Lab Assignment Report - Tracking

Yaqi Qin*

8 December 2022

1 Implementation of the Condensation Tracker

1.1 Computation of the color histogram

The `color_histogram` function calculates the normalized histogram of RGB colors occurring within the bounding box within the current video frame. The main steps are:

- **Check boundaries.** To ensure the bounding box lies inside the current frame, $xmin$ and $xmax$ should be within $[0, width - 1]$, $ymin$ and $ymax$ should be within $[0, height - 1]$. If not, then clip them to the closest boundary using the `numpy.clip` function. The results are converted to `int` to ensure validness.
- **Get the bounding box from the frame.** Given the valid corner coordinates, get the bounding box from $frame[ymin : ymax, xmin : xmax, :]$
- **Set bin edges.** For each RGB channel, set $hist\_bin$ equal-width bins over $[0, 255]$, using the `numpy.linspace` function to get $hist\_bin + 1$ evenly spaced bin edges.
- **Construct the normalized histogram.** Bin each channel of the bounding box using the `numpy.histogram` function. Then concatenate them all into a 1-D vector (length $= 3 \times hist\_bin$). The resulting histogram vector is then divided by the sum of all its elements to get the normalized histogram where all its elements sum up to 1. (For histogram computation, I have initially tried to combine the RGB channels and bin them into $hist\_bin^3$ bins using the `np.histogramdd` function, but the tracking results are bad where the trajectories fluctuate drastically. The reason might be the resulting histogram is quite sparse but high in dimensions, leading to drastic change in χ^2 distance)

1.2 Derivation of matrix A

When we assume no motion, the location of the sample in the next frame (x_t, y_t) should directly depend on the location of the current sample (x_{t-1}, y_{t-1}) perturbed by the random noise w_{t-1} i.e.

$$x_t = x_{t-1} + w_{t-1}^x$$
$$y_t = y_{t-1} + w_{t-1}^y$$

So the matrix A should be an identity matrix, i.e.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

*Student ID: 21-946-819

When we assume constant velocity, than the location of the sample in the next frame should depend both on the previous location and the amount of movement defined by the velocity (assume the velocity is the amount of movement between two consecutive frames), i.e.

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1} + w^x_{t-1} \\ y_t &= y_{t-1} + \dot{y}_{t-1} + w^y_{t-1} \\ \dot{x}_t &= \dot{x}_{t-1} + w^{\dot{x}}_{t-1} \\ \dot{y}_t &= \dot{y}_{t-1} + w^{\dot{y}}_{t-1} \end{aligned}$$

In this case,

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.3 Propagation

Let S_{t-1} and S_t represent the sample set at two consecutive frames. The python implementation utilizes the matrix operation.

In the no motion case:

$$\begin{aligned} S_t &= S_{t-1}A^T + \sigma_p * W_{t-1} \\ &= S_{t-1} + \sigma_p * W_{t-1} \end{aligned}$$

where S_t and $S_{t-1} \in R^{N\times 2}$, $W_{t-1} \in R^{N\times 2}$ is the stochastic component where each element follows a standard normal distribution, i.e. $W_{t-1}[i,j] \sim \mathcal{N}(0,1)$. σ_p is the deviation of the position.

In the constant velocity case:

$$S_t = S_{t-1}A^T + W_{t-1} \otimes [\sigma_p, \sigma_p, \sigma_v, \sigma_v]$$

where S_t and $S_{t-1} \in R^{N\times 4}$, $W_{t-1} \in R^{N\times 4}$ is the stochastic component where each element follows a standard normal distribution similar to the previous case. σ_v is the deviation of the velocity (the last two terms of $s^{(n)}_t$). In particular, here $\otimes$ means broadcast-multiplication, i.e. for each row of W_{t-1} perform element-wise multiplication with $[\sigma_p, \sigma_p, \sigma_v, \sigma_v]$.

After the prediction, the first two columns of S_t (the x and y coordinates) are clipped within the range of $[0, width - 1]$ and $[0, height - 1]$ to ensure the new samples are inside the frame.

1.4 Observation

In the `observe` function, the weight of all the particles are computed as follows:

Iterate over each sample $s^{(n)}_t$:

- Get the bounding box boundaries based on the sample center (px, py) and the box size, i.e.

```
xmin = px - 0.5 * bbox_width
ymin = py - 0.5 * bbox_height
xmax = px + 0.5 * bbox_width
ymax = py + 0.5 * bbox_height
```

Here whether the values are inside the frame is not checked, but rather checked in the `color_histogram` function.

- Compute the color histogram on this bounding box using the `color_histogram` function.
- Compute the χ^2 distance between the histograms of the current sample and the target, using the `chi2_cost` function, the result is denoted by $dist_i$
- Compute the un-normalized weight of the current sample according to the following equation:

$$particle\_w[i] = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{dist_i^2}{2\sigma^2})$$

Finally, the weights of all samples are normalized by dividing the whole array by its sum.

1.5 Estimation

In the `estimate` funciton, the new mean state of the sample set S_t is simply the weighted sum according to the $particle\_w \in R^N$, which can be realized utilizing matrix multiplication, i.e.

```
def estimate(particles, particle_w):
    mean = particle_w.T @ particles
    return mean
```

1.6 Resampling

In the `resample` function, same number of particles are resampled from S_t based on their weights $particle\_w$ with replacement. The weights of the new samples are again normalized by dividing the sum, i.e.

```
def resample(particles, particles_w):
    n_particles = particles.shape[0]
    sample_indexes = np.random.choice(n_particles, n_particles, p = particles_w, replace=True)
    return particles[sample_indexes], particles_w[sample_indexes] / sum(particles_w[sample_indexes])
```

2 Experiments

2.1 Track a moving hand with a uniform background

2.1.1 Assume no motion

The tracker is tested with the default parameters using 30 particles, 16 bins for each channel, $\alpha = 0$, $\sigma\_observe = 0.1, \sigma\_p = 15$. The trajectory is shown in Figure 1 where the the hand disappears at about frame 36 and the tracker is able follow the hand stably until it disappears.

2.1.2 Assume constant velocity case

The tracker with the default settings can roughly follow the hand with slight shift over some frames, e.g. frame 36 in Figure 2(b) where the posteriori mean state is on the left side of the hand. So I further test with **more particles** (from 30, to 60 and 120). As shown in the Figure 2(d) and (f), using 60 and 120 particles can accurately locate the hand at frame 36, which indicates that more particles yield more stable trajectory and more accurate tracking results. And the discrepancy between the posteriori and priori mean state is diminished.

In particular, in both cases, the tracker seems to follow the center of the wrist, rather than the hand with fingers in the initial bounding box.

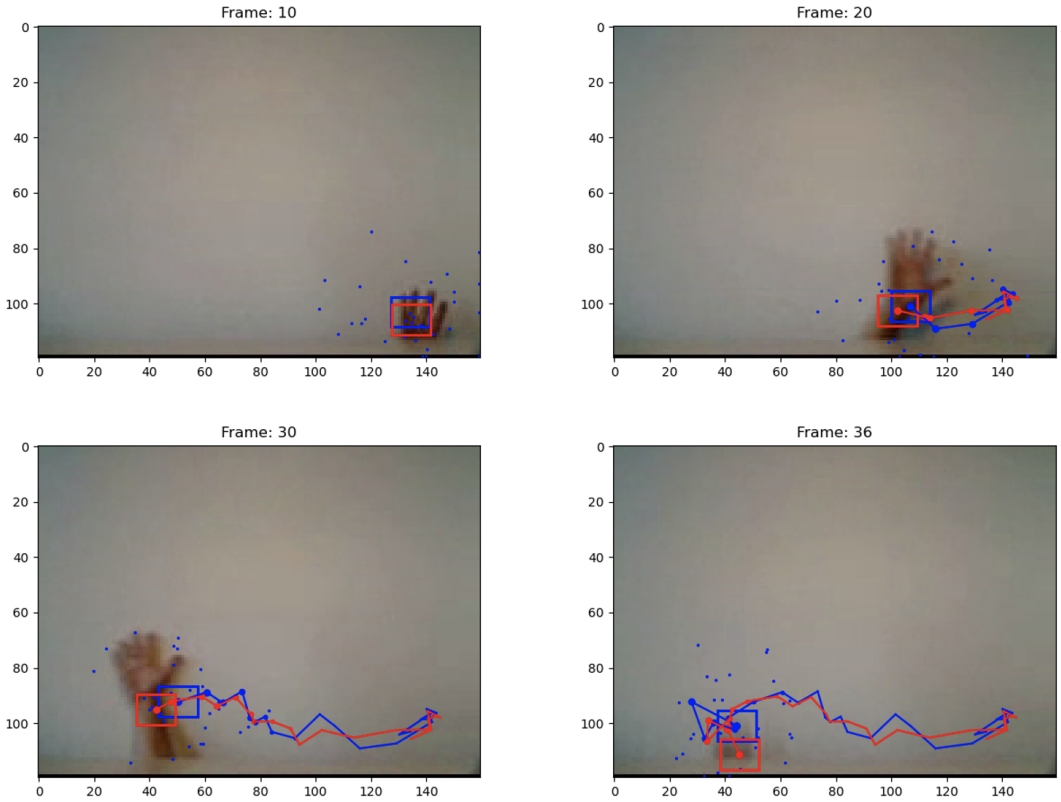

Figure 1: The trajectory of the tracking results on video 1 with no motion and default parameter setting: using 30 particles, $hist\_bin = 16$, $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$.

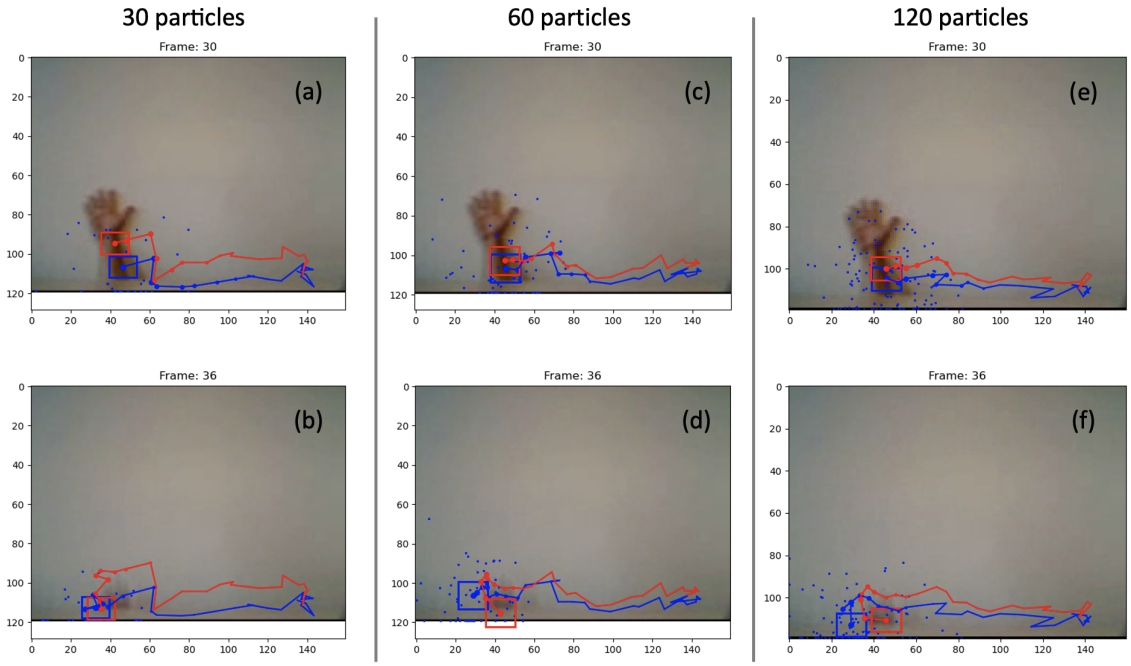

Figure 2: The trajectory of the tracking results on video 1 with constant velocity tested with different number of particles. Other parameters are set to be $hist\_bin = 16$, $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$, $\sigma\_v = 1$, $initial\_velocity = (1, 10)$.

2.2 Track a moving hand with some clutter and occlusions

2.2.1 Assume no motion

The tracking results with the default parameters are shown in Figure 3, where the trajectory is trapped around the box and fails to track the hand when approaching or being occluded by the box.

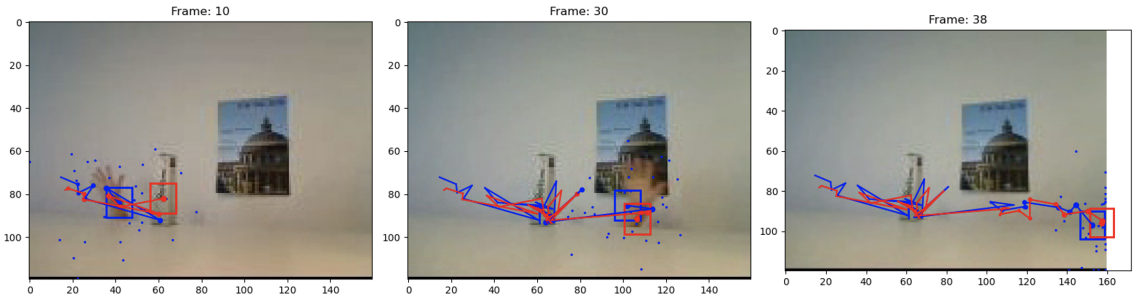

Figure 3: The trajectory of the tracking results on video 2 with no motion tested with 30 particles, $hist\_bin = 16$, $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$.

Test the effect of particle numbers. To alleviate the zigzag fluctuation due to occlusion, I increased the number of particles to 60 and 120, and the results are shown in Figure 4. With more particles, the occlusion issue is obviously improved, the bounding box is roughly around the hand when being occluded. From the figure, **60 particles** are enough to generate better tracking results (more accurately locate the hand e.g. in frame 30) and relatively smooth trajectory.

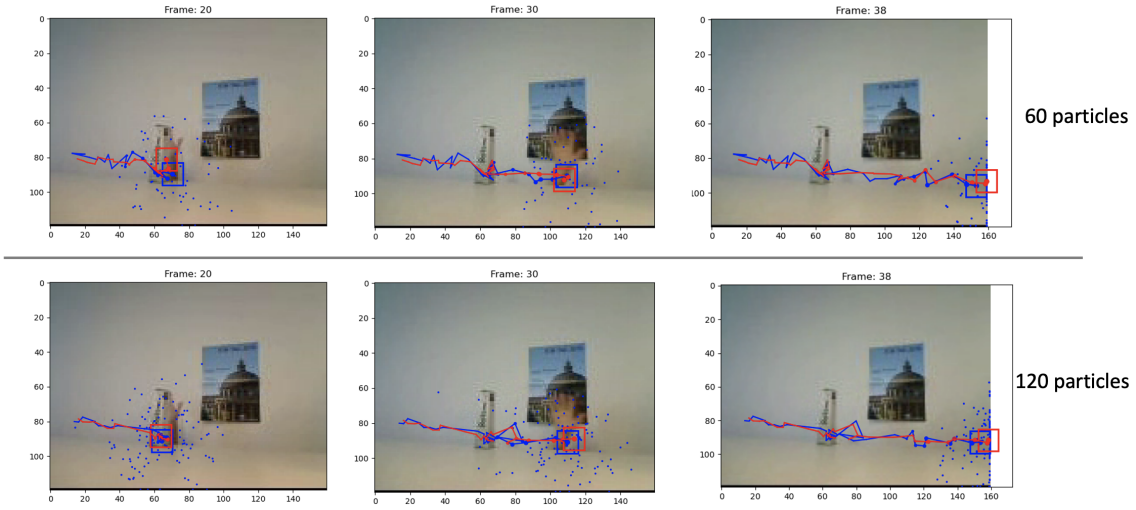

Figure 4: The trajectory of the tracking results on video 2 with no motion tested with 60 and 120 particles, $hist\_bin = 16$, $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$.

Test the effect of number of bins. For further improvement, I tested the same model with different number of bins (4, 8, and 64) using 60 particles. As show in Figure 5, when number of bins decreases, the trajectory tend to be smoother around the box, the reason might be that the colors in this video are not diverse, so that small bins are enough to capture the color changes. But too small bins (=4) might hurt the ability to distinguish the object from the background (like the fluctuation towards the picture on the wall). Larger number of bins would result in sparse histograms that might be too sensitive to change of background with large χ^2 distance. In this case, **8 bins** works best.

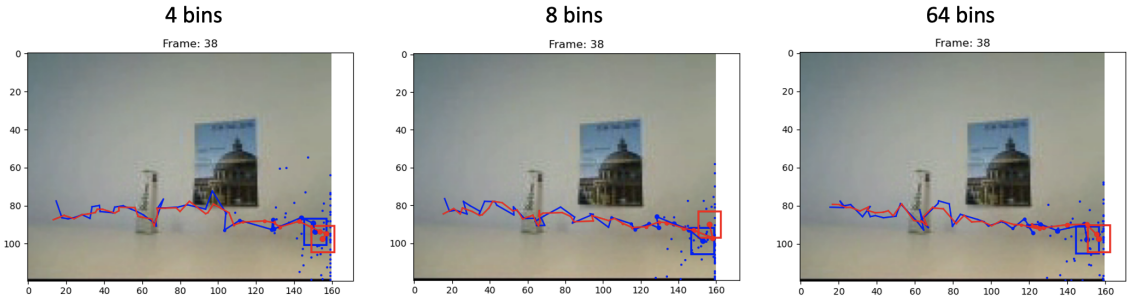

Figure 5: The trajectory of the tracking results on video 2 with no motion tested with different number of bins, and 60 particles, $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$.

2.2.2 Assume constant velocity

Test the effect of the velocity model. In this model, I fix the number of particles to be 60, and number of bins to be 8 to test other parameters. As shown in Figure 6, the posterori mean state could roughly follow the hand, but lost the target around the box again. Compared with the no motion model, the discrepancy between the posteriori and the priori mean state increases a lot, which indicates that the movement of each particle between frames is larger so that the color histogram of the bounding box differs more from the previous step.

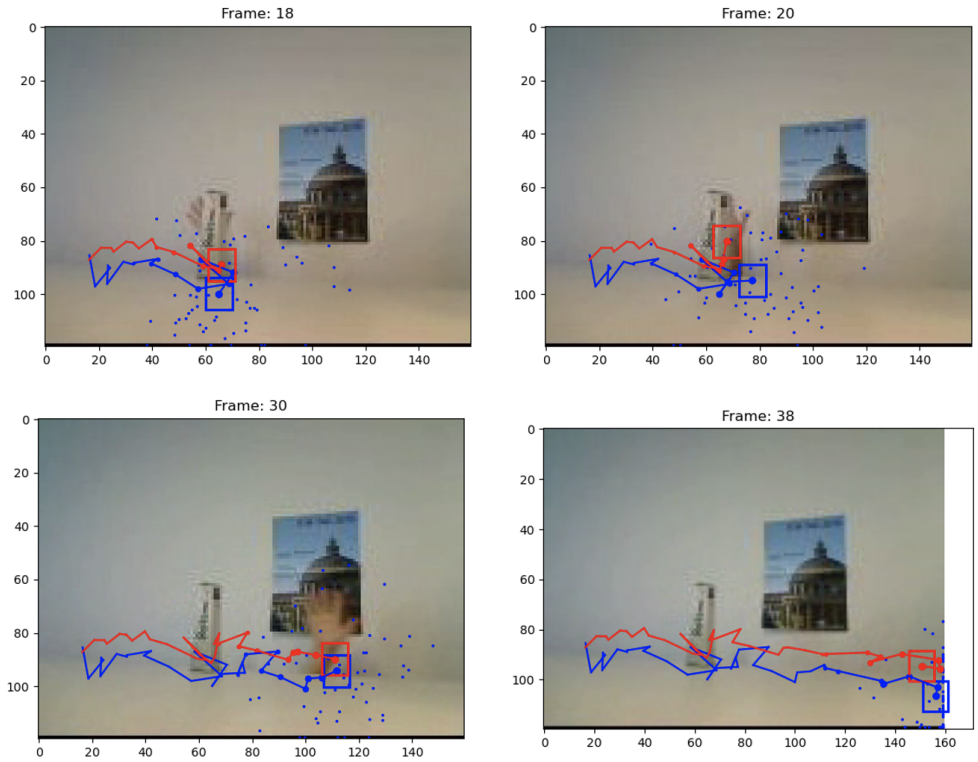

Figure 6: The trajectory of the tracking results on video 2 with constant velocity using 60 particles, 8 bins, and default parameters: $\alpha = 0$, $\sigma\_observe = 0.1$, $\sigma\_p = 15$, $\sigma\_v = 1$, $initial\_velocity = (1, 10)$.

Test the effect of the system noise. In this part, I test smaller and larger system noises on both position (σ_p) and the velocity (σ_v). From Figure 7, as noises becomes larger, the trajectory is more unstable and fluctuates dramatically. But when noises become too small, the movement of the particles might become too small to capture the moving hand. As in the $\sigma\_p = 5$ and $\sigma\_v = 0.1$ case, the trajectory trapped around the box and loses the hand

afterwards. In this case, $\sigma\_p = 10$ and $\sigma\_v = 0.5$ works best in that it follows the hand with relatively stable movement.

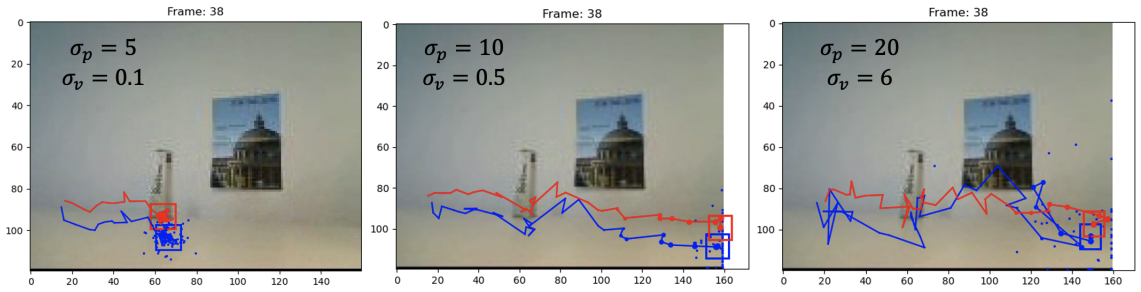

Figure 7: The trajectory of the tracking results on video 2 with constant velocity using 60 particles, 8 bins, and tested with different system noise. Other parameters: $\alpha = 0$, $\sigma\_observe = 0.1$, $initial\_velocity = (1, 10)$.

Test the effect of the measurement noise. In this part, I further fix the system noise to be $\sigma\_p = 10$ and $\sigma\_v = 0.5$, and test settings of the measurement noise $\sigma\_observe$, as shown in Figure 8. As measurement noise increases to 0.5, the Gaussian w.r.t. the χ^2 distance is more flat, meaning that the weights over all the particles tend to be uniform. In this case, resample and update tend to ignore the most information from the χ^2 distance, causing he trajectory quickly lose the target. As measurement noise decreases, the weights of particles center more around small χ^2 distance, reducing the variance in the new sample set. As shown in the $\sigma\_observe = 0.05$ case, the trajectory tends to fluctuate more and susceptible to the background picture. $\sigma\_observe = 0.2$ works best in that it follows the hand and the trajectory is robust against the occlusion and the background picture.

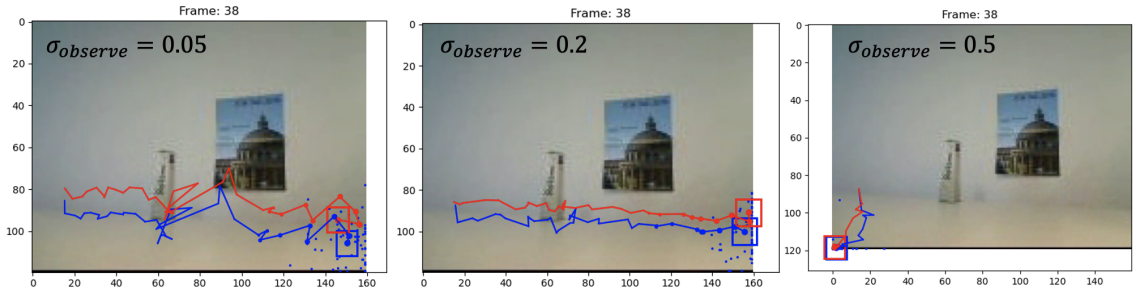

Figure 8: The trajectory of the tracking results on video 2 with constant velocity using 60 particles, 8 bins, $\sigma\_p = 10$ and $\sigma\_v = 0.5$, and tested with different $\sigma\_observe$. Other parameters: $\alpha = 0$, $initial\_velocity = (1, 10)$.

2.3 Track a a bouncing ball

Using the best parameters tested in video 2 with constant velocity, the tracking result is shown in Figure 9 where the posteriori mean state is able to stably track the ball moving forward and also backward. The reason might be that the ball is clearly separable from the background (almost white and grey), so even though the prior mean is away from the ball, but those small portion of particles around the ball can be updated with much higher weights according to their chi^2 distance, and thus dragging the posteriori mean close to the ball.

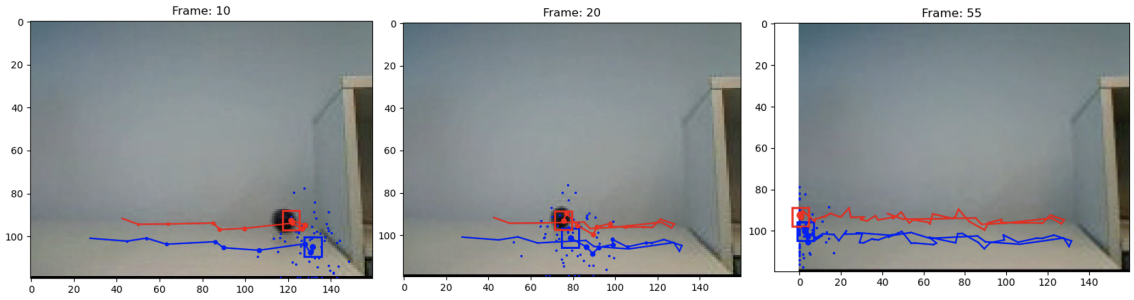

Figure 9: The trajectory of the tracking results on video 3 with constant velocity using 60 particles, 8 bins, $\sigma\_p = 10$ and $\sigma\_v = 0.5$, $\sigma\_observe = 0.2$, $\alpha = 0$, $initial\_velocity = (1, 10)$.

Test the effect of the velocity model. Keeping all the parameters the same, I test the model with no motion. As shown in Figure 10, similar to the analysis for video 2, the model with no motion can still track the ball with closed prior and posterior mean state.

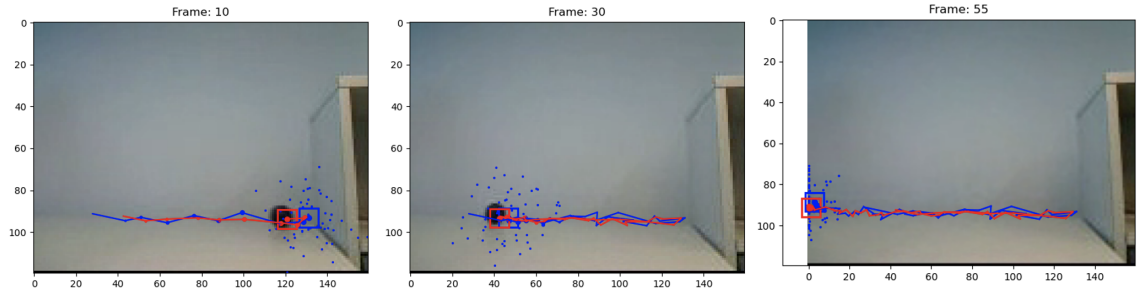

Figure 10: The trajectory of the tracking results on video 3 with no motion using 60 particles, 8 bins, $\sigma\_p = 10$, $\sigma\_observe = 0.2$, $\alpha = 0$.

Test the effect of the system noise. In this part, I fix all other parameters with constant velocity as in Figure 9, and vary only the system noise. From figure 11, when system noise decreases ($\sigma\_p = 5$, $\sigma\_v = 0.1$), the ball moves faster than the particles, so that the trajectory quickly loses the target when it moves forward. When system noise increases ($\sigma\_p = 15$, $\sigma\_v = 1$), the movement of the particles is too large, so that the particles could barely capture the ball when it moves backward and all the particles are trapped in the rightmost corner.

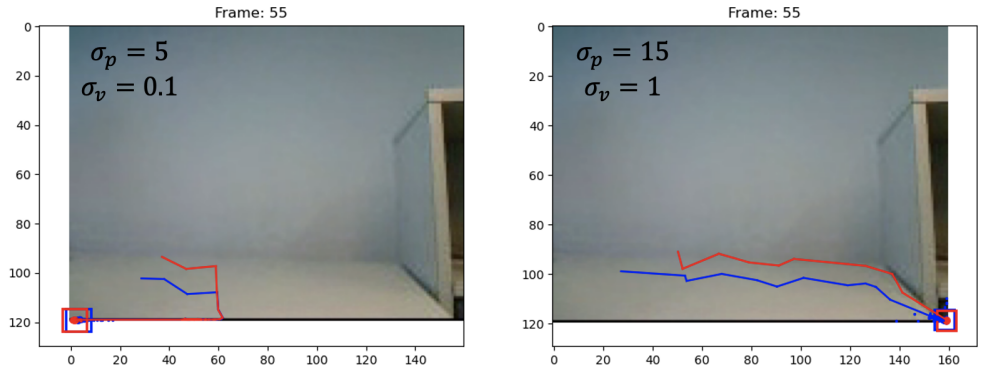

Figure 11: The trajectory of the tracking results on video 3 with constant velocity using different system noise. Other parameters: 60 particles, 8 bins, $\sigma\_observe = 0.2$, $\alpha = 0$, $initial\_velocity = (1, 10)$.

Test the effect of the measurement noise. In this part, I fix all the parameters from Figure 9 and vary only the measurement noise. As shown in Figure 12, when the noise increases to 0.5,

the tracker fails to track the ball when it bounces back, similar to the analysis for video 2, with large noise, the sampler became insensitive to the χ^2 distance and thus might lose the target.

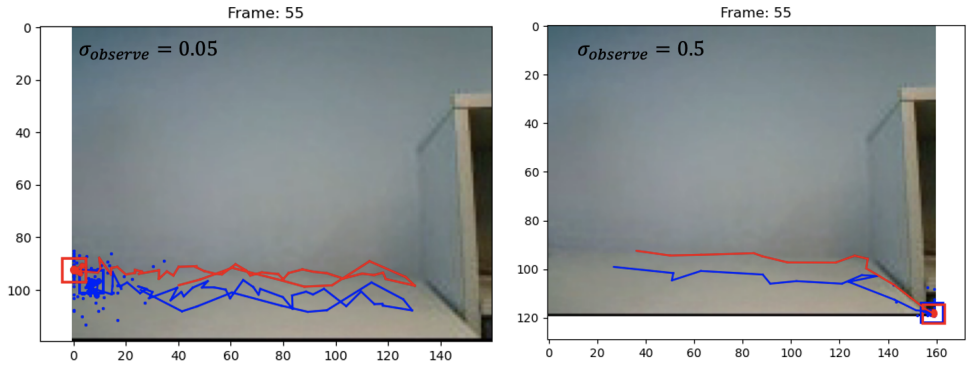

Figure 12: The trajectory of the tracking results on video 3 with constant velocity using different measurement noise. Other parameters: 60 particles, 8 bins, $\sigma\_p = 10$, $\sigma\_v = 0.5$, $\alpha = 0$, $initial\_velocity = (1, 10)$.

Test the effect of α. In all the previous models, the α is set to be 0, meaning that the target histogram will always be the initial one without updating. In this part, I test the effect of tuning α using video 2 and video 3 as examples. All the other parameters are the same as the best parameters tested for video 2: 60 particles, 8 bins, $\sigma\_p = 10$, $\sigma\_v = 0.5$, $\alpha = 0$, $initial\_velocity = (1, 10)$. The results are shown in Figure 13.

When $\alpha = 1$, the target histogram will depend only on the current posterior mean, in both videos, the tracker loses the target halfway, in this case, if the mean state wanders away from the target, there is almost no way for correction.

When α is between 0 and 1, e.g. $= 0.5$, the target histogram will depend both on the current posterior mean and the original one. Allowing such update is beneficial especially when the lighting or surrounding vary a lot along the way. From Figure 13 we can see that for video 2, the tracking trajectory is smoother compared with $\alpha = 0$. But for video 3, there is still a risk of losing the target (towards the end of videos).

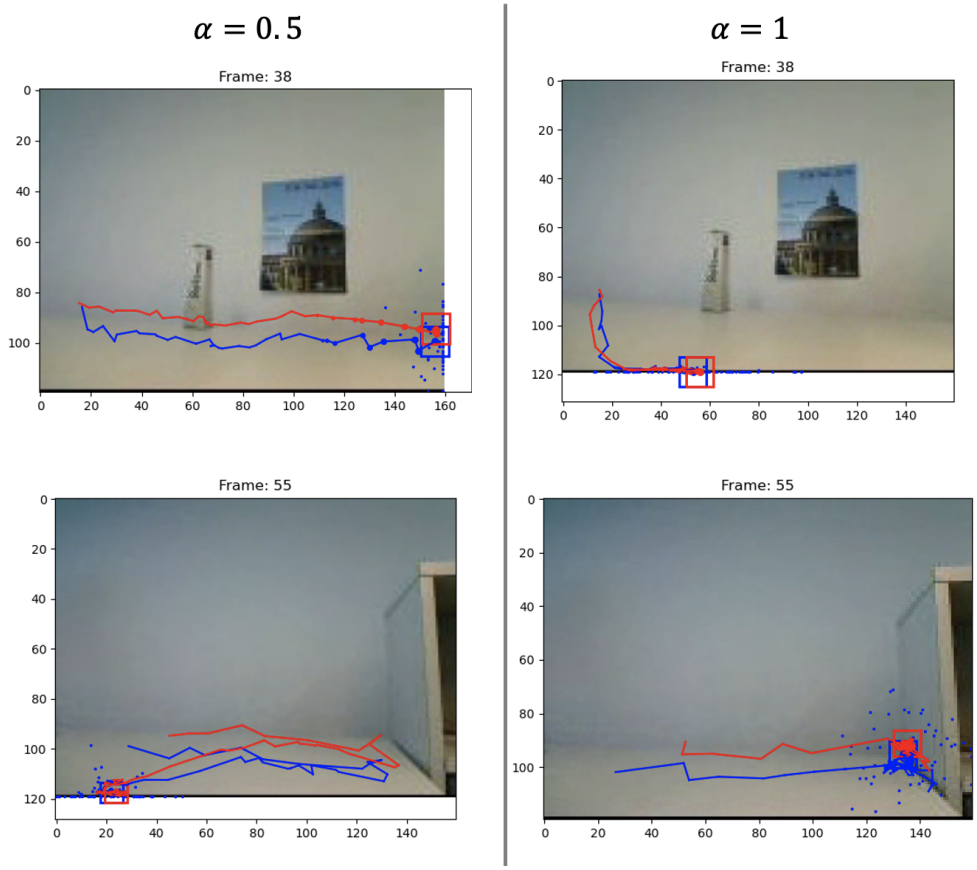

Figure 13: The trajectory of the tracking results on video 2 and video 3 with constant velocity using different α. Other parameters: 60 particles, 8 bins, $\sigma\_p = 10$, $\sigma\_v = 0.5$, $\sigma\_observe$, $initial\_velocity = (1, 10)$.