Igor Martinelli: Assignment 6 Report

maigor@ethz.ch, 19.916.048.

1 CONDENSATION Tracker based on color histograms

1.1 Color histograms

The first step in the implementation of the CONDENSATION tracker was about computing the function color_histograms(), which returns the normalized color histograms of the pixels inside a specified bounding box (the bounding-boxes represent the objects to be tracked, and the initial one is drawn by the user). This was done through the Matlab function cv2.calcHist(), applied separately to the 3 channels of the bounding box (R, G and B values). The 3 obtained histograms are then stacked together in a single vector and then normalized with respect to their sum.

1.2 Derivate Matrix A

Let us consider first the case of the no-motion model. In this case, the state is given only by the position of the samples, namely $s = [x, y]^T$, and we take into account only the position noise $w = [\sigma_P, \sigma_P]^T$ on the two components x and y. Since we expect the particles not to move, matrix A is assumed to be the 2x2 identity matrix, and the system model is described as in the following:

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

In the case of the constant velocity model, instead, the state of each bounding-box is expressed by 4 components, namely the 2 components of the samples' position and the 2 components of their respective velocities, as in the following $s = [x, y, \dot{x}, \dot{y}]^T$. For the noise, we take into account two different terms respectively for the position and the velocity, having the following expression $w = [\sigma_P, \sigma_P, \sigma_V, \sigma_V]^T$. In this case, since we expect the particles to move with constant velocity, we have to set their velocity not to change throughout the evolution of the system (since it is constant), while we have to set their position to change in the following way:

$$x_{t+1} = x_t + \dot{x}_t dt + \text{noise, with } \dot{x}_t = \frac{dx_t}{dt} = v \tag{2}$$

Hence, the system model was described in the following way:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

In this specific case, I assumed $dt = 1$, since I consider a time step to be the time between two consecutive frames of the video. In this way, velocity is expressed in pixels per frame (PPF).

1.3 Propagate

After establishing the system model, the particles' movement was determined using the function propagate(). This function calculates the updated state of the particles based on the previously defined model. Initially, the particles and their corresponding weights were initialized (this part is already present in the provided code). Additionally, a verification was conducted to ensure that the particles' positions fall within the frame. If any particle was found outside the frame boundaries, I assigned the limit values of our frame, considering both height and width, to those particles.

1.4 Observation

observe() computes the weights for particles based on color histogram. At first, ensure the bounding box for each particle lies inside the image. Then for each particle, compute the color histogram of the bounding box with the center at particle. The similarity between the color histogram of particle and target color histogram is computed by χ^2 distance. Then the probability for each particle is obtained with the following formula:

$$\pi = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{\chi^2(s,target)^2}{2\sigma^2}} \tag{4}$$

Here $s, target$ means the color histogram of a certain particle and target.

1.5 Estimation

estimate() computes the eman states given the following formula:

$$E[s_t] = \sum \pi_t s_t \tag{5}$$

It means the mean state at a certain time is the weight average of all particles at this time (use weights from observation step).

1.6 Resampling

resample() tries to resample particles with replacement based on their weights from observation step. For resampled new particles, their weights are normalised. To summarize, those steps explained above can be used in the following sequence to form CONDENSATION tracker:

1. compute the histogram of the selected bounding box as target histogram
2. initialize particles
3. propagate particles and estimate the mean state right now
4. observe the weights for current particles and update mean state based on weights
5. update color histogram based on histogram of target and mean state
6. resample particles based on weights
7. repeat step 3-6 until the end of video

2 Experiments

The second portion of the task involved evaluating the CONDENSATION tracker that had been previously applied to three videos specified in the assignment instructions. The objective was to assess the tracker's performance by modifying different parameters. Upon running the condensationTracker() function, we observe a blue line (accompanied by blue particles) denoting the mean state of the priori particles, and a red line (with red particles) indicating the mean state of the posterior particles. Subsequently, I will analyze each of the three videos individually.

2.1 video1.wmv

Tracking with default parameters on the first video is somewhat good for both models, but not very accurate. In the first frame we set the box to be on the fingers of the hand, but the trajectory shows us that the tracking algorithm followed the wrist in the subsequent frames (see Figure 5).

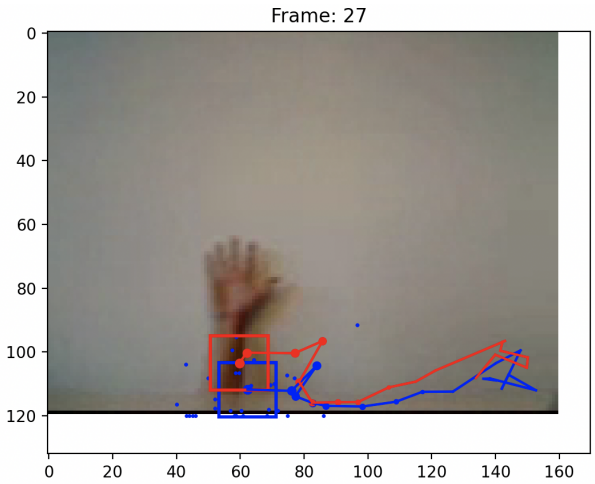

Figure 1: Trajectory half way through video 1, constant velocity model

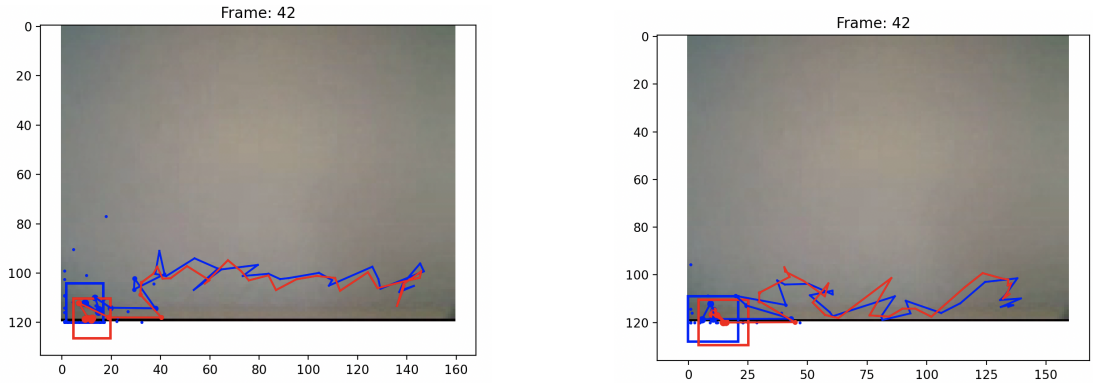

Figure 2: Trajectory of tracker applied to video1 with default parameters, with no-motion model (on the left) and constant velocity model (on the right).

This is may due to the fact that the illumination changes during the motion, thus when we first draw the bounding-box around the fingers, the area is more darkened than when the hand is moving. Rather, the shading of the wrist's bottom portion is quite comparable to that of the fingers in the first frame.
One way to improve the accuracy is by setting the initial velocity parameters to a negative x and y velocity, since the hand is moving up and to the left (the origin is on the top left corner). By setting the initial velocity to [-3,-12] we get better results, the model with constant velocity assumption being the most accurate one (see figures below).

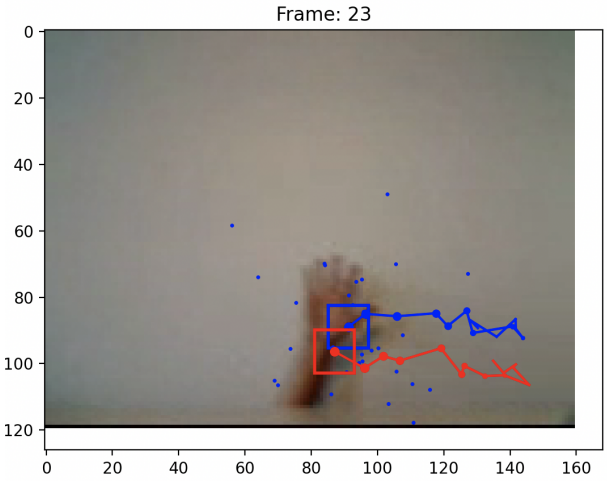

Figure 3: Trajectory half way through video 1, constant velocity model

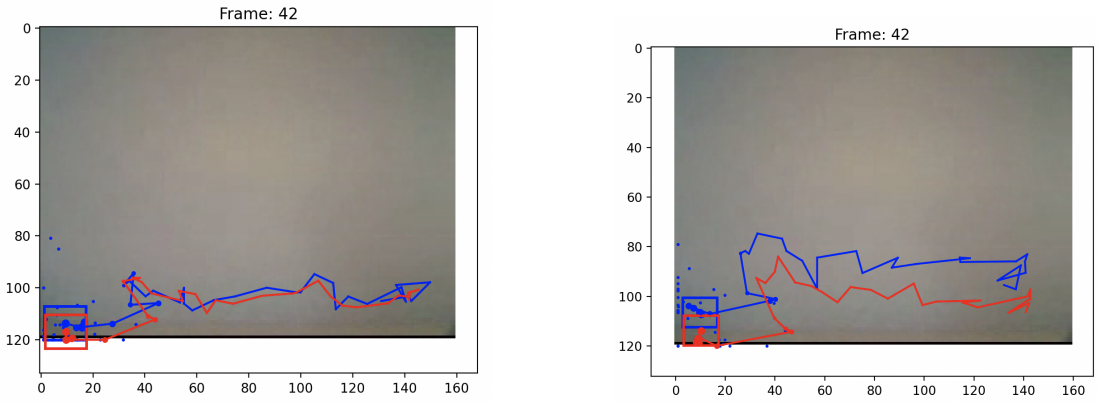

Figure 4: Trajectory of tracker applied to video1 with initial velocity set to [-3,-12], with no-motion model (on the left) and constant velocity model (on the right).

2.2 video2.wmv

2.2.1 Constant velocity model

Using the constant velocity model with default parameters to track the hand on video2 leads to the tracker getting stuck on the object in the middle of the frame once the hand passes behind it. Once the hand is past the object the tracker goes back to tracking the wrist.

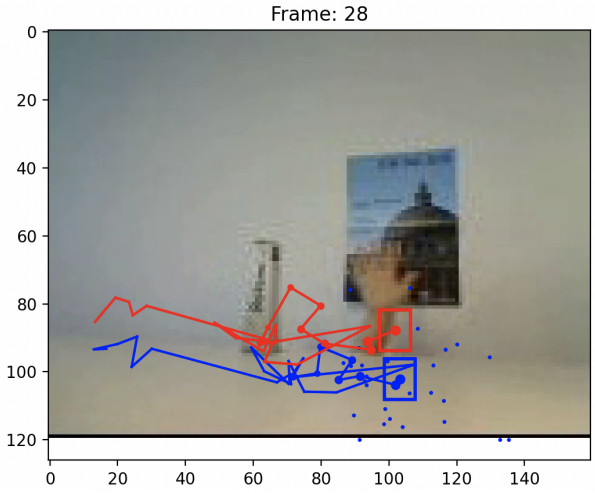

Figure 5: Trajectory half way through video 2, constant velocity model

2.2.2 System noise

If we assume low system noise, we notice that the tracker becomes more accurate in tracking the hand and doesn't get stuck on the object for too long (see left Figure 6). However, if we assume high system noise, when the hand gets near the object, the tracker always tries to track that instead of the hand (see right Figure 6).

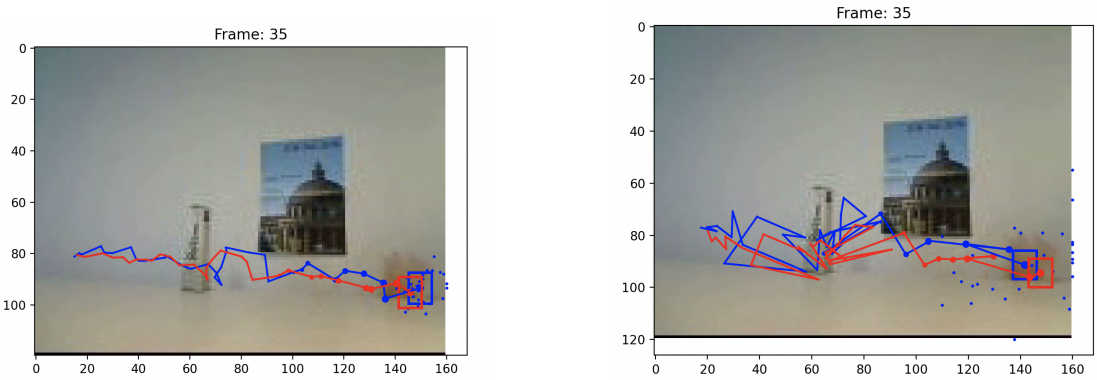

Figure 6: Trajectory of tracker applied to video2 with initial velocity set to [1,0], with constant velocity model with low system noise assumption(on the left) and high system noise assumption (on the right).

2.2.3 Measurement noise

The measurement noise was another critical parameter to modify because it determines how the weights are associated to the relevant particles. When the measurement noise is low, say $sigma = 0.01$, the tracking fails totally and the particles may disappear or follow very random pathways. This is because all of the particles have relatively little weights, therefore none of them are ideal candidates to follow. High quantities of measurement noise, on the other hand, provide the opposite consequence. In this scenario, all of the particles have high weights, implying that they are all suitable candidates for following. As a result, the tracker becomes entirely stuck and stops following the hand (see Figure 7).

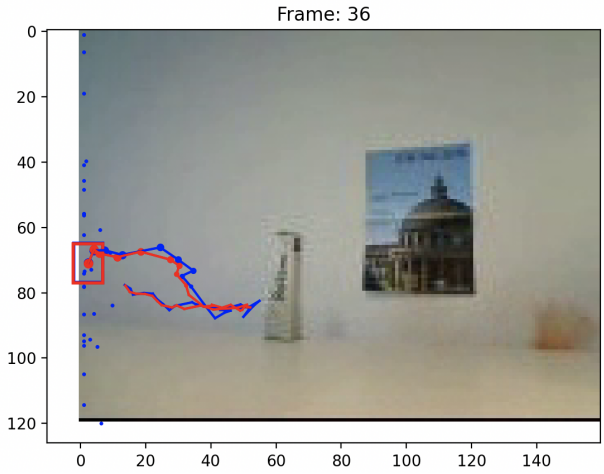

Figure 7: Full trajectory on video 2 assuming high measurement noise.

2.3 video3.wmv

The third video differs from the other two as it shows a ball bouncing on a wall instead of a moving hand. Thus, applying the parameters used for the previous videos, like Video 2, isn't entirely accurate due to the changes in motion and colors. This drawback of the

CONDENSATION tracker is significant because it relies heavily on the specific scene, always needing adjustments.

When attempting to use the best parameters from Video 2, I couldn't track the ball properly (see Figure 8). Either the particles stopped following the ball before it reached the wall, or after the wall impact, they continued updating to the right, ending up in the bottom right corner.

2.3.1 Constant velocity

In this case, the constant velocity model may not always be the best choice. When the ball hits the wall, it changes direction and slows down after the impact, meaning the velocity is no longer constant. During the impact, the particles kept updating to the right while the object moved in the opposite direction. Generally, for this third video, the preference was for the no-motion model.

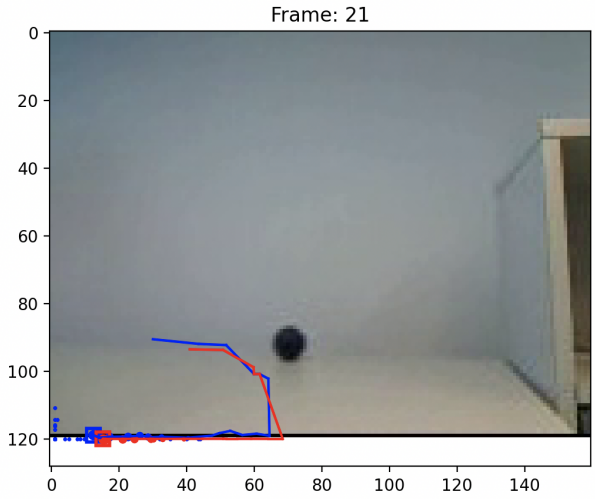

Figure 8: Full trajectory on video 3 with best parameters for video 2 with constant velocity model.

2.3.2 Noise

In this case assuming an high value for system noise solved the problem. In this way, being the particles spread on a wider area, they were able to trace back to the ball also after the collision with the wall (see Figure 9). As far as the measurement noise is concerned, the same reasoning for Video 2 applies also now.

2.3.3 Bins, particles and alpha

At the same time, having lots of particles really helps make the tracker more accurate. The number of bins is also important for tracking the ball. If you choose too few bins, the tracker might mess up, especially when the ball is black like the bottom right corner. Having a reasonable number of bins, like 15, works well.

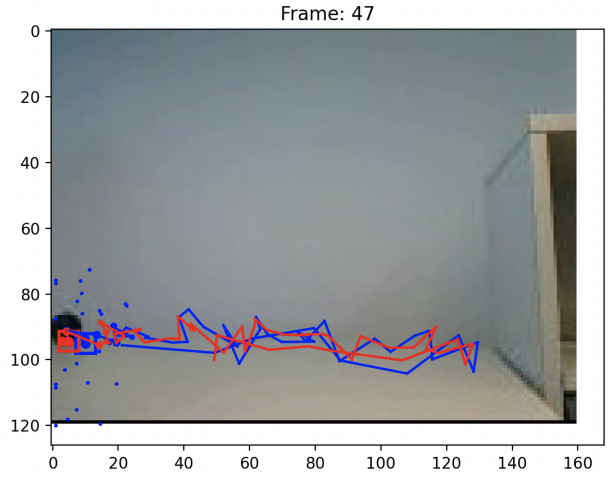

Figure 9: Full trajectory on video 3 with no-motion model and high system noise assumption.

For all three videos, a good balance is setting α to 0.5. This choice is like finding a middle ground between updating the target histogram a bit and not updating it too much. By letting the appearance model change over time, the target histogram adjusts to different situations. This makes sure the tracker stays on track and adapts to what's happening. So, when you fine-tune the bins, increase the particles, and pick the right alpha value, it all adds up to make the tracker work better (see Figure 10).

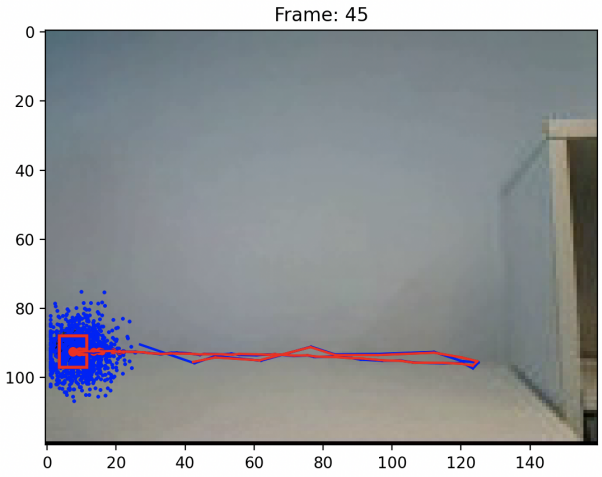

Figure 10: Full trajectory on video 3 with no-motion model, 1000 particles, 15 bins and α = 0.5.