CS141: Particle Filters

Prithvi Shahani

March 2024

1 Introduction

In this assignment, I implemented a 2D image-based particle filter for localization against a known map. The particle filter is a non-parametric implementation of the Bayes filter. Intuitively, particles are spread across a map and based on some image comparison metric, particles are given a weight by how close they resemble the observed image from the agent, and are resampled accordingly to represent areas that are most likely to be where the drone is. The particles move according to the expected movement of the drone and are weighed and resampled accordingly as they attempt to localize the drone position on the map. Particles can localize both position and direction, but in this assignment, I assumed that direction is fixed throughout (and points North).

The function $P(z|x)$ to determine each particle's weight comes in many flavors. Some common approaches are the *color histogram* and the *structural similarity index measure* (SSIM).

The color histogram method represents the distribution of colors in an image by counting how many pixels of a color (or a bin of colors) are present and using a method like correlation to determine similarity. For this method, higher correlation is deemed to be more probable of being close to the true drone position.

On the other hand, SSIM measures similarity in texture, luminance, and contrast instead of color (and is typically done in grayscale comparisons). Similarly to the color histogram method, a higher correlation is deemed to be more probable of being close to the true drone position.

SSIMs tend to be more expensive than histogram-based methods, but tend to be more accurate considering image structure, while histograms are simple and focus on comparing image palettes.

Along with that, the resampling methods that could be taken also came in many flavors. Some common approaches are *roulette resampling* and *systematic resampling.* In both cases, resampling noise is added to the resampled particles so that they cluster around probable areas instead of just representing fixed areas deemed likely in previous samples of particles.

In both cases, the probability distribution can be thought of as a strip of paper of length 1m. Each particle's distance on this paper is determined by its probability of being representative of the image (where the total probability of all particles sums to 1). So a particle with 0.5 probability, would take up half of this strip of paper.

Roulette resampling can be simply thought of as randomly spinning the roulette wheel (in a uniform distribution from 0 to 1) and picking the particle who appears on the strip of paper at the specific value the wheel landed on. This is repeated for the desired number of resamples (typically the same as the starting number of particles). This represents the weight distribution well, but heavy weights can dominate selection. It can become computationally expensive depending on the number of resamples as it has a runtime complexity of O(n log n).

On the other hand, systematic resampling is thought of as having a ruler lined on the strip of paper with equal intervals (for the number of samples desired). The starting point of the strip of paper is randomly selected and the particles are chosen based on it lining up with the interval lines on the ruler. The ruler is wrapped around to the start of the paper strip (if the ruler ends up misaligning with the paper) and the resampled particles are also picked from the wrapped around portion. It is less intuitive than the roulette approach but is much more efficient at a runtime complexity of O(n) and avoids having weights that dominate the resampled list.

Overall, this paper will compare both resampling methods and both probability $P(z|x)$ methods to understand their relative performance in image-based localization particle filters.

2 Simulation Environment

The simulation environment was modeled exactly after the specifications of the assignment. Specifically, drone movement is calculated by calculating an angle in the uniform interval $\theta = [0, 2\pi]$ and the dx, dy are taken by $dx = cos(\theta), dy = sin(\theta)$ respectively as $sin^2(\theta) + cos^2(\theta) = 1$. The movement is then added (along with noise) to the drone if the resulting observation image is still within bounds. If this property is violated, the movement is ignored and the agent stays at the same spot in the subsequent timestep. The noise added was relatively small (computed in trial and error), but enough that the movement vector isn't completely indicative of the drone position.

Generally, the constants picked in this such as the observation image size, noise, and bins per channel (for the color histogram) were determined via trial and error by visually seeing the particle filter in action. They were arbitrarily chosen to promote exploration for particles (to avoid localizing to a visually similar but different location on the map), to minimize unnecessary bottlenecks in runtime (more bins per channel means sharper comparisons but much slower performance and this also applies to image size).

3 Particle Filter

The particles were generated the same way the drone location's original position is generated and two implementations of weighted importance sampling were done as discussed in the introduction. Similarly, two resampling methods were also done for the particles with added resampling noise to form clusters to tackle the issue of resampling degeneracy of particles. Furthermore, movement of particles weren't constrained the same way that the drone movement was, but particles that can't form a full image (due to not being within image bounds) or are out of bounds are automatically assigned a weight of 0. The noise was also small and determined to encourage some degree of exploration, but not high enough that the particles don't cluster around relatively precise areas. The visual weights of the particles are just a product of the image width and the normalized particle weight.

In this implementation, 100 particles were used in the filter. 1000 particles had better localization, but due to runtime concerns, it wasn't feasible to run all the experimental cases with 1000 particles as a parameter. This is due to there being 100 trials and 100 timesteps per trial for each experimental case. However so, 100 particles still performed adequately well, converging towards the true drone position in many cases.

4 Experiments and Evaluation

As discussed earlier, for all the provided images, 4 cases were considered. The 4 cases are the combinations of the different resampling methods and probability functions considered in this paper. All cases were run with 100 trials and 100 timesteps per trial

Evaluation metric Mean Squared Error (MSE) of the drone location vs the average particle location (the average of all particles of the average of all trials) was used for each time step. This was chosen over the MSE metric in skimages because having a relatively similar image isn't indicative of good particle filter localization as seen in some cases (like the Mario Map) where different spots of the map look very similar but aren't necessarily the same location).

4.1 Bay Map

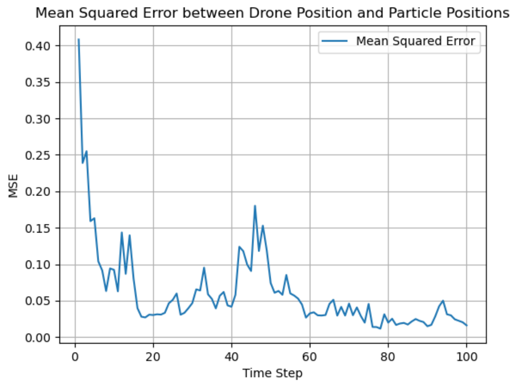

Figure 1: MSE of color histogram and systematic resampling

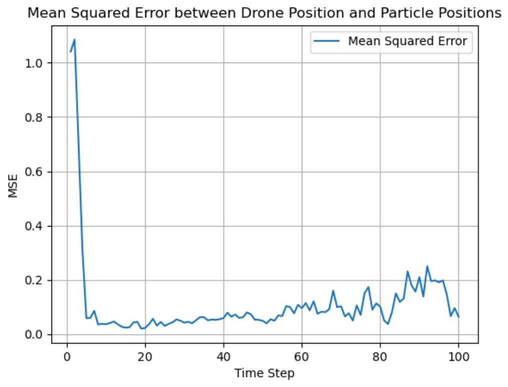

Figure 2: MSE of color histogram and roulette resampling

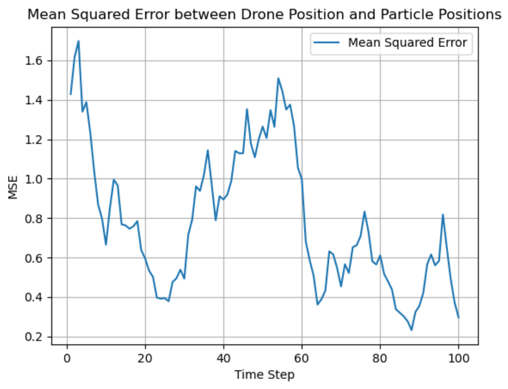

Figure 3: MSE of SSIM and systematic resampling

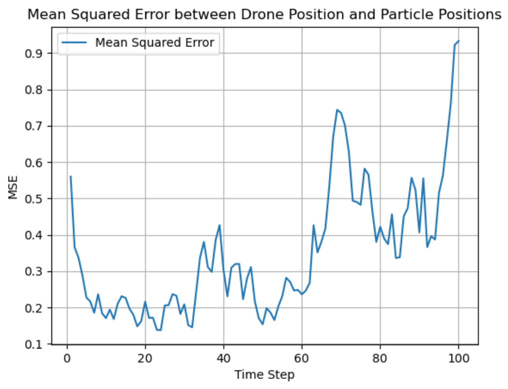

Figure 4: MSE of SSIM and roulette resampling

4.2 City Map

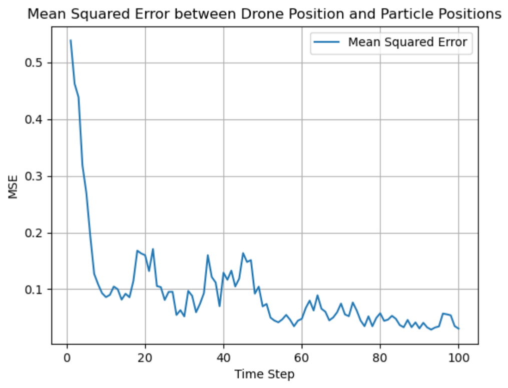

Figure 5: MSE of color histogram and systematic resampling

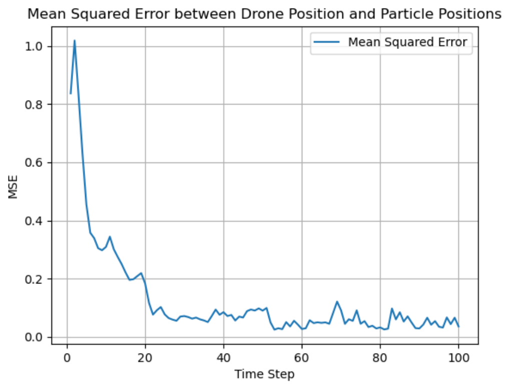

Figure 6: MSE of color histogram and roulette resampling

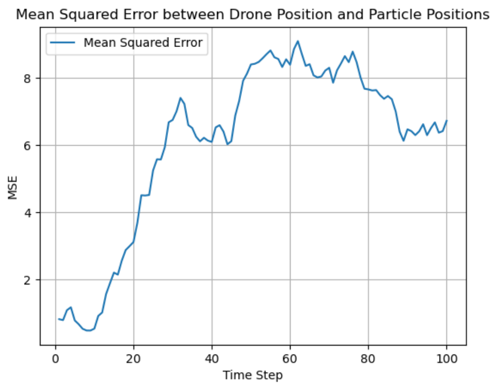

Figure 7: MSE of SSIM and systematic resampling

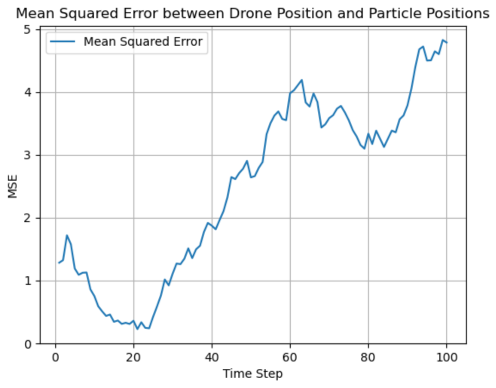

Figure 8: MSE of SSIM and roulette resampling

4.3 Mario Map

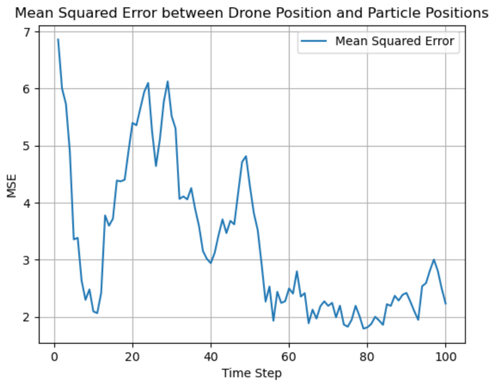

Figure 9: MSE of color histogram and systematic resampling

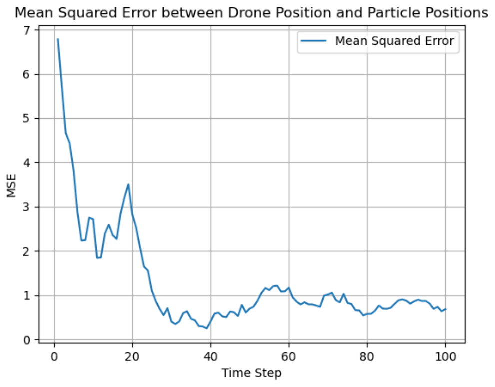

Figure 10: MSE of color histogram and roulette resampling

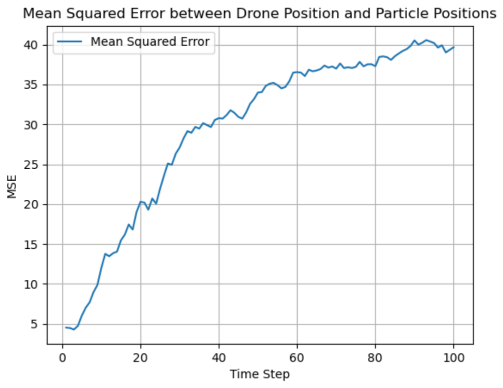

Figure 11: MSE of SSIM and systematic resampling

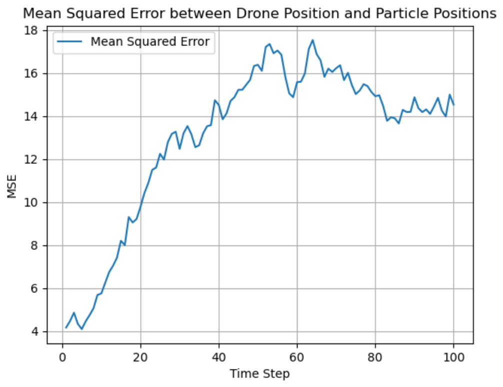

Figure 12: MSE of SSIM and roulette resampling

4.4 Discussion of results

4.4.1 On comparing color histograms with SSIM

As can be seen in the results, in all cases, it appears that color histogram weighting is superior to SSIM (keeping the resampling the same between comparisons). In the bay map, the MSE drops over time, especially when in comparison with roulette resampling (Figure 2) for color histograms, while for SSIM the MSE is more erratic. When visualizing this in the manner like the demo video, it appears that the particles get stuck in the water. This could be due to it perceiving textures in water as noise, so perceiving two observation units of water as structurally different while the color histograms perceive them as relatively the same. This trend is also observed for the city map, however SSIM performs worse over time seemingly for the same reason. It considers the blocks of the city as very different due to textures, while they're visibly the same to the color histograms. This is problematic as it would make particles more likely to localize to very specific blocks that have similar textures to the true location. This is even more exaggerated in the Mario map where the overall image has small structural differences throughout but for the most part the same color. Overall, for these images, it's quite clear that color histograms are superior to SSIM. It's potentially due to the nature of these images which have very similar overall colors but distinct textures throughout. SSIMs may perform better in images with distinct shapes or images with distorted features (like when drones have noise)

4.4.2 On comparing systematic and roulette resampling

In the Bay map it appears that roulette resampling performs poorly as compared to systematic resampling for color histograms. This is potentially due to systematic resampling promoting more exploration (by taking particles at regular intervals rather than randomly selecting particles which would cause a bias towards particles with greater weight) which allows for the particles to avoid being trapped by poorly localized but closely resembling particle images. This phenomena isn't very clear in the SSIMs case which could be due to SSIMs negatively impacting how the weights are determined for the particles. This same pattern is for the most part observed in the city map. It appears that over time, the The magnitude of MSE for systematic resampling is once again smaller as compared to roulette resampling, and this again demonstrates the spread out nature of systematic resampling. Lastly, interestingly enough, the roulette resampling performs better in the Mario map with more stability and lower magnitude MSE nearing the end of the timesteps.

When reading of the consequences of systematic resampling, it was evident that it can introduce a problematic bias when the weight distribution of the data aligns with the sampling intervals. This is seen in the Mario map case and makes sense because the Mario map's particles all have relatively similar weights at the start which could be aligned with intervals, which does explain why the

MSE fluctuates more at the later time steps, than it does for roulette resampling which doesn't force this uniform approach onto the resampling process.

5 Extra Credit

1 As discussed earlier, one of the $P(z|x)$ functions implemented earlier is the color histogram comparison. The other one was the structural similarity index measure (which is also not a distance-based function).

2 The attached video is a demonstration of the particle filter in action for the bay map with the color histogram and roulette resampling combination.

3 Lastly, the topic of SSIMs being suited to distorted images came up in the discussion. Hence, a third experiment exploring this was conducted for comparing SSIMs and color histograms for roulette resampling on the Bay map was done. Small noise was added to drone readings (but not particle readings) to change the RGB values.

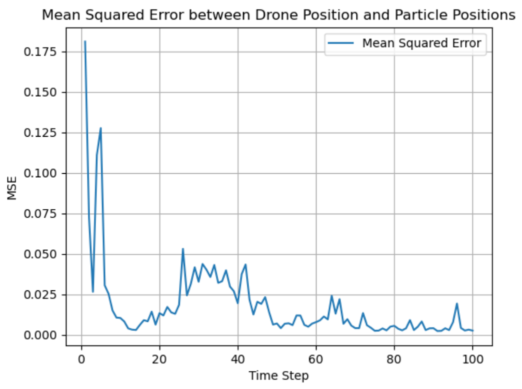

Figure 13: With uniform noise from [0,3] for color histograms

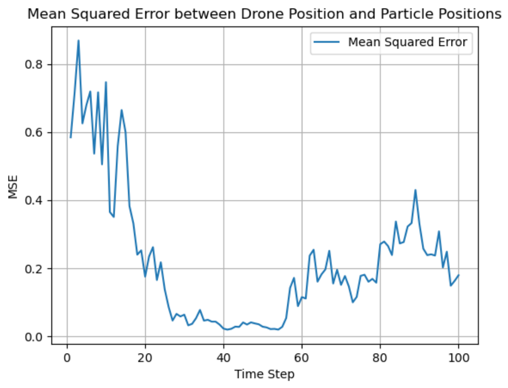

Figure 14: With uniform noise from [0,3] for SSIM

Evidently, color histogram weighting ended up being noisier/erratic compared to a noise-less version. On the other hand, it appears that SSIM performed better with noise as compared to no noise with a better MSE performance compared to figure 4.

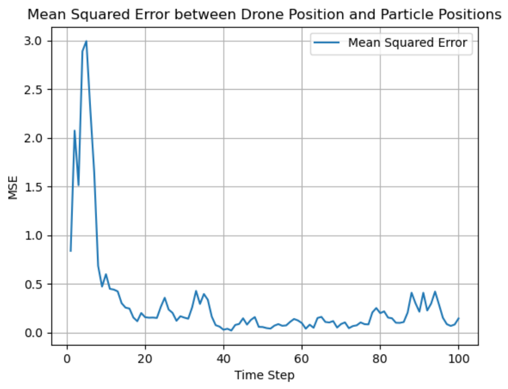

Figure 15: With uniform noise from [0,20] for color histograms

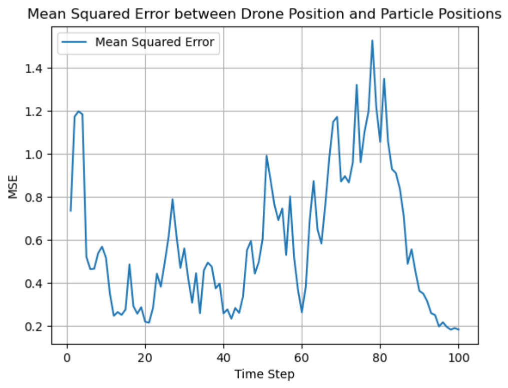

Figure 16: With uniform noise from [0,20] for SSIM

Evidently, more noise lead to a higher MSE for the color histograms, but surprisingly, SSIM was negatively impacted much more than color histograms with a more erratic MSE over the timesteps. It does perform better than it did with less noise near the end but evidently performs worse around the 60-80 timesteps.

It's quite interesting how the color histograms still perform better overall, given that SSIMs are expected to shine when the image to be compared is distorted. However, compared to itself, SSIM tends to get better (although noisier in the process) when there is more noise especially at timestep 100. Evidently, uniform noise of 20 (on a pixel basis) is enough to throw off SSIM which recovers at the end, but surprisingly, color histograms remain relatively unaffected in comparison.