Filippo Ficarra: Lab 6 - Condensation Tracker

fficarra@student.ethz.ch, 22–938–062.

08/12/2023 - 16:38h

1 Implementation

In this exercise we implemented the CONDENSATION algorithm, a sampled-based solution of the recursive Bayesian filter. We needed to implemt 5 functions:

- color_histogram
- propagate
- observe
- estimate
- resample

All of this functions are implemented in the corresponted .py files.

1.1 Color histogram

Listing 1: color histogram

```
def color_histogram(xmin, ymin, xmax, ymax, frame, hist_bin):
    bounding_box = frame[ymin:ymax, xmin:xmax]
    hist_channels = []

    for channel in range(bounding_box.shape[2]):
        hist, _ = np.histogram(bounding_box[:,:,channel], bins=hist_bin)
        hist_channels.append(hist)

    histogram = np.concatenate(hist_channels)
    norm_histogram = histogram / np.sum(histogram)

    return norm_histogram
```

In this function I am creating for each channel (RGB) an histogram with hist_bin number of bins for the pixels. What the function does is to take the patch of the image described by the bounding box, create the histogram for each channel, concatenate and then normalize.

1.2 Propagate

Listing 2: propagate

```
1
2   def propagate(particles, frame_height, frame_width, params):
3     delta_t = 1
4     sigma_p = params["sigma_position"]
5     sigma_v = params["sigma_velocity"]
6
7     A = np.identity(particles.shape[1])
8     model_noise_std = np.array([sigma_p, sigma_p])
9
10    if params["model"] == 1:
11        A[0, 2] = delta_t
12        A[1, 3] = delta_t
13        model_noise_std = np.array([sigma_p, sigma_p, sigma_v, sigma_v])
14
15    w = np.random.randn(particles.shape[0], particles.shape[1]) * model_noise_std
16
17
18    particles = np.matmul(particles, A.T) + w
19
20
21    particles[:, 0] = np.clip(particles[:, 0], 0, frame_width - 1)
22    particles[:, 1] = np.clip(particles[:, 1], 0, frame_height - 1)
23
24  return particles
```

This is the pivotal function of the algorithm. Here each sample s'_{t-1} is propagated with the following formula: $s_t^{(n)} = As_{t-1}^{\prime(n)} + w_{t-1}^{(n)}$. First, each sample is of this form: $s = \{x, y, \dot{x}, \dot{y}\}$. The matrix Thge matrix A is taken as the identity for the no motion case, or the identity plus an identity with the diagonal shifted up and multiplied by dt. For the no motion model the sampling gives this result:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \implies (x_t^{(n)}, y_t^{(n)})^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} (x_{t-1}^{(n)}, y_{t-1}^{(n)})^\top + (\sigma_{pt-1}^{(n)}, \sigma_{pt-1}^{(n)})^\top =$$

$$\begin{cases} x_t^{(n)} = x_{t-1}^{(n)} + \sigma_{pt-1}^{(n)} \\ y_t^{(n)} = y_{t-1}^{(n)} + \sigma_{pt-1}^{(n)} \end{cases}$$

For the constant velocity we are using the following contruction:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \implies (x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)})^\top =$$

$$\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (x_{t-1}^{(n)}, y_{t-1}^{(n)}, \dot{x}_{t-1}^{(n)}, \dot{y}_{t-1}^{(n)})^\top + (\sigma_{pt-1}^{(n)}, \sigma_{pt-1}^{(n)}, \sigma_{vt-1}^{(n)}, \sigma_{vt-1}^{(n)})^\top =$$

$$\begin{cases} x_t^{(n)} = x_{t-1}^{(n)} + \dot{x}_{t-1}^{(n)} dt + \sigma_{pt-1}^{(n)} \\ y_t^{(n)} = y_{t-1}^{(n)} + \dot{y}_{t-1}^{(n)} dt + \sigma_{pt-1}^{(n)} \\ \dot{x}_t^{(n)} = \dot{x}_{t-1}^{\prime(n)} + \sigma_{vt-1}^{(n)} \\ \dot{y}_t^{(n)} = \dot{y}_{t-1}^{\prime(n)} + \sigma_{vt-1}^{(n)} \end{cases}$$

I assumed dt to be 1, and w is a random matrices with covariance matrix given by σ_p and σ_v. At the end the particles are clipped to be in the frame.

1.3 Observe

Listing 3: observe

```
def observe(particles, frame, bbox_height, bbox_width, hist_bin, hist, sigma_observe):
  particles_w = np.zeros(particles.shape[0])
  for i, particle in enumerate(particles):
      x_center = particle[0]
      y_center = particle[1]

      x_min = np.clip(int(x_center - bbox_width/2), 0, frame.shape[1]-1)
      y_min = np.clip(int(y_center - bbox_height/2), 0, frame.shape[0]-1)
      x_max = np.clip(int(x_center + bbox_width/2), 0, frame.shape[1]-1)
      y_max = np.clip(int(y_center + bbox_height/2), 0, frame.shape[0]-1)

      histogram = color_histogram(x_min, y_min, x_max, y_max, frame, hist_bin)

      chi_2 = chi2_cost(histogram, hist)

      particles_w[i] = 1/(np.sqrt(2*np.pi)*sigma_observe) *
                          np.exp(-chi_2**2/(2*sigma_observe**2))

  particles_w = particles_w / np.sum(particles_w)

  return particles_w
```

For each particle we compute the Chi2 distance between the color histogram of the box around the particle and the otriginal color histogram. Then each weight is computed using the gaussian function on this Chi2 distances. Everything at the end is normalized to mantain this weight as probabilities.

1.4 Estimate

Listing 4: estimate

```
def estimate(particles, particles_w):
  estimate = np.zeros(particles.shape[1])

  for i, particle in enumerate(particles):
      estimate += particle * particles_w[i]

  estimate = estimate / np.sum(particles_w)

  return estimate
```

This function compute the estimated particle, computing its weighted mean using the particle weights.

1.5 Resample

Listing 5: resample

```

def resample(particles, particles_w):
    index = np.random.choice(particles.shape[0], particles.shape[0], p=particles_w)
    return particles[index], particles_w[index]
```

With this function then we resample the particles given the weights computed in the observe function.

2 Results

2.1 Video 1

The algorithm works pretty well with the first video since the movement is very simple and the color are pretty distinguishable. The tracker worked, in a sense, with the default parameters but we, as we can see in the picture below, it tended to follow the wrist and it tended to loose the tracking almost at the end of the video, when the hand moved quicker. Below the results for the video 1 and the default parameters.

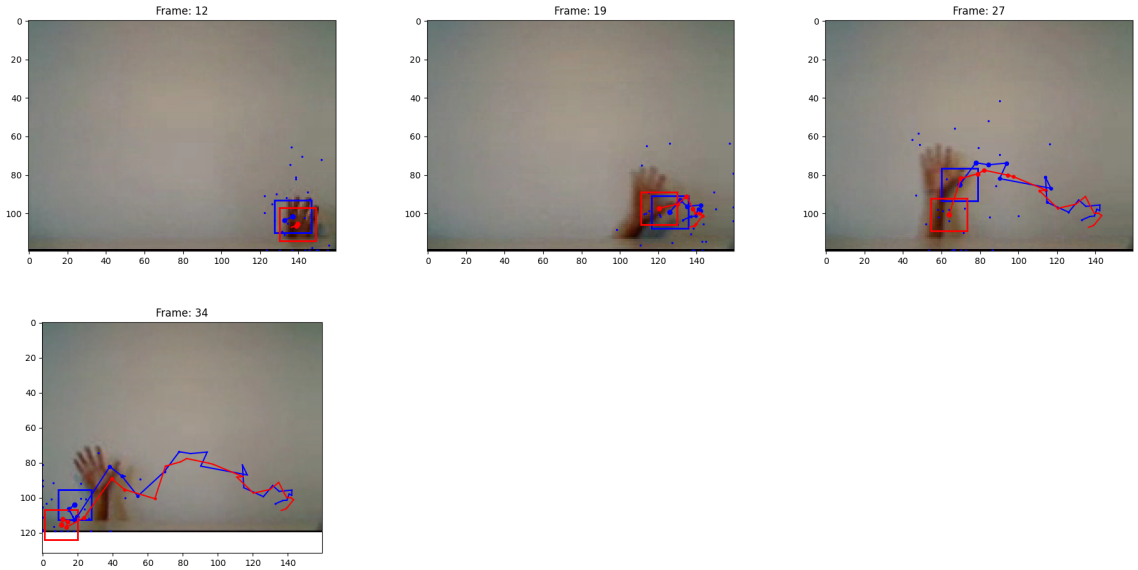

Figure 1: Video 1 with default parameters, no motion model

If we increase more the numbers of particles we should get more accurate results. As shown below the tracker gets better in the last part of the video when it doesn't loose the hand. One of the problem is that the tracker still follows the wrist, and this can be due to the no motion model or the hist_bin, since the image contains the shadow of the hand.

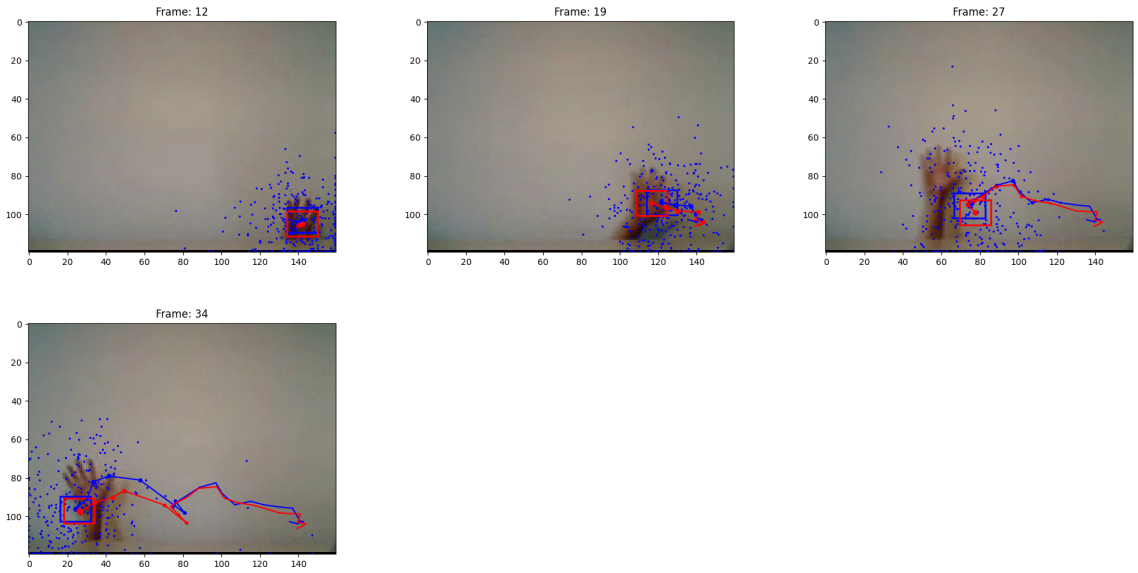

Figure 2: Video 1 with default parameters, no motion model, 300 particles

What happens if we play with other parameters like the number of bins? Below I show that reducing the number of hist_bins impact on the tracking and make it fail. This is due to the fact that we are grouping very different colors together, and this makes the algorithm not robust to light changing for example.

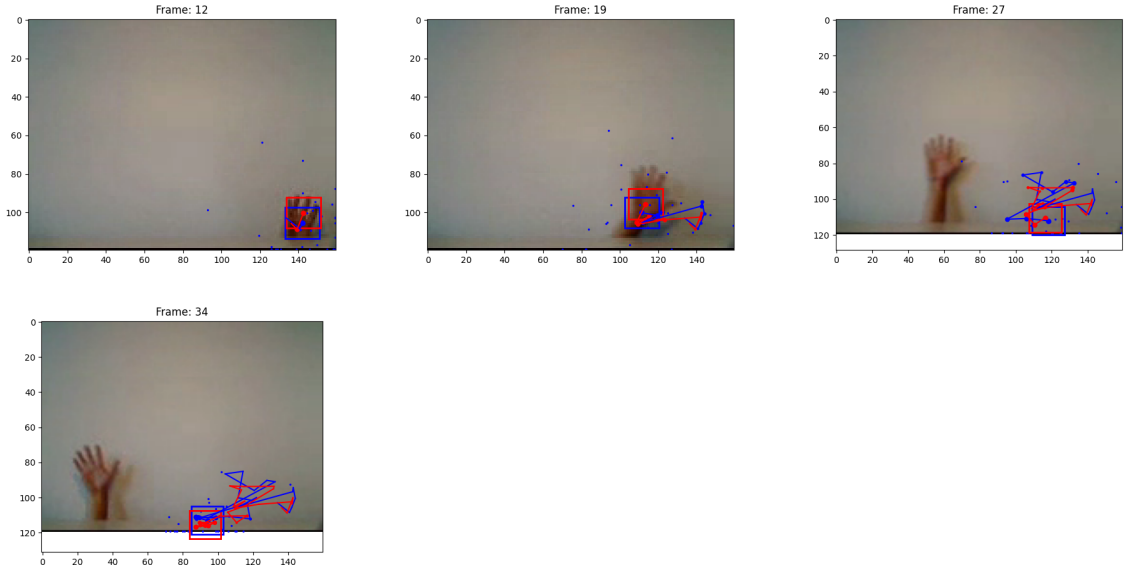

Figure 3: Video 1 with default parameters, no motion model, hist_bin 8

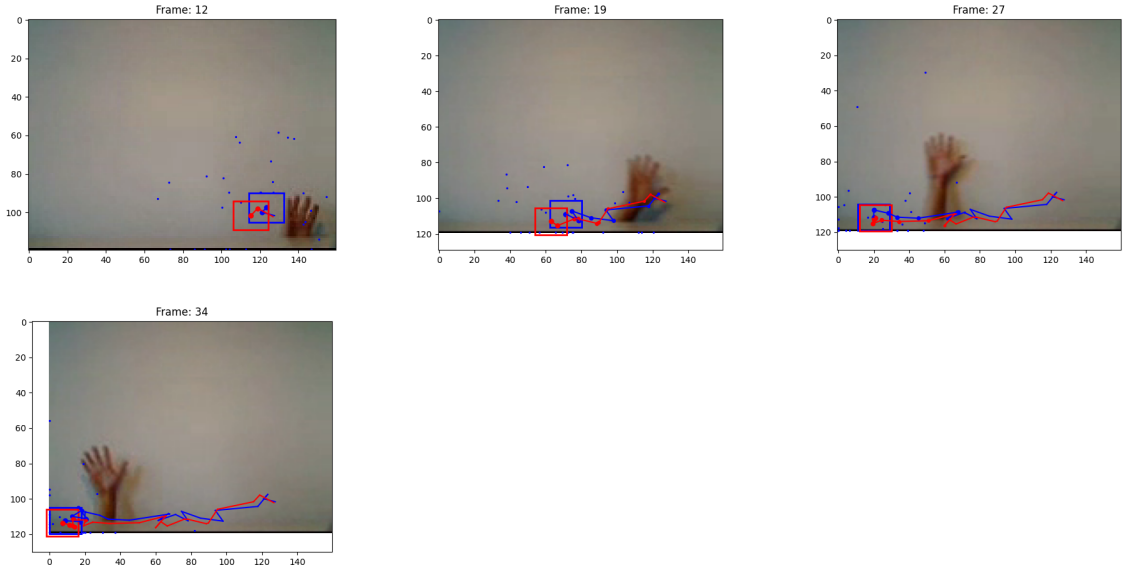

Figure 4: Video 1 with default parameters, no motion model, hist_bin 4

If we increase the number of hist_bins to 32 we see an improvement on the tracking. The model is able to track better the hand and not only the wrist. Also the model is more stable to the sudden disappearance of the hand at the end.

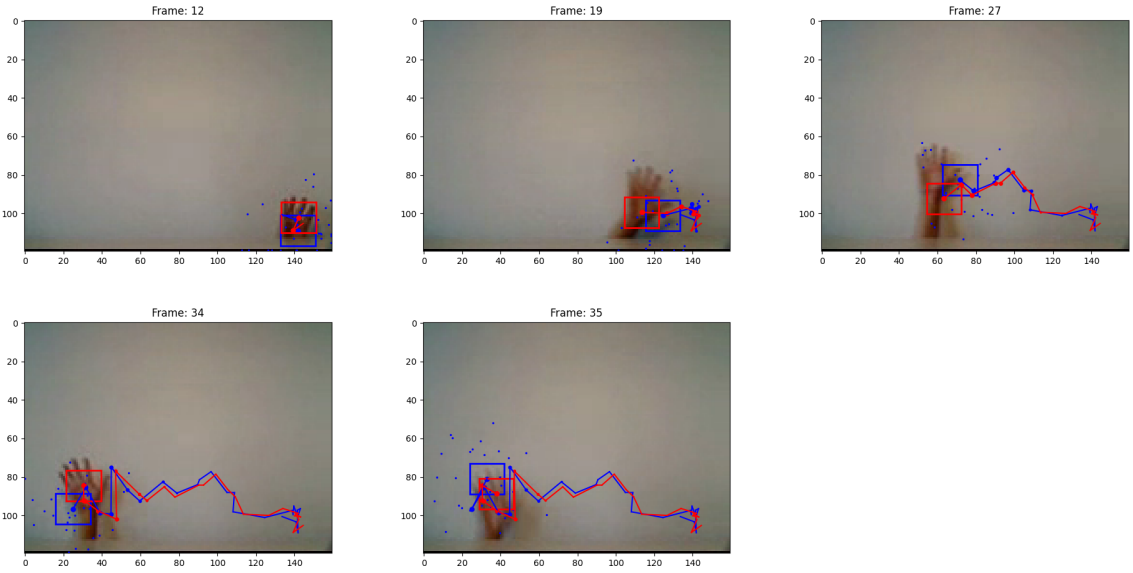

Figure 5: Video 1 with default parameters, no motion model, hist_bin 32

While no motion model is working, thanks also to the sigma_position, the result can be improved using a constant velocity model. The only parameter changed here are the initial velocity to (-1, -5) ,so the tracker is initialy directed to top left, and the sigma_position reduced to 10.

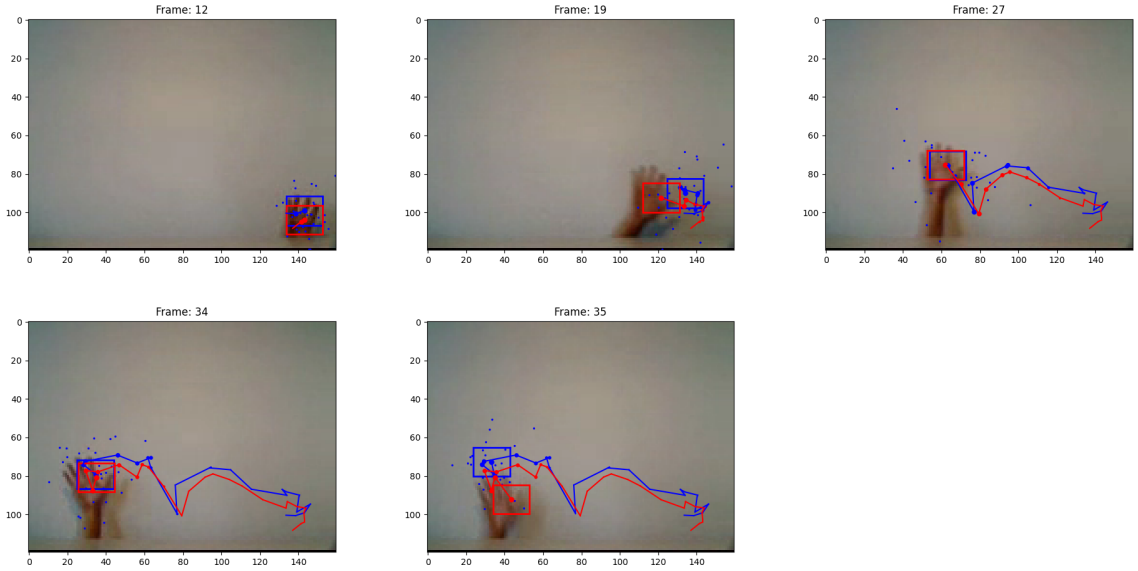

Figure 6: Video 1 with default parameters,constant velocity, initial velocity (-1, -5), sigma_position 10

2.2 Video 2

The second video was more complex than the first one, since there is an object that performs occlusion and the background is not constant as in the first video. As first model I tried the one with default parameters. Below the results.

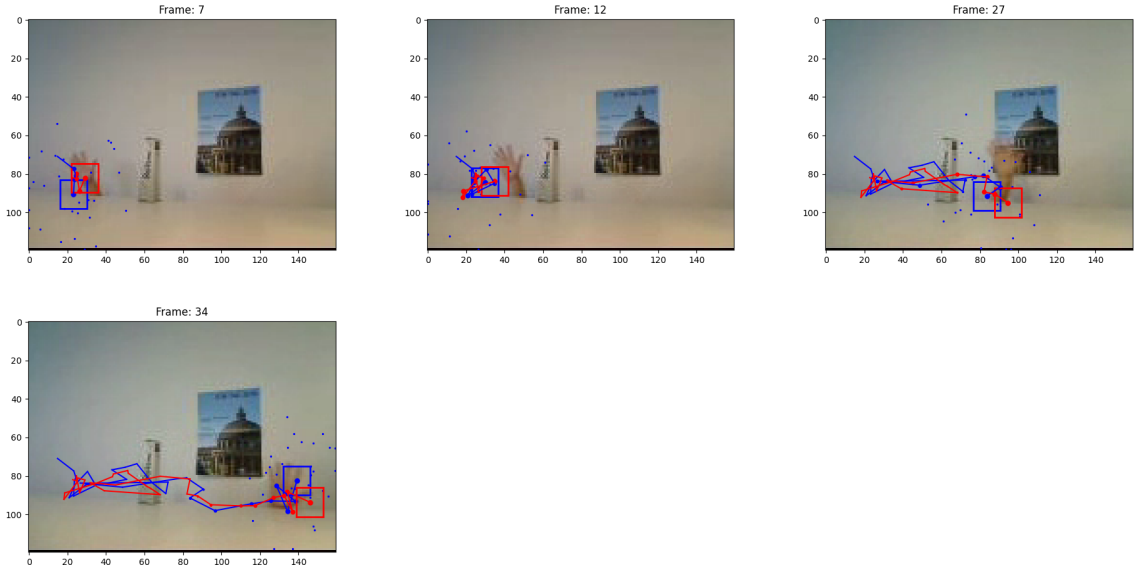

Figure 7: Video 2 with default parameters, no motion model

The model almost track the hand for the whole video. As shown in the images, we have high inaccurancy near the obstacle, but the model is still able to follow the hand.

Lets now try the constant velocity. Since the object is moving right without any y movement, we can set the inital velocity to (3, 0).

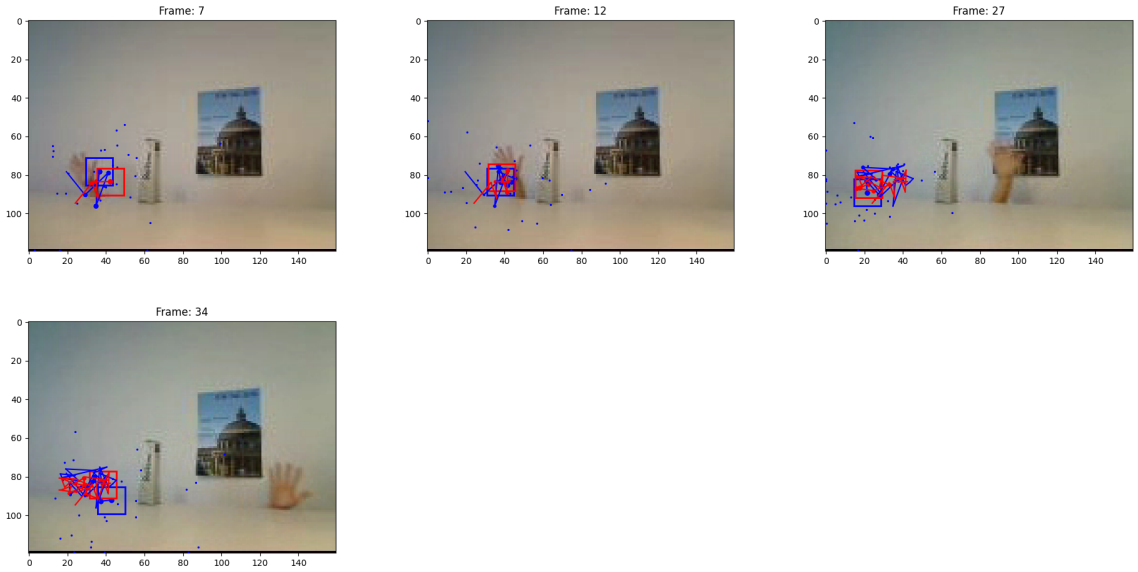

Figure 8: Video 2 with default parameters, constant velocity model, initial velocity (3, 0)

This tracking fails, due to the occlusion. We can imporve the tracking sertting α to 0.5, so that the color histogram is updated with with the current histogram. Despite this change the result are not so different from the constan motion.

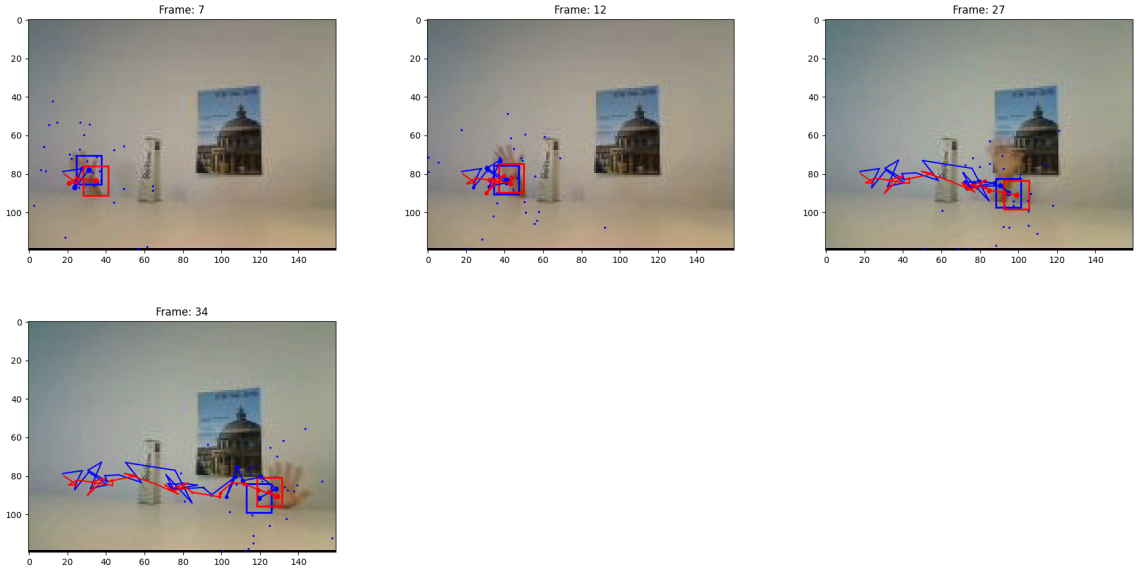

Figure 9: Video 2 with default parameters, constant velocity model, initial velocity (3, 0)

If we further change the parameters we can find an optimal setting for tracking. What performed best was 300 particles, sigma_position = 1, sigma_velocity = 0.1, $\alpha = 0.5$ and initial velocity (4,0). Below also a comparison of the result when using different sigma_observe values.

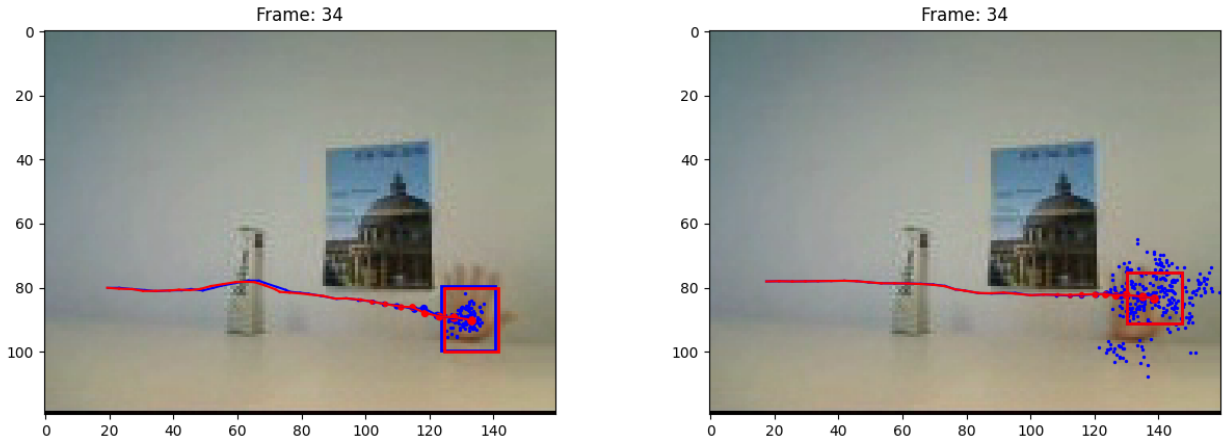

Figure 10: Best params for Video 2, sigma_observe 0.1 vs sigma_observe 1

The second image seem to be a better tracking, but the problem is that, due to the high observation uncertainty, the tracker tend to be ahead the hand and not tracking it for the middle part of the video.

If we increase sigma_position to 30, the tracking result in failure as shown in the image.

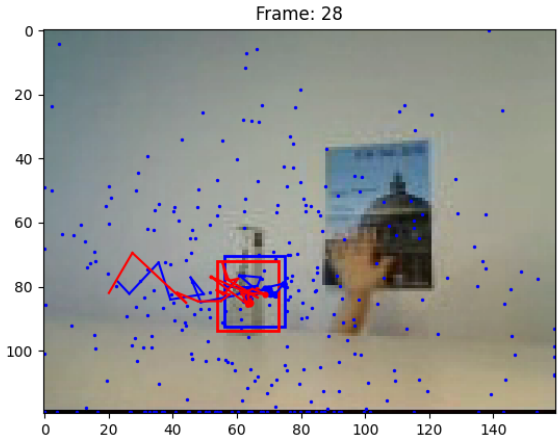

Figure 11: Best params for Video 2, sigma_position 30

The outcome of this experiment for now tell us that:

- Reduced system noise in the model typically leads to more confidence in the predicted motion.
- Higher system noise makes the predictions less reliable and more susceptible to errors.
- Lower measurement noise implies more accurate and reliable measurements of the object's position. This leads to better updates in the algorithm.
- Higher measurement noise introduces more uncertainty into the measurements. This makes the update less accurate.
- Constant velocity is very useful when we know that the object is moving at a constant velocity, since the assumption of constant velocity helps to estimate next frames in which the object is more likely to appear

2.3 Video3

I started looking at the effect of video 3 with the best parameters of the video 2. These are the results: Clearly

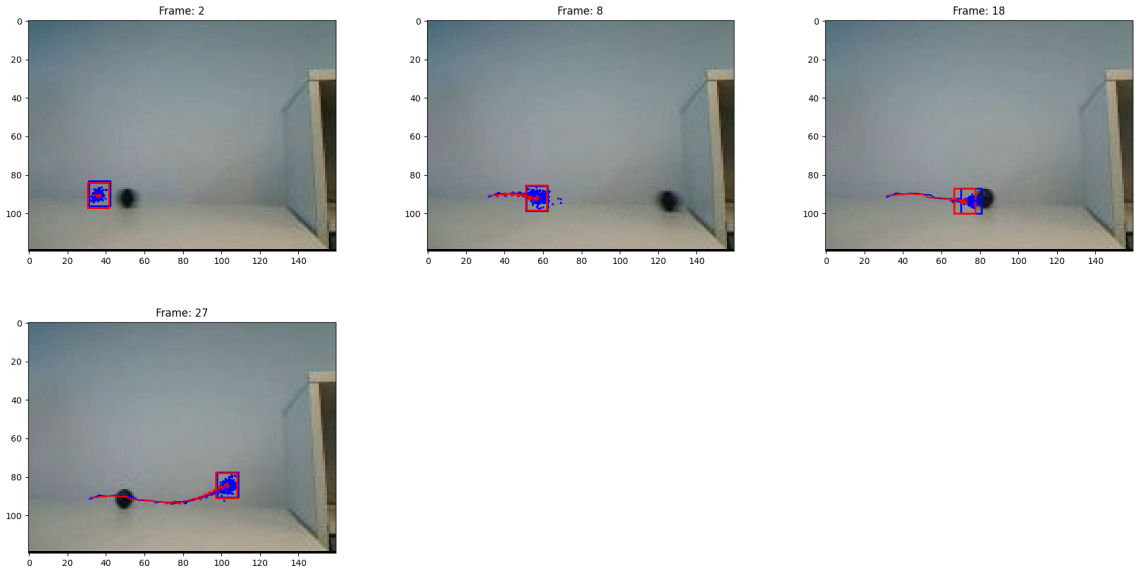

Figure 12: Video 3 with best parameters for model 2

the approach have some problems. First we notice that the initial velocity is not adequate for the video in analysis. We can impove this just increasing the velocity on the x axis to 10. We see a slightly improvement on the tracking, but still the algorithm is not capable to follow the ball for the entire video.

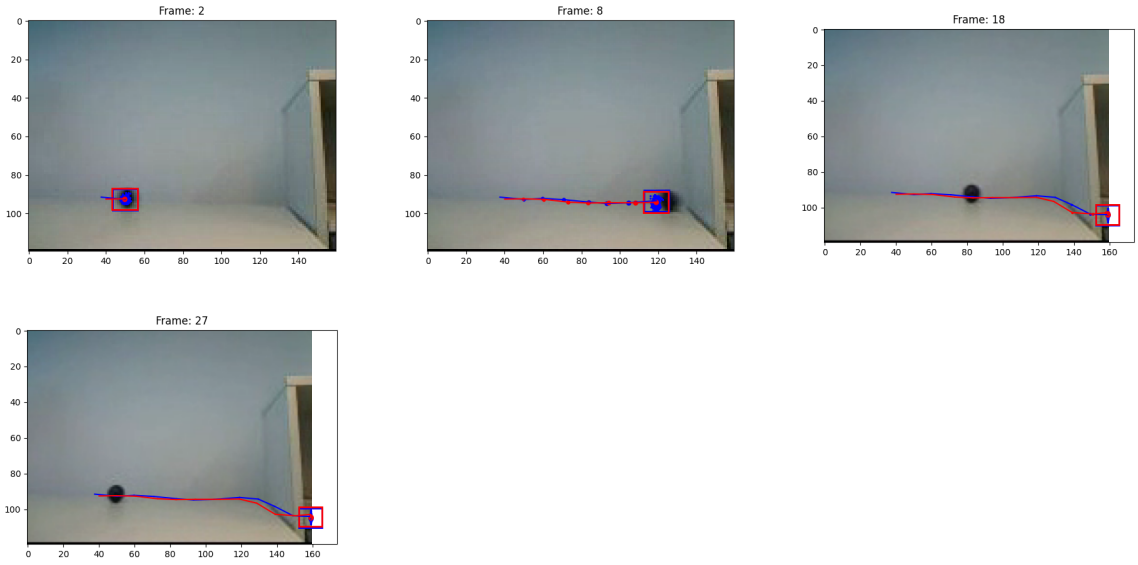

Figure 13: Video 3 with best parameters for model 2, initial velocity (10, 0)

This is because the ball changes direction suddenly, so the constant velocity is not able to follow, we can go back to the no motion model, but we need to increase the system noise.

As expected the system noise, when increased, helps tracking in a no motion model. In the first image, due to the very low sigma_position the tracker didn't move, while increasing its value gave an almost perfect tracking.

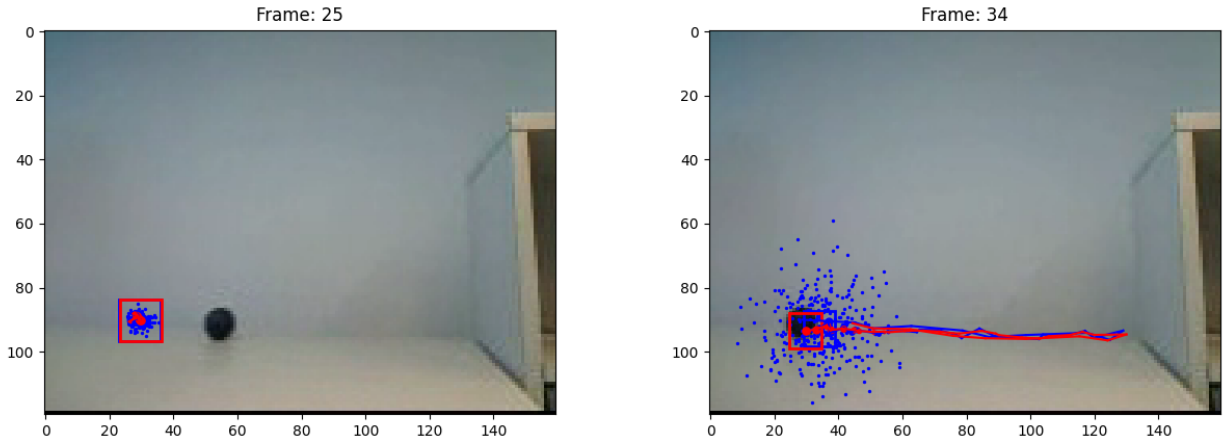

Figure 14: Video 3, no motion model, sigma_position 1 vs 10, sigma_observe 0.1

We can also look at the effect of decrasing and increasing the observation uncertainty.

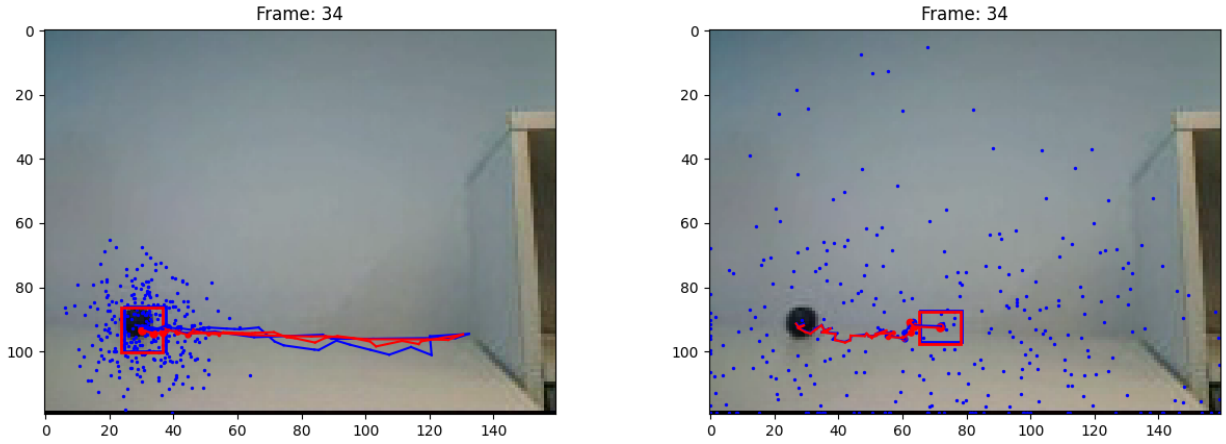

Figure 15: Video 3, no motion model, sigma_obeserve 0.05 vs 1.0

In summary:

- Number of particles when increased makes the tracker more accurate since we have more samples from which apporximate the prediciton distribution
- Increasing the number of bins in a histogram color model provides a more detailed representation of color information. This for better discrimination between different colors and shades, potentially leading to more accurate object recognition and tracking.
- Allowing appearance model updating enables the tracker to adapt to changes in the object's appearance over time. This adaptation helps in handling variations in lighting conditions, occlusions etc...