MAE 4180/5180, ECE 4772/5772, CS 3758

Autonomous Mobile Robots

Homework 5 - due Wed. April 7 by 11:59PM
(available until Sun. April 11 by 11:59PM)

A document containing the answers and figures should be uploaded to Gradescope as `HW5writeup.pdf`. All the code and the files needed to run the code are to be uploaded to Canvas as a zip file named `HW5code.zip`. Note, the code zip file should contain ONLY .m files, config files and files containing data (if needed) and **not any other file** (writeup, assignment, simulator, toolboxes etc). Specific functions, as indicated below, should be uploaded to their corresponding assignments on **Canvas**.
Notes about autograded assignments:

- We **highly** recommend you develop and debug the autograded functions in Matlab. The error messages that the autograder provides are not as explicit as in Matlab
- Before submitting, we **highly** recommend running your own test case with "Run Function" as it can reveal syntax errors, missing functions, etc. when transitioning to Canvas.
- You may submit as many times as you like until the deadline
- Make sure to also include the functions in the zip file you upload to Canvas
- **Reusing code:** We encourage you to reuse your prior functions in the new functions. For the autograded assignments, if you want to use a previous function you have written (for example robot2global), simply copy-paste it below the function you are writing

Introduction and Setup

In the Simulator, load `loopMap` and run a control program that will drive the robot into walls (you may use the function `backupBump.m` or write a new function). Make sure you are collecting the following sensor data in `dataStore`: truthPose, bump and depth. Allow the robot to move around the environment and bump into most (if not all) walls. Access the sensor data in the MATLAB workspace by typing `global dataStore`, and make sure the data was collected properly. You will use this data in the two mapping sections. Remember that in the simulator, the robot radius is .16, and the sensor origin is [.13 0].

As you have seen, each bump sensor of the Create returns a value of 1 if the sensor is triggered, and 0 otherwise (with no false alarms). Initialize a 25×25 occupancy grid of the environment with $p_0(\text{occ}) = 0.5$ as the prior occupancy probability for all cells (i.e. $\ell_0 = 0$). The boundaries of the environment in the x and y directions are $[-2.5, 2.5]$. You may find the MATLAB function `meshgrid.m` helpful.

Mapping - Occupancy Grid with Bump Sensor (45 points)

1. Plot the occupancy grid boundaries and the robot's trajectory. Indicate with a marker (e.g. star, triangle) the locations where a bump sensor was triggered.
2. Write a function `logOddsBump.m` to calculate the log-odds $\ell_t(\text{occ})$ of cells in the grid using only bump sensor measurements. Make sure you allow non-square grids ($n \times m$, where n is the number of cells along the X dimension and m the number of cells along the Y dimension). What are the inputs? what are the outputs?
3. Explain the sensor model you used, assumptions you made in `logOddsBump.m`, and why they make sense.

4. Write a function `plotOccupancyGrid.m` that given the log odds and the grid, plots unoccupied cells in white, occupied cells in black, and uncertain cells in shades of gray. You may use `colormap(flipud(gray))`, `pcolor` or `image` to help plot the grid cells. Plot the occupancy grid at three different time steps using the log odds obtained from `datastore.truthPose` and `datastore.bump`.
5. If you had to choose for each cell whether it is occupied or free (for example, to use the map in a planning algorithm), what would you do? Apply this method to the grid at the last timestep and plot the result (each cell is assigned either 0 or 1).
6. Repeat (4,5) with a 50×50 grid. What do you observe?
7. What differences might you expect to observe between the map generated using simulated bump sensors and one using real bump sensor data? Why?

Mapping - Occupancy Grid with Depth Information (45 points)

1. Write a function `logOddsDepth.m` to calculate the log-odds $\ell_t(\text{occ})$ of cells in the grid using only depth sensor measurements. What are the inputs? what are the outputs?
2. Explain the sensor model you used, assumptions you made in `logOddsDepth.m`, and why they make sense.
3. Plot the occupancy grid at three different timesteps using the log odds obtained from `datastore.truthPose` and `datastore.rsdepth`.
4. Repeat (2,3) with a 50×50 grid. What do you observe?
5. What are the pros and cons of using a finer grid resolution?
6. Compare the maps generated using depth with those using the bump sensor. Is one map "better" than the other? Why or why not?
7. What differences might you expect to observe between the map generated using simulated depth sensors and one using real depth sensor data? Why?

Test function (10 points)

Edit the function `TestOccupancyGrid.m` that takes as input,

- dataStore
- ℓ_0 (scalar value for the prior occupancy probability of grid cells)
- NumCellsX (number of cells in the X axis)
- NumCellsY (number of cells in the Y axis)
- boundaryX (1x2 vector representing the interval of possible values for X, for example [-2.5 2.5])
- boundaryY (1x2 vector representing the interval of possible values for Y, for example [-2.5 2.5])

outputs,

- lFinalBump (final occupancy grid using bump sensor)
- lFinalDepth (final occupancy grid using depth sensor)

and plots final occupancy grids, one for the bump (as in part 4 of the mapping with bump sensor section) and one for the depth information (as in part 3 of the mapping with depth information section). Plots generated should have meaningful titles and labels.

Submit this function in the autograded assignment `Homework 5 TestOccupancyGrid.m` on Canvas. Course staff will use this function to test your code with different inputs, and we are using the autograder as a way to ensure that the test function runs. We are NOT testing for correctness of the algorithm. Make sure the zip file you submit on Canvas contains this function and all of the functions necessary to run it.