COMP417: Assignment 2
Due Saturday, Feb 18 at 6pm

February 5, 2017

1 A* implementation (5pts)

Implement the A* algorithm for an omnidirectional robot on a 2D plane. You are given starter code that implements Dijkstra's algorithm in Python at the following repository: `https://github.com/florianshkurti/comp417.git`, under the directory `comp417/path_planning_and_control_assignment/python`. You need to modify the function `plan()` in the file `astar_planner.py`. You can run this file as follows:

```
cd path/to/comp417/path_planning_and_control_assignment/python/
./astar_planner.py ../worlds/map.pkl
```

What you need to submit: 3 images of paths produced by your planner. Use the same starting state that is currently provided, and 3 distinct destination states that are far from each other. Your images should be named `astar_result_[0|1|2]_firstname_lastname.png`

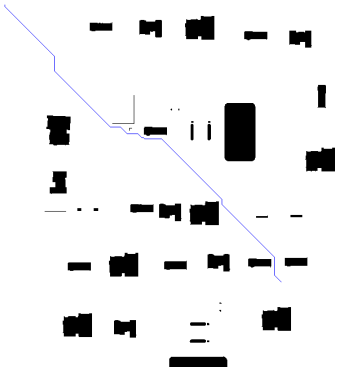

Figure 1: How the A^* planner will look like once you implement it and run it on the provided map.

2 RRT implementation (5pts)

Implement the RRT algorithm for an omnidirectional robot on a 2D plane. You are given starter code that implements some of the RRT functionality. You

need to modify multiple functions which are annotated with TODOs in the file `rrt_planner.py`. Note that this version of the RRT is the simplest version to implement in the sense that we are not requiring kd-tree-based nearest neighbor queries and complicated collision queries. We use occupancy grids to simplify collision detection. Once you are done implementing the required functionality you can run this file as follows:

```
cd path/to/comp417/path_planning_and_control_assignment/python/
./rrt_planner.py ../worlds/map.pkl
```

What you need to submit: 3 images of paths produced by your planner. Use the same starting state that is currently provided, and 3 distinct destination states that are far from each other. Your images should be named `rrt_result_[0|1|2]_firstname_lastname.png`

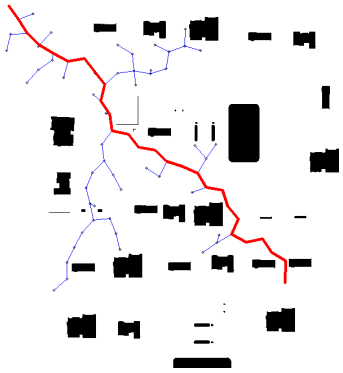

Figure 2: How the RRT planner will look like once you implement it and run it on the provided map.

3 LQR problem formulation (2.5pts)

Recall the example of the double integrator system with friction, i.e. the analogy of the curling stone that we saw in class:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

where α is the friction coefficient, $\mathbf{p}$ is the 2D (point) position vector of the stone, and $\mathbf{u}$ is the external control applied to the stone. You are given two curling stones of equal mass m. They start from different starting positions, with different starting velocities. You are tasked with finding an LQR controller/policy that receives feedback on the joint state of the two curling stones and outputs command vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ so that the two stones end up very gently hitting each other and not bouncing away from each other. In other words, define a joint linear system

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t \tag{1}$$

and an instantaneous cost function

$$g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T B \mathbf{u}_t \tag{2}$$

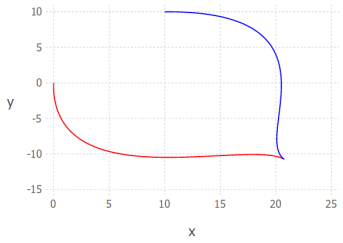

Figure 3: LQR control of two stones starting from different points with different velocities, such that they meet at the end of the provided time horizon.

such that when the state $\mathbf{x}_t$ stabilizes around $\mathbf{0}$ the stones are touching each other and their individual velocities are also stabilized around $\mathbf{0}$. Note that you are not asked to decide on the time horizon (number of time steps) for which you will need to run the controller to make the stones meet.

What you need to submit: a file called lqr.pdf with the definition of the state $\mathbf{x}$, the control $\mathbf{u}$, the matrices A, B, Q, and R. Justify that your choice of Q, and R are positive definite matrices. Include the steps you took to arrive at your formulation of these quantities. Do not write more than a page of justifications if you are typesetting. Feel free to use LaTex or scan your handwriting.

4 How to submit

Similarly to assignment 1, you will submit all your work in a file called `path_planning_and_control_assignment.zip` which will contain your extensions to the provided starter code, as well as the six images and the pdf file. Submissions will be done on myCourses.