Robot Learning

Lab Exercise 4

Wednesday 8th February 2023

Edward Johns

Welcome to the fourth lab exercise for the Robot Learning course! In this lab, you will get to know the code that you will be using for Coursework 2, which will be released later today, after the lab.

There are the following five Python files: *robot-learning.py*, *robot.py*, *environment.py*, *graphics.py*, and *model.py*. And you will need to install the following packages: *pyglet, numpy, scikit-image, scipy,* and *perlin-noise*.

To begin, run *robot-learning.py* and observe what happens. You will see a lot going on, so take your time. First, you will notice the robot randomly moving around in the environment, as you have seen before. However, there are some new additions. (1) The background consists of a random map, which is randomised each time the program is run. (2) There are now buttons on the top-right to enable you to turn on visualisations if you so wish. (3) The visualised model appears to be changing very rapidly.

The model that is visualised, emulates the model that the robot has learned. In this lab and in Coursework 2, you will not actually be learning the model, because this can be more complex for complex environments such as these. Instead, the function *Robot.model.predict()* emulates making a prediction using the learned model, and also emulates uncertainty in this model. As the robot physically moves around the environment, this uncertainty decreases for those states and actions nearby the states and actions that the robot has visited. As such, physically exploring a certain part of the environment will result in the uncertainty decreasing in that part of the environment. You should be able to observe this as the visualised model begins to settle in the region around the initial state, where the robot has been exploring more.

Now take a look at *robot-learning.py*. There is a timer, which considers two types of timing information. First, is *cpu_time,* which is how long your code has taken to run so far, such as with your planning algorithm. Second, is *action_time*, which is what you can consider to be the time taken to execute all the physical steps. You can think of this as if each physical step takes one second. The *time_elapsed* combines *cpu_time* and *action_time* to compute the overall time that has been taken. Note that this means that physical steps take significantly "longer" than virtual steps computed during the planning, and therefore there is more of a penalty in taking physical steps than in simulating steps during planning. During the coursework, you will try to combine physical steps and planning steps in an optimal way, within a time budget.

For the remainder of the lab, try to become familiar with this code, and try to implement a planning algorithm which will enable the robot to reach the goal. You should not edit any files apart from the file *robot.py*.

End of exercise