Robotics Localisation Lab Information Sheet

October 21, 2014

This worksheet assumes you have a basic understanding of Matlab and object oriented programming. If you have questions about how BotSimLib works (not general matlab questions), please post them on blackboard.

0.1 BotSimLib

This is a library designed to make your life easier. It includes a simulation of a wheeled robot inside an arena that can perform simulated ultrasound readings of the surrounding walls. It also includes an automarker.

The following will explain how to configure and use the robot movement and ultrasound scanning and how to use the automarker. A more detailed explanation is included in the code files as comments.

1 Example 1 - Movement

The library works by having a robot object that accepts move, turn and scan commands. When you are debugging your program you can also get and set the global position and angle of robot but while marking these features are disabled. During marking you can only give the robot instruction for moving and turning relative to its current position. You are not able to access the robots current position or angle during marking. If you attempt to access this information in your program while it is getting marked, the program will crash and you will get 0 marks. This will all be explained in more detail in example 5 and 6. Example 1 shows you how to move and turn the robot

1. Log in
2. Open Matlab. Press start button > Type "Matlab"
3. In matlab select File > Open > Navigate to the BotSimLib folder, open the file EXAMPLE1movement.m
4. Run the file by pressing F5
5. If you are asked if you want to change folder click "Change Folder".
6. Follow Onscreen instructions.
7. Read the file.

2 Example 2 - Scanning

The robot can perform simulated ultrasound scans using the ultraScan() function. There are several configuration options you can set which are explained in this example.

1. Open the file EXAMPLE2scanning.m and run it.
2. Read the file.

3 Example 3 - Inside Map

This example shows you how to test if your bot is inside the map and generate a matrix of movable space.

1. Open the file EXAMPLE3insideMap.m and run it.
2. Read the file.

4 Example 4 - Particles

This example shows you how to set up and move particles. This example uses the BotSim class for its particles. You are not allowed to modify the BotSim class as it will interfere with automarking. If you want your particles to have extra variables or functions, you can make a copy of the BotSim class and modify that, leaving BotSim intact. For example, copy BotSim.m and rename it particle.m, then modify particle.m to suit your needs. This is not necessary however to complete the assignment.

1. Open the file EXAMPLE4particles.m and run it.
2. Read the file.

5 Example 5 - How to run localise.m

This example is very similar to the previous one, but shows you how to format your code for marking. IMPORTANT: Your assignment is to complete the file localise.m. An example version has been provided for you to point you in the right direction. In this example EXAMPLE5localise.m calls localise.m. This is the reccomended way of developing your algorithm. Modify the localise.m function, and call it by running EXAMPLE5localise.m.

While localise.m is running, it plots information and prints debug messages. The printing and plotting are inside if statements, so that they only plot the information during the debug process. This is because plotting is very slow and would increase the time it takes for your algorithm to run during marking. You can check if you are in debug mode by using the function botSim.debug().

1. Open the file localise.m and read it.
2. Open the file EXAMPLE5localise.m and run it.
3. Read the file.

In the next example localise.m will be run through the automarker. Here, the debug information will not print so the file will run much faster.

6 Example 6 - Automarking

Everyone's localization simulator will be marked automatically with a Matlab program. This is to ensure that the work is marked fairly, consistently and in greater depth than would be possibly by hand.

Your assignment is to write a localisation function localise.m that accepts a botSim object, map and target. The function returns the same botSim object that was passed to it so that it can be analyzed during marking. You have been provided with a template of this function, which you should complete. The botSim object will be analysed by looking at:

- The total time taken to reach the target point

- The final distance from the target point
- The length of the path taken from the start position to the target point
- Percentage of the time that the robot hits a wall or leaves arena

Marking algorithm Each student will get a vector score based on the above criteria. E.g.

Student	Time	Distance	Path Length	Collisions
1	10.1	1.5	201.7	10.7%
2	23.4	2.8	127.2	8.3%
3	5.9	0.4	79.3	4.2%
⋮	⋮	⋮	⋮	⋮

During marking, your algorithm will be tested on a variety of maps and parameters that will test various aspects of the algorithm. These include but are not limited to:

- Large maps
- Maps with repeated geometry
- Long winding maps (for testing path planning)
- Maps with non perpendicular walls.
- various levels of movement and turning noise
- various levels of sensor noise

The map edge will always be one continuous loop, with no islands in the middle.

Do not worry if your algorithm is not able to do all of these things perfectly, all you have to do is perform better than what other people can do.

1. Open the file EXAMPLE6templateMarking.m and run it.
2. Read the file.

7 VERY IMPORTANT - SUBMISSION PROCEDURE

You need to submit a zipped file with localization.m and any other functions you have written. Do not include BotSim.m or any other functions included in the library.

The file will be marked with the following method:

1. Extract your zip into a folder.
2. Add the files from an unmodified version of the library. This is to ensure you cannot modify any of the existing functions. If you have (incorrectly) included library files, they will be overwritten.
3. Run EXAMPLE6templateMarking.m

You should test that your code runs after going through this procedure. If it does not run, you will get 0 marks.

8 FAQ

1. Q: Can I use feature X on Matlab?

   (a) Yes, as long as the automarking function works when run on a machine in 1.07 or 1.08.

2. Q: Can I use mex files?

   (a) See question 1.

3. Q: Can I use multiple cores?

   (a) See question 1.

4. It's too slow.

   (a) Welcome to Matlab, also the library has not been written with performance as the primary objective. Unfortunatly, there are several cases I chose ease of use over performance. If you have a suggestion for how to improve performance without reducing usability please let me know and I will change it.