Comp 150 PR HW2

Andre Cleaver

March 10th, 2019

1 Simulation Environment

In this homework, the task is to create a Particle Filter to localize the position of a drone over a given map. The tools I used for this experiment are Python with OpenCV. The maps are provided as 2D colored images and can be assigned as variables. The images' width and height are obtained and used to shift the origin from the top left corner to the center of the image. The coordinate of the initial drone position is displayed displayed along with the number of particles used in the simulation. Instead of hitting enter to proceed to the next time-step and display the updated drone and particle position, I allowed the simulation to wait $500 microseconds$ to proceed to the next time-loop for ease of use. The simulation will terminate once a goal condition is met. The goal is to have at least 80% of the particles in view of the drone. That is, the bulk of the particles must fall within the boundary of the drone's reference image. The terminal window will display the final image along withe time step of convergence.

The following steps occur to simulate the environment in the script:
1) Randomly pick a location position (x, y) within the chosen image that will be the location of the drone.
2) Generate the particles and set whether they are in a circular distribution around the drone, or if the particles are evenly distributed throughout the image.
3) Establish a cropping function that will take inputs of an image and a (x, y) position.
4) Establish a display function that will mark the location of the drone and region of interest on the image map.
5) Calculate the RMSE for all particles, normalize weights, resample 'good' particles after removing 'bad' particles.
6) Randomly determine the next position of the drone by randomly choosing an angle, $theta$, and extracting the dx and dy components. This $theta$ value is under constraint to avoid moving the drone out of the image frame.

Parameters that are initialized:
$Imagesize = 100$

$Number of Particles = 100$
$DroneSpeed = 50$ The drone will move a distance of 1 unit which is equivalent to 50 pixels.
$N(\mu = 0, \sigma = 5 pixels)$ Noise characteristic with average and standard deviation to drone and particles movement.

2 Particle Filter: Implementation

Calculating the probability of the measurement given the position of the particle, $P(z|x)$, is carried out by using the Root Mean Square Error (RMSE) method. After normalization this probability corresponds to the 'weight' of the particle, with a value of 1 being a perfect match. The equation goes are follows:

$$P(z|x) = RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(d_i - f_i\Big)^2}$$

Here, d_i is the reference 'drone' image and f_i is the particles' image. So in this equation, images that are identical will equate to a RMSE value of 0. The RMSE value is treated as the 'weight' of each particle after normalization. Particles that are or become out of range are assigned a weight of 0. Particles with weights of 0 are removed, and the remaining particles become part of a batch for resampling. Resampling is choosing particles based on weighted importance; therefore, particles with greater weights will be chosen more frequently.Particles and the drone then 'move' a distance of 1 unit which corresponds to 50 pixels with added noise, $N(\mu = 0, \sigma = 5 pixels)$. For visualization, the full map is displayed along with the location of the drone and particles. The code operates with a for-loop that treats each iteration as a time-step, in this case $dt = 1$.

Figure 1 shows a timelapse of the particle simulation at times $t = 0$, $t = 12$, and $t = 37$. The particles are initially distributed around the drone with equal weights. After each iteration, the particles with the highest probability on the location of the true position of the drone get selected for resampling. After an adequate amount of time, the particles will converge near to the drone's position.

3 Experiments and Evaluation

To evaluate the particle filter, I will look at the average time (K,steps) it takes for the largest cluster of particles to localize within the square of drone image that represents the 'true' position of the drone. Here the largest cluster must be 80% of the total number of particles. In my initial case 100 particles are generated and 80 of them must fall within the drone's 'line of sight'.

I will also run the experiment comparing two different conditions. I will evaluate how the particle filter performs when changing the crop size. Here, the initial crop size is 100 pixels. I will run the same experiment but reduce the

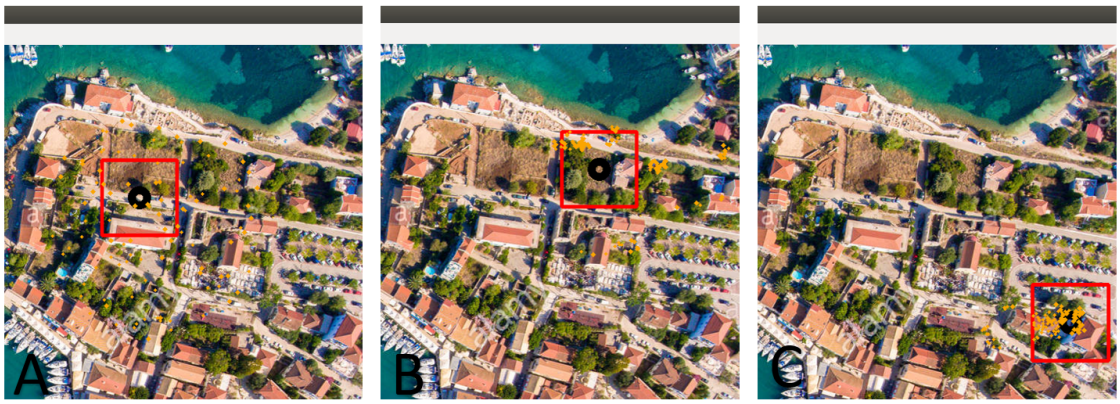

Figure 1: This figure shows a timelapse of the particle filter closing in on the location of the drone shown as a black ring. The red square represents the field of view the drone sees and will serve as the reference image. Here at Timestep = 0 sec (A) the drone spawns in random location within the the map along with a set of particles that are uniformily distributed within a 100 pixel radii of the drone all with equal weight. Timestep = 12 sec (B) The particles have now formed clusters that represent the predicted location of the drone. Both the drone and the particles move at the same relative distance with some noise. Timestep = 37 (C) The particles have met the goal on converging on the true position of the drone.

crop size by 50 pixels. By reducing the crop size I would expect the algorithm to converge at much longer time-steps and even fail to converge more frequently.

Evaluating convergence, I allowed my particle filter to run up to 100 iterations, ($K = 100$) for 20 runs. Under the parameters above, the particle filter managed to converge 19 out of the 20 runs with an average convergence at $K = 35.4sec$ and a standard deviation of $\sigma = 24.02$ sec.

As for comparing two different crop sizes ($crop_size = 100$) and ($crop_size = 100$), I performed the simulation with small and large particle distribution. In Figure 2, there is an increase in convergence time as you decrease the image size. This supports my claim as the amount of pixels for comparison is insufficient. This suggests that there is a minimum image size that can be assigned to allow convergence for every simulation. When distributing the particles throughout the image rather than a smaller circular distribution, the amount of successful convergences within the 100 timesteps decreases. In the case of smaller image size and larger distributions, less than half of the 20 runs converge.

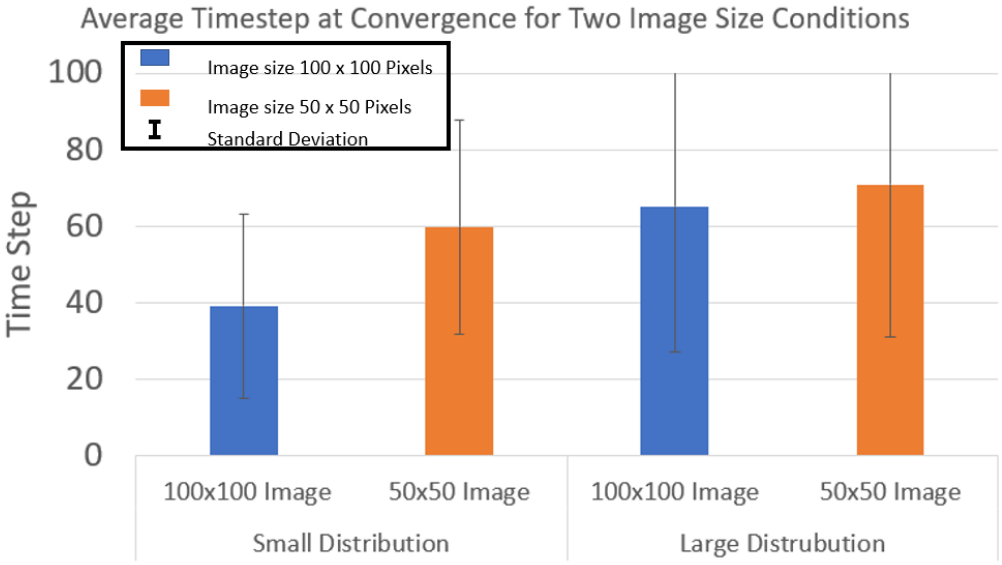

Figure 2: This figure shows the average timesteps of convergence of the simulation considering the 2 conditions of image size. These results suggest that decreasing the image size will result in a relatively longer convergence time. The size of the standard deviation bars also suggests that the simulations are not precise as some simulation will complete as little as 7 time steps and as long as 98 timesteps.

4 Extra Credit

As part of the simulation section, I added noise to the RBG image captured for both the drone and particles' cropped image.