Kyle Vedder

CS-603 Particle Filter

kvedder@umass.edu

March 17, 2017

Background

The goal of this assignment was to implement a particle filter to do localization of a robot, given a map as well as real odometry and laser scan data. The challenges faced are that neither the laser scan nor the odometry are noise free, and thus treating them as the ground truth will lead to incorrect localization.

For this specific implementation, odometry is run at 20hz and laser scans are run at 10hz. In addition, odometry data is particularly noisey, meaning that the distribution of possible locations after an odometry move is fairly large, and that without the aid of laser data to facilitate convergence of possible locations, the uncertainty of positions can grow to be rather large.

Motion Model

The motion model captures the uncertainty in the movement of the robot. For example, if the robot reports moving forward $0.1m$, there is a chance that it only move $0.99m$ or it actually went $0.11m$, and so this model captures that fact. Thus, when we are forward projecting particles, we randomly sample from this model.

For my model, I knew that the robot that captured the data was holonomic, and thus added noise to the x, y, and $theta$ positions as a function of the amount changed. Specifically, I used the following functions. Note that all units are in meters.

$$\begin{aligned} proj(x) &= 0.35x + 0.005 \\ proj(y) &= 0.35y + 0.005 \\ proj(\theta) &= 0.1\theta + 0.01 \end{aligned}$$

Note that this will cause the particles to move even if the odometry reports no movement. I initially tried the motion model without these terms and I had little success.

Laser Beam Model

The laser beam model captures the likelihood of an single laser beam's view given a robot's location and the map of where the robot's environment. The goal of this beam model is to be able to score particles according to their likelihood; that is, if none of the beams actually viewed match the particle's supposed beams very closely, then it's not very likely that the particle is the true location of the robot. To encode that, we try to find the likelihood of a single laser beam given that we already know our distance from the obstacle.

For my laser beam model I tried to capture a few different pieces of information. We know that the laser sensor has a very low variance, so that can be captured as a gaussian distribution with a mean around the actual distance from the wall. In addition, we know that things might get in front of the robot and block the beams from the wall, so I modeled this with an exponential decay. We also know that due to too long of distances or errors in the sensor, it may read as max length, and thus I added a uniform distribution at max range of the sensor. Finally, to account for unmodeled noise, I added a uniform distribution over the entire output range of the sensor.

Ultimately, to score the likelihood of these, we took the probability density of the composition of these four functions, which looks like as follows. Note all units are in meters.

$$\begin{aligned} likelihood(\text{reading}|\text{distance from wall}) = {} & sampleExponential(\text{reading}, 0.01) \\ & + 4 \cdot sampleGaussian(\text{reading}, \text{distance from wall}, 0.01) \\ & + 0.1 \cdot sampleUniform(\text{reading}, 0, RAYMAX) \\ & + 0.02 \cdot sampleUniform(\text{reading}, RAYMAX - 0.01, RAYMAX) \end{aligned}$$

Note that this highly values the gaussian far above all other functions, and thus can cause overconfidence in readings. I believe that this is potentially the problem with my final product, and that if I were to lower these values and adjust the motion model accordingly, I'd get better results.

Resampling Procedure

To score each of the 100 particles, I subsampled the beam to $\frac{1}{5}$th of the initial beams and took the geometric mean of the likelihood of the samples. This gave me non-normalized weights, which I then used to do naive resampling. The choice of naive resampling over low variance resampling was for engineering simplicity rather than for better algorithm performance; the obvious implication of this is that my particle filter could possibly fail the dual room test.

To then get a final position, I took an average of all of these freshly resampled particles.

Tuning

By far the most unscientific of the work I did, tuning this system boiled down applying what I knew about the data in order to point me in the correct general direction, and then using trial and error to get better results.

For example, I knew that the odometry had quite a bit of error in it, as was evident from my attempts to treat it as ground truth. Ultimately, I realized odometry should serve as more of a general guideline rather than anything concrete, hence the large amount of noise, both in the stationary and moving aspects. I also found that that the angular data was quite noisy, often causing the system to careen into walls if taken too literally, so I spent a good amount of time tuning that as well.

The sensor model I knew should reflect the fact that the laser scanner has very little noise in actual readings, which is somewhere on the order of a half millimeter. As such, I highly valued the readings from the gaussian representing these readings. This is what I believe was the issue causing behavior in the latter half of the dataset, as the sensor readings didn't line up with the walls, and thus the system overvalued particles that matched the readings. I spent very little time tuning this part of the system as it worked fairly well from the very beginning.

Future Work

As the system does not currently work well given the latter half of the data, I believe that tuning the sensor model to be less confident would be fruitful.

I also believe that the system uses too much data; 100 particles with approximately 120 rays each does not work for realtime domains. While the code could be further optimized, that will not give nearly as large of a performance gain as would limiting the algorithm's use of data.

Finally, I think it'd be interesting to try this without the use of odometry data; simply using laser scan data and simply placing a gaussian around each of the particles for the motion model.