Comp 150 Probabilistic Robotics Homework 2: Image Based Particle Filter for Drone Localization

Aparna Penmetcha

March 3rd 2020

In this assignment, the task was to create an image-based particle filter that can localize the position of a drone over a known aerial map. The drone flies over the map and can take small images, and the map is an image of the entire area. To achieve this goal, Python with the 2D plotting library matplotlib were used.

1 Particle Filter: Simulation

The first part of the assignment was to create a simulation environment which allows the drone to move in random x and y directions, take sensor measurements before each movement and generate a reference image for a specific position on the map.

The initial step to simulate this environment was to create a function that when given the position of the drone, the aerial map and the size of the desired image, would return a subset of the whole map that the drone will see which is equivalent to the current position of the drone (observation image). The reference image is the image from the particle, assuming that the drone is in the position where the particle is. The particles are guesses of where the drone is, and ideally the particles will be seeing what the drone sees.

Finding the error between the reference and observation image tells us that if the error is high, then the particle's estimate is incorrect and thus a bad particle. If the error is low, then the estimate is close to the correct position and considered a good particle. Several different methods to calculate errors were considered and in the end, Root Mean Square Error (RMSE) was determined to be the best method. RMSE is a common way to measure the error of a model which predicts quantitative data [1] with the following equation:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

with $\hat{y_i}$ as the predicted values and the y_i as the observed values. In our case $\hat{y_i}$ represents the reference image and y_i is the observed image. The RGB differences between the reference and observation images are squared, summed and then the square root of them are taken to obtain the errors.

The simulation was also created in a way that to advance to the next time step, the enter key should be pressed. As the drone moves in random x and y directions, the particles will also move in the same direction as the drone and attempt to populate around the region of the drone. In the simulation the drone is shown as a black circle, while the particles are shown as red dots. Some parameters that were initialized in this simulation were:

- 1 unit of distance travelled by drone is equivalent to 50 pixels
- Desired image size from drone is 100
- Both the observation and reference image should be of size $m \times m$ (in this case 100x100)
- Number of particles for localization of drone is 100

2 Particle Filter: Implementation

The second part of the assignment was to actually implement the two main steps of the particle filter:

1) Sensing and re-sampling of particles according to the likelihood of seeing that observation
2) Moving the particles according to the known movement vector (with some added noise

At each time-step, you start with the true position of drone and set of particles. For each particle the reference image is captured and the error between the reference image and observation image is found. For each particle we have a value of error, and the bigger the error, the further the particle is away from the actual position.

To be able to re-sample the particles, each particle must be assigned weights in terms of probabilities. As previously mentioned, the higher probabilities are matched to the particles that have smaller errors, and lower probabilities to larger errors. This inverse relationship between the errors and probabilities for the particles are useful in determining the re-sampled particles. According to the errors, we can assign the probabilities for the particles to be used to re-sample for the next time-step. After assigning weights, we re-sample our particle set with replacement of probabilities according to weights.

The following figures illustrate the particle and drone simulation at different time steps

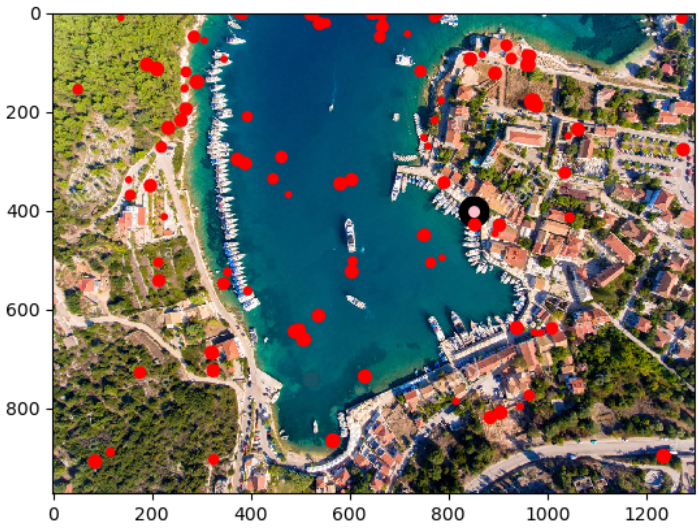

Figure 1: Simulation at time step $N = 0$

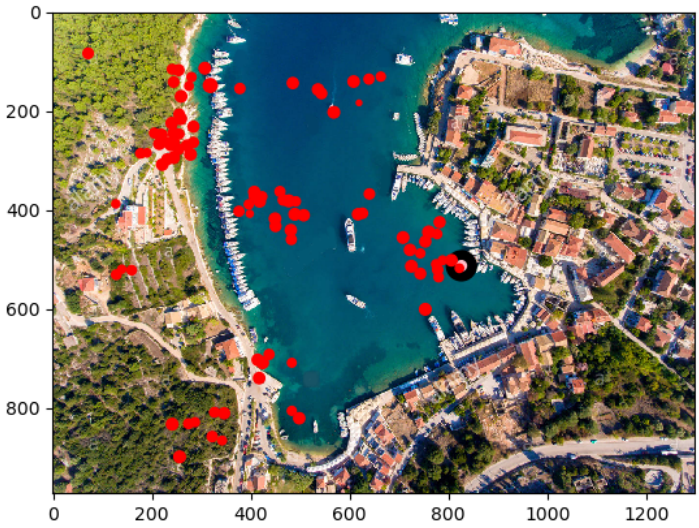

Figure 2: Simulation at time step $N = 8$

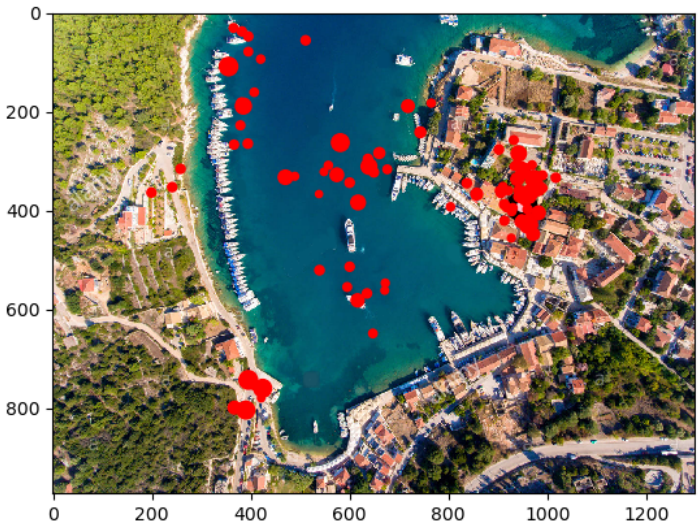

Figure 3: Simulation at time step $N = 15$

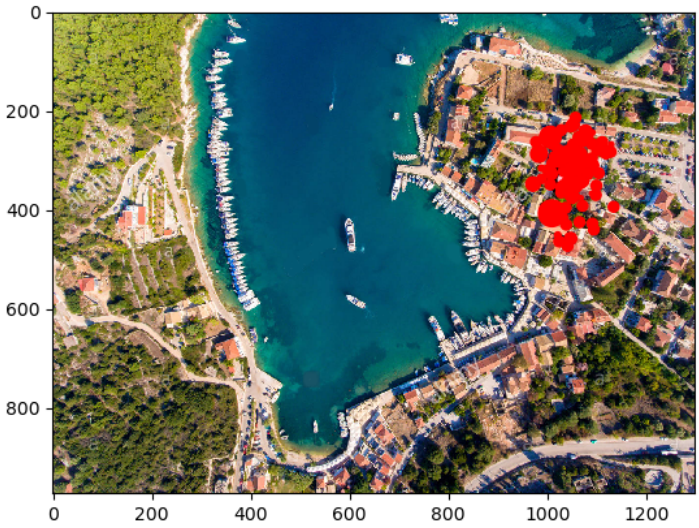

Figure 4: Simulation at time step $N = 24$

As seen from the above Figures 1-4, the particles are initially spread out throughout the map. After each time step, particles that have a high probability associated with them (as in highest probability that they are close to the true position of the drone) will get re-sampled.

Each time-step consists of particles, a reference image, finding the errors between the reference and observation image, finding the weight for each particle and finally re-sampling those particles. After the re-sample, since the drone moves by a random vector (random x and y directions), the particles also need to move according to that vector. However, each particle also needs to have some associated noise. This allows each particle to "explore" different positions without being restricted to certain positions.

3 Experiments and Evaluation

To evaluate the particle filter, the metric of number of time steps for the particles to converge is considered. A maximum of 100 time-steps are allowed for convergence, and the program will end if the particles do not converge before 100 time-steps. For evaluation, an experiment has been set up as 70% of the particles have to be within 1.5x of the observation image for our localization to be deemed a success. The experiment will compare two different conditions: image size (m) and number of particles (N). 10 trials were run for this experiment for each four sets of conditions (N=100, m=100**;** N=100, m=200**;** N=50, m=100**;** N=50 m=200) which is represented in the figure below:

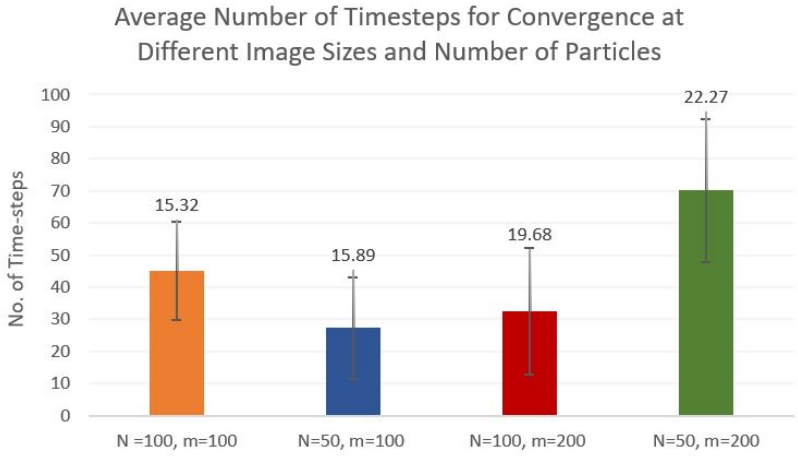

Figure 5: Average Timesteps for Convergence of Particles for Drone Localization

For image size (m), 100x100 and 200x200 was tested with the number of particles (N) set at 100. As expected, the larger image size of 200x200 was able to converge the particles together quicker with a time step average of 27.3, while the image size with 100x100 converged at 45 steps.

When the number of particles (N) was decreased to 50 with the image size still at $m = 100$, the average number of time-steps was found to be 32.4 which is a smaller average than that of the N=100, m=100 condition. This was not an expected result as I assumed that the more particles there were, the better the program would be at re-sampling the particles with more accuracy. One possible explanation for this discrepancy could be that since the condition was that 70% of the particles had to be in within 1.5x of the observation image, the less the particles, the faster that 70% is achieved.

When the image size was changed to 200, the number of time-steps decreased. It also seems that when the number of particles decreased to 50, the number of time-steps for convergence decreased. However, those two conditions of m=200 and N=50 combined resulted in the highest number of time-steps necessary for convergence.

Overall the particle filter does fairly well with the three maps that were given, with the best results from the BayMap. However, there is definitely an issue of consistency with the results of the particle filter. Specifically in the m=200 and N=50 condition, 4 out of the 10 trials did not converge. For calculation purposes, the numbers for those trials were set to the maximum trial number of 100, but if the simulation was able to run for longer, the average time-step would have been much higher. In all four conditions, there were incredibly low time-step results (e.g 7 time-steps for N=100,m=200) as well as extremely high time-step results such as the conditions without any convergence.

4 References

[1] Moody, James. "RMSE." Medium, Towards Data Science, 6 Sept. 2019, towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e.

5 Index

This section contains all of the raw data collected from the 10 trials for each of the 4 conditions.

Trials	No. of Time-steps
1	39
2	69
3	24
4	52
5	31
6	28
7	62
8	39
9	59
10	47

Trials	No. of Time-steps
1	17
2	31
3	15
4	54
5	17
6	11
7	29
8	10
9	41
10	48

Trials	No. of Time-steps
1	63
2	26
3	13
4	25
5	7
6	37
7	62
8	49
9	22
10	20

Trials	No. of Time-steps
1	61
2	58
3	DNC
4	43
5	47
6	73
7	DNC
8	DNC
9	66
10	54