Computer Vision
Lab 11
Tianqi Wang

Condensation Tracker

1. Implementation

1.1 Color Histograms

This function aims to calculate the normalised histogram of RGB colours occurring within the bounding box selected. It is achieved by first selecting the pixels within the bounding box and then apply imhist() function to get the histogram for each RGB chanel. Then we concat these histograms together and normalise the result.

1.2 Derive matrix A

When there's no motion, the matrix A should simply be the identity matrix, since we want (x, y) to be the same and not affected by the velocity. When there's constant motion, we want (x, y) to be $(x + \dot{x}dt, y + \dot{y}dt)$. In our case, dt should be 1 and thus this gives us A = [1 0 1 0; 0 1 0 1; 0 0 1 0; 0 0 0 1].

1.3 Propagation(Prediction)

This function propagates the particles given the system prediction model and the system noise. The formula used is as below:

$$s_t^{(n)} = As_{t-1}^{'(n)} + w_{t-1}^{(n)}$$

Matrix A is derived from last step, and W is simply generated by normrnd() function, which generates random numbers from gaussian distribution given mean and variance.

1.4 Observe(Update)

This function aims to update the weight for particles, according to the distance of their colour histogram to the target histogram.

For each particle, we first computes its color histogram, which describes the bounding box defined by the center of the particle and W and H. Then we compute the chi-square distance between the colour histogram and the target histogram, and then update the weights according to this distance using the equation below.

$$\pi^{(n)} = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{\chi^2(CH_{s(n)}, CH_{target})^2}{2\sigma^2}}$$

1.5 Estimation

This function estimates the mean state given the particle and their weights, using the equation $E[s_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}$. Then the target histogram is updated as a convex combination of the old colour histogram and the colour histogram of the mean state.

1.6 Resampling

With this function we would like to resample the particles with replacements based on their weights.The resampling is simply done with randsample() function in Matlab.

2. Experimentation

2.1 video 1

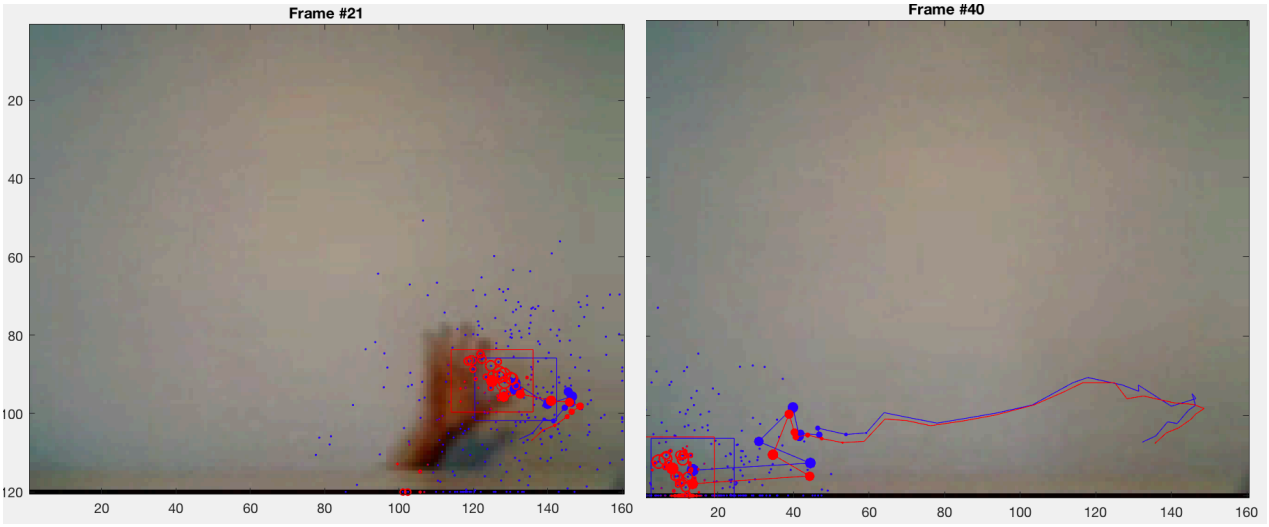

Figure 1: video 1, non-motion model, default parameters

The hand in this video is easy to track, and from the above plots, we can see that it has been tracked relatively well. However, it appears that in the later frames, the arm rather than the hand is tracked. The reason might be that due to the lighting condition, the colour of the hand changes but the arm becomes more similar to the initial colour of hand with time.

2.2 video 2

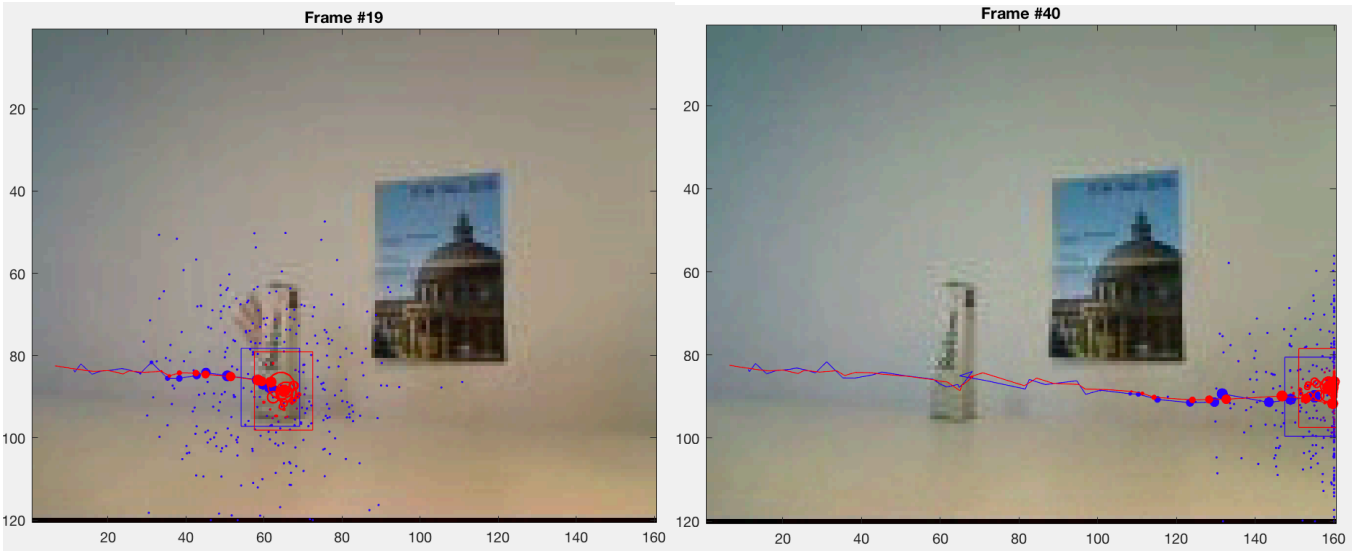

Figure 2: video 2, non-motion model, default parameters

From the above plot we can see that the hand can be tracked very well even with occlusions and clutter.

If we decrease the system noise, we can see from the figure 3 that the condensation tracker with non-motion model does not work at all. The reason might be this: if the system noise is large, in the update step, the particles could span in a relatively wide range and some of them might still be in the right position, even if when it is occluded. However, if the system noise is too small, we will possibly lose the object.

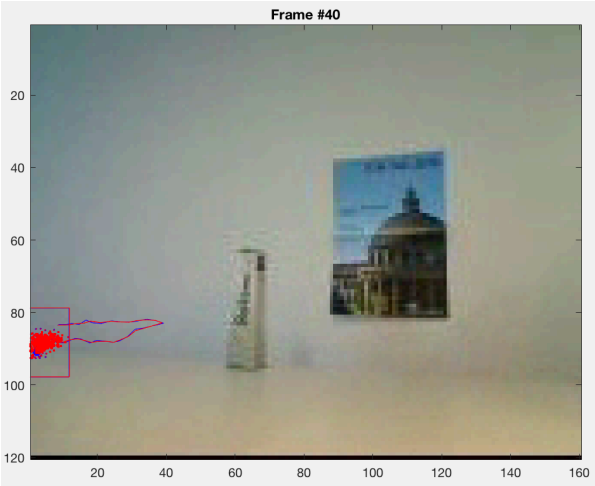

Figure 3: video2 , non-motion model, system position noise = 1

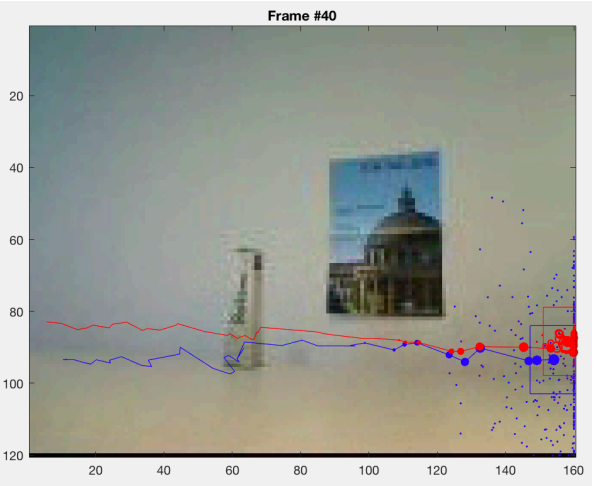

Figure 4: video 2, motion model, default parameters

Next, we check the constant motion model. From figure 4 we can see that the constant motion model is successful. By assuming constant velocity we can move our prior estimation in the direction of the motion, without having to increase the system noise. This can be particularly useful when the object is occluded.

Now we consider the effect of measurement noise. When the measurement noise is large, the weights are widespread and the importance can be assigned to some irrelevant particles, and this will cause the tracking to fail when it becomes too large, as shown in the picture below. But the pro of a moderately large measurement noise can be increased smoothness.

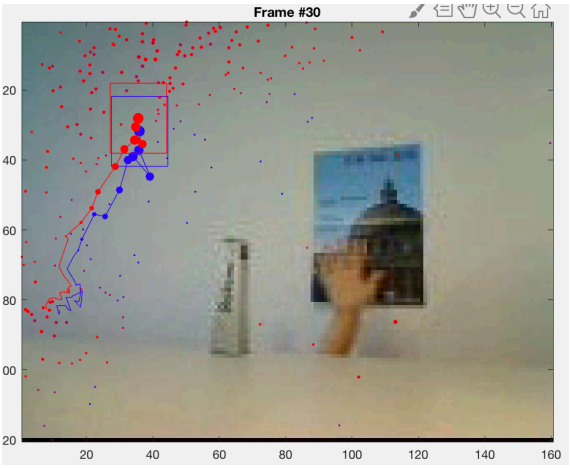

Figure 5: video2, non-motion model, measurement noise = 0.5

2.3 video 3

Now we try the bouncing ball example with the same parameter as before. As we see from figure 6 and 7, this same configuration doesn't work for the bouncing ball example, as the velocity and position of the ball changes drastically.

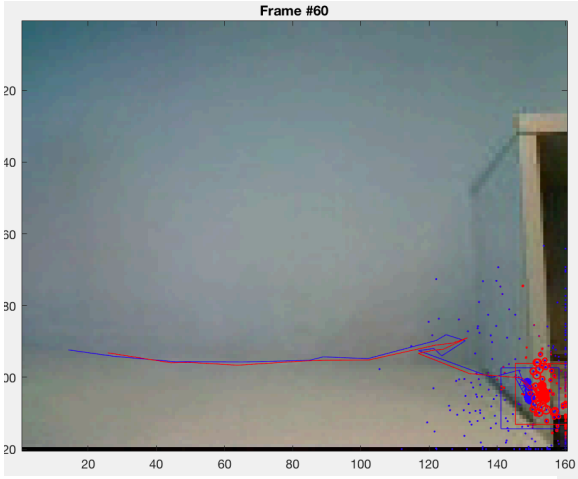

Figure 6: video3, non-motion model with default configuration

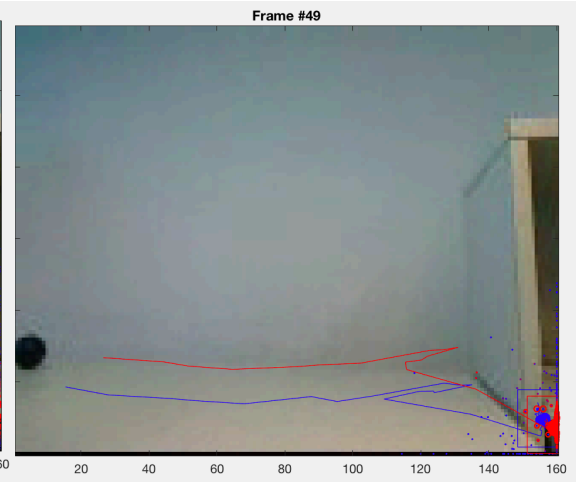

Figure 7: video3, motion model with default configuration

To make it work, we first try to increase the system noise of position and velocity. From figure 8 we can see that it works a bit better.

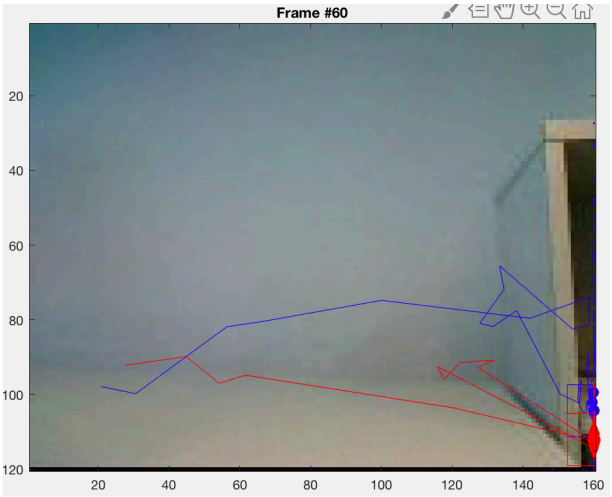

Figure 8: motion mode, system noise of velocity = 10, system noise of position = 30

Here we also try to explore other parameters. Firstly, we try more particles. With more particles, the tracking becomes more stable. From figure 9, we can see that the tracking is successful. But the downside would be that computation takes longer.

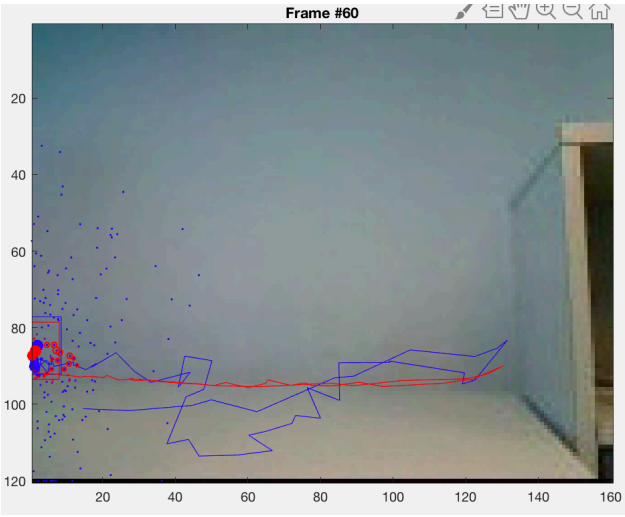

Figure 9: motion model, particles = 1000

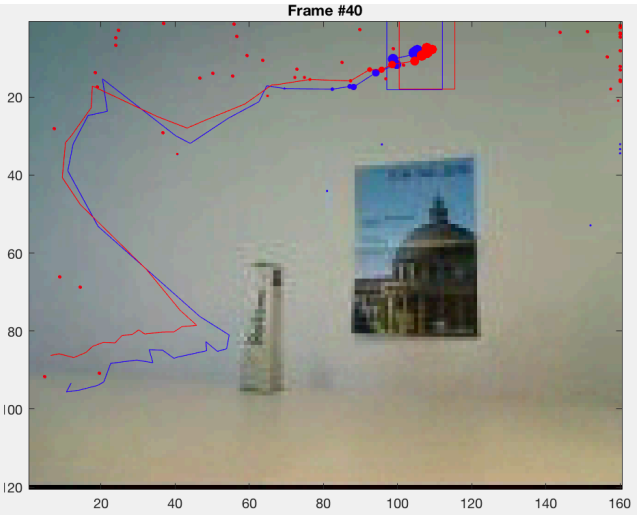

Figure 10: video 2, motion mode, 5 bins

Then, we change the number of bins. With an increased number of bins, the histogram will more more precise. However, this precision will cause problems since it will make the difference between two similar histograms appears to be larger. On the contrary, if the number of bins becomes too small, then all histograms will appear to be similar and the tracking will be very inaccurate. From figure 10, we can see that with too few bins the tracking is not successful at all.

Finally we change α, which is the weight assigned to the old target histogram with updating the target histogram. When α is increased, the histogram will incorporate more knowledge of the updated states. If the colour(position, etc.) of our tracked object changes, increasing α can actually help, as shown in the figure below.

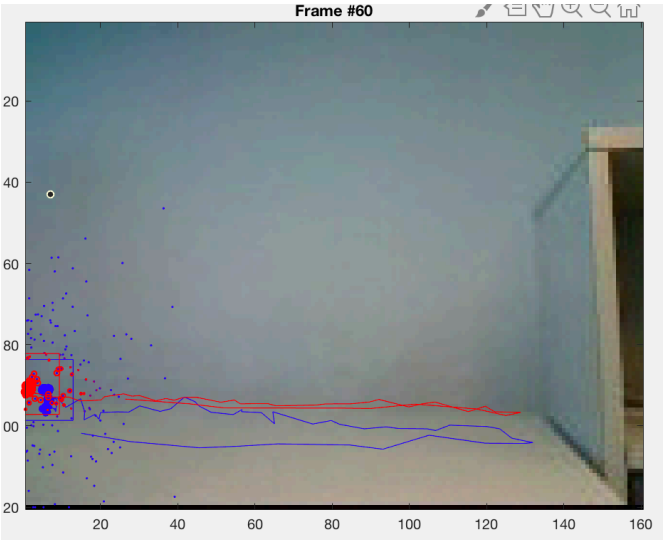

Figure 11: motion model, alpha = 0.5