ASSIGNMENT 9 - CV REPORT

CONDENSATION Tracker Based On Color Histograms

In the implementation of the condensation tracker based on the color histograms, the steps proposed in the assignment were followed. In the newly defined ***color_histogram*** function, first a quick check on the coordinates of the center of the bounding box was done. Then, the 3 color histograms of the bounding box with respect to the 3 RGB channels are computed and stacked together in a single vector. The final histogram is then normalized with respect to its sum.

In the ***propagate*** function the system model is designed and implemented. In this assignment, it was asked to take two models into consideration (no motion and constant velocity).

As suggested in the assignment, for a given particle p, let S_t be the state of the model at time step t, A be the deterministic component of the system model and S_{t-1} and w_{t-1} respectively the state of the bounding box and its noise at the previous step. We can adopt the following discrete model:

$$\mathbf{S_t' = A\, s_{t-1}' + w_{t-1}}$$

- ***NO MOTION MODEL***

First of all we need to consider the case of no-motion model. We can describe the state of each box only via two variables **[x,y]**, where x,y represent the location of the center of the bounding box. The position of the box is only affected by the noise. Therefore we can proceed as follows:

The dynamic matrix is of course the 2x2 identity matrix, since in absence of noise, the position is expected to be constant.

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Then we can add a positional noise with zero mean and standard deviation equal to σ_p, provided in the assignment. In formulas:

$$\mathbf{x'_t = x'_{t-1} + w_{t-1}}$$

$$\mathbf{Y'_t = y'_{t-1} + w_{t-1}}$$

Where we can model w as random samples of a gaussian with zero mean and σ_p standard deviation.

- ***CONSTANT VELOCITY MODEL***

In the case of constant velocity model, each particle is defined by 4 state-variables $\mathbf{[x,y,v_x,v_y]}$, where x,y once again represent the location of the bounding box, and v_x,v_y the velocities in the x and y direction. We can model the system via the following equations:

$$\mathbf{X'_t = x'_{t-1} + v_x\, dt + w_{t-1}}$$

$$\mathbf{Y'_t = y'_{t-1} + v_y\, dt + w_{t-1}}$$

$$\mathbf{V_x = V_x + n_{t-1}}$$

$$\mathbf{V_y = V_y + n_{t-1}}$$

The first two equations arise from the analysis of a simple dynamical system: the new position of a particle after a time step dt is obtained by adding to the starting position the product of the velocity and dt. A Gaussian noise is then added to model the uncertainty. The noise *w* is sampled from a Gaussian distribution with 0 mean and standard deviation $\sigma_{\mathbf{p}}$.

The last two equations are explained by the fact that we make the hypothesis of constant velocity. Once again a Gaussian noise is added to take uncertainty into account. The noise *n* is sampled from a Gaussian distribution with 0 mean and standard deviation $\sigma_{\mathbf{v}}$

As a consequence the matrix A takes the form:

$$A = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We can assume dt = 1 if we want to express velocity in pixel per frame. A timestep dt = 1 is assumed to be the time between two consecutive video frames and not the time in the International System (second).

In the ***propagate*** function, a check on the particles' position was implemented in order to make sure that they lie inside the frame. Whenever the particles are outside the frame, they take the values of the edges of the frame. This applies to both the width and the height.

In the ***observe*** function, we compute for all particles their color histograms after having computed their respective bounding boxes. We compute the weight of each particle by taking into account the χ^2 distance between the target model color histogram and each bounding box histogram.

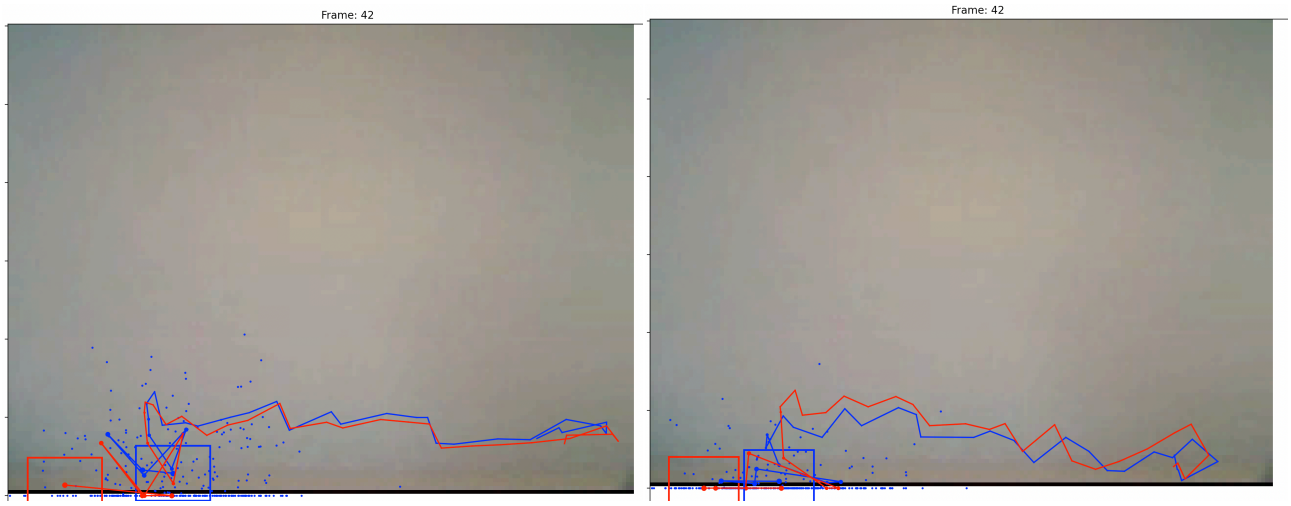
In the ***estimate*** function, the mean state is calculated via a weighted average of the particles.

Finally, in the ***resample*** function, the particles are sampled with replacement according to their weights. The function random choice is used to sample the indices.

Experiments

VIDEO 1

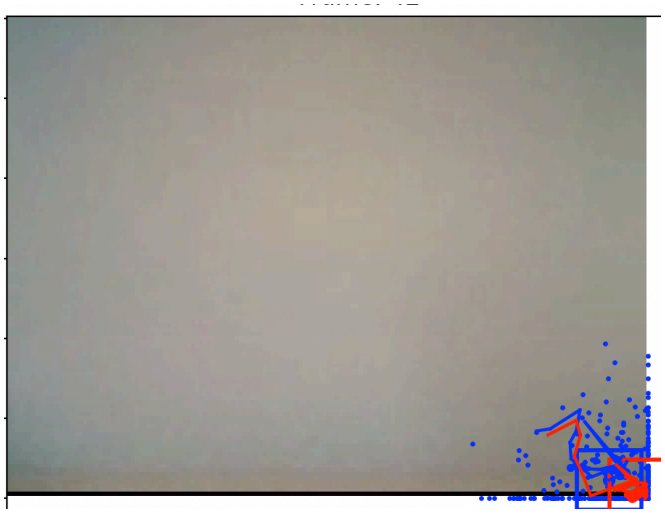
The first video shows an hand moving around in a uniformly colored background. Here the results are reasonable both with the no motion and with the constant velocity model.



Here are two trajectories obtained with no motion model (on the left) and constant velocity model on the right. Default parameters were used for these experiments.

The tracking is working perfectly fine, with the slight inconvenience that sometimes the forearm is tracker rather than the hand. This is an expected behavior since the arm and the hand are chromatically similar.

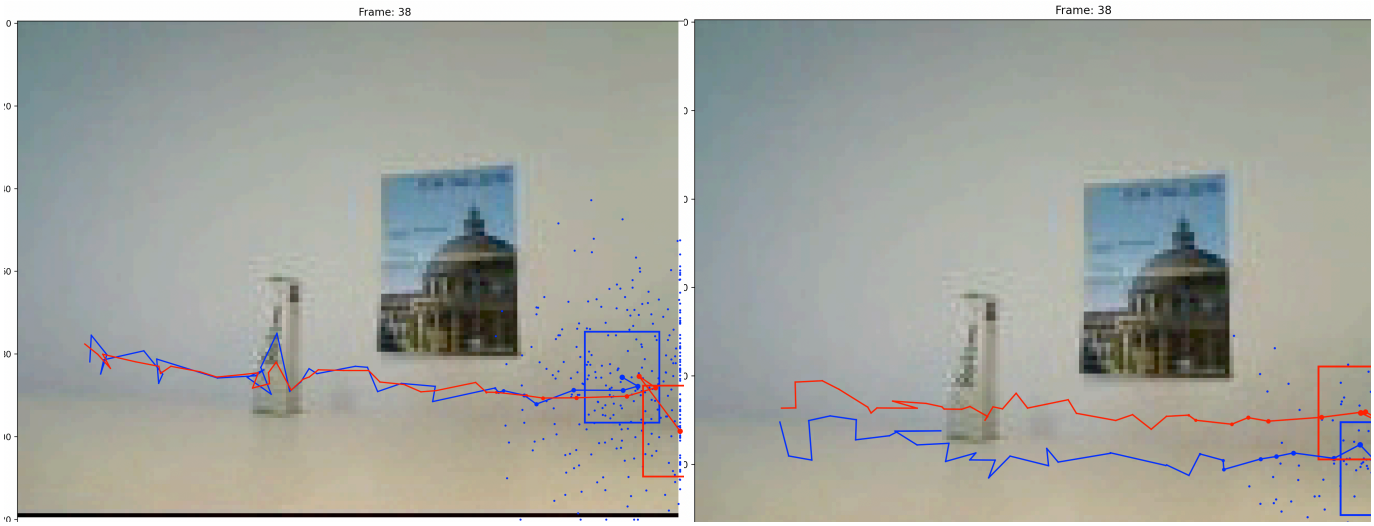
An interesting experiment can be done by changing the value of α. In particular, setting $\alpha = 1$ means to update the target histogram only with the last estimated mean state, without taking into account the initial histogram of the original bounding box. Ceteris paribus, the algorithm completely fails to track the hand:



In the figure on the left, the result obtained with the no motion model, when setting $\alpha = 1$.

VIDEO 2

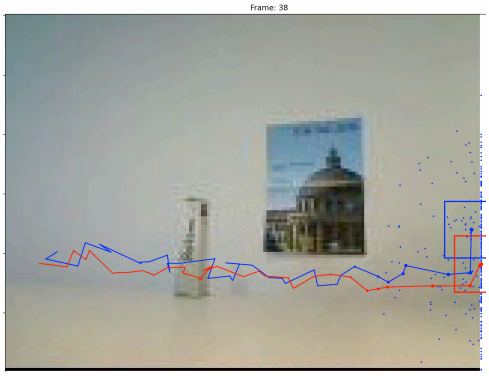
In the second video, the goal is still to track the motion of an hand. In this case, however, a couple of objects show up on the scenes which may cause issues of occlusion and clutter.



Here are two trajectories obtained with no motion model (on the left) and constant velocity model (on the right). Default parameters were used for these experiments.

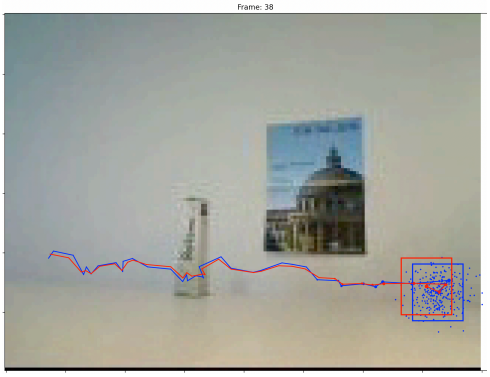
In the latter case, we can observe that the mean state a priori (blue line) is lower with respect to the mean state a posteriori (red line). This is probably due to the dubious initialization of the velocity in the motion model. The components of the velocity with the default parameters were set to $\mathbf{v_x} = 1$ and $\mathbf{v_y} = 10$, although the hand barely moves along the y-direction.

By initializing $\mathbf{v_x} = 1$ and $\mathbf{v_y} = 0$, a more reasonable result is obtained for the mean-state a priori.



In the figure on the left, the result obtained with the constant motion model, when setting the initial velocity vector:

v = [1 0]

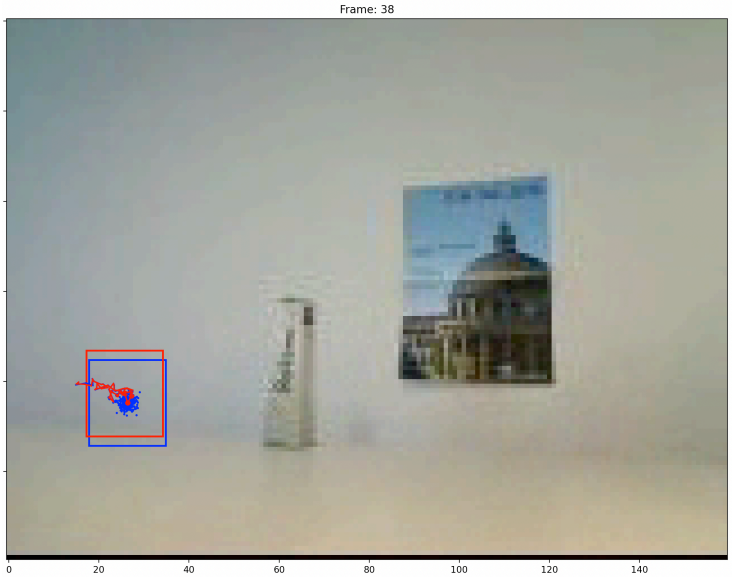


After having fine-tuned the notion model, it is possible to obtain an even smoother and linear tracking, as shown in the figure on the left

ANALYSIS OF SYSTEM NOISE

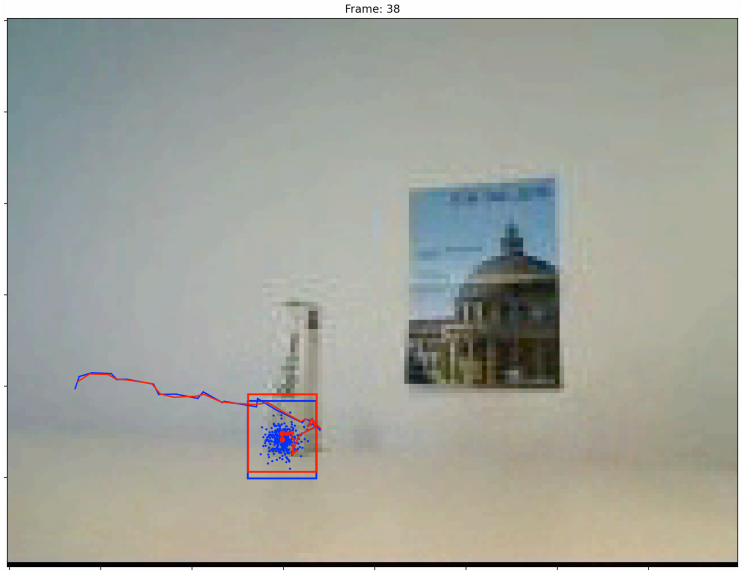
It's now interesting to analyze how the tracking performances vary when changing the **system noise**. Our only hyperparameter to tune the system noise is the standard deviation of the Gaussian distribution modeling the noise itself. If we take into consideration the no motion model, we can then tune σ_p, which by default was set to 15.

Let's start by setting *sigma position* = 1. In this case, the standard deviation of the noise is so low, that the particles fail to move from the starting point. It's important to remember that the noise mean is set to zero. Therefore, with a low standard deviation, most of the times the particles tend to stay put, in particular when using the no motion model.



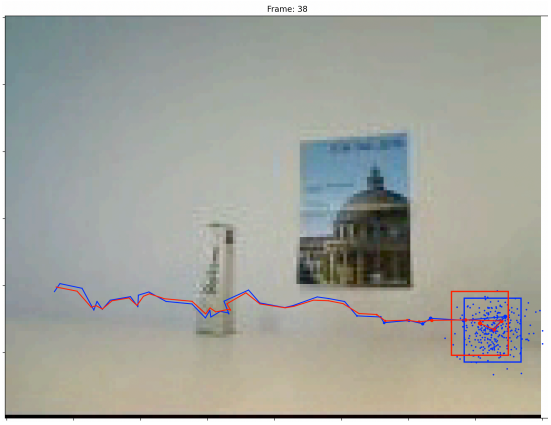
In the figure on the left, the result obtained with the no motion model when setting $\sigma_p = 1$.

An interesting result is obtained by setting *sigma position* = 2, while using the no motion model. Tracking fails in presence of occlusion. As soon as the hand disappears, the tracker gets stuck on the occlusive object.



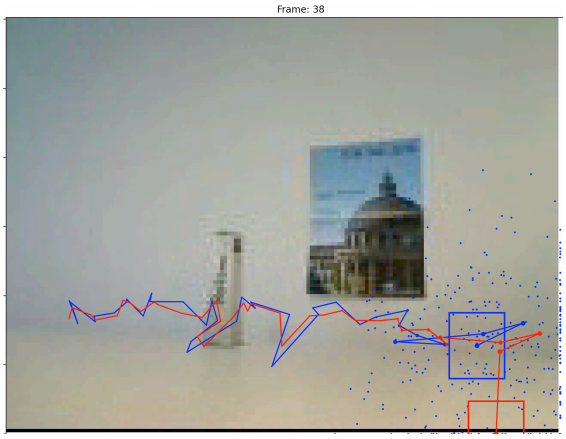
In the figure on the left, the result obtained with the no motion model when setting $\sigma_p = 2$.

Setting *sigma position = 5* seems to yield smoother result: the tracking is correct and does not fail in presence of occlusion and clutter. Moreover the trajectories of the mean-state look cleaner and smoother.



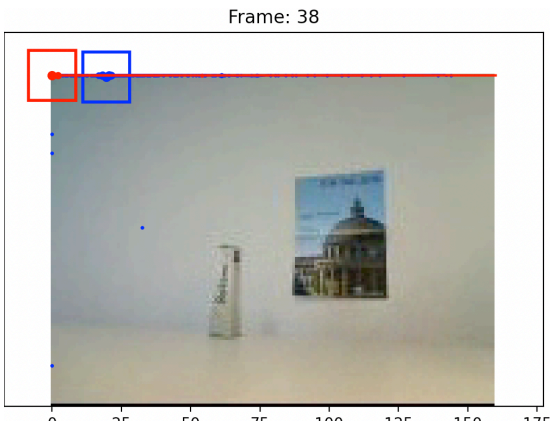
In the figure on the left, the result obtained with the no motion model when setting $\sigma_p = 5$

Increasing *sigma position = 20 can already lead to the complete failure of the tracking system. However, most of the times the generated trajectories look irregular and erratic.*



In the figure on the left, the result obtained with the no motion model when setting $\sigma_p = 20$

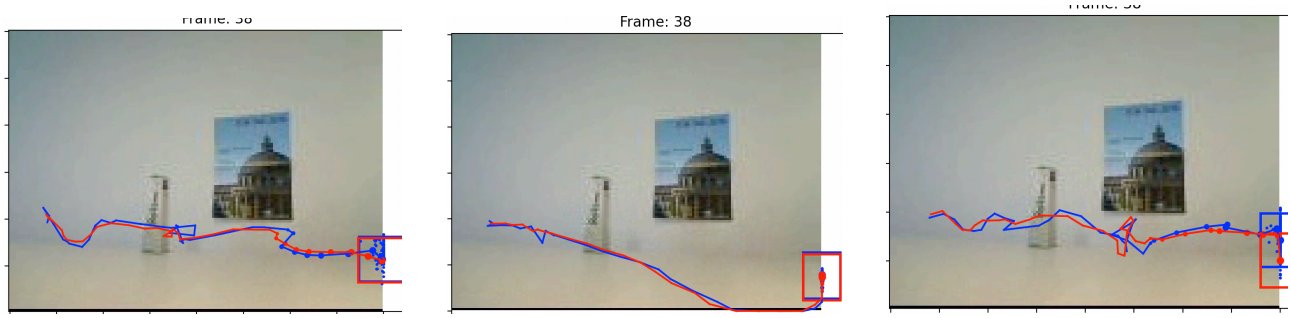
Finally, setting sigma position = 50, which is an extremely elevate value for the modeling of the system noise yield to the complete failure of the tracker. The particles are basically sampled at random and most of the time they are picked outside the frames. Since we have forced the particles to stay inside the boundaries of each frame, the boxes are usually located along the borders of the frame.



In the figure on the left, the result obtained with the no motion model when setting $\sigma_p = 50$

Overall, there is a clear **tradeoff** when setting the system noise parameter. On one hand, setting sigma too low may lead the tracker to fail since it may not keep up with the movement of the object to track. Occlusion and clutter issues usually worsen this problem. On the other hand, setting sigma too high may lead to erratic and unforeseeable tracking trajectories, as the particles are basically sampled at random. The deterministic component of the system becomes irrelevant if compared to the noise.

The analysis on the system noise can be performed also with the constant velocity model, where we have initialized the velocity vector v = [1,0], since the hand barely moves on the vertical direction. In this case, as outlined before, we have defined the standard deviation of the position noise as $\sigma_{\mathbf{p}}$, while we will call $\sigma_{\mathbf{v}}$ the standard deviation of the velocity noise. Similar consideration on the system noise can be made. There is, however, a small difference. First, since the deterministic part of the system model allows the particles to move (thanks to the velocity), tracking ***often*** works fine even when $\sigma_{\mathbf{p}}$, $\sigma_{\mathbf{v}}$ are low.



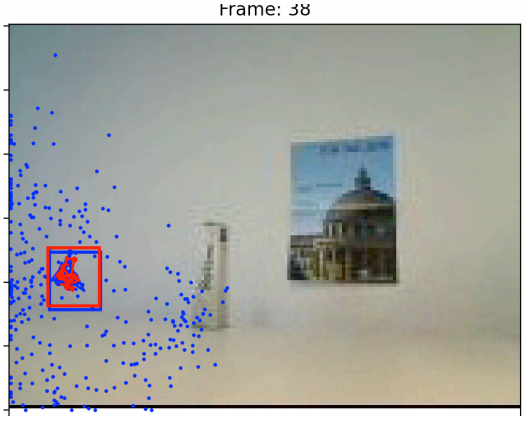
Here are three results obtained with the constant motion model, setting $\sigma_{\boldsymbol{p}}$, $\sigma_{\boldsymbol{v}}$ **=** (1,1). *This setup often yield good results.*

When σ_{p} , σ_{v} are assigned high values the results instead are the same of the no motion model, yielding to the total failure of the tracker. In particular, it's important to avoid choosing an high-value for the velocity noise, since the particles will tend to explore different direction, therefore resulting in ineffective tracking.

ANALYSIS OF MEASUREMENT NOISE

The hyperparameter *sigma observe* is involved in the computation of the weights and it is therefore a significant hyperparameter to be tuned. As it is shown in the following formula, we can basically interpret *sigma observe* as the length-scale of an RBF kernel. If *sigma observe* assumes a small value, basically all the particles end up having small weights, whereas if *sigma observe* assumes a high value, all the particles have high weights. Moreover, numerical issues related to the exponential may occur if *sigma observe* is too small.

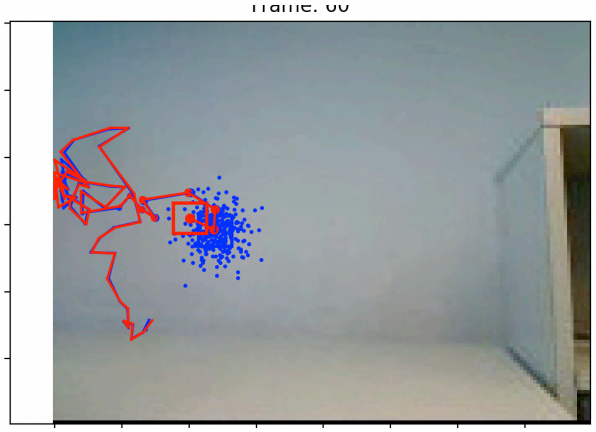
$$\pi^{(n)} = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{\chi^2(CH_s{}^n, CH_{target})^2}{2\sigma^2}}$$



In the figure on the left, the result obtained with the no motion model when setting σ(sigma_observe) = 1

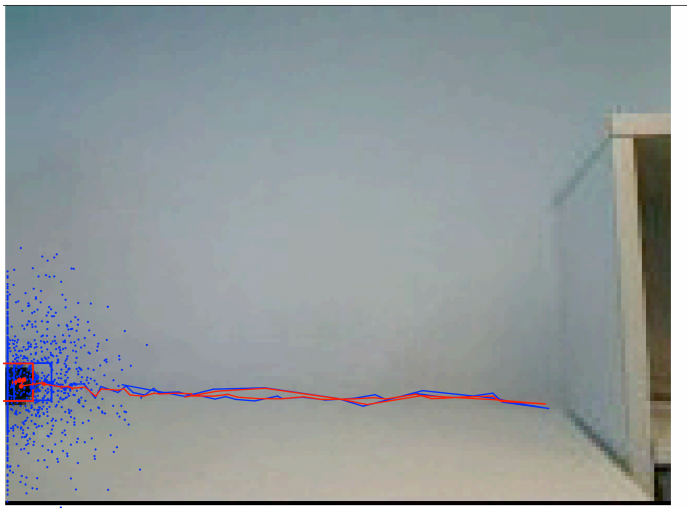
VIDEO 3

The third clip consists in tracking a ball bouncing off a wall. After having tried to implement the tracker with the previous optimal parameter (no motion model), I have found out that most of the times tracking fails. Tracking in this video is in fact a more complex task due to the fact that not only the velocity of the ball changes, but it also changes direction after the bounce. Moreover there are a few spots in the images that are chromatically similar (in terms of RGB values) to the ball. The reason for this failure lies in the fact that the no motion model, with low noise standard deviation ($\sigma_p = 5$), is simply unable to keep up with the speed of the ball.



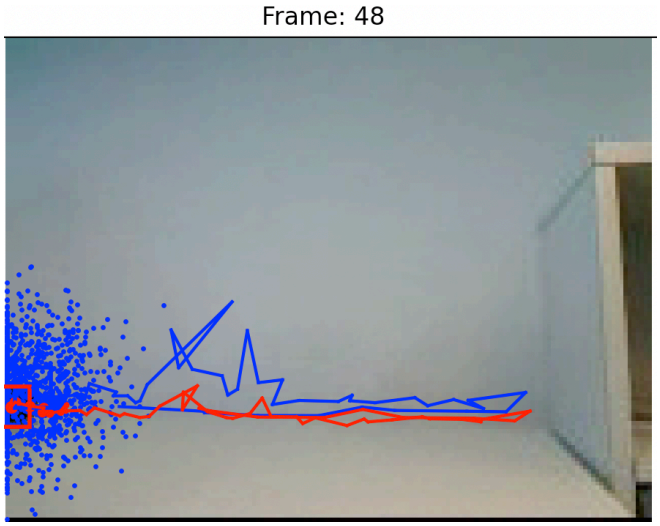
In the figure on the left, the result obtained with the no motion model optimized with the parameters of the previous video ($\sigma_p = 5$).

Using an increased number of particles (1000) and an increase noise position, we manage to achieve the correct tracking.



In the figure on the left, the result obtained with the no motion model using 1000 particles and $\sigma_p = 10$.

In this case, I preferred to use the no motion model, since the constant velocity model does not theoretically fit well. As outlined before, the velocity of the ball is not constant and also changes direction after the bounce. After having performed different experiments, the **constant velocity motion** may still work, but its results are a little bit more unstable.



In the figure on the left, the result obtained with the constant velocity model using 1000 particles, $\sigma_p = 10$ and $\sigma_v = 1$.

For the system noise and the measurement noise the same observations of video2 can be made.

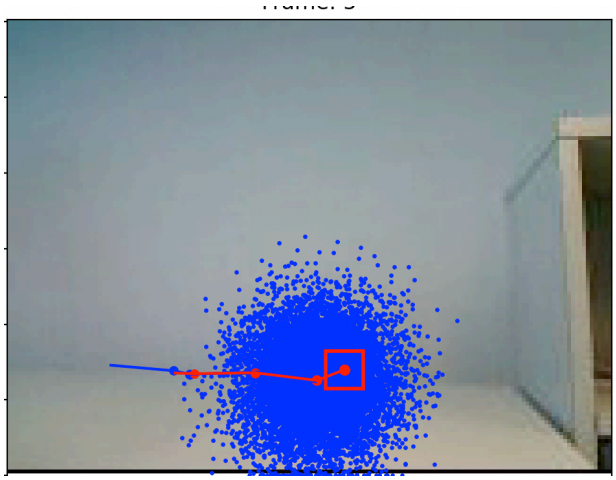
Once again there is a **tradeoff** when setting the **system noise** parameter. On one hand, setting sigma too low may lead the tracker to fail since it may not keep up with the movement of the ball. On the other hand, setting sigma too high may lead to unpredictable tracking trajectories, as the particles are basically sampled at random. The deterministic component of the system becomes irrelevant if compared to the noise. In this case, the change in the velocity of the ball makes the problem more complicated and the choice of the sigma parameters becomes more critical. An optimal choice of these parameters should guarantee good tracking performances before and after the bounce of the ball.

For what concerns the **measurement noise** parameter, the same reasoning for Video 2 applies now. First of all, choosing *sigma_observe* too small often results in numerical problems related to the exponential. If that issue may be solved thanks to increased machine precision, it would still yield incorrect results: with a low sigma, basically all of the particles are assigned minimal weights. On the other hand, if *sigma_observe* is chosen too big, the model is unable to distinguish which particles have an histogram similar to the target. The similarity measure is simply not as effective with an unreasonable choice of sigma_observe, leading to all the particles to have very similar weights.

QUESTIONS

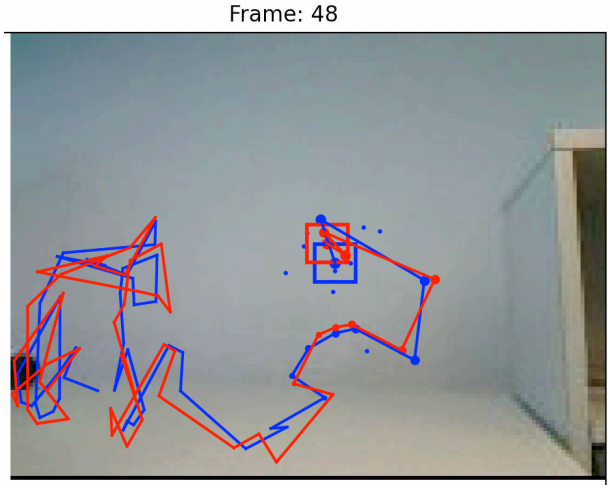
1. What is the effect of using more or fewer particles?

Increasing the number of particles yields improved accuracy of our tracker. Sampling more particles basically means to explore a greater area of the frame. By looking into a wider area of the frame, it becomes unlikely to miss the object to track. On the other hand, the algorithms becomes computationally more expensive. Time performance of the condensation algorithm decays.



In the figure on the left, an example of what happens with 10000 particles. The blue dots are spread on a wide area, covering a significant portion of the frame. The algorithm however becomes significantly slower.

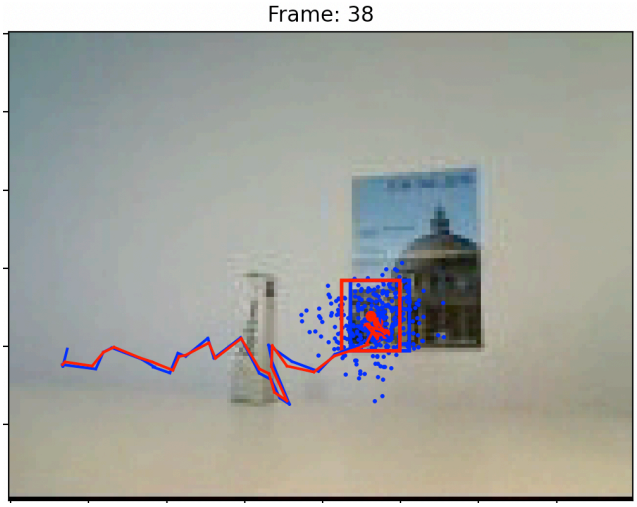
If the number of particles becomes too small, however, tracking is going too fail because of the lack of statistical significance. The condensation algorithm is based on sampling particles in those areas which are considered more likely to be covered by the object. With fewer particles it can easily happen that none of them actually follows the object, thus resulting in the breakdown of the tracker.



In the figure on the left, an example of what happens using 10 particles. The mean state is calculated only averaging among 10 samples. With such a low number of particles, the average is not as effective.

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

The number of bins chosen to form the histograms also plays an important role in the algorithm. By choosing a small number of bins, the tracker is in fact very likely to fail its task. We can interpret the number of bins as an instrument sensitivity: if the bins are not enough, the condensation algorithm may not be able to differentiate areas with similar RGB values.



In the figure on the left, an example of what happens using 2 bins. The clutter of the poster causes the tracker to fail. The histogram approximation becomes too coarse, leading to the inability to track the hand.

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

The advantage of updating the model, setting $\alpha > 0$, is that we allow the algorithm to update our target histogram, not only relying on our initial, self-outlined, bounding box. The condensation algorithm proposed in the assignment allows us to update the target color histogram as a convex combination between the old color histogram of the target and the color histogram of the mean state:

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

This may be useful whenever **illumination changes** is a factor throughout the video. If our object slightly changes its color due to illumination changes, by updating the target histogram, the tracker is able to keep up with the pixels changing their RGB values. The model is more sensitive and robust to chromatic changes in the motion.