CS8803: ACRL, Spring 2019: Homework 3

This assignment ay be completed in teams of up to 3 students.

Due: Monday, April 22nd, 3pm

1 The Problem

In this assignment you will be controlling a simple differential drive car in a world filled with walls and open space. In addition to fixed walls, there will be road blocks which may or may not be open. **Your task is to execute a series of commands that will transition the car across the map to the goal without colliding with either a wall or road block**. To accomplish this goal, you may use any strategy that we discussed in class including learning a policy or using a fast (re)planner.

Maps and Roadblocks There are three maps provided with the assignment on which you can evaluate your algorithm. Maps consist of gray scale pixels, each representing 1m in the world. Black pixels (values of 0) represent filled spaces where the car cannot go, white spaces (values of 1) represent open spaces, and gray pixels (values between 0 and 1) represent the probability that this space is open. These spots are considered road blocks. A value of .9 means that the the square will be open 90% of the time. During a run, all spaces will either be open or closed, but the car will not know that until it gets close enough to observe the road blocks. Until observing the space, you should consider it unknown. Included in the assignment are `map_1.mat`, `map_2.mat`, and `map_3.mat`, which contain MATLAB structs.

Within the matlab structures, the terminology "bridge" and "road block" should be understood to be interchangeable, as the problem was originally posed as having to cross "bridges." `map_struct.bridge_locations` contains the (x, y) locations of all of the road blocks in the map, and `map_struct.bridge_probabilities` contains the probability that those road blocks are open. `map_struct.start` and `map_struct.goal` contain the start and goal locations on the map, respectively.

The Car and Motion Model The motion model of the car will consist of a restricted differential drive base with only one control parameter. The state of the car will be continuous, defined as $X = [x, y, \theta]^\top$, where (x, y) is the location and θ is the heading. The border of the car will be modeled as a rectangle centered at (x, y), angled by θ. The physical car

parameters are width, length, wheel base, `left_radius`, and `right_radius`. These values are set in the file `load_sim_params.m`, and have been set as follows:

```
params.length = 3;
params.width = 2;
params.wb = 2;
params.l_radius = .25;
params.r_radius = .25;
```

At each time step, the wheels will rotate a nominal angle $\Delta\phi_{\text{nom}}$, which can be modified by a single input parameter, $u = -2 \cup [-1, 1]$. For $u = [-1, 1]$, we set the final rotation for each wheel to

$$\begin{aligned}\Delta\phi_{\text{right}} &= \Delta\phi_{\text{nom}} + u\Delta\phi_{\text{max.dev}} \\ \Delta\phi_{\text{left}} &= \Delta\phi_{\text{nom}} - u\Delta\phi_{\text{max.dev}}\end{aligned}$$

Values of u greater than 0 will cause the car to turn in an arc to the left, while values of u less than 0 will turn the car to the right. When $u = 0$, both wheels turn equally and the car goes straight. In the special case of $u = -2$, the car moves in reverse, we set

$$\Delta\phi_{\text{right}} = \Delta\phi_{\text{left}} = -\Delta\phi_{\text{rev}}.$$

From here, given the radius of each of the wheels (r_{right} and r_{left}), we can determine the distance each wheel travels across the ground

$$\begin{aligned}R &= r_{\text{right}}\Delta\phi_{\text{right}} \\ L &= r_{\text{left}}\Delta\phi_{\text{left}}.\end{aligned} \tag{1}$$

Finally, using a constant velocity differential drive model, we can calculate the motion of the car:

$$\begin{aligned}x' &= x + \frac{wb}{2}\frac{R+L}{R-L}\left(\sin\left(\frac{R-L}{wb} + \theta\right) - \sin\theta\right) \\ y' &= y - \frac{wb}{2}\frac{R+L}{R-L}\left(\cos\left(\frac{R-L}{wb} + \theta\right) - \cos\theta\right) \\ \theta' &= \theta + \frac{R-L}{wb}\end{aligned} \tag{2}$$

where wb is the wheel base in meters.

2 What to Turn in

The simulation environment provided with this assignment is written in MATLAB, but you are under no obligation to use MATLAB. Most of the provided scripts and functions

are simple, and can be ported to your programming language of choice (subject to TA approval). The only requirements are that the motion model and collision calculation remain unchanged, and you must implement the strategy yourself (you cannot using planning or RL libraries). The MATLAB scripts provided are commented, and should get you started in the right direction.

Turn in a zip file containing your code and a 4 page report describing the algorithm and methodology you used to solve the assignment. In the writeup, for each of the three maps, please include your average performance (across all samples), best performance, worst performance, and on which maps your robot collided with the wall, if any.