ETH ZURICH, D-INFK

Computer Vision Exercise 10: Condensation Tracker

Xingchang Huang

January 7, 2018

1 CONDENSATION TRACKER BASED ON COLOR HISTOGRAMS

1.1 INTRODUCTION

This part shows the main step I implemented for the CONDENSATION tracker. Color histogram is a feature descriptor for describing the image patch of the bounding box around the particles, so it is important when we need to re-weight the particles to see which are more important for tracking. Then we need to move the initial particles for tracking the hand using velocity and noises σ_v, σ_p, which mean the standard deviation of velocity and position. These two params are important for moving (or called propagation) to track the goal object. Each time we move the particles, we can use estimate function to visualize the mean state and evaluate it by drawing plots.

Next step is to re-weight the particles by measurement model (observation) after propagation, which is to add more weights to those particles can track the goal by comparing the histogram between the particles and target.

Then we can re-sample the particles according to the particles weights calculated in the previous step. After iterating these main three steps, we can finally get the particles that can track the object (e.g., hand) well and filter those far away particles. This is the main concept and explanation of all the steps of CONDENSATION algorithm.

1.2 COLOR HISTOGRAMS

Given an image patch (bounding box) around each particle, where each particle locates in the center of the bounding box, we can calculate the RGB color histogram just using bins from values 0-255. Thus, the histogram has size of $hist\_bin^3$.

1.3 DERIVE MATRIX A

Here we just need to define the matrix A as shown in the "Prediction" (propagation) step in CONDENSATION algorithm, which is used to sample new particles, or move particles to next position. As each particle has position and velocity, we just need a linear computation to add velocity of each time step to the original position and then we can get the new position and therefore new particles. So for state of particles $[x, y]^T$ and $[x, y, \hat{x}, \hat{y}]^T$, we need matrix A as the following, respectively.

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

A here represents no motion and constant velocity, respectively. In this way, As_{t-1}^n means $[x, y]^T$ and $[x + \hat{x}, y + \hat{y}]^T$ and the later part we can add standard normal distribution with variance to both of them for getting new particles.

1.4 PROPAGATION

As shown in the previous step, propagation can use the derived matrix A and normal distribution noise, with pre-defined σ_v (sigma velocity) and σ_p (sigma position) to propagate particles to get new particles. For implementation, I use a while loop to propagate the particles until all of them (e.g., 300) update the position inside the size of frame.

1.5 OBSERVATION

After propagation, you have to evaluate which particles track the object well. Thus, I use color histogram again to check which particles' color histogram of bounding box can match the bounding box the user defined at the beginning and the updated one well using χ^2 cost. The updated bounding box is calculated by a convex combination that combine the previous target and the current estimated histogram of mean state. Hence the param α can be adjusted in different videos. Then the cost is fed into a normal distribution, whose probability would be smaller for larger cost. Update the weights of particles starting from uniform one using the calculated probability and then normalize the weights to get a new distribution.

1.6 ESTIMATION

Briefly, estimation step is to calculate the mean state or the mean of x, y position and velocity. What we need to do is to calculate the mean of these two or four values among all the particles, which is used to draw plots later to show the performance of this tracking method.

1.7 RESAMPLING

The last step is to re-sample from the new particles according to their calculated weights. The randsample function with replacement in MATLAB can easily do this thing, which may sample the same particles during sampling process. And again, we need to normalize the weights after sampling and the number of particles is still the same as the one before re-sampling. The difference is that some particles with small weights may be discarded and some particles may be sampled multiple times. As you can see later in the plots, the red dots represent the posteriori mean state, which would be more accurate for tracking than the blue ones.

2 EXPERIMENTS

After implementation, we then go to the part of experiments. There are totally 3 videos for tracking hand and ball under different settings of background.

Video 1. The first video is to track a hand in a uniform background, which made this task easier. At first we need to draw a bounding box for this hand. The main problem of the first video is that the initial hand is only half of the whole one so the parameter α is quite important in this video. If we set α to be 0, the target color histogram (CH) would only contain half shape of the hand, which can bring mismatch when we need to track the whole hand and I found that $\alpha = 1$ is much better because it utilized the color histogram of later frames. The parameters like hist_bin, number of particles, sigma_position are set to be the default one but I set the sigma_observe to be large (e.g., 3 with no normalization) because it is quite essential to update the particles' weights. With smaller sigma, it is easier to filter the particles with smaller weights and concentrate on the particles with larger weights but at the same time smaller sigma may result in 0s weights. Also, I found that the initial bound box should be carefully drawn that contain the hand and less background. The tracking performance can be shown in Figure 2.1. More details about the parameters are shown in the code. As we

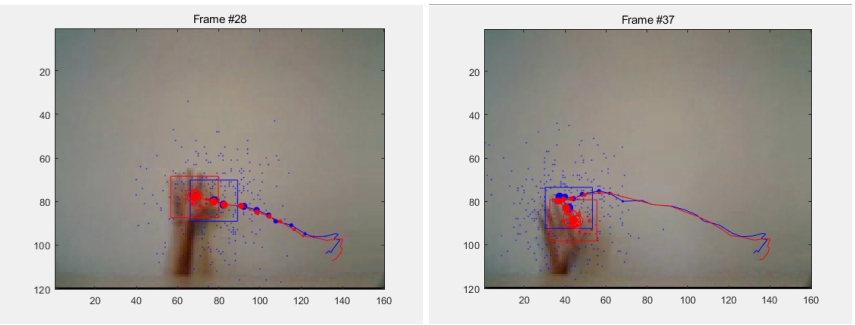

Figure 2.1: Tracking a hand using no motion model.

track the hand based on color, it is possible that we check the arm or finger tips rather than the hand. But later this problem may be modified during hand moving.

Notice that we can use model 0 (no motion) for video 1 because we only track the hand based on the color histogram under the uniform background. However, for the next video 2, we can not only rely on color histogram for the case of occlusion that would make the particles fail to track the real hand. In such case, we need to move the particles with velocity or momentum to avoid the lost of hand and its color histogram.

Video 2. As discussed above, we set $\alpha = 0.1$ and use velocity model for tracking a hand in background with clutter and occlusion. The hist_bin is set to be default 16 again and I use a high velocity $x_v = 16$ in x direction and a small one $y_v = 1$ in y direction. The following results show that it works for these parameters after occlusion. Here we track the hand mainly based on the bounding box defined by user at the beginning and use velocity to ensure the particles move with momentum along the original direction and not missing the hand. Then we have

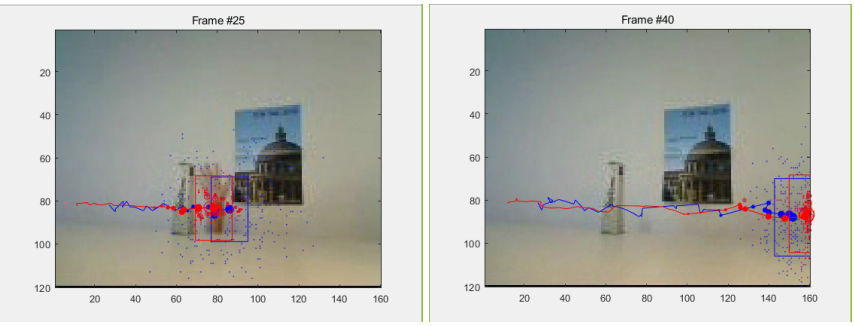

Figure 2.2: Tracking a hand in background with clutter and occlusion using velocity model.

three questions about the changing of parameters.

- What is the effect of using a constant velocity motion model? As discussed above, we use velocity to ensure the particles move with momentum along the original direction and not missing the hand. Without using it, the particles have the probability not to catch up with the moving hand after occlusion as shown in the following figure.

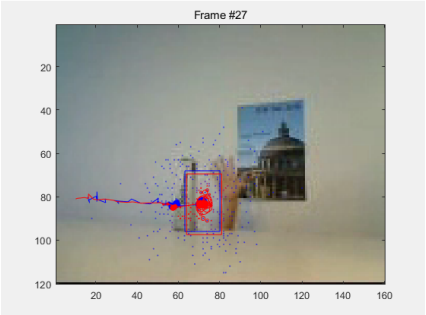

Figure 2.3: Fail to track the hand after occlusion using no motion model.

- What is the effect of assuming decreased/increased system noise? System noise includes position and velocity noise, which may influence the propagation of particles. The decreased or increase of the noise may lead to too slow and fast movement of particles, which directly decide the position of all particles, so I use the default setting for both of them.

- What is the effect of assuming decreased/increased measurement noise? The measurement noise includes observation noise. During the experiments, I found that the observation noise has great influence on the re-sample of particles, which is based on the calculated weights in the observation step. Too large observation noise may make the normal distribution flatter, which is not good to re-weights the particles. But too small one make normal distribution like a pulse, which may bring to 0s in most weights. Thus, I choose the observation noise to be 3 to ensure that the particles with small weights can be filtered and there is no problem of all 0s weights.

Video 3. This task is to track the ball. When I use the parameters tuned in video 2, I found that it fails after the ball bounce back from the wall. I assume that this problem may come from the velocity so that the particles may still go forward and cannot turn back in time. So I try to decrease the velocity along x direction from 16 to 8 and it works. Compared with video 2, I still suppose that the main problem should be the setting of velocity because I also use the default value for system noise. And here are the failure and success case for video 3 using two different velocity.

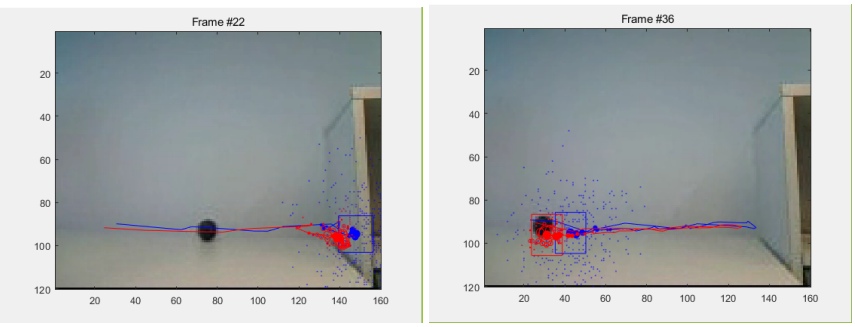

Figure 2.4: Tracking a ball with bouncing using velocity model, where $v_x = 16, 8$, respectively.

3 Questions Answering

Then we have three more questions for this experiments.

- What is the effect of using more or fewer particles? Obviously, fewer particles may result in not accurate result because it is possible that not enough particles can successfully track the goal while more particles can bring better accuracy but with more intensive computation. It is also possible that more particles may bias the tracking because some chunks of particles may concentrate on tracking another object with similar features.

- What is the effect of using more or fewer bins in the histogram color model? Too few bins may lead to faster tracking but the color histogram may not exactly describe the color feature. I have tried fewer bins in video 1 but the performance is worse than the case when hist_bin = 16. But more bins may also lead to intensive computation and we can not also ensure better performance with number of bins higher than 16.
- What is the advantage/disadvantage of allowing appearance model updating? The appearance model here means the color histogram, which is a convex combination of the estimated one and the previous one. Thus, I think the parameter α is quite an important parameter. For example, in the first video, I use $\alpha = 1$ because I suppose that the first frame has just half of the hand we need to track and we therefore need to update CH during movement. But in video 2, I changed it to 0.1 because the hand may disappear due to occlusion so the particles may lost. When you update the appearance model (color histogram), you can mostly rely on the first frame that contains the whole shape of hand, which makes it more accurate as shown in experiments of video 2.

4 BONUS

For my own video, I design a difficult scene where I need to track a key under the background with similar pattern and color (book and chess board), even with light source and the generated shadow. Besides the design of background, I also move the key not following the constant velocity or direction, which may challenge the constant velocity model a bit. Therefore, it is quite a difficult task. Before the experiments, I have compressed the video recorded by my cell phone and changed it to .avi format. The size of frame is 272×480 and the length of the video is 14s and I use the frames from 1 to 600 with step frame of 10.

I tried the velocity model by changing speed to 16 for both direction. Also, I increase the sigma observe in order to keep more particles and relieve the problem of fast, sudden move of the key. But the initial performance still shows that the particles lost when the key move irregularly (suddenly goes up or down), where the track of key is moving right, left, down and up.

The first two figures show that the initial tracking is good because of the smooth movement. However, the next two figures show the failure case when the key suddenly moves up with a little bit left and goes down with a little bit right, and the particles cannot move so fast to catch up with the key, where the constant velocity is to blame for. The velocity model can only ensure constant velocity in a certain direction so it doesn't work well in the case of sudden movement. The improvement here I think would be improve the constant velocity model to a variable velocity motion model and ensure that the particles can move to any direction instead of one specific one, which can theoretically track the hand moving to any direction and in any velocity.

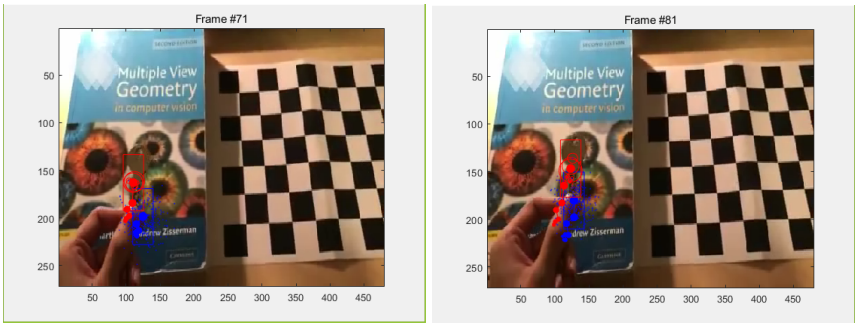

Figure 4.1: Track a key in the complex background using constant velocity model.

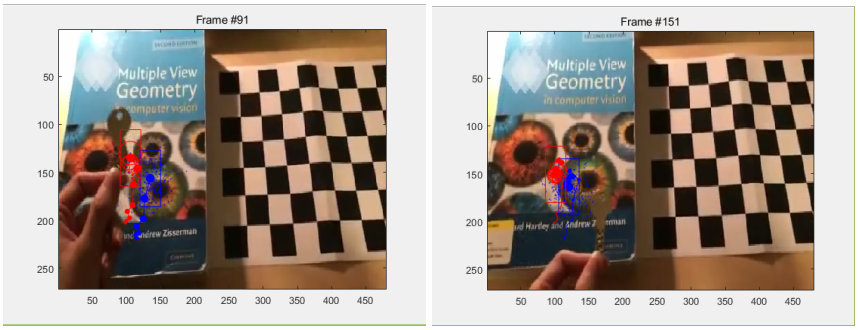

Figure 4.2: Fail to track the key with sudden and fast movement using constant velocity model.