CS/ME 469: Machine Learning and Artificial Intelligence for Robotics

Assignment #1: Search and Navigation

Code A Due: Oct. 21st **Code B due:** Oct. 28th **Writeup (A+B) due:** Oct. 28th

The focus of this assignment is on the implementation of an A* variant, able to generate 2-D paths that navigate between start and goal locations while avoiding obstacles, and a simple controller that enables a wheeled mobile robot to drive the generated paths. Items marked with [*] have more weight in the final grade determination.

Environment

Obstacles: The landmarks from ds1 of homework #0.
Partial observability: The robot is only able to observe obstacles located within its 8 neighbor cells.
Connectivity: The 8 cells surrounding a given cell are its neighbors. (And theses are the only allowable cell transitions.)
True cost: +1 to move into a neighbor cell that is unoccupied, +1000 to move into a neighbor cell that is occupied.

Implementation

PART A

1. Build a grid with cell sizes of $1 x 1 m$, and a range of $x:[-2,5],\ y:[-6,6]$. Mark each cell that contains a landmark as occupied.

2. Implement the A* algorithm. You can assume that you have knowledge of which cells are occupied. Design an admissible heuristic, given the true cost function described above. Write up all relevant equations, and justify why your heuristic is admissible.

3. Plan paths between the following sets of start (S) and goal (G) positions. Provide a visual display of your results, that shows each grid cell, distinguishes occupied cells, and also shows the planned paths.

$$A:\ S=[\ 0.5, -1.5\ ],\ G=[\ 0.5,\ \ 1.5\ ]$$

$$B:\ S=[\ 4.5,\ \ 3.5\ ],\ G=[\ 4.5, -1.5\ ]$$

$$C:\ S=[-0.5,\ 5.5\ ],\ G=[\ 1.5, -3.5\ ]$$

4. Now modify your A* algorithm to be "online": that plans as the robot moves and does not have *a priori* knowledge of the obstacles (but rather, can observe them only when physically in a neighbor cell to the obstacle). What does this mean for the set of expanded nodes? Describe any modifications made to your A* algorithm from Step 2.

5. Plan paths between the start (S) and goal (G) positions from Step 3. Provide the same visualization.

6. Now let's work with a slightly more realistic grid. Decrease your cell sizes to $.1x.1m$ (keeping the same overall grid dimensions). A common way to (concisely) account for the physical size of a robot when path planning is to inflate the size of any detected obstacles by the radius (or diameter) of a virtual shape (e.g. a circle) able to encompass the entire robot footprint. Inflate the amount of space each landmark occupies by $.3m$ in all directions (with the final result of a square, or an approximation to a circle – either is acceptable).

7. [*] Plan paths between the following sets of start (S) and goal (G) positions. Provide a visual display of your results, that shows the occupied cells and the planned paths. (It is okay to not display demarcations for the unoccupied grid cells.)

$$A:\ S=[\ 2.45,-3.55\ ],\ G=[\ 0.95,-1.55\ ]$$

$$B:\ S=[\ 4.95,-0.05\ ],\ G=[\ 2.45,\ 0.25\ ]$$

$$C:\ S=[-0.55,\ 1.45\ ],\ G=[\ 1.95,\ 3.95\ ]$$

PART B

8. [*] Design an inverse kinematic controller able to drive a path generated by your online-A* implementation, based on the model you developed in Step 1 of Homework #0. That is, the function should output translational and rotational speeds $[v,\omega]$ that will allow the robot to achieve a target 2-D position $[x_T, y_T]$, given the robot's current 2-D position and heading $[x, y,\theta]$. (The target positions will come from the paths generated by your online-A* implementation.) Make this controller more realistic by considering also the robot's current speeds, and restricting maximum accelerations to the following values $\dot{v}=0.288\frac{m}{s^2},\ \dot{\omega}=5.579\frac{rad}{s^2}$ (observed from the ground truth datasets of homework #0). Use a sampling rate of $dt=0.1s$. Report the maths of your formulation, with explanations as needed.

9. [*] Drive the paths generated in Step 7. The robot should always begin with speeds $[v=0\frac{m}{s},\omega=0\frac{rad}{s}]$, and heading $\theta=-\pi/2$. When you execute the speeds output by your controller of Step 8, make things even more realistic by adding on some noise. You can assume that your robot always knows where it is. What you cannot assume, however, is that it gets exactly where you asked it to go. Provide a visual display of your results, that shows the occupied cells and the planned paths. (It is okay to not display demarcations for the unoccupied grid cells.) Display also the position and heading of your robot at each step of the execution. Discuss any challenges, inconsistent or surprising behavior.

10. Putting it all together: Plan the paths while driving them. That is, no longer use your paths from Step 7 (which assumed perfect world dynamics). Instead, at each step of your online planning, execute the next chosen step using your inverse kinematic controller. Provide a visual display of your results, that shows the occupied cells, planned paths, robot position, and robot heading. Discuss any challenges, inconsistent or surprising behavior.

11. Drive while planning for the start/goal positions of Step 3, using your finer grid. Repeat this procedure, now using the coarse grid (of Step 3). Provide a visual display of your results, that shows both planned paths, as well as the position and heading of your robot at each step of the execution (for each path). Show also each grid cell and distinguish occupied cells (for this, choose a single grid resolution; either is fine). Discuss any behavior

differences you see between operating on a coarser versus finer grid decomposition. What are advantages and disadvantages of each?

12. What additional factors would confound driving these paths if our robot was operating within the real world? (What sort of assumptions and simplifications does our simulated world and controller make?)

Code

To be submitted with the report. Development in Matlab, Octave or Python is a requirement. A single executable script file named “run.m” or “run.py” should output, when executed, all of the data (plots, table values) reported in the write-up (as instructed above), with clear presentation and labels. *Code that does not run will receive zero points.* The run file should be easy to read – a sequence of calls to standalone functions, that hide away the meat of code. Clear commenting for all functions and function calls is encouraged. (No need for excessive verbosity, but the quicker and easier we are able to read the code, the better for your grade...)

Write-up

For each step above, explain thoroughly and with clean consistent maths all design decisions (heuristic function, inverse kinematic controller, how parameters were set...) and reported assessment results. Remember to label all plot axes, title all plots, provide captions for any figures, include (as appropriate) units for all reported numbers, etc. Language counts – specifically with respect to clarity. If you use other sources be sure to cite them.