Computer Vision

Condensation Tracker

Name: hanxue Liang

Student ID: 16-912-156

In this exercise, I implemented a condensation tracker algorithm based on color histograms and test it on videos.

1.1 Color Histogram

In order to get the normalized histogram, firstly, we need to check whether the bounding box is within the frame. If this is the case, we add one to the histogram matrix at the position x, y and z which correspond to the RGB values of the pixel we are currently checking. Then to normalize the histogram matrix, we will divide the matrix by the number of elements inside.

1.2 Derive matrix A

The dynamic matrix A transfer the old state into new state by the function:

$$s_t' = As_{t-1} + w_t$$

In the model with no motion at all, the state only have two attributes: their x- and y-position. Because it doesn't move, so A is identity:

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

For the model with constant velocity, the state has four attributes: the first two are positions and the last two are velocities. Since the speed is constant, the lower part of the matrix is identity. And the position should change over time, so we get the transfer matrix A as:

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

1.3 Propagate

For this part, firstly we need to identify whether we are dealing with a model without motion(model=0) or with constant velocity(model=1). Then we propagate particles using the linear stochastic differential equation $s_t' = As_{t-1} + w_{t-1}$. The stochastic element w is the system noise from normal distribution with $\sigma_{postion}$ and $\sigma_{velocity}$ as standard deviations. After propagation, we need to make sure that the particles still within the frame ad set it to nearest points on the edge if it doesn't.

1.4 Observe

In this part, for every particle, we calculate its color histogram using the current particles as center of the bounding box. We will calculate the distance between its histogram and the target histogram with χ^2 function. The distance will be used to calculate the particle weight by equation (6) given in exercise sheet. Finally, the weights will be normalized to make their sum to be one.

1.5 Estimate

To estimate the mean weight, we simply compute the dot product of the particles and their weights. This will multiply each particle's attributes by their corresponding weight.

1.6 Resample

In this part, I simply use *randsample* function of Matlab to draw a random sample out of current particles based on their weight as a probability distribution. After I get the index of selected particles, I generate the new particle and particle weights matrixes from original ones. At the end, I normalize particle weights so that sum of them equals to one again

2.1 Experiments-Video1

For video 1, we try to track a moving hand on a uniform background,

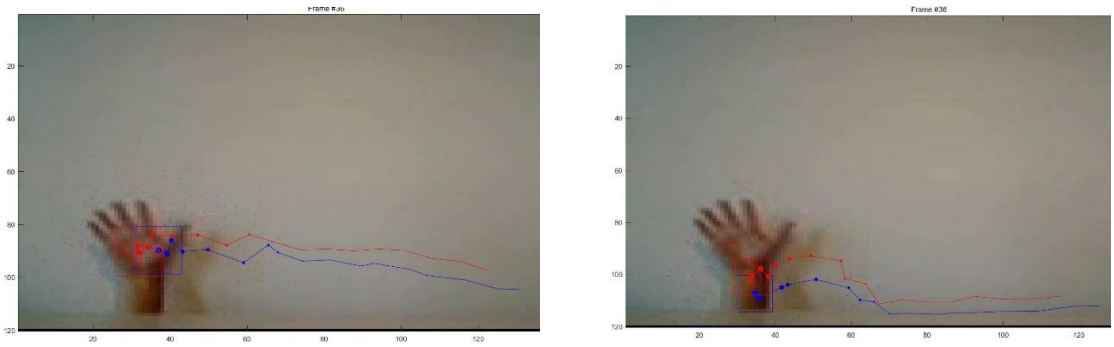

Figure 1: a. successful and b. unsuccessful tracking of hand in video1. The blue points are the a priori particle and the blue line their mean state. The red points are the a posteriori particles and the red line their mean state.

As we can see on figure 1, 1.a the tracking is relatively good as the particles follow the hand. However, in 1.b, the result is not good as the trackings are not always precisely on the hand but rather on the forearm. This is because when we first select hand on the first frame, the select box has a high contrast. And in the following frames, the forearm has a big contrast due to the shadow and light. While the performance can be improved by changing *params.alpha*.

2.2 Experiments-Video2

In this part, we track a moving hand with some clutter and occlusions.
Firstly, I use no motion model to track the hand.

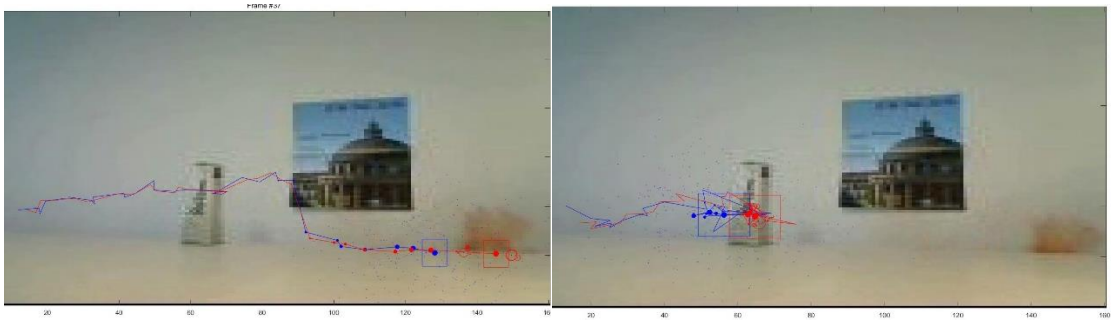

Figure 2: a: use no motion model 0 with $\sigma_{postion}$=15 b: use no motion model 0 with $\sigma_{postion}$=1
The blue points are the a priori particle and the blue line their mean state. The red points are the a posteriori particles and the red line their mean state.

Even though the hand is moving with a constant velocity and is occluded by an object at one point, the no motion model still manages to track through the entire sequence. While the tracking is not accurate after the occlusion(see Fig.2-a), it still can track the arm. During the update of the particles, the stochastic term is relatively big so the particles are spread on a big surface. This enables some particles to still be on the hand when it "reappears" from behind the obstacle and therefore the tracking can continue. However, when the system noise is decreased, the particles are much closer to each other, and the tracking does not succeed with the no motion model: they get stuck when

the object is hidden (see Fig.2-b).
Now, I use the constant velocity model and the result is shown below:

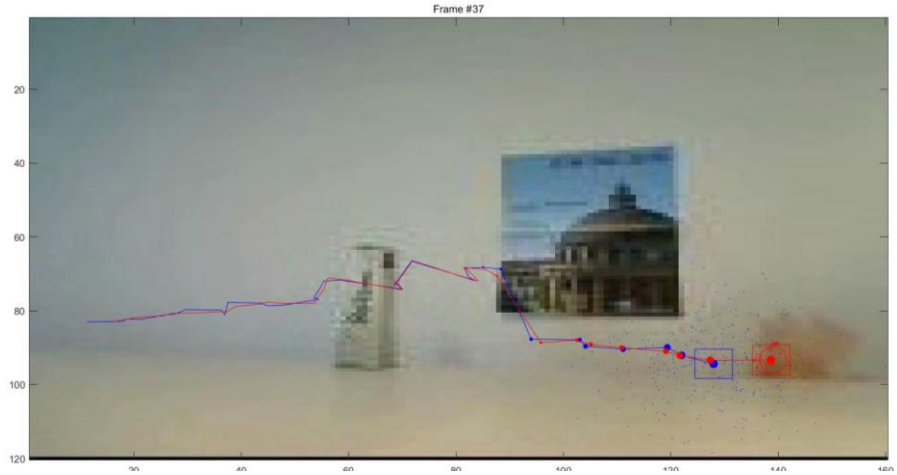

Figure 3: Tracking with clutter and occlusions with the constant velocity model. The blue points are the a priori particle and the blue line their mean state. The red points are the a posteriori particles and the red line their mean state. $\sigma_{postion}$=5 and the velocity is [6,0].

a) What is the effect of using a constant velocity motion model?
   By assuming constant motion we can move the particles into the direction of the motion, without increasing the uncertainty of the objects position. This is especially useful when an object becomes occluded since we have a guess on how and to where the object is moving.
b) What is the effect of assuming decreased/increased system noise?
   By increasing the system noise, we spread our a priori estimate more, this results in a more stable and smooth tracking compared to a decreased system noise. This can be found from our experiments based on no motion model as shown in figure2.
c) What is the effect of assuming decreased/increased measurement noise?
   Increasing the measurement noise increases the smoothness of the tracking since we are more likely to find the same location of the corresponding object.

2.3 Experiments-Video3

In this part, I try to track a bouncing ball with the same parameters as before. The main difference is that its direction changes.

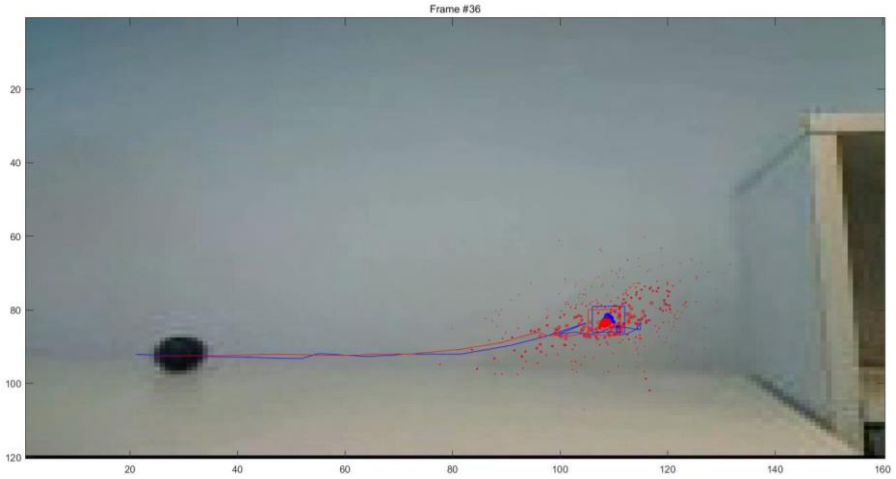

Figure 4: Tracking of a bouncing ball with same parameters as before. It uses constant velocity model with a speed to the right [6,0].

As we can see on Fig.4, when I use the same paramter as in video2, they do not work with the bouncing ball. Its direction changes drastically and the particles are all updated to its right and never find it back. In order to fix that, we can change several parameters.

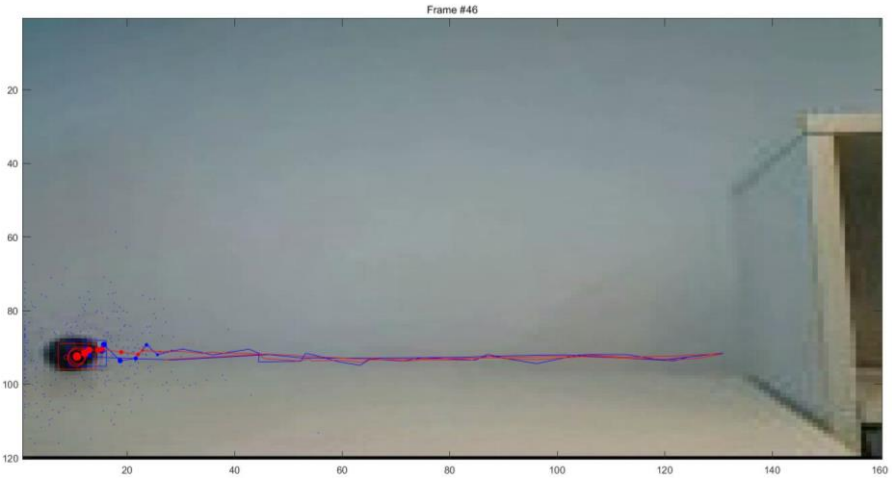

Figure 5: Tracking of a bouncing ball with same parameters as before except we change $\sigma_{postion} = 15$.

One strategy is to increase the $\sigma_{postion}$ so that there still are some particles in the ball region even when it bounces.We can see its effect in Fig.5.

Another strategy is improving $\sigma_{velocity}$ which allows the particles' velocity to change. In this case it worked but the results are not perfect as the velocity change is sometimes too drastic and too random.

a) What is the effect of using more or fewer particles?
   Using more particles gives us a better spread over a greater area, meaning we can cover more ground and are less likely to lose track of the object, using less particles causes the opposite. But more particles means more computational cost.
b) What is the effect of using more or fewer bins in the histogram color model?

The more bins we use, the more number of distinct colors we are able to detect, and the more delicate our target histogram will be. The advantage is that we can track a very distinct object but on the other hand, the algorithm is also more susceptible to color noise from the background.

c) What is the advantage/disadvantage of allowing appearance model updating?
The advantage is that we can track an object which is changing in appearance. In this sense, the disadvantage is also clear, because if the object is occluded by something similar, the tracker is likely to change its tracking object.