16831 RoboStats Lab 1: Particle Filter Robot Localisation

Kumar Shaurya Shankar
kumarsha@cs.cmu.edu

October 7, 2014

1 Approach

The general approach is to follow the standard paradigm for Monte Carlo localisation using particle filters, as presented in [1]

2 Implementation

2.1 Code

The code is written as an Eclipse project with Makefiles created in the subdirectory Default. Map is a class that reads and stores the map as an OpenCV matrix. Particle is the class that contains the observation model measurement probability and propogation method. MCFilter is the class that orchestrates the sampling loops.

2.2 Sampling Distribution

For the initial sampling step, I determine valid robot locations by thresholding pixels in the map with occupancy probability between 0 and 0.001. This threshold value was used because of a few spurious low probability obstacles in the lower right corner of the main corridor. Once the valid locations are obtained, indexes are chosen at random from the list of valid locations, and an arbitrary angle is chosen out of 180 possible orientations.

2.3 Motion Model

I use the odometry motion model as presented in [1]. This model samples from the relative transformation between two consecutive odometry estimates. Each transformation is broken down to three components involving an initial rotation followed by a translation and a final orientation rotation. Gaussian noise depending on a few parameters added to each of these components provide the motion model the descriptive power to represent a very wide range generic motion models.

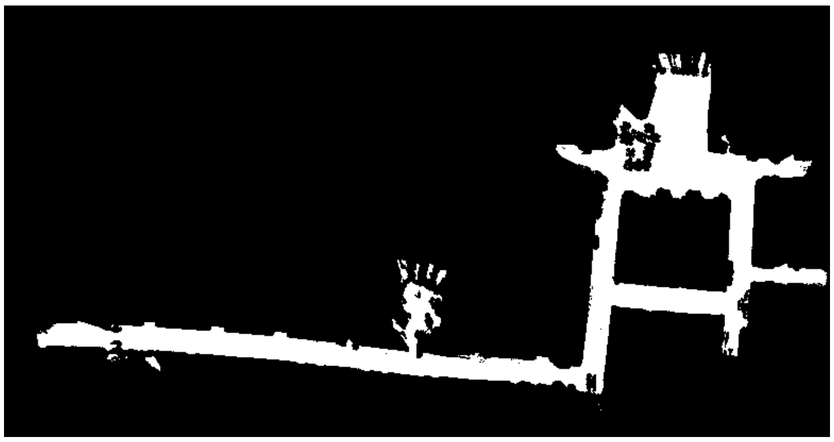

Figure 1: Valid particle locations in the map after being thresholded for particles with probability between 0 and 0.001

2.4 Sensor Model

For ray tracing, I use the standard DDA algorithm to find the grid location of intersection for a ray starting from the query point. I also build a lookup table discretized over 360 degrees of rotation using the ray tracing for speedup. Finally, I use OpenMP for parallelizing the particle weight estimation routine. Surprisingly, the lookup method did not offer a significant speedup over ray tracing. For randomization, I used the boost libraries with a Mersenne Twister pseudo random generator.

For the sensor model, due to the noise in the laser data readings and in ray tracing, the sensor model consists of primarily a uniform distribution that overlaps with a wide Gaussian distribution centered on the ray traced reading. The relative magnitude of the two distributions is tuned such that the overall distribution looks like a small bump around the ray traced measurement, and uniform everywhere else. This works by effectively allowing noisy potentially misaligned particles to survive a few resampling iterations and get attracted towards true orientations.

2.5 Resampling Procedure

The resampling procedure used is a slightly modified version of the low variance resampler as presented in [1]. The only difference in the algorithm is that if a particular particle index is being resampled, then a small additional gaussian perturbation is applied to the orientation of the propogated particle. This ensures that there is some variance between the resampled particles.

2.6 Parameters

This project was a parameter tuning hell. Lots of parameters.

2.6.1 Motion Model Parameters and Parameter Sensitivity Testing

Please see corresponding table. The videos linked in the description were produced without any gaussian noise in the motion model, but for testing parameter sensitivity, I chose a specific set of parameters that primarily add noise in yaw and a little translation (video at dataset1tweak, and dataset2tweak). This specific choice of low variance gaussian noise was chosen so as to accommodate the comparative larger jumps between the odometry between laser readings as I throw out the Odometry entries in the data log. We still converge with this motion model on both the datasets tested (1 and 2), but it takes longer for the algorithm to settle to the final distribution.

Videos	α_1	α_2	α_3	α_4
	0	0	0	0
Tweak	α_1	α_2	α_3	α_4
	0.05	0	0.1	1

Table 1: Parameters for the motion model. Tweak was a separate set of parameters used for sensitivity testing.

2.6.2 Sensor Model Parameters

As mentioned earlier the corresponding weights between the uniform and gaussian distribution were chosen such that the gaussian acts as a small bump that allows particles to survive longer.

Ray Tracing Gaussian Standard Deviation	5 dm
Weight for Gaussian	10
Weight for Uniform	1

Table 2: Parameters for the sensor model

2.6.3 Resampling Perturbation Model

Since the primary source of error is in the yaw due to skipped odometry measurements, I perturb only the theta with a standard deviation of 0.2 radians. This ensures that the particles on being resampled do not all get populated at the same location, and allows fine tuning of the particle orientation thanks to the bump in the observation model.

Standard Deviation for Rotation	0.2

Table 3: Parameters for the Gaussian perturbation at resampling

3 Results

The presented code succesfully localises on all the datasets. In particular, using only 1000 starting particles sampled uniformly across the valid locations in the map, the particles converge to the proposed location within 30 time steps (excluding dataset 4, where two proposal locations exist until towards the end). These videos are available at Dataset 1 and Dataset 2. A folder containing all these videos is available at this Dropbox folder

4 Future Work

In future work I can be smarter about using an adaptive number of particles, seeing that the filter rapidly converges to a candidate set fairly quickly, it will probably be more efficient to start sub sampling the number of particles after an initially large set of particles. This could also be a part of the kidnapped robot problem where, in the presence of a only a few candidate particle sets, some particles should be randomly sampled across the map.

It would also probably be useful to work on future lab projects with partners.

References

[1] Thrun, Sebastian and Burgard, Wolfram and Fox, Dieter, *Probabilistic Robotics*, 2005

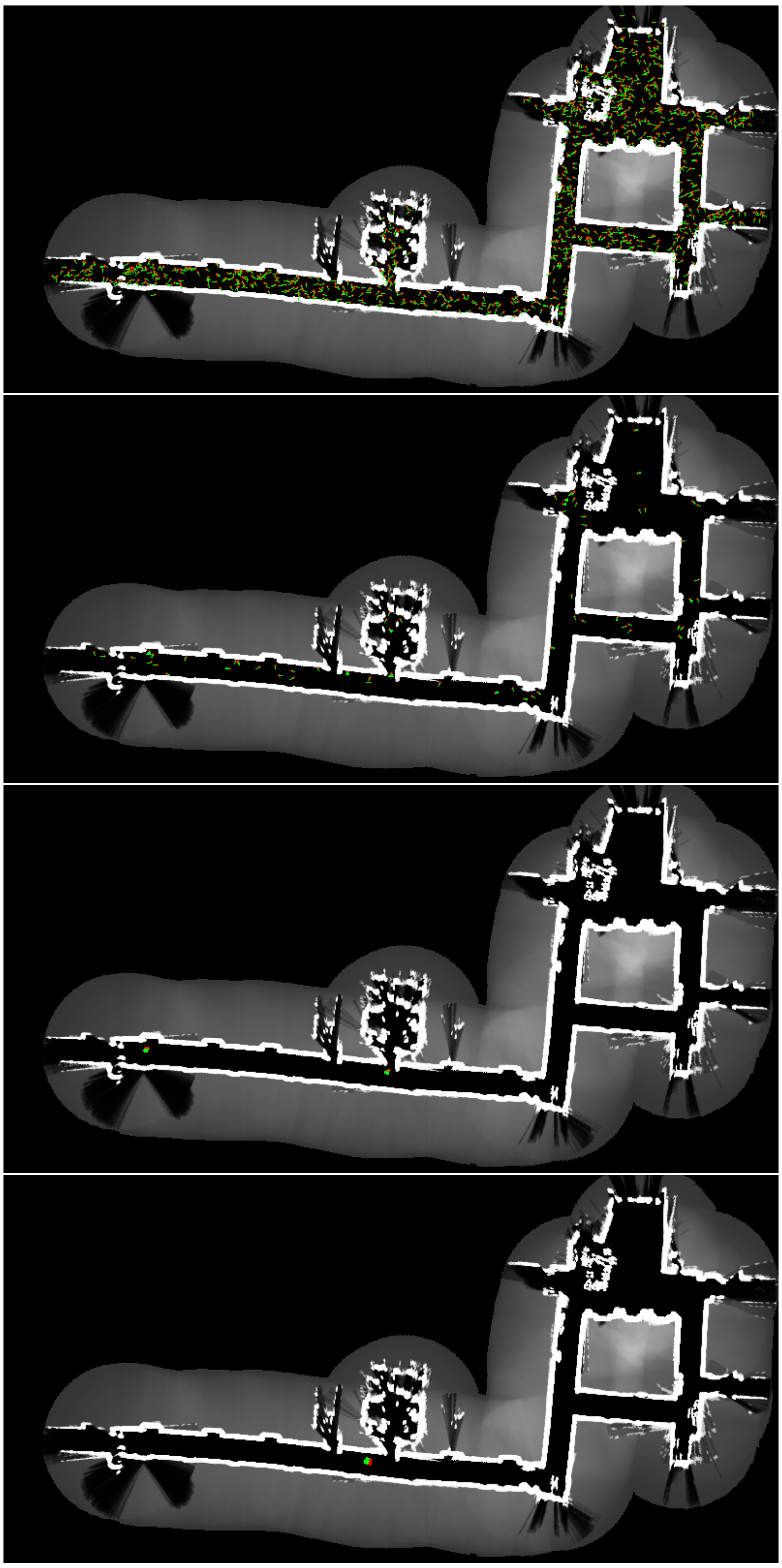

Figure 2: Sequence of the particle filter running at timesteps 0, 25, 63 and 117 respectively for the second datalog. Notice the rapid convergence at timestep 25, two proposal distributions at timestep 63 and finally resolution to just one candidate particle set at timestep 117