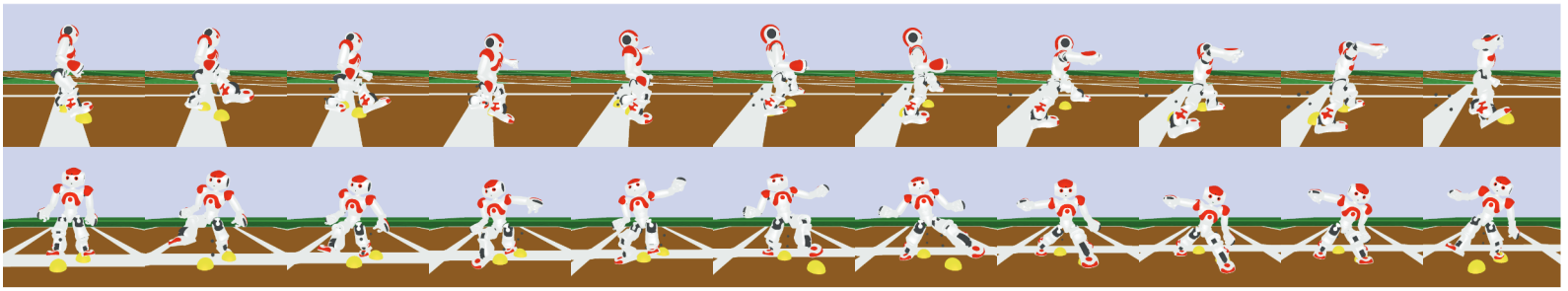
Tecnológico de Monterrey
Campus Ciudad de México
Escuela de Ingeniería y Ciencias
Departamento de Mecatrónica


Alfonso Brown González
A01335743 ITS


Fernando Cossío Ramírez
A00759499 ITS


Dr. Edgar Omar López
Asesor


Dr. David Christopher Balderas
Co-asesor



$$r_t = 0.25e^{\frac{-(pose\_diff)^2}{10}} + 0.3e^{\frac{-(pose\_accel)^2}{10}} + 0.2e^{\frac{-(d\_steps)^2}{10}} + 0.05e^{\frac{-(ankle\_accel)^2}{10}} + 0.05e^{\frac{-(feet\_parallel)^2}{10}} + 0.02e^{\frac{-40|height\_diff|^2}{10}} + 0.02e^{\frac{-10|pitch|^2}{10}} + 0.01e^{\frac{-10|yaw|^2}{10}} + 0.1e^{\frac{-10|roll|^2}{10}}$$

1. Introduction

Reinforcement Learning has become a huge topic of interest in the **Artificial Intelligence** community, but there are relatively few applications to **robotics**. This is because it can be very hard to implement RL to solve **large problems**, and with robots this is often the case. We believe that the best way to maximize AI's contribution to robotics is through the **democratization** of RL. To do so, we decided to develop this project as a tool and guide to get started in the research and implementation of Reinforcement Learning in a NAO humanoid robot.

2. Objective

Integrate a **programming framework** with a **simulation environment** for the **NAO robot** that is:
- Based on **Reinforcement Learning**.
- Programmed through **reward functions**.
- Able to **evaluate** the performance of the **learning algorithm**, **hyperparameters**, and **reward functions**.

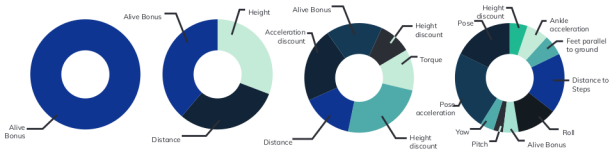


Figure 2. Evolution of the weighted reward functions tested, starting from very general in the left, to the very specific rewards. The best results came from the most specific functions, as they had more information and were less likely to get stuck in local optima.

3. Methodology

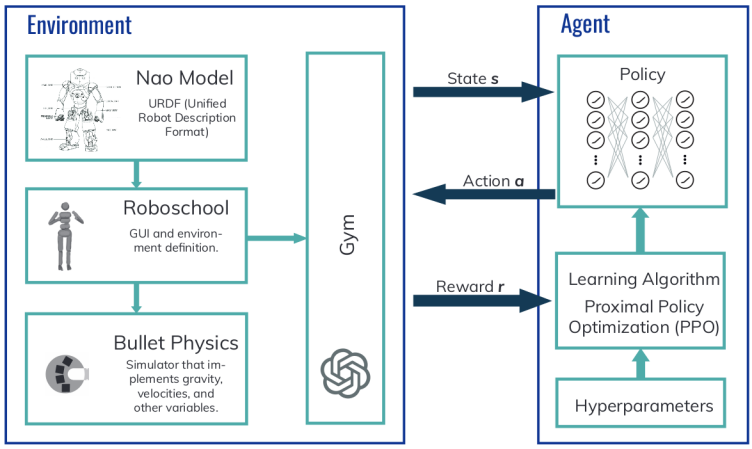


Figure 1. Robot simulation framework created. At every timestep, the agent receives an observation from the environment containing information of the current state. The agent then decides on an action based on its policy, which results in a reward. The learning algorithm adjusts the agent's policy depending on the reward. The process is then repeated.

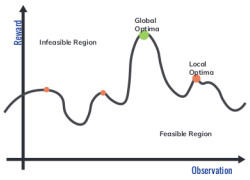


Figure 3. Comparison between Local Optima and Global Optima in an arbitrary reward function.

Defining Reward Functions

Ideally, with sufficient exploration, an agent should always be able to find the global optima. In reality this doesn't happen. The reward function is the best guide an agent has of how to act optimally and a function that doesn't give enough information will get stuck at local optima.

4. Results

Vectorized simulation environment based on Roboschool that allows multiprocessing. Over 16x training speed factor in a 48-core AWS EC2 Instance running Ubuntu.

Average time standing up of 1.62 seconds. Maxmimum reward of 44.7, average reward at convergence of 17.1. Convergence was reached after 3 million timesteps

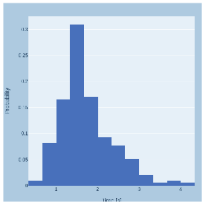


Figure 4. Episode duration (time standing up) distribution.

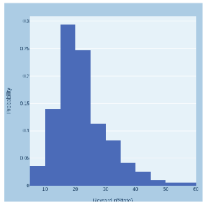


Figure 5. Distribution of rewards obtained per episode.

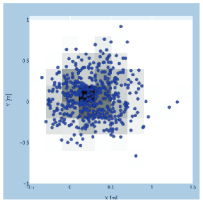


Figure 6. Distribution of the agent's final position.

5. Conclusions

The **most important** factor to reach an appropriate solution is the **reward function**, which has to be:
- Non-sparse
- Non-negative
- Continuous
- Normalized

Reinforcement Learning **always optimizes rewards**, but the solution is not necessarily the expected.

6. Future Work

Implement the trained agent in the real robot.

Implement a hierarchical control scheme that allows to program more complex tasks.