Particle Filter for Drone Localization

Srisharan Kolige

1 Simulation Environment Setup

Programming Language and Libraries

The simulation of particle filter localization was done with Python.

For image processing, OpenCV and Numpy were used; for visualization, Tkinter was used.

Mapping between Continuous and Discrete Spaces

For the convenience of mapping between the x-y coordinates and pixels, every image was resized so that its width and height are odd number of pixels.

This way, the origins of both spaces can be centered together as illustrated in figure 1 (the origin of the pixel space was set to the center).

The mapping function from x-y coordinate to pixels is as follows:

$$(\,i,\ j\,) = (\,round(x * scale),\ round(y * scale)\,)$$

where i and j are row and column of the pixels, x and y are coordinates on the x-y plane, and the scale is 50 pixels per unit distance on the x-y coordinate.

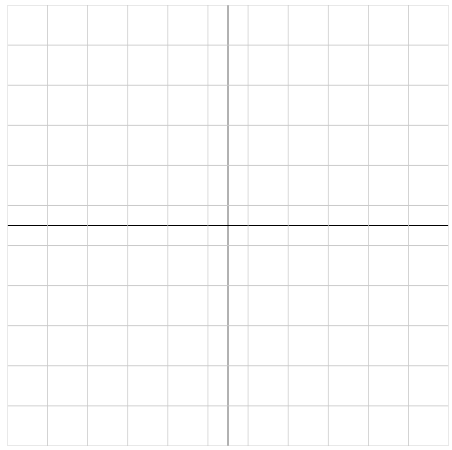

Figure 1. The illustration of centering the origins of both spaces.

Drone Movement and Observation

Initially, the drone is positioned randomly throughout the map according to uniform distribution.

Then, upon key press, the drone makes a random movement with distance of 1.0 unit length in the x-y coordinate plus a Gaussian noise of zero mean and 0.1^2 variance at both x and y directions (the drone never moves outside the boundaries of the map).

At each time steps, an $m \times m$ image, centered at the location of the drone, is cropped.

Cropped image size, m, was set to several different values, such as 31, 51, and 101.

Positioning Particles

At $t = 0$, N number of particles, with the same weights, are uniformly distributed throughout the map.

When key is pressed and the drone is moved, each particles is moved according to the drone's belief of its movement (without the drone's movement noise) with Gaussian noise added.

The number of particles, N, was set to either 500 or 1000.

Weighing Particles

When the particles move, the weight of each particles is re-calculated.

First, a reference image with size of $m \times m$ is cropped at each particles' position.

Then each of the reference images is compared to the observation image.

To compare images, RGB color histogram with size of $4 \times 4 \times 4$ bins was used.

This method, specifically RGB color space, was chosen because the maps show a clear difference in red, green, and blue pixels.

In addition, color spaces such as HSL and HSV were unnecessary because the color intensity of the map is always consistent.

The histograms of the observation and reference images were compared by each bins.

Finally, the weights were calculated as following:

$$w = 0.5 * w_t + 0.3 * w_{t-1} + 0.2 * w_{t-2}$$

Two previous weights were included in the calculation in order to prevent sudden change in weights that causes significant errors.

According to its weight, each particles is resized at every time step.

2 Simulation Result

Simulation Overview

For the project, all three maps were used instead of choosing one specific one.

Due to their distinct aspects, three maps had different probability of success.

The first map had clear difference in red, green and blue pixels, making localization the easiest among the three.

However, the consistent blue value at the ocean sometimes makes it difficult to localize.

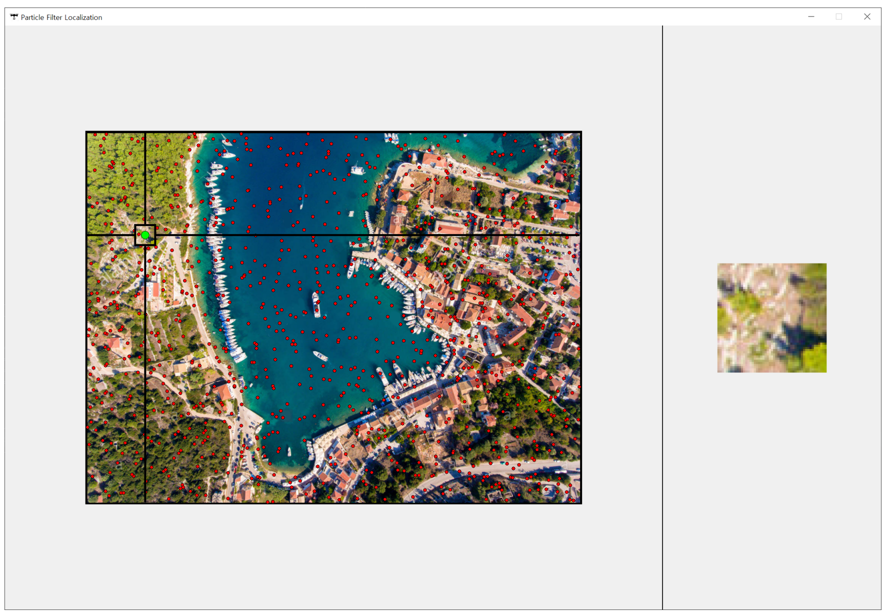

Figure 2. First map at $t = 0$.

Figure 3. *First map at* $t = 5$.

Figure 4. *First map at* $t = 20$.

Figure 5. Starting at the ocean ($t = 0$).

Figure 6. 20 time step after starting at the ocean ($t = 20$).

The second map has a similar pattern all over the map, making it most difficult to localize.

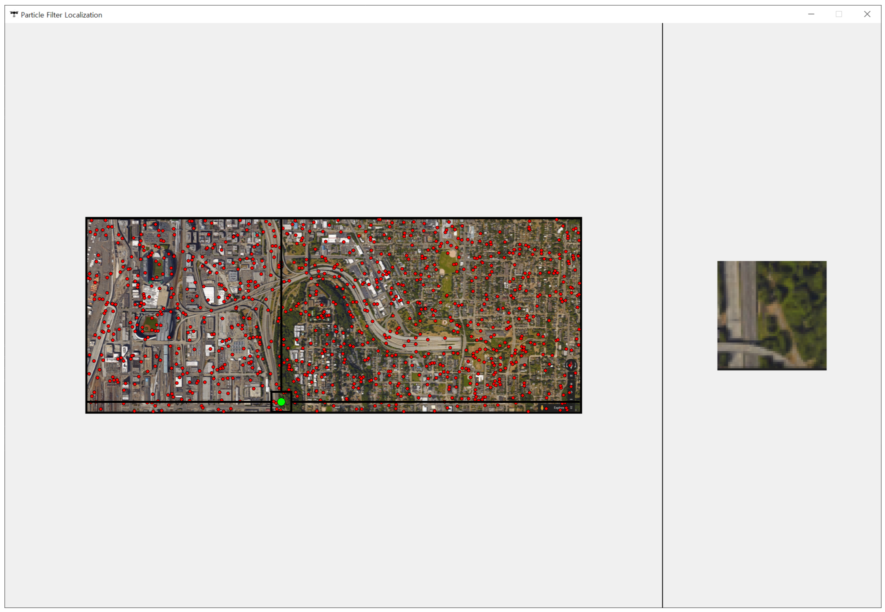

Figure 7. Second map at $t = 0$.

Figure 8. Second map at $t = 20$.

The third map has the most distinct colors by each pixels.

This makes it very easy to make clusters, but difficult to make multiple clusters into one.

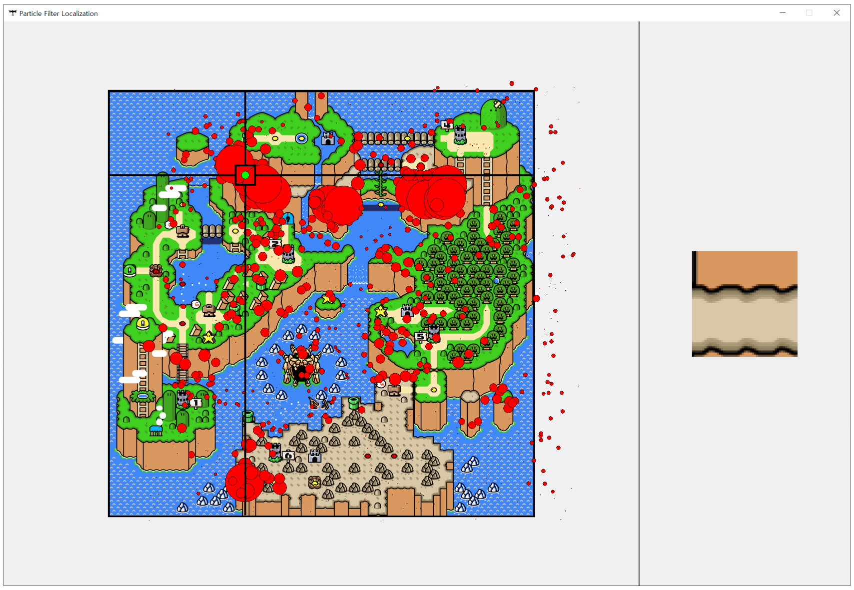

Figure 9. Third map at $t = 10$.

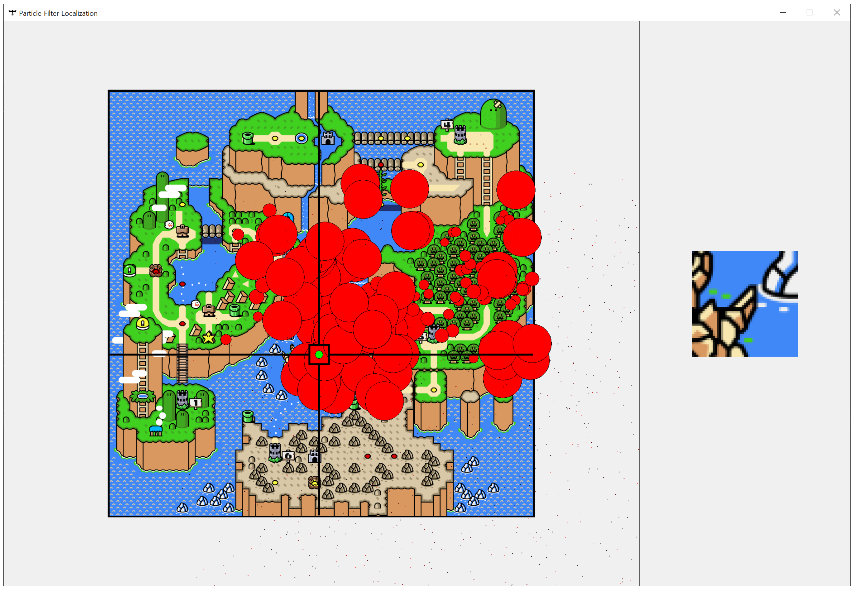

Figure 10. Third map at $t = 20$.

Experiment Evaluation

In order to have a metric of evaluation, the number of localization successes were counted under multiple conditions.

The conditions are
i) map = map1, map2, map3
ii) width/height of cropped image: $m = 31, 51, 101$
iii) histogram comparison method: Chi-Square, Bhattacharyya distance

The probability of localization was calculated for each conditions by counting the number of times the particles cluster at true position in 20 time steps out of 20 trials.

The probability of localization is shown in the table below.

	Chi-Square	Bhattacharyya Distance
m=31	0.55	0.5
m=51	0.6	0.7
m=101	0.6	0.6

Table 1. Success rate of localization in Map 1

	Chi-Square	Bhattacharyya Distance
m=31	0.05	0.1
m=51	0.05	0.15
m=101	0.1	0.1

Table 2. Success rate of localization in Map 2

	Chi-Square	Bhattacharyya Distance
m=31	0.5	0.45
m=51	0.55	0.5
m=101	0.5	0.4

Table 3. Success rate of localization in Map 3

According to the experiment, the best success rate was at Map 1, when $m = 51$ and the histogram comparison method is Bhattacharyya distance.

It seems like when the cropped image is too small, the probability of the particle being near the true position is small.

And when it is too big, the observation and the reference images tend to include too various points of interest, such as red roof or green trees, making the color histogram inaccurate.

Although localizing with particle filter was successful in certain conditions, implementing just the color histogram method for image processing had many limits.

For a more robust method of localizing with flying agents, other methods of image processing in addition to color histogram must be implemented.