Title: COMP 150 Probabilistic Robotics Homework 2: Image-Based Particle Filter for Drone Localization

Name: Ramtin Hosseini

Algorithm

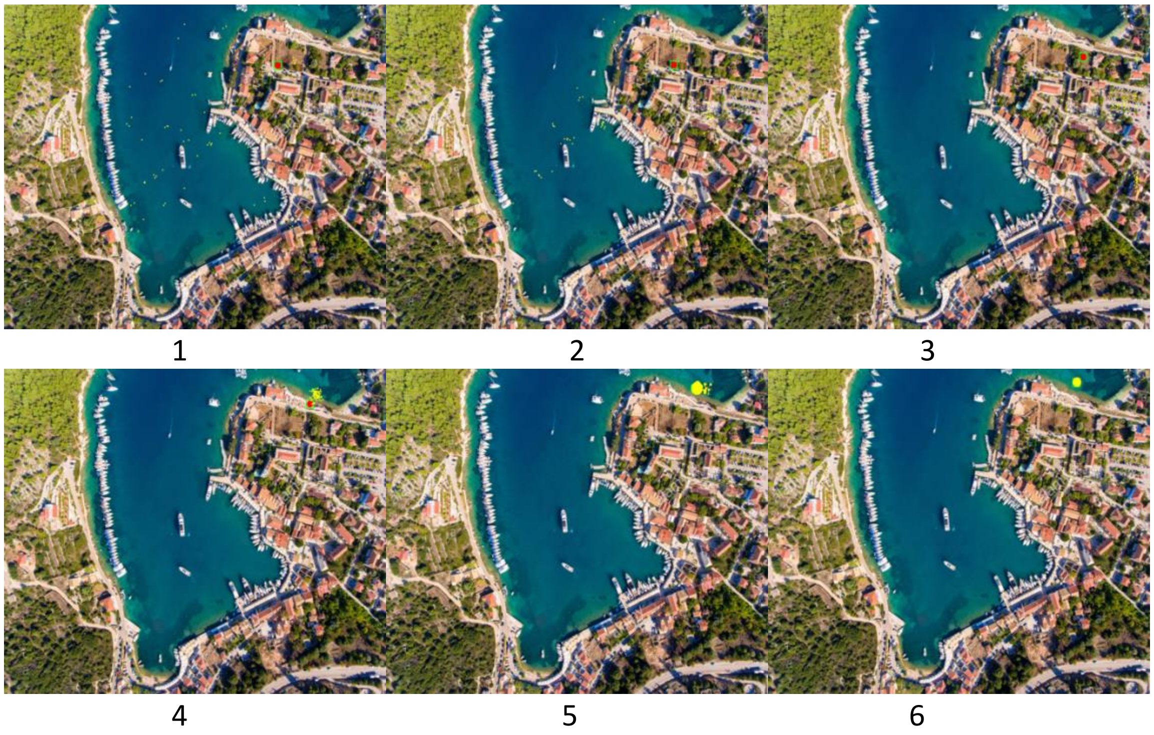
At the beginning, the map is generated with drone at random and particles in random positions. Then the reference image (a square image around the drone position) and the observation images (square around the particle positions) are created. Each image is a 3D NUMPY array with RGB channels.
I used Root Mean Square Error (RMSE) as my loss function to compute the difference between the pixels of the reference image and the observation images. Based on RMSE, I used negative exponential function: *exp(-x)* to assign weights (probabilities) to particles.
Note that the higher, the distance, the lower the weight should be. Thus, if we consider the exponential function in <u>negative domain</u>; for all $x, x' < 0$, if $|x| > |x'| \Rightarrow exp(-x) < exp(-x')$.
After applying above mentioned operation, I normalized the probabilities that they sum to 1. The user can move the agent by pressing "a" key on keyboard. The drone moves randomly in any direction. The movement as described in the spec is $dx^2 + dy^2 = 1.0$. Note that 1 UNIT distance is 50 pixels. Moreover, I added noise to this movement which is drawn from a gaussian distribution. Note that some movements are invalid; e.g. it shifts the drone or particles to the outside of the map, so they need to be ignored by considering a few conditions.
Note that the agent knows about the movement vector, but it has not access to the noise.
After drone movement, based on the movement vector, the particles will be moved with the same movement vector but each of them with different random noises which are sampled from gaussian distribution. Finally, based on the probability of each particle, weighted importance sampling will be done for generating new particles (particles with higher probabilities have higher expectation to be repeated, but particles with lower probabilities will be less or removed during time).
This process will be continued until localization (most of the time, it converges well, especially for BayMap and CityMap but sometimes the agent gets confused, mostly it happens in MarioMap, and cannot converge correctly to the correct location. The reason is the randomly assigning particles in locations which have very similar pattern to the location of the drone, so it cannot localize itself correctly.)

Experiments and Results

I did several experiments. In most cases, the agent could localize itself; however, it might happen that drone could not localize itself. I investigated to figure out the reason. I realized that if at beginning, particles located at random positions which have very similarity with true drone positions, it is very difficult for agent to be able to localize itself. In the following, we have example of both cases that localization successfully happens, or it fails.

Good example (Agent could localize itself)

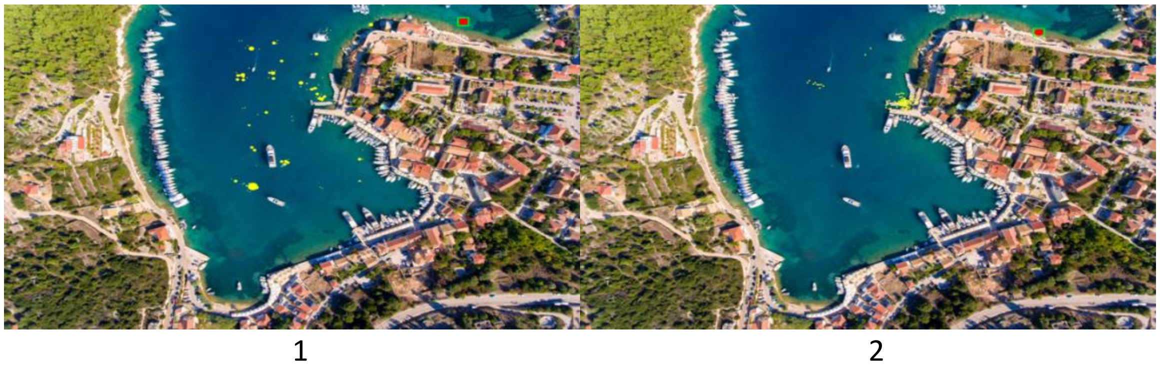
In this experiment, the drone could localize itself over time.



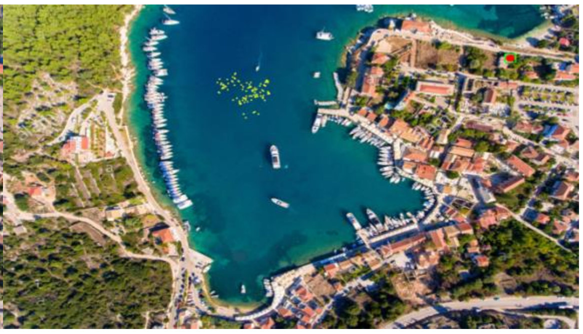
Localization in BayMap. From left to right at different time steps. Top left shows time step = 0. The next time steps are 10, 30, 120 and 125 respectively. The Hyper-parameters are window size 25 and 100 particles.

Bad example (Agent could not localize itself)

Due to random initializing to area which have very similarity pattern to true position, localization failed.



3


4

Localization in BayMap failed. From left to right at different time steps. Top left shows time step = 0. The next time steps are 10, 50 and 120 respectively. The Hyper-parameters are window size 25 and 100 particles.

I have defined two metrics for evaluating the performance of particle filter. For each metric, the experiments have been done 10 times.

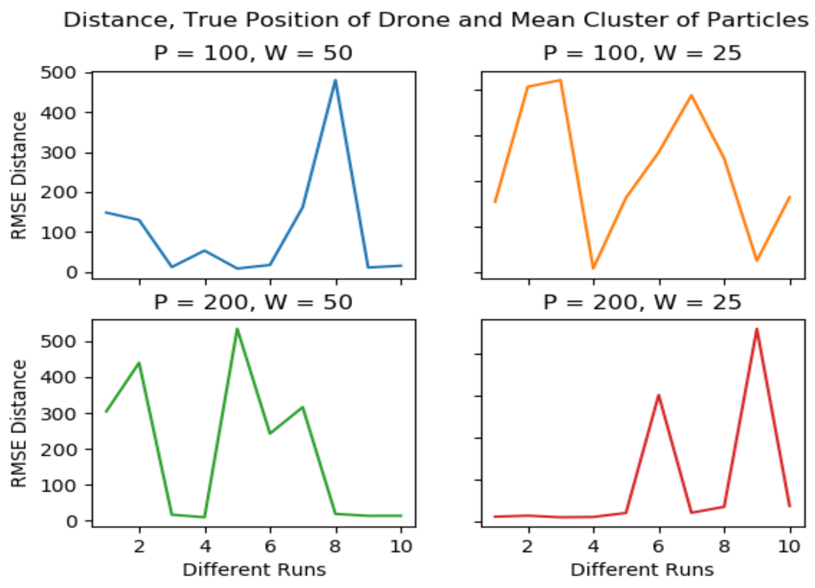
- Distance between <u>mean</u> cluster to true position after 60 time-steps
- Distance between <u>nearest</u> cluster to true position after 60 time-steps

The conditions that I investigated are:

- Window size: 25 vs 50
- # Particles: 100 vs 200

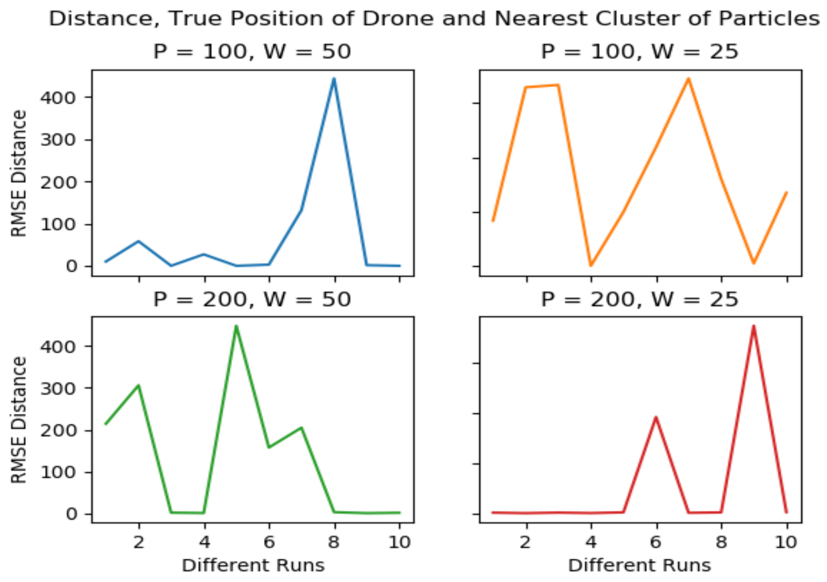
Distance between <u>mean</u> cluster to true position after 60 time-steps

In most experiments, larger window size contributed that drone could localize faster and in a fixed amout of time, on avarage the particles get closer to the true positions. In most experiments having more particles could help faster convergence and mean cluster gets closer to the true postion BUT if the random initilaization of particles are bad, then having more particles could cause that it becomes more difficult for the drone to localize itself.



Distance between nearest cluster to true position after 60 time-steps

The same conclusion as above, larger window size causes that drone could localize faster and in a fixed amout of time, the nearest particles get closer to the true positions. In most experiments having more particles causes faster convergence and nearest cluster gets closer to the true postion BUT if the random initilaization of particles are bad, then the drone face difficulties to localize itself, as it should handle more particles which are far away from true position and have stucked in areas with high similarity to the true position. Thus, it would be more challenging to get rid of these area and the likelihood of failure will be increased.



Note that, in these experiments, out of 10 runs for each configuration of window size and number of particles, the success/ failure of localizing are in reported in the following tables.

Window-Size	**# Particles**	**Success**	**Failure**
50	100	9	1
25	100	7	3
50	200	8	2
25	200	8	2

Note that, in these experiments, the random initializations were seeded that we avoid the effect of it for doing comparisons.