Robotics Project

16.06.2020

Guillaume Vandecasteele
guillaume.vandecasteele@gmail.com
Erasmus Brussels University of Applied Sciences and Arts - Postgraduate A.I.

Preface

A video of the agent in action can be found here:
https://youtu.be/_wWXsIcgnmU

All source code can be found on GitHub in this repository:

https://github.com/guycastle/robotics-rl

The repository contains the following folders and files

- **agents**: various scripts to train and test agents
    - **x_test.py**: Test the model
    - **x_learn.py**: Initial training of the model
    - **x_continual_learn.py**: Continue training previously trained model
- **envs/rpi_led_env/rpi_led_env.py**: My custom environment
- **logs**: Tensorboard logs and model training checkpoints
- **raspberrypi/roboticspi.py**: Script to run on Raspberry Pi to receive and send data and control the LED
- **test**:
    - **testenv.py**: Script to test the custom environment without a model
    - **testnotebook.ipynb**: Jupyter notebook to test some small stuff
- **utils**:
    - **DetectPixelScript.py**: Script to test the LED detection
    - **HLSSlider.py**: Script to display window with sliders to see which values to use to filter objects/colors in HLS colour space
    - **HSVSlider.py**: Script to display window with sliders to see which values to use to filter objects/colors in HSV colour space
    - **HSVSliderContours.py**: Script to display window and add contour to detected objects, with slider to adjust HSV values
- **ppo*.zip**: Saved trained models.

Choices

Environment

While developing the environment, I encountered a lot of difficulties to get the red LED to be recognized by the webcam. Having also moved to a new apartment 2 weeks prior to the deadline, my previous color boundary settings were no longer working, and lacking any curtains, I need different settings depending on the time of day, which is why I created a few tools to find correct values on the fly.

Once this problem was more or less done, I focussed on the environment. I wanted to try working with an environment with 9 possible actions; up, down, left, right, idle plus the diagonals. The observation space will contain the [x,y] coordinates of the detected LED, as well as the [x,y,z] data we obtain from both the gyroscope and accelerometer sensors of the Raspberry Pi, totalling 8 values altogether.

Because the data from the raspberry pi seems to be values ranging from -1 to +1, I decided to also normalize the LED coordinates to values between -1 and +1 with [0,0] as the center.

The reward is calculated as the distance from the center (diagonal of the square created from both points) in pixels. The lowest reward being half the diagonal of the image resolution, and the highest being 0.

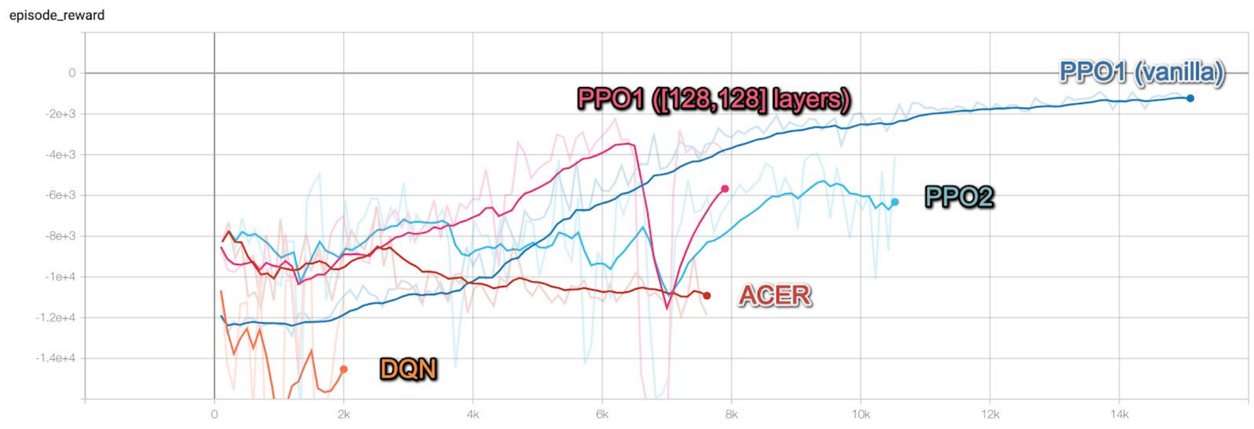
In order to further reward the agent when it gets close to the center, I decided to create a configurable “bull’s eye” zone around the center that would grant additional reward points if the LED is moved to that zone.

Algorithms

I browsed the algorithms available from Stable Baselines, and tried out most algorithms that fit my environment (Box observation space and Discrete action space); ACER, DQN, PPO1 & PPO2 (all vanilla). After disappointing early results, especially DQN which just stopped moving after 2000 timesteps, PPO1 seemed to show the most success.

I also attempted to train a BDPI agent by running an altered train.py script but after 10 minutes, my laptop was overheating and getting low on resources without noticeable improvement so I abandoned it.

Once I settled on the PPO1 model (having no decent GPU on my laptop, it seemed PPO2 wouldn't offer much improvement), I started tweaking the neural network of the MlpPolicy, and seemed to have a faster learning rate with a [128,128] layer.



Issues

LED Detection

Due to my webcam and the intensity of the LED, I had a hard time detecting it as a red object. I also had trouble getting it to ignore the Raspberry Pi power LED on the side which is also a bright, intense red, until I discovered there was a way to turn it off.

Learning

- To properly train a model, I required approximately 10000 timesteps (30-40 minutes) depending on the RL algorithm settings. I needed to keep the Sense LED board statically centered on the webcam in order to get the best results. If I moved the Raspberry Pi too much, or held too far away, the agent did not seem to be improving at all over time

- I was unable to train a model to correctly keep moving the LED after rotating the Raspberry Pi by more than 60°. I attempted to continue training the model in 90° rotation increments, but the result was a model that couldn't move the LED towards the center in any orientation.