Particle Filter Assignment

Rocco Diverdi and Alexander Crease

What was the goal of your project?

Our goal was to use the Neato's lidar and odometry to find and keep track of the robot's position within a previously mapped region and given a reasonable guess.

How did you solve the problem?

We used a particle filter with a reasonable initial guess to give an approximate location and orientation of the robot. The initial guess initiated a gaussian distribution of particles with some standard deviations to spread out the particles but localize them primarily around the guess. When the robot moves, the particles move according to the motion indicated by the robot's odometry with some additional random motion. This was determined by another gaussian distribution operating independently on the X,Y, and theta values.

Given the distance readings the lidar was outputting at certain angles around the robot, each particle took the theoretical obstacle would be if it were the "correct" guess and compared it to the map obstacle closest to that point. This difference was calculated, and particle's weights were determined using an exponential decay function based on the distance error. For each update, we took a fraction of the particles (sampled according to their weights with the given function), and for each of those selected particles, put a small distribution of new particles around those guesses to fill up the new particle array to be the same size as the old one.

Describe a design decision you had to make:

We made a bunch of design decisions relating to where and when to add randomness, and how much to add. We found that the variance for the laser scan had to be much higher than the deviations we saw when watching the laser scan reading because a higher variance in the laser scan allowed close, but slightly off guesses to get higher weights. We also realized that there was a balance between randomness added when updating from odometry and when resampling particles. For initial testing, we eliminated all variance when resampling particles, but this led to clumps of guesses which weren't getting closer to the robot. Adding a small amount of variance into resampling the particles led to much nicer clumps of particles which migrated towards the robot more consistently.

What if any challenges did you face along the way?

One of the biggest challenging we faced was figuring out how to best debug our system given all of the randomness we had introduced, and what parameters we should increase or decrease based on each test run. For example, it was very hard to tell if increasing the number of laser readings we were factoring in to our obstacle detection code was benefiting our system or

hurting it. The randomness made it hard to figure out whether or not our function had a bug, or if our standard deviations were just too large. For each change, we had to go through and test each function with no standard deviation and then slowly increase the deviations in certain sections of the code to see how much of an affect they had. We also applied a similar incremental testing to the number of particles and the number of directions to look for obstacles.

What would you do to improve the project if you had more time?

We would explore different methods of inserting randomness to try and more closely follow the strengths and weaknesses of both odom and the laser scanner. We would also like to find ways to filter out obstacles which would not be visible from the robot's location as well as particles which were driving through obstacles.

One interesting direction to go in would be to determine the robot's location not based on estimations and obstacles the way we did it, but based on the geometries it sees and how they match with geometries on the map. So if the robot recognized a corner, it would limit its guesses to just corners of that size and shape. This would help get much better guesses on the orientation of the robot and eliminate a lot of the noise.

Did you learn any interesting lessons?

Randomness sucks, so get rid of as much of it as you can for as long as you can, and only add it in once you are confident in your system. Checking the behavior and setting discrete waypoints to show functionality also helps with debugging. Also, particle filters are hard but fun!