ME598: Deep Reinforcement Learning **Fall 2018**

Project Report

Ehsan Saleh *Professor Matthew West and Timothy Bretl*

Introduction

Reinforcement Learning can provide policies for controlling robotic movements. Recent progress in robotics devices development [1] raised the need for intelligent robot controllers. In this project, we will try to design a robotic controller for a single-foot robot by simulation. The robot was intended to avoid obstacles that come across while running on a conveyor belt. The reinforcement learning model used for this project was the proximal policy optimization [2].

Implementation Keys and Details

For the purpose of simulation, we chose to use bullet as the the physics engine, and the pybullet API for creating simulation environments. We will highlight some of the challenging parts of using this physics engine properly:

- Enabling self collision including the parent links can make the simulation more realistic. Especially since non-trained agents tend to make the robot hit itself.
- Defining the right friction models has high impact on the behavior of contacts. The default contact properties of this engine seem to be slippery. The robot description file can be found in the "urdf" folder of the supplementary code and material.
- Deactivating the joint motors is also an important step. The joint motors are turned on by default.

Furthermore, reward shaping is also a very crucial step for training a successful agent. We will highlight some of the components used for training agents on similar robots.

- The simplest reward function maximizes the potential energy of the robot. However, the optimal policy for this reward could make the robot jump too high distances.
- To account for the instability of this reward function, we may need to add motor energy consumption cost.
- The motor energy consumption may cause the robot to lean on itself for movement or standing, which is inappropriate. Therefore, adding self collision cost would also be appropriate.
- Adding other costs such as stall torque cost, or penalizing for the angular displacement may also help stabilize the robot movement.
- Colliding with obstacles should also be considered separately for a more significant penalty in order to be avoided properly.

Sample Training curves

In the following figures, you can see the learning curves produced during training the agent.

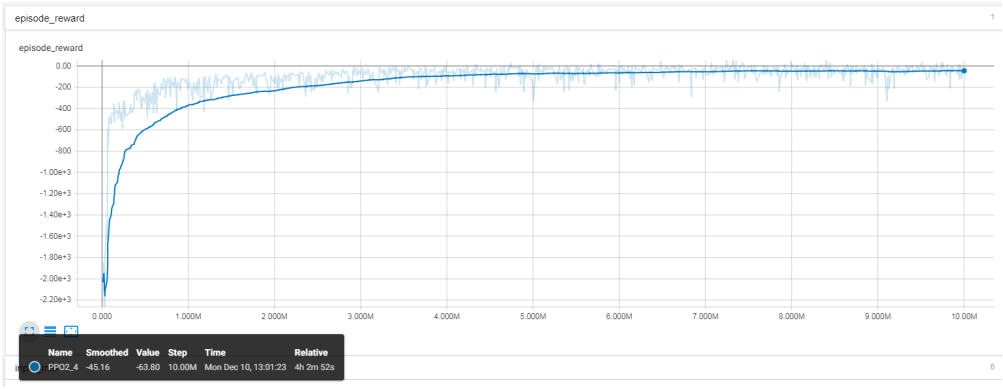

Figure 1: The learning curve for the total episode reward

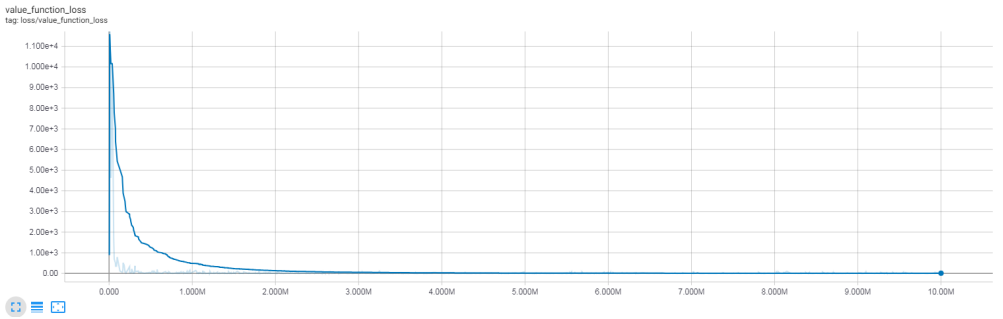

Figure 2: The learning curve for the value function loss

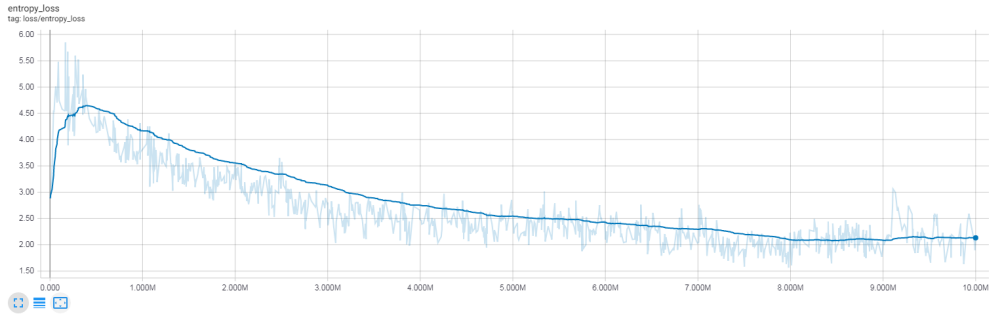

Figure 3: The learning curve for the entropy loss

References

[1] Park, H.-W., Wensing, P. M., and Kim, S. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research 36*, 2 (2017), 167–192.

[2] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms.