8 Computer Vision – Pierre Beckmann

8.1 Implementation

In this part I will go quickly through the main steps of the implementation of a Condensation tracker. We tried to track an object in a given bounding box using random particles that we give different weights specified by a gaussian that uses the a similarity function using histograms and resample them accordingly.

8.1.1 Color histograms

In this function we compute the normalized histogram of RGB colours in a frame within a bounding box defined by $[x_{min}, x_{max}]$ x $[y_{min}, y_{max}]$. For each colour channel we call the matlab function `imhist()` which returns a histogram and obtain 3 separate histograms fig1b. We put them one after the other into a vector and normalize it to obtain our desired histogram fig1c.

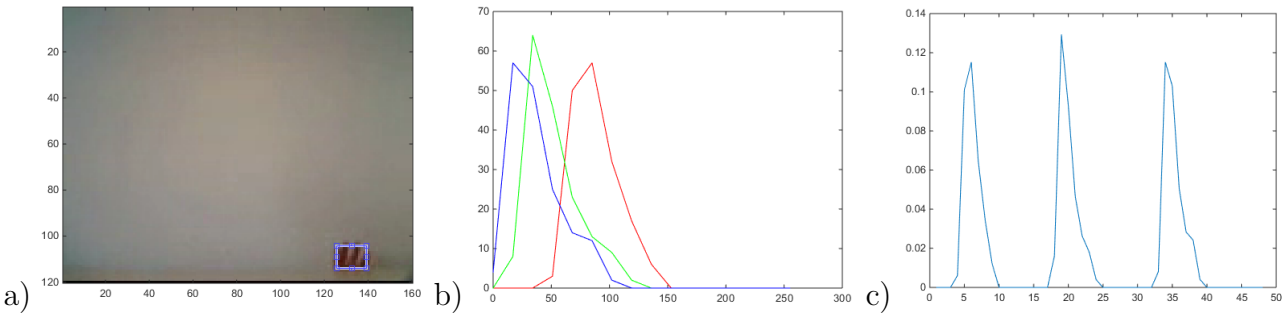


Figure 1: **Color histogram.** a) Chosen bounding box. b) Correspondent histograms for each colour channel. c) Stacked and normalized histogram.

8.1.2 Derive matrix A

Next we derived the dynamic matrix A for no motion (i) and for constant velocity (ii). This matrix propagates each sample s_{t-1} to s_t by a linear stochastic differential equation $s_t = As'_{t-1} + w_{t-1}$. With w being the stochastic component acting as the noise. This is what we derived:

No motion (i):

$$\begin{cases} x_t = x_{t-1} + w_{x,t-1} \\ y_t = y_{t-1} + w_{y,t-1} \end{cases} \implies A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Constant velocity (ii):

$$\begin{cases} x_t = x_{t-1} + \dot{x_{t-1}}dt + w_{x,t-1} \\ y_t = y_{t-1} + \dot{y_{t-1}}dt + w_{y,t-1} \\ \dot{x_t} = \dot{x_{t-1}} + w_{\dot{x},t-1} \\ \dot{y_t} = \dot{y_{t-1}} + w_{\dot{y},t-1} \end{cases} \implies A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We set $dt = 1$ so our velocities will be in per frame.

8.1.3 Propagation

Using the computed A matrix we implemented the next function wich proagates the particles with a given prediction model and the system model noise with a standard deviation for both the position and the volicity. We also make sure all the particles are in the frame and put them on the closest point on the border if they get out.

8.1.4 Observation

This function updates the weight of the particles. For each particle we compute the similarity of the corresponding histograms χ^2 and use it to compute the weight:

$$\pi = \frac{1}{\sqrt{(2\pi\sigma_{obs}^2)}} e^{-\frac{\chi^2(CH_{curr}, CH_{target})^2}{2\sigma_{obs}^2}}.$$

With $CH_{target} = (1-\alpha)CH_{target} + \alpha CH_{E[s_t]}$, σ_{obs} the standard deviation of the observation, $E[s_t]$ the mean state and χ^2 a given function that computes the distance between the two histograms. Finally we normalize the particle weights.

8.1.5 Estimation

This function estimates the mean state of a set of particles by suming the multiplication of the weights and the position for each coordinate.

8.1.6 Resampling

Next we implemented the function that resamples the particles based on their weights. We decided to use the low variance resampler already used in the previous assignement 3.

8.2 Experiments

In this part the experiments are shown. Only parameters that were changed from the original parameters are specified (to avoid listing them again for each video).

8.2.1 Video1

Overall the hand is tracked pretty good but it tends to track the wrist (or middle section of the arm) maybe because the lighting changes slightly through the video.

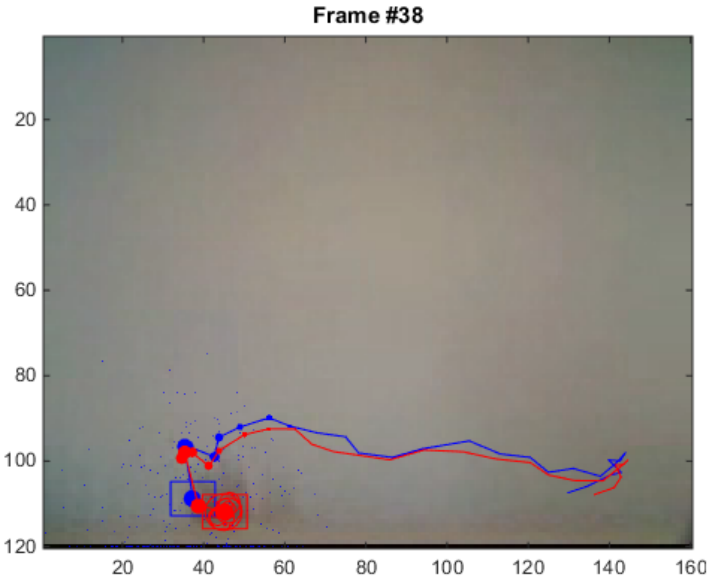
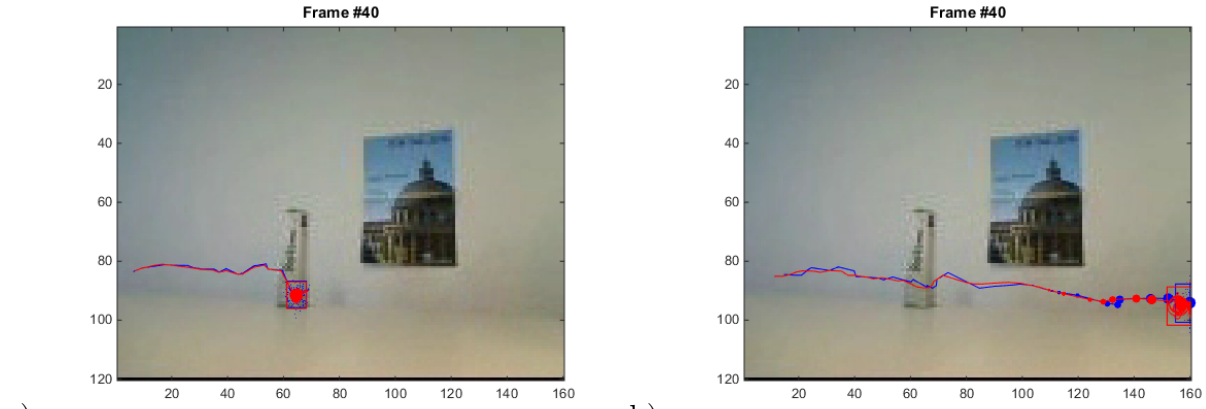


Figure 2: **Video 1.** Blue is a priori mean state and red is a posteriori mean state. For this tracking we used the given parameters of the assignement.

8.2.2 Video2

In this part we varied the paramters in order to have a standard deviation for the position that leads to a narrower cloud of particles than the ocluding object. Therefore we chose $sigma_{position} = 2$ (instead of 15 in the initial paramters). A velocity of 8 pixels/frame in the x direction was chosen for the prediction model 2 (ii).

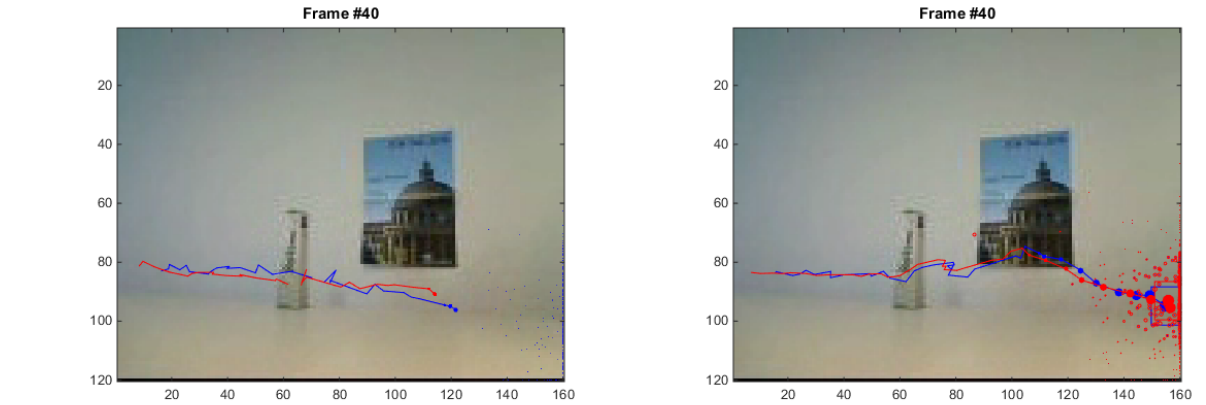


a) b)

Figure 3: **Video 2.** Blue is a priori mean state and red is a posteriori mean state. a)First prediction model: no motion. b) Second prediction model: constant velocity motion, with a velocity of 8 pixels/frame in the x direction.

We observe that the tracking only works with the constant velocity motion prediction model. This seems logical because the hand has more or less a constant velocity and this way we allow the tracker to get trough the occluding object. We also notice that a big system noise for the position enables the tracker to work even without the constant velocity model if the nois is big enough to generate a larger particle cloud than the occluding object.

Now when we change the standard deviation of the measurement noise we decide how wide spreaded the weights are. With a very small $\sigma_{observe}$ the tracker is not very flexible to changes in the bounding box because it only weighs the particles with a very good distance. The smaller the standard deviation the scricter is the tracker. Therefore when the hand passes before the poster it can't succesfully track the hand whith a very small standard deviation because the background changed. Therefore it stays under the map. In case a the tracker was too strict and lost the hand when it reached the poster, in case b it was more flexible and tracked the hand even better than with initial parameters when it passes before the poster:



a) b)

Figure 4: **Video 2.** Blue is a priori mean state and red is a posteriori mean state. With $\sigma_{position} = 15$, $V_x = 8$ and $\sigma_{observe} = 0.008$ for a and $\sigma_{observe} = 0.5$ for b.

8.2.3 Video3

When tracking the ball witht the same parameters that worked for video 2 ($\sigma_{position} = 2$ and $V_x = 8$) we succesfully identify it for the way to the wall from the left to the right but we loose track of it on it's way back when it bounces of the wall fig5. This makes sense because for the first part the estimated velocity is at least in the right direction before it bounces of the wall but then it is in the opposite direction of the ball's motion.

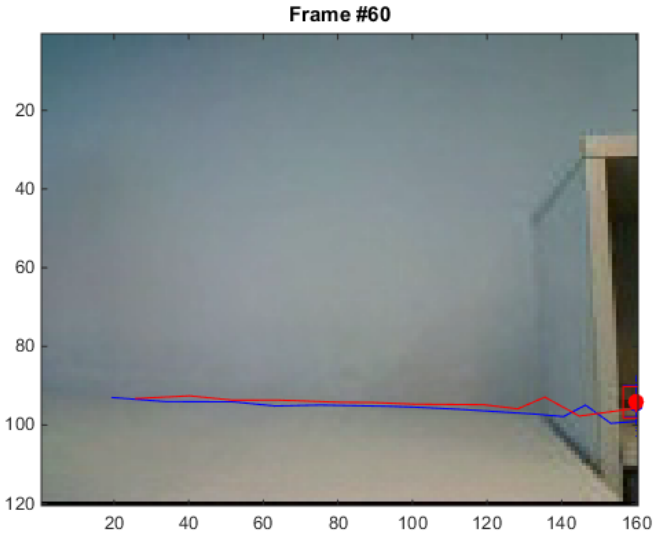


Figure 5: **Video 3.** Blue is a priori mean state and red is a posteriori mean state. $\sigma_{position} = 2$ and $V_x = 8$ with prediction model 1.

However the ball is succesfully tracked through the whole sequence with initialy given parameters and prediction model 0:

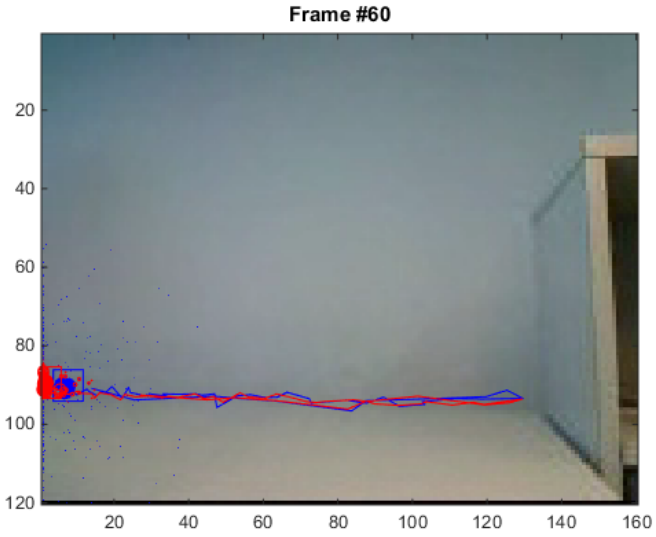


Figure 6: **Video 3.** Blue is a priori mean state and red is a posteriori mean state.

Another option would have been to use model 1 with a small initial speed in x direction and a high standard deviation for the velocity.

8.2.4 Additional questions

Number of particles Increasing the number of particles means a more accurate mean state and therefore a more fluid motion and better quality of the tracking:

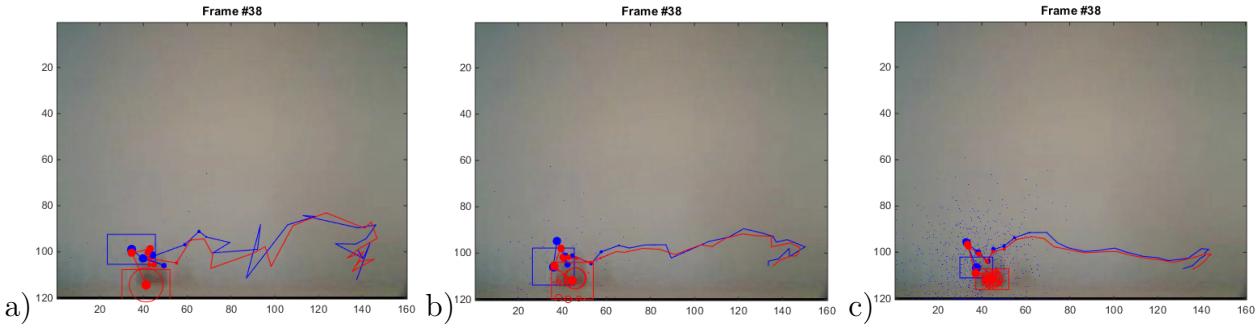


Figure 7: **More or fewer particles.** Blue is a priori mean state and red is a posteriori mean state. Initial paramters with a) 10 particles, b) 100 particles and c) 1000 particles.

Number of histogram bins Increasing the number of bins for the histogram increases the overall quality of the tracking. However even for only 3 bins the result is not so bad:

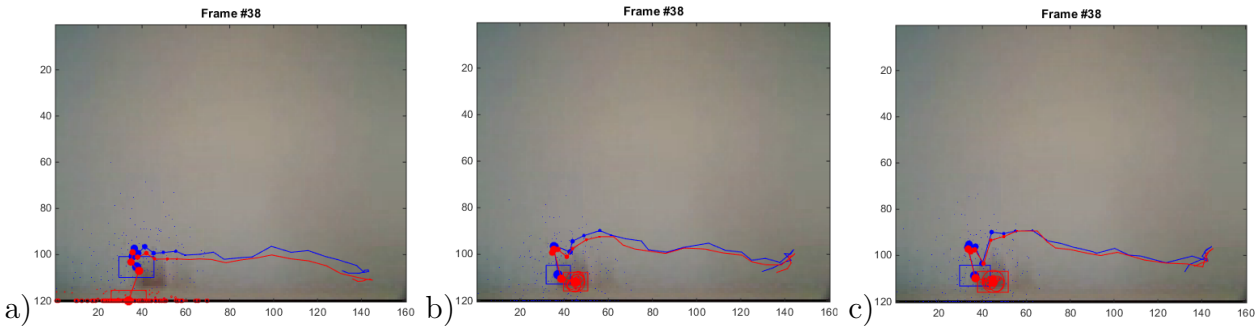


Figure 8: **More or fewer histogram bins.** Blue is a priori mean state and red is a posteriori mean state. Initial paramters with a) 3 histogram bins, b) 16 histogram bins and c) 100 histogram bins.

Between 16 and 100 bins the difference is very small so even though the number of bins improves the quality it seems like we have a logarithmic evolution (or a threshold reaching situation) of tracking quality in fuction of the number of bins.

Appearance model updating with α When increasing α the tracker becomes dynamic and can adapt to luminosity changes for example, however in the case of an occlusion it can't recover if it looses track of the object.

8.2.5 Own video

For my own video I decided to track a ball in a football penalty kick. The tracking was very tricky because white football shoes and shirts and the white lines are in the ball trajectory and were confused with the ball. This is why I decided to take the velocity model with $V_x = -10$pixels/frame and $V_y = -1$pixels/frame. Then I had to find the optimal $sigma_{observe}$ and α to let the tracker adapt to the background. The optimum was: $sigma_{observe} = 0.2$ and $\alpha = 0.1$. Also because the tracked object is small in the complete frame I chose a relatively small $sigma_{position}$ of 3.

Figure 9: **My own video.** Blue is a priori mean state and red is a posteriori mean state. Given paramters with following changes: model 1, $sigma_{position} = 3$, $V_x = -10$, $V_y = -1$, $sigma_{observe} = 0.2$ and $\alpha = 0.1$.

In this case the difficulty when tracking in the ball comes from all the white lines on the field aswell as players equipments. Other than changing th color of the ball it would be interesting to make sure that the tracker constantly follows the ball by introducing shapes. This way it wouldn't follow a line or a shirt because they don't have a round shape. This could be done with feature extraction (Harris corners) for example to get the contour of the object. This way we could maybe even track the ball without having to set an initial velocity which would be much more interisting for using it in the football world.