Parrot AR Drone (source: Wikimedia Commons; CC BY-SA 4.0)

Project Overview

The **Quadcopter** or **Quadrotor Helicopter** is becoming an increasingly popular aircraft for both personal and professional use. Its maneuverability lends itself to many applications, from last-mile delivery to cinematography, from acrobatics to search-and-rescue.

Most quadcopters have 4 motors to provide thrust, although some other models with 6 or 8 motors are also sometimes referred to as quadcopters. Multiple points of thrust with the center of gravity in the middle improves stability and enables a variety of flying behaviors.

But it also comes at a price–the high complexity of controlling such an aircraft makes it almost impossible to manually control each individual motor's thrust. So, most commercial quadcopters try to simplify the flying controls by accepting a single thrust magnitude and yaw/pitch/roll controls, making it much more intuitive and fun.

The next step in this evolution is to enable quadcopters to autonomously achieve desired control behaviors such as takeoff and landing. You could design these controls with a classic approach (say, by implementing PID controllers). Or, you can use reinforcement learning to build agents that can learn these behaviors on their own. This is what you are going to do in this project!

Project Instructions

decide to work from the GitHub repository, make sure to edit the provided `requirements.txt` file to include a complete list of pip packages needed to run your project.

The concepts following the workspace are optional and provide useful suggestions and starter code, in case you would like some additional guidance to complete the project.

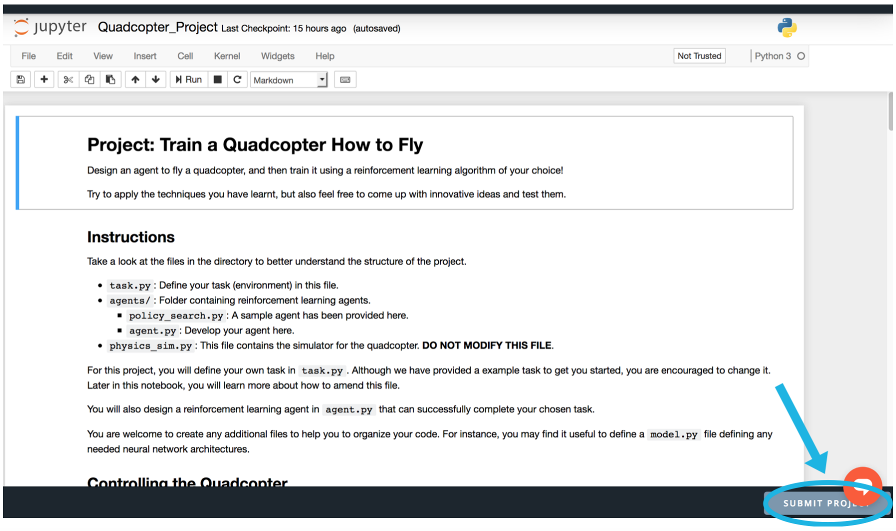
Evaluation

Your project will be reviewed by a Udacity reviewer against the project rubric. Review this rubric thoroughly, and self-evaluate your project before submission. All criteria found in the rubric must meet specifications for you to pass.

In this project, you will design your own reinforcement learning task and an agent to complete it. Note that getting a reinforcement learning agent to learn what you actually want it to learn can be hard, and very time consuming. For this project, we *strongly* encourage you to take the time to tweak your task and agent until your agent is able to demonstrate that it has learned your chosen task, but this is not necessary to complete the project. As long as you take the time to describe many attempts at specifying a reasonable reward function and a well-designed agent with well-informed hyperparameters, this is enough to pass the project.

Project Submission

If submitting from the workspace, simply click the "Submit Project" button at the bottom of the workspace window.



Project: Train a Quadcopter How to Fly

Design an agent to fly a quadcopter, and then train it using a reinforcement learning algorithm of your choice!

Try to apply the techniques you have learnt, but also feel free to come up with innovative ideas and test them.

Instructions

Take a look at the files in the directory to better understand the structure of the project.

- `task.py` : Define your task (environment) in this file.
- `agents/` : Folder containing reinforcement learning agents.
  - `policy_search.py` : A sample agent has been provided here.
  - `agent.py` : Develop your agent here.
- `physics_sim.py` : This file contains the simulator for the quadcopter. **DO NOT MODIFY THIS FILE.**

For this project, you will define your own task in `task.py` . Although we have provided a example task to get you started, you are encouraged to change it. Later in this notebook, you will learn more about how to amend this file.

You will also design a reinforcement learning agent in `agent.py` that can successfully complete your chosen task.

You are welcome to create any additional files to help you to organize your code. For instance, you may find it useful to define a `model.py` file defining any needed neural network architectures.

Controlling the Quadcopter

upload. This should include the **Quadcopter_Project.ipynb** file with fully functional code, all code cells executed and displaying output, and all questions answered.

NEXT