Computer Vision Assignment 9

Luca Ebner (ebnerl@student.ethz.ch)

December 11, 2020

In this weeks assignment we program an algorithm that is able to track objects in an image by using a particle filter based on color histogram measurements.

In the following, the state of a particle is described as

$$s = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T \tag{1}$$

1. CONDENSATION Tracker Based On Color Histograms

In order to track particles between different frames, we need an additional quantity to compare them. In this exercise, we use color histograms as measurements and implement a CONDENSATION Tracker (CONDitional DENsity propagaTION over time). For this, the following code parts are needed:

- **Color Histograms**: We need a Matlab function that computes the color histogram measurement for a given pixel patch.
- **Derive Matrix A:** In order to predict (propagate) the particles of interest, we need to derive the dynamic matrices for the respective models a) no motion and b) constant velocity.
- **Propagation:** Propagate the particles according to their dynamic model matrix **A**.
- **Observation:** After propagating, we need to compare the histograms of the propagated *a priori* particles with the target particle histogram and assign an according probability weight to the particles.
- **Estimation:** We compute the estimation for our target particle.
- **Resampling:** Afterwards the *a priory* particles need to be resampled according to their weights in order to receive a new set of particles, the so-called *posteriori* particles.

1.1 Color Histograms

To compute the color histograms of our target, we create a Matlab function
`histogram = color_histogram(xMin, yMin, xMax, yMax, frame, hist_bin)`.
The inputs are the bounding box parameters `(xMin, yMin, xMax, yMax)` for the object to track, the current video frame `frame` and the number of bins for each RGB channel to use for the histogram `hist_bins`. To compute the histogram counts in each bin, the Matlab function `histcounts` can be used. An example for such a histogram can be seen in Figure 1.

Figure 1: Example histogram with size 3x16. (3 channels for RGB, 16 histogram bins each.)

1.2 Derive Matrix A

The dynamic propagation (without noise) looks as follows:

$$s_t = As_{t-1}. \tag{2}$$

In case of the constant velocity model, the explicit equation can be written as

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_{t-1}, \tag{3}$$

where dt in this exercise is set to 1.

In the case of no motion, the state vector simply reduces to $s = \begin{bmatrix} x & y \end{bmatrix}^T$, which leads to

$$\begin{bmatrix} x \\ y \end{bmatrix}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_{t-1} \tag{4}$$

1.3 Propagation

To propagate the set of particles, we implement a function
`propageted_particles = propagate(particles, sizeFrame, params)`.
The inputs are the current set of particles `particles`, the size of the video frame `frame` to make sure the particles stay within the image and the model parameters `params` that tell us whether to use the no motion or constant velocity assumption. Each particle can be propagated by equation (3) and some added noise, where the noise in this exercise is approximated as a normal distribution. The parameters for the normal distribution are also given as an input within `params`. The resulting propagated particles are called the *a priori* particles, because they give a first idea of where our tracking object could be prior to consider the measurements.

1.4 Observation

It is now to compare the *a priori* particles with the target and set their importance weight accordingly to the equation

$$\pi = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(\chi^2(CH_{particle}, CH_{target})^2}{2\sigma^2}}, \tag{5}$$

where the χ^2 distance can be computed with the provided function `chi2_cost`. The Matlab function
`weights = observe(particles, frame, H, W, hist_bin, hist_target, sigma_observe)`
computes the particle weight of every particle as described in equation (6). The inputs to this function are the *a priori* particle set `particles`, the current video frame `frame`, the bounding box width `W` and height `H`, the number of histogram bins to use for each channel `hist_bins`, the target histogram `hist_target` and the σ to use in equation (6) `sigma_observation`.

Once the weights are computed, it is important not to forget to rescale them such that their total sums up to 1.

1.5 Estimation

The current estimation for the target particle state can easily be computed by summing up the state vectors from all the particles multiplied by their weight:

$$E[s_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}. \tag{6}$$

1.6 Resampling

In the end, the *a priori* particles need to be resampled with replacement according to their weights. One way to do this is to implement a so-called *Resampling Wheel*, where I reused the one we learnt from the Udacity videos from *Lab Assignment 3 - Optical Flow Particle Filtering* and adapted it to this assignment. The resulting resampled particles are called the *posteriori* particles, which will be the input for the propagation step in the next loop iteration.

2. Experiments

2.1 Video 1

In Figures 2 we can see a first tracking result with with the standard parameters.

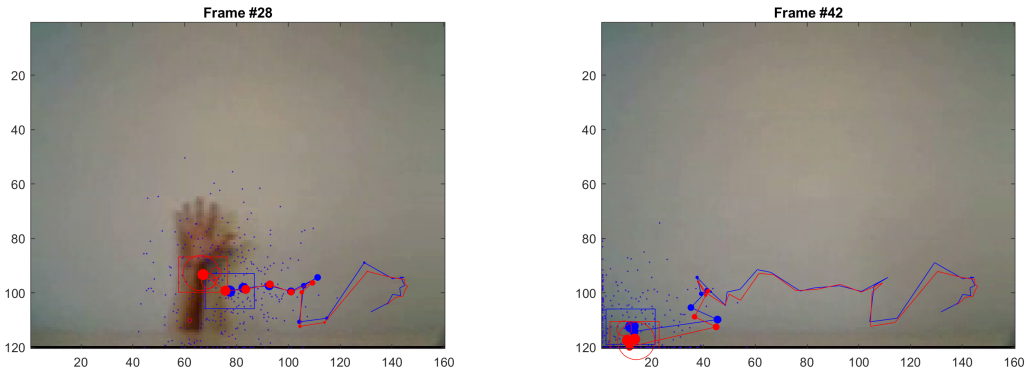

Figure 2: Tracking result with standard parameters. ($\alpha = 0, \sigma_{obs}, model = 0, numparticles = 300, \sigma_{pos} = 15, histbin = 16$)

As we can see, the tracking result in the beginning is quite bad. Upon closer investiation we notice that the tracking gets thrown off as soon as the illumination of the hand, and therefore the RGB color, changes. To cope with this, we can allow the target histogram not to be constant but also evolve over time by the formula

$$CH_{target} = (1 - \alpha) \cdot CH_{target} + \alpha \cdot CH_{E[s_t]}. \tag{7}$$

If we set $\alpha = 0.5$ for example, the tracking result seems to be much more robust to changes in illumination. The respective result can be seen in Figure 3.

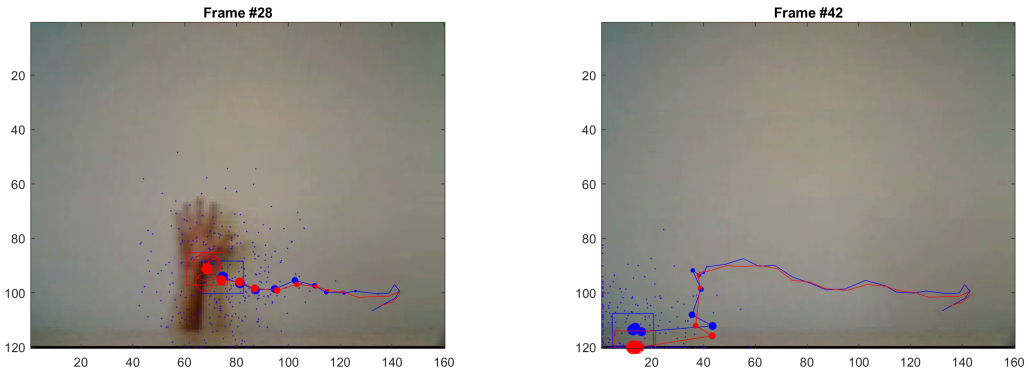

Figure 3: Tracking result with changed α. ($\alpha = 0.5, \sigma_{obs}, model = 0, numparticles = 300, \sigma_{pos} = 15, histbin = 16$)

For this part of the exercise, we did not yet experiment with other parameters than α.

2.2 Video 2

If we run the algorithm again on the second video with the same parameters as above, we notice that the tracking gets stuck on the occlusion object as seen in Figure 4.

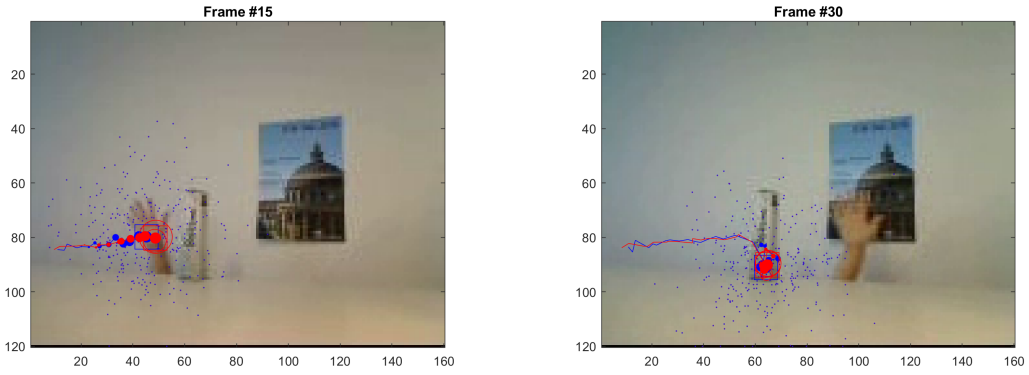

Figure 4: Tracking result of video 2. ($\alpha = 0.5, \sigma_{obs} = 0.1, model = 0, numparticles = 300, \sigma_{pos} = 15, \sigma_{vel} = 1, histbin = 16$)

We can now try to investigate on the effect of our parameters and answer the questions from the exercise.

- What is the effect of using a constant velocity motion model?
- What is the effect of assuming decreased/increased system noise?
- What is the effect of assuming decreased/increased measurement noise?

The most obvious thing to change here is that we should start using the constant velocity model, which makes sense since the hand is indeed moving with a more or less constant speed. This allows us to propagate the particles in a more meaningful manner even if the tracking object is occluded. Since the particles still propagate according to their velocity, it enables us to catch our tracking object once it becomes visible again.

Another thing we notice is that the system noise during propagation is way too high for this scenario. As we can see in Figure 4, the propagated particles (blue) are very widespread. If we reduce the system noise, the propagated particles are more narrowly distributed, which leads to a more precise prediction of where our tracking object could be. In our experiment, we reduce the system noise to $\sigma_{pos} = 1.5$ and $\sigma_{vel} = 0.5$. Additionally, we can handle our measurements a little more 'generous'. Before, our measurement noise was very low, which means that we trusted very much in our comparison between the color histograms. However, it can be beneficial to be bit more generous and increase the assumed measurement noise. This means that we do not trust so much in our measurements and assign the weights a bit more even. For our experiment, we increased the measurement noise to $\sigma_{obs} = 0.3$.

As we can see in Figure 5, by using the aforementioned parameters, we can indeed achieve successful tracking.

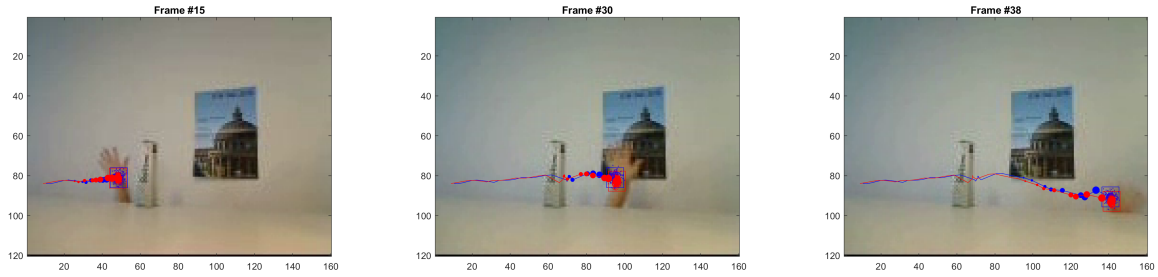

Figure 5: Tracking result of video 2 with adjusted parameters. ($\alpha = 0.25, \sigma_{obs} = 0.3, model = 1, numparticles = 300, \sigma_{pos} = 1.5, \sigma_{vel} = 0.5, histbin = 16$)

2.3 Video 3

If we track the ball in Video 3 with the same parameters as in Video 3, we can see that it fails already in the second frame, as displayed in Figure 6.

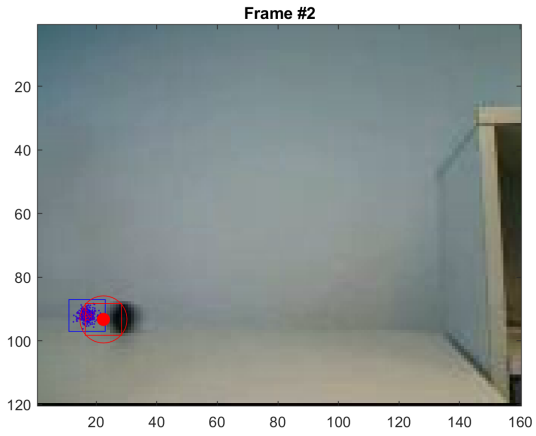

Figure 6: Tracking result of video 3. ($\alpha = 0.25, \sigma_{obs} = 0.3, model = 1, numparticles = 300, \sigma_{pos} = 1.5, \sigma_{vel} = 0.5, histbin = 16$)

The reason for the failure here lies in a wrong assumption for the initial velocity. If we adapt the initial velocity and try again, we get a result as seen in Figure 7.

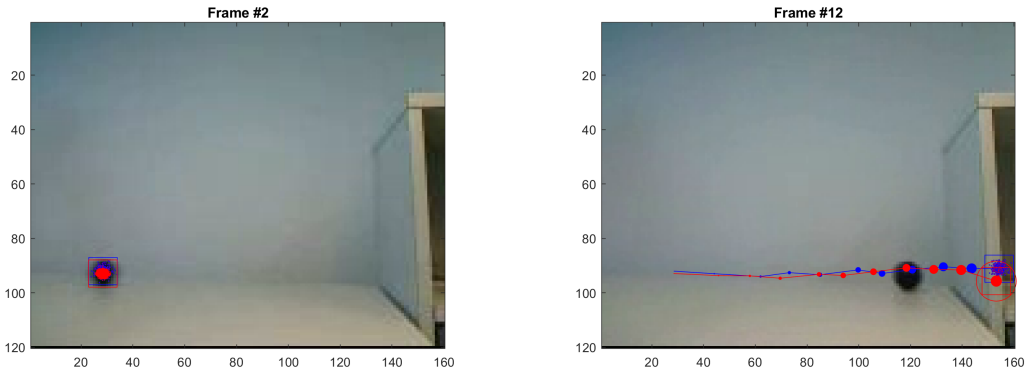

Figure 7: Tracking result of video 3. ($\alpha = 0.25, \sigma_{obs} = 0.3, model = 1, numparticles = 300, \sigma_{pos} = 1.5, \sigma_{vel} = 0.5, histbin = 16$)

We can see that here the initial tracking went fine. However, at the moment where the ball bounces off the wall the tracking fails. The reason for this lies in our model, where we are still assuming constant velocity. At the moment where the ball bounces off the wall, the velocity of the ball instantly changes its direction, which is why the constant velocity model does not make much sense here. The model assumption for no motion is not really correct neither, but it is a more general solution and in this case less counterproductive. If we now set the model assumption back to no motion and again increase the system noise to make the propagated particles more widespread we can get much better results. Additionally, we notice that the ball in this video has a very distinct color form everything else and is never occluded. This means we can put more trust in our measurements and reduce the change of our target histogram. In Figure 8 we can see a successful tracking result with the suggested parameter changes.

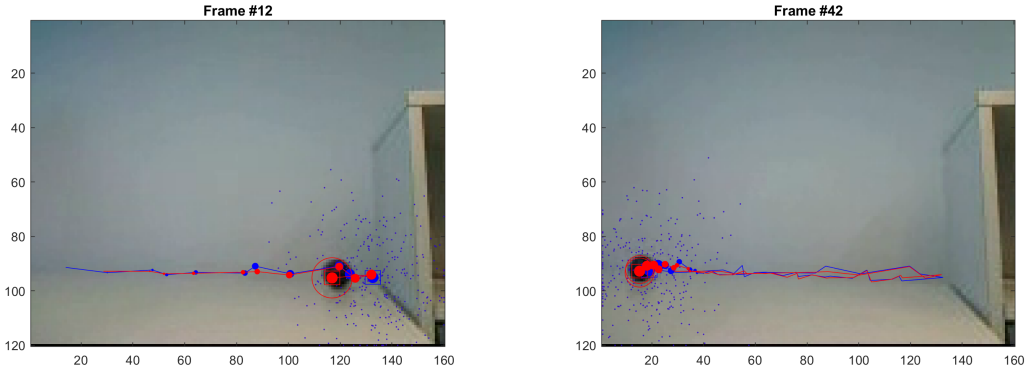

Figure 8: Tracking result of video 3. ($\alpha = 0.1, \sigma_{obs} = 0.1, model = 0, numparticles = 300, \sigma_{pos} = 15, histbin = 16$)

2.4 Other Parameters

What is the effect of using more or fewer particles?

We have to keep in mind that the resampling of the particles is random. If we therefore increase the number of particles we can get a more robust representation of the underlying probability density function and we have an increased chance of drawing particles that fit very well to our target. This of course comes with the cost of longer computation times. Contrarily, if we reduce the number of samples can increase computation

speed, but we make the algorithm less robust.

What is the effect of using more or fewer bins in the histogram color model?

If we increase the number of bins used for our color histograms we can achieve a higher uniqueness of the histograms, but we also are less robust against noise. On the other hand, if we use only a few bins, then the histograms of different patches of the image can start to look very similar when in fact they correspond to completely different objects.

What is the advantage/disadvantage of allowing appearance model updating?

This question was already partly answered in section 2.1. If the illumination of an object changes over time, the RGB values also change. If we allow the target histogram to update slightly from frame to frame, we can get more robust to illumination changes as seen in Video 1. However, if the model updating happens too fast we risk our target to update in a wrong way. For example when our object is occluded as in Video 2, the target histogram would very quickly update to the histogram of the occlusion itself and we would not recognize our real target anymore once it becomes visible again.