

CSE 397/498-013
Introduction to Mobile Robotics

Homework: Monte Carlo Localization

Report Due Date: Tuesday, 22 Nov 05 submitted via Blackboard
PRIOR TO THE START OF CLASS (NO ANALOG SUBMISSIONS)

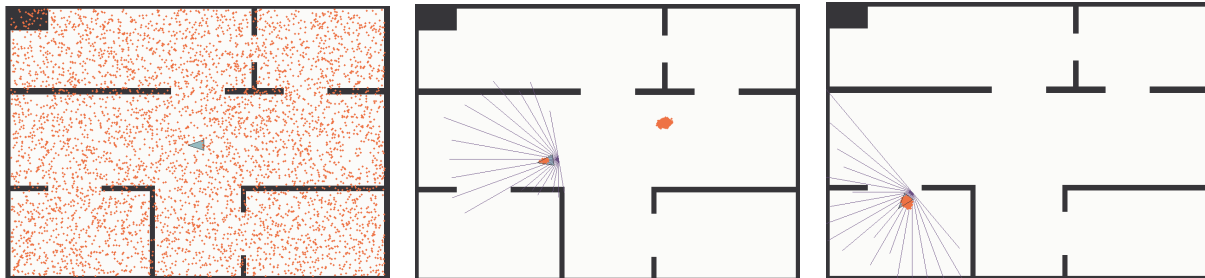


Figure 1: Snapshots from an MCL Simulation

A. **Objectives:** The goal of this assignment is to implement the MCL algorithm using a particle filter as discussed during class.

B. **Robot Model:**

1. **Motion Model:** Again, we are using a differential drive kinematic model. However, we will assume that there is uncertainty in the robot motion model.

- a) Assume that both the left and right wheels velocities are subjected to independent random noise. Values of 10-20% of the actual velocity for the standard deviation are reasonable. NOTE: Since the wheel will have separate noise inputs this will also result in errors accumulating in orientation.
- b) Assume a constant linear robot velocity of 5 m/s
- c) Assume the update rate for the odometry is 10Hz.

2. **Sensor Model:**

- a) Assume that your robot is fitted with a 360-degree ring of sonars spaced 20 degrees apart.
- b) The sonar update rate is 1Hz.
- c) Assume that the sensors have a maximum range of 750 cm.
- d) Assume that the sensitivity of the sonars is 10cm.
- e) Using this information, you will need to generate a probability density function for *each possible sensor range*.

1. Your PDF should account for the fact that a given percentage of measurements will exceed the maximum range.
2. You may assume as simplified obstacle model so that every range is modeled with an initial (non-zero) probability.
3. You do NOT have to account for changes in the environment.
4. Your PDF should be represented as a 75x75 element look-up table where each entry would correspond to a probability of that range being measured given the actual range. NOTE: It may be more efficient to implement the LUT as the probability distribution rather than the PDF.
5. A sample PDF from the class notes is shown below.

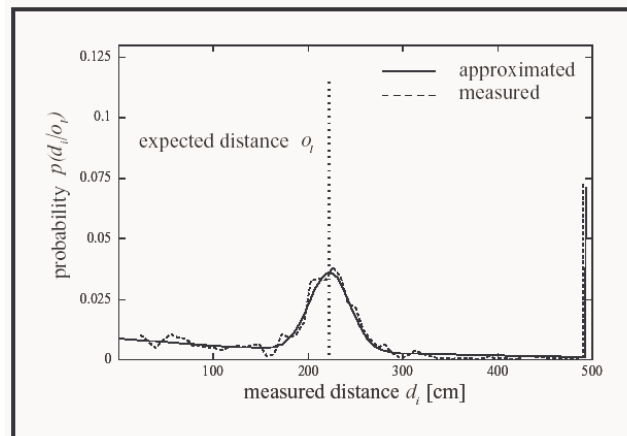


Figure 2: Sample PDF for a given sonar range.

- f) Modify your FireSonar function so that when it receives an array of ranges, it will return a corresponding array of corrupted ranges as sampled from the PDFs.

C. Environment Model:

1. The function which generates the a priori map is provided on the web site.
2. Assume a continuous environment representation.
3. The resolution of the map is such that each pixel corresponds to 10cm.

D. Requirements:

1. You MUST use Matlab to complete this assignment.
2. This is an individual assignment. Each student is required to submit his/her own work in order to receive credit.
3. As stated on the web page, if you turn this assignment in late without coordinating with me first you will receive a 0.

E. The Assignment:

1. Using a particle filter, you are to implement the MCL algorithm discussed in class.

- a. Refer to the second version of the algorithm listed to implement this efficiently in Matlab. You should also employ vector operations whenever possible to accelerate runtime operations.
- b. Your main function `hwk(numParticles)` will take as an argument the number of particles that will be used to estimate the PDF for the robot's pose.
- c. Each particle corresponds to a hypothetical robot position and orientation.
- d. Initially, the particles will be distributed at random poses across the map. You can plot the particles very quickly using

```
pHandle = plot(particles(:,1),particles(:,2),'r.') ;
```

This assumes that `particles` is an array with the `x` and `y` coordinates in its first two columns. DO NOT TRY TO PLOT INDIVIDUALLY OR USE THE `MAKEROBOT` COMMAND FOR THE PARTICLES.

- e. The initial robot position will be taken as mouse input from the user. You can generate the initial robot orientation randomly.
- f. You will need to write a motion planner function

```
robot = MotionPlanner(robot, sonars, map)
```

that provides a means by which the robot can negotiate the `map`.

- i. This can be a simple reactive controller that always heads in the direction of greatest open range given by `sonars`, but you must handle the case for when the robot is about to collide with a wall.
 - ii. It should change direction in this case so that it always stays within the bounds of the map. While you may allow particles to pass through walls the robot **MUST NOT!**
 - iii. The update rate of `MotionPlanner` should be 1Hz.
- g. You will also need to implement a raytracing function

```
ranges = RayTrace(robot, map)
```

Given the `map` and the `robot` as arguments, this will return an array of range values for perfect sonar measurements.

MCL In Action: Once you have implemented all of the above functionality, MCL is performed as follows:

- h. Move the robot taking into account uncertainty in the robot's motion.
- i. Move each particle identically to the robot WITH RESPECT ITS OWN coordinate frame, PLUS THE ADDITION OF RANDOM NOISE as introduced from the motion model.
- j. Use your `RayTrace` function to generate an array of sonar measurements.
- k. Pass this array to `FireSonar` to generate the corrupted measurements.
- l. Weight each particle with respect to these corrupted measurements using the PDFs you previously generated.
- m. Resample the particle set based upon the weights.
- n. Plot the particles on the map.
- o. Generate a motion plan for the robot
- p. Repeat.

F. Turn in: A write-up, to include images from all simulation trials, as well as your Matlab source code. You are to also to make arrangements to meet with me IN PERSON for 15 minutes to present and discuss your homework.