CS/ME 301: Introductory Robotics Laboratory

Assignment #2: Navigation: Localization, Planning, and Mapping

Code, Part A Due: February 10th (3:29pm) **Demo, Part A Due:** February 12th (12:30pm)
Code, Part B Due: February 19th (3:29pm) **Demo, Part B Due:** February 24th (12:30pm)
Report Due: Mar. 1st (11:59pm)

This assignment will focus on robot navigation: mapping, localization and planning. At the end of this assignment, your robot will be able to keep track of its location within a grid-based map while walking (*Localization*); build this map by observing walls with its sensors (*Mapping*); generate a path from any arbitrary starting cell to any arbitrary ending cell on this map, that passes through only open space *(Planning*); and traverse the generated path.

Read through the entire assignment sheet before getting started.

Start your lab report early! This is a much larger assignment than the previous one, and *a lot* of it can be written up before the Demo Days are complete.

Write code outside of lab hours! This assignment has much more logic to code up than the previous assignments; especially in Steps IV and V. Testing your logic in standalone python programs will allow you to develop code outside of lab hours.

Pay attention to the target completion dates! Be sure to move on to the next step according to this schedule; even if your code and behaviors are not working as well as you would like. It is better to have something working poorly for all of the steps of this assignment than to have one step working great and another step not working at all.

Submitted code must run without edits on demo days! We will be stricter about this than on the previous assignment. On demo days, all code must run out-of-the-box, with zero file edits. Any parameters or function selections must be either (1) passed in at the commandline, or (2) interactively prompted as the code runs. Do *not* submit code that has not been tested on the robot hardware, and make sure your submitted code is on the robot when you leave class on the code submission deadline days.

Instructions – Part A

I. Getting started: Software

Download the map.py file from Canvas.

First: **Read all of the comments in the map.py code**, so that you know which functions will help you complete the rest of the lab.

Then: Create a asn2_grpX.py file, import map.py, and add a line of code to print the obstacle map (i.e., the walls). It should print out a map that is hard-coded into the file, which will be used in the first parts (II-III) of the lab. In the final part (V), your robot will build a new map, from observations it makes of walls in the world.

A position on the map is given by ***x(i,j,k)***. The ***i*** index runs north-south, the ***j*** index runs east-west and the ***k*** index is the heading. The position of the map origin ***x(0,0,1)*** is the upper left-hand corner facing North. The enum defining heading directions is defined at the top of the map.py file.

The size of a cell in the map is the same as the size of a 2x2 square of 4 tiles on the floor.

Target completion: Middle of class, Tuesday February 3rd.

II. Localization

In this lab, your robot will need to be able to traverse a map represented as a grid. The simplest way to do so is to create discrete commands that have your robot walk one cell in each of the {North, South, East, West} directions. Note that one of these given commands might be a composition of the walking and turning gaits developed in the Assignment 1 lab.

To help keep your robot oriented, try to end up in the center of the subsequent cell, with a heading that is aligned with an imaginary line stretching from the center of the starting cell and center of the subsequent cell. The robot should be keeping track of which cell it is in. Practice executing arbitrary sequences of commands: see where the robot actually ends up, and also where it "thinks" it is.

Your code should be able to take target start and goal positions from the commandline. (Either interactively, while the code is running, or when it is launched.) Don't forget to include heading in your positions.

Assessment: Test your robot's ability to walk 3 pairs of start/goal positions: ***x(0,0,3) → x(3,0,1)*** and ***x(0,0,3) → x(0,3,2)*** and ***x(0,0,3) → x(2,2,4)***. Start the robot in the middle of the starting cell, facing "South". (You can do this on any appropriately-sized set of floor tiles; no need to be using a physical map just yet.) After it is done moving, measure the error (distance *and* orientation) from the center of the final square.

Repeat as needed for good results. (Expectation: 3 trials of each.)

Going further: Instead of needing to stop at each single cell traversal, try to make a smooth motion that takes advantage of groups of "walk forward" commands.

Target completion: Middle of class Thursday, February 5th.

III. Planning

Next, your robot will need to be able to generate and follow plans, that specify a path from any given starting position $\boldsymbol{x}_s$ to any ending position $\boldsymbol{x}_g$. The big pieces needed to achieve this are the following:

1. Be able to accept a starting position $\boldsymbol{x}_s$ and an ending position $\boldsymbol{x}_g$ from the command line. (Either when launching the program, or interactively while it is running. Don't forget to include heading in your start/goal positions!)

2. Set the cost of each grid cell. This value should depend on the presence of walls, and on the locations of $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$. To set these costs, you are free to implement any path planning algorithm and cost metric of your choosing.

3. Generate a path from $\boldsymbol{x}_s$ to $\boldsymbol{x}_g$. This should take into consideration the cell costs, and the result should look like a sequence of grid cells to visit.

4. Generate a command sequence to achieve your path. This will call the commands developed in Step II. Make sure this sequence prints to the terminal on demo day.

5. Walk the path. (By following the command sequence.)

Assessment: Generate and execute plans between multiple sets of $\{\boldsymbol{x}_s,\boldsymbol{x}_g\}$ on the maze set up in the lab. Choose pairs of $\{\boldsymbol{x}_s,\boldsymbol{x}_g\}$ that challenge your robot, in addition to those that demonstrate good behavior, and/or that highlight interesting aspects (good or bad) of your robot's performance. Collect at least 2 repetitions of each pair. Measure things like:

(i) execution time
(ii) error in the final position/heading (or in the case of failure to reach the goal position: the final grid cell achieved and reason for the failure)
(iii) anything else that seems important or of interest

Repeat as needed for good results. (Expectation: At least 6 trials in total.)

Target completion: End of class Tuesday, February 10th.

Demo Day: Thursday, February 12th

Instructions – Part B

IV. Mapping

The final step in this assignment is for your robot to *build* a map, by exploring the world. You can assume that (i) the robot always starts at the origin, i.e. $\boldsymbol{x_s = x(0,0,1)}$, and that (ii) the world is an 8x8 square. The big pieces needed to achieve this the following:

1. A wandering algorithm. When the algorithm is run, the robot should explore (move in) the world in a way so that (i) it explores unknown regions and (ii) never hits any walls. It should keep track of which cell it is currently in while wandering.

2. A wall recording mechanism. In step III of this lab, you were querying the map for information about whether there is a wall between neighboring cells. There are functions in the map.py file to instead *set* this information, which is how the new map should be stored. Walls should be detected using the sensor data from your robot.

Your code should print to terminal the map that it builds after and during exploration.

Assessment: Build a map of the laid out in the lab. Measure things like:
(i) execution time
(ii) the number of times a wall is touched/hit
(iii) the portion of the environment explored (i.e., fraction of cells visited)
(iv) anything else that seems important or of interest

Repeat as needed for good results. (Expectation: At least 4 trials.) Make sure to report the maps your robot builds in your writeup.

Going further: Combine your planning and mapping code, and plan within and then traverse a map built as a result of exploration. Gather data that assesses your performance and report these results.

Target completion: End of class Thursday, February 19th.

Demo Day: Tuesday, February 24th

Demo Days

Demo Day will involve a *maze traversal competition* between groups, and also a *maze exploration*. Assessments will consist of races to first reach goal positions (metrics: time, accuracy), and the exploration of unknown mazes (metrics: coverage, time).

Your planning code should be able to accept start/goal positions interactively.

Your mapping code should print out the map that it builds after exploration.

Code

It is okay to submit one asn2.py file with your code for both Parts A and B, or to separate them into two files. If the former, just make sure it is possible to call either planning or mapping from the commandline, by passing arguments either (1) when launching the code or (2) interactively while the code is running.

Report

Each group member is to submit their **own** report, in PDF format. While it is expected that the raw data for each student will match that of their partner, the presented figures, tables and analyses must be performed by each student individually, and be their own work. (See the Syllabus for more about plagiarism/cheating.)

For this assignment only, the page limit is increased to 7 pages.

In addition to the guidelines set out in the Lab Report Guide, the following questions/considerations should be addressed in the Assignment #2 lab report:

I. INTRODUCTION:
When introducing the topics of this lab, include a discussion of some alternative (more complicated) ways to perform Mapping, Localization, and Planning (many were mentioned in the book and in lecture). Present their advantages and disadvantages.

II. METHODS
Make sure to provide a thorough description of your implementation of each of Steps II-IV, giving special attention to your algorithms from Steps III and IV.
Make sure to talk about any assumptions being made, and any simplifications.

III. RESULTS:
Be sure to describe any relevant initial conditions, measurement mechanisms, failures, etc. (How was distance measured? Angles?)
Be creative in your reporting mechanisms; e.g. using plots when applicable, rather than just data tables.
Report your demo day performance.
Report your assessment data from each of Steps II-IV.

IV. DISCUSSION:
Identify any reasons for poor performance, challenges, sources of uncertainty, areas for improvement, any surprising results, and anything else you think merits discussion. Make sure to discuss each of Steps II-IV.