Computer Vision Assignment 11: Condensation Tracker

Charlotte Moraldo

1. CONDENSATION Tracker Based on Color Histograms

1.1 Color Histograms

In order to compute the desired normalized histogram, I begin by extracting the bounding box from the frame, after checking that is within the frame. If it isn't, I just crop the bounding box so that it fits inside the frame. I then compute the histograms using the Matlab function `imhist` and make sure to specify the number of bins `hist_bin`. This is performed three times, for each of the color channels: R, G, and B. The 3 resulting histograms are then concatenated in a vector returned in the variable `hist`.
In figure 1 below, we can see an example of the plot of a color histogram:

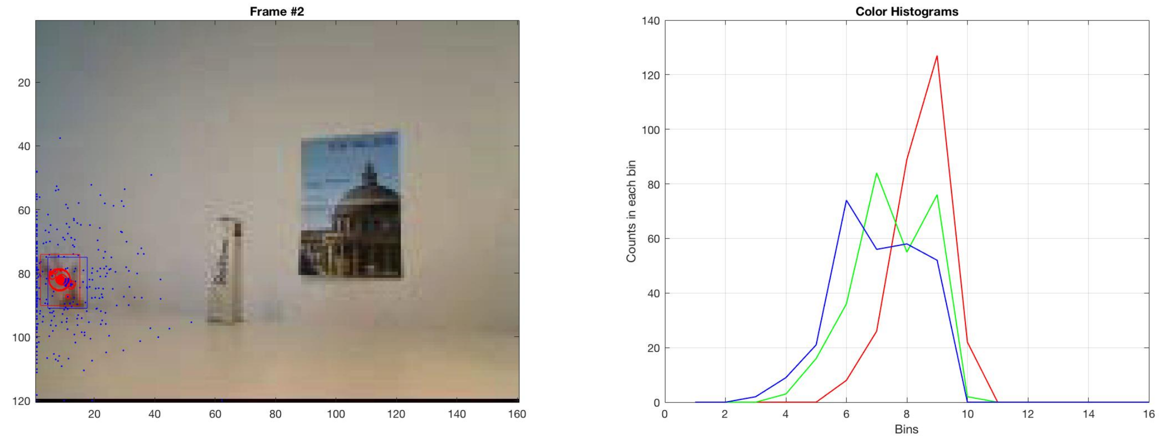


Figure 1: Example of a color histogram: left shows the frame with the bounding box, right shows the color histogram.

1.2 Derive matrix A

The dynamic matrix A allows to go from the old state s_{t-1} to the new state s_t following the linear stochastic differential equation: $s_t = As_{t-1} + w_{t-1}$, where w_{t-1} is the stochastic component. In this exercise, we consider two prediction models: no motion at all (i.e. just noise), and constant velocity model.

In the first model (no motion), only the samples' position varies. As we are in a 2D environment, the coordinates x and y are considered, and the dynamic matrix is therefore of dimension 2x2. As the samples are not supposed to move, we can conclude that this matrix is simply equal to the identity:

$$A_{nomotion} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

In the second model (constant velocity), the position and the velocity of the samples must be taken into account, which corresponds to: x, y, v_x, and v_y. A is therefore of sixe 4x4. Since the velocity is constant, the lower part of the matrix is identity. However, the position changes according to the following formula:

$$v = \frac{\Delta x}{dt} = \frac{x_1 - x_0}{dt} \implies v \cdot dt = x_1 - x_0 \implies x_1 = x_0 + v \cdot dt.$$

The new position is equal to the old one plus the time elapsed dt times the speed in the corresponding direction. In the case of this exercise, dt is set to 1. Therefore:

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

1.3 Propagation

In the function `propagate`, the linear stochastic differential equation $s_t = As_{t-1} + w_{t-1}$ has to be implemented. I begin by defining the dynamic matrix A as derived previously, and then define the stochastic element w_{t-1}, for both models (no motion / constant velocity). We know that the noise is normally distributed: $w_{t-1} \sim \mathcal{N}(0, \sigma^2)$, and we are given the position variance $\sigma_{position}$ as well as the velocity variance $\sigma_{velocity}$ (and since it is noise, $\mu = 0$).

- In the case of the no motion model, w_{t-1} can therefore be defined as: $\begin{bmatrix}\sigma_{position} & \sigma_{position}\end{bmatrix}$, multiplied by a random element following the normal distribution $\mathcal{N}(0, 1)$, defined in Matlab by `randn`.
- Similarly, in the case of the constant velocity model, w_{t-1} is defined as: $\begin{bmatrix}\sigma_{position} & \sigma_{position} & \sigma_{velocity} & \sigma_{velocity}\end{bmatrix}$, multiplied by a random element following the normal distribution.

Finally, before returning the propagated particles, I jsut check that they are contained within the frame. If it is not the case, instead of discarding them I simply move them to the edges of the frame.

1.4 Observation

The function `observe` computes a weight for each particle. In order to do so, for each particle I compute the color histogram in a bounding box of height H and width W, centered in the position of the particle. Then, the weight π is computed using the following formula:

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

Where:

- $CH_{particle}$ is the color histogram in a $[W, H]$ bounding box around the particle
- CH_{target} is the color histogram of the target we stored
- σ_{obs} is the variance of a Gaussian which defines the probability of each sample
- The χ^2 distance is computed with a provided function `chi2cost`

Finally, I normalize the weights before returning them in the variable `particles_w`.

1.5 Estimation

In order to estimate the mean weight of the particles, I simply compute: $E[s_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}$ by summing the dot product of the particles and their weight. The dot product allows to multiply each particle by its corresponding weight. As we've normalized the weights while computing the observations, there is no need to divide this by the sum of the weights.

1.6 Resampling

In order to resample, I used the same technique (low variance) as in the particle filtering assignment (assignment 3). I simply adjusted some parts of the algorithm for our problem, which is in 2D (while assignment 3 was in 1D). For more details, please refer to assignment 3.

2. Experiments

2.1 Video 1

The first video shows a moving hand with a uniform background, which we try to track. The following pictures illustrate the obtained results:

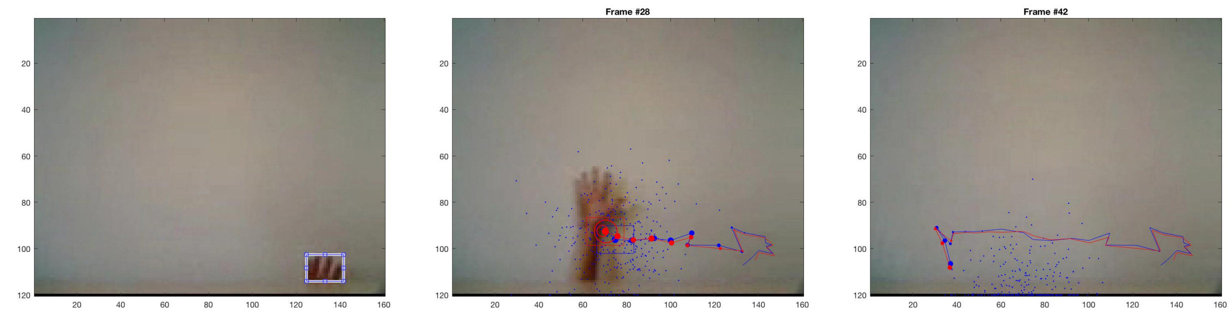


Figure 2: Tracking on a uniform background. Left image shows the initial target box, middle image shows a frame in the middle of the movement (frame 28), right image shows the end result of the tracking (frame 42).

The blue points are the a priori particles, and the blue line is their mean state. Similarly, the red points are the a posteriori particles and the red line their mean state.
We can see that the tracking is quite good, as the hand is being followed. However, we can see that even if we set the initial target box to the tip of the fingers, the tracker ends up following the wrist or the upper part of the forearm. This is due to the fact that the initial target box in the first frame the fingers have high contrast, and as the hand moves the lights and shadows change. The fingers become brighter and the right part of the forearm gets more contrast. This could be improved for example by varying α in the update of the color histogram of the target: $CH_{target} = (1-\alpha) \cdot CH_{target} + \alpha \cdot CH_{E[s_t]}$. In Fig.2 below, α is set to 0.5 ($\alpha = 0$ in Fig.1), and as we can see, the result is already better:

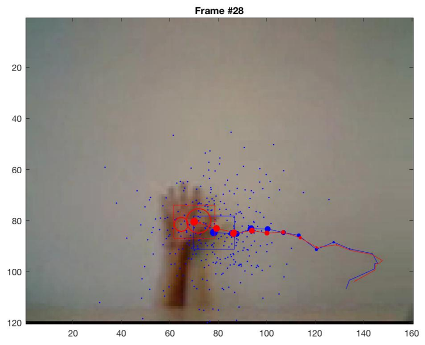
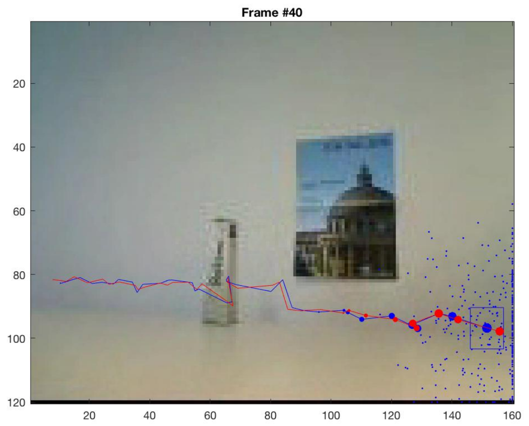


Figure 3: Frame in the middle of the movement (frame 28), with $\alpha = 0.5$

2.2 Video 2

We now track a hand in motion with some clutters and occlusions. Firstly, the model with no motion is used.



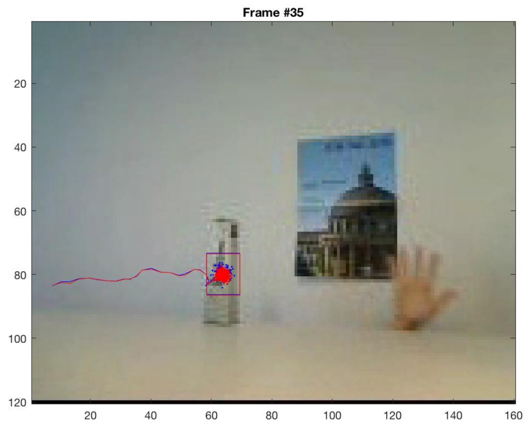


Figure 4: Tracking with clutters and occlusions. The left image shows shows the end result of the tracking, with large system noise ($\sigma_{position} = 15$). The right image shows the tracking with a reduced system noise ($\sigma_{position} = 1$)

We can see in the left image of Fig.3 than even though the hand is being occluded by an object at some point, the no motion model still manages to track the hand throughout the entire video. This can be explained by the quite high system noise ($\sigma_{position} = 15$): when the hand is occluded, since the stochastic term is quite big, the particles are spread on a large surface. Therefore, when the hand reappears after the occlusion, some particles will still be on the hand and this will allow the tracking to continue. However, when I decrease the system noise to $\sigma_{position} = 1$ (right imag of Fig.3), the particles are much closer to each other, meaning that as soon as the hand disappears behind the object, the particles get stuck on that object and the tracking fails.

In order to overcome this problem, I tried using the constant velocity model:

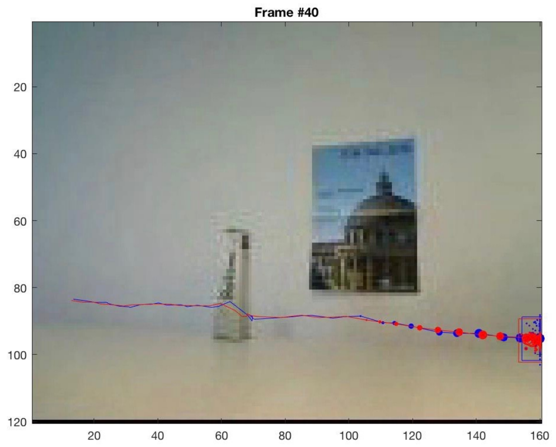
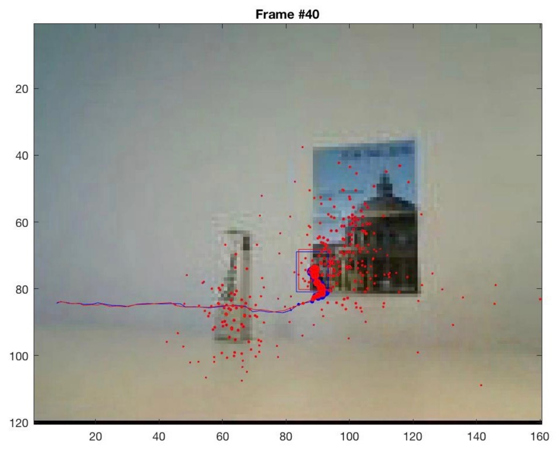


Figure 5: End result of the tracking with constant velocity model. Initial velocity equal to $[5, 0]$, $\sigma_{position} = 1$ and $\sigma_{velocity} = 1$

I decided to set the velocity in the y-axis equal to 0 as the motion of the hand is msotly in the x-axis. We can see that the results presented in Fig.5 are very accurate, even with small system noise. The problem of the particles getting stuck at the occlusion has been solved by using the constant velocity model.

Now, I will test the effect of increasing / decreasing the measurement noise:



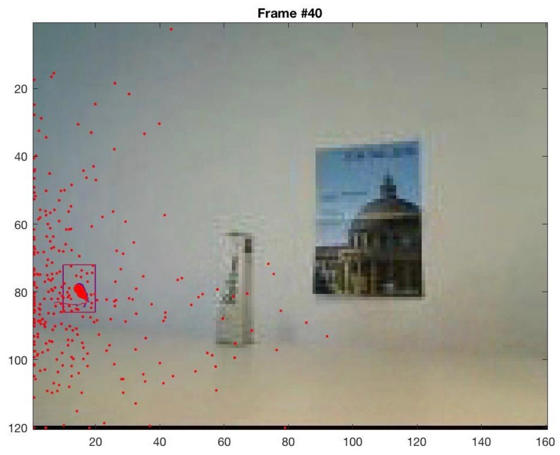
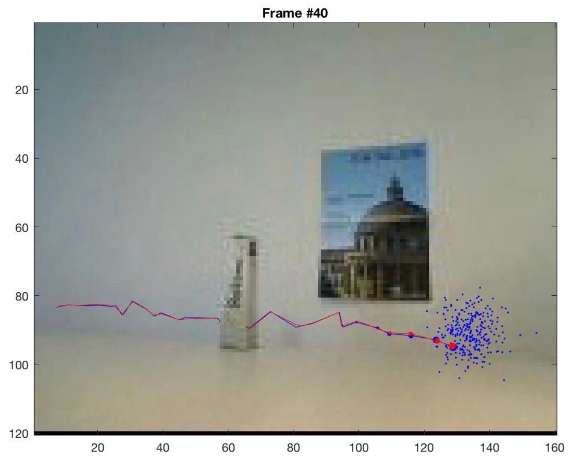


Figure 6: Large measurement noise. Left image: $\sigma_{observe} = 0.6$, right image: $\sigma_{observe} = 1$



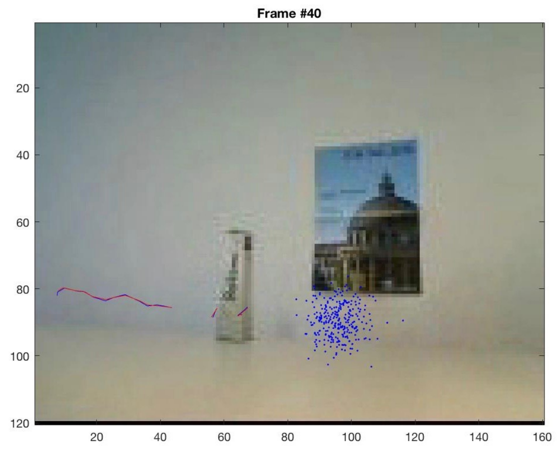


Figure 7: Small measurement noise. Left image: $\sigma_{observe} = 0.008$, right image: $\sigma_{observe} = 0.005$

In the case of a large measurement noise, the weights are widely spread so importance is given even to particles that shouldn't be relevant. All the particles are good candidates. For instance, with $\sigma_{observe} = 0.6$ (Fig.6 left), as soon as the hand is occluded, some particles stay stuck on the object - but the tracking still manages to find the hand when it reappears. But when the background changes, too much importance is given to the poster and the particles start tracking a pattern there: the tracking fails. As we increase further the measurement noise ($\sigma_{observe} = 2$, Fig.6 right), this phenomena is amplified: the particles stay stuck on the uniform wall and totally lose track of the hand.

In the case of a small measurement noise, the weight distribution is very narrow so particles that are slightly different will have a very small weight. Small changes therefore mess up the tracking completely, as it can be seen in Fig.7. With $\sigma_{observe} = 0.008$, when the hand gets occluded the tracking fails momentaneously, but as soon as the hand reappears the tracking continues. When decreasing further $\sigma_{observe}$ to 0.005, this effect is amplified and we can see that the occlusion makes the tracking fail completely. Since the measurement noise is very small, none of the particles are good candidates and we end up with starvation (i.e. the sum of the weights is equal to 0).

2.3 Video 3

The main difficulty of this video is the fact that the ball bounces and therefore its velocity changes direction, unlike the hand which always moves towards the same direction.

Firstly, I tried to perform the tracking while keeping the same parameters as what gave the best result with video 2: small noise, constant velocity model, and constant speed to the right. The result obtained can be seen in Fig.8 below. As we can see, the result isn't as good as with video 2: the moment the ball bounces, the particles keep being updated to the right and they never find the ball again.

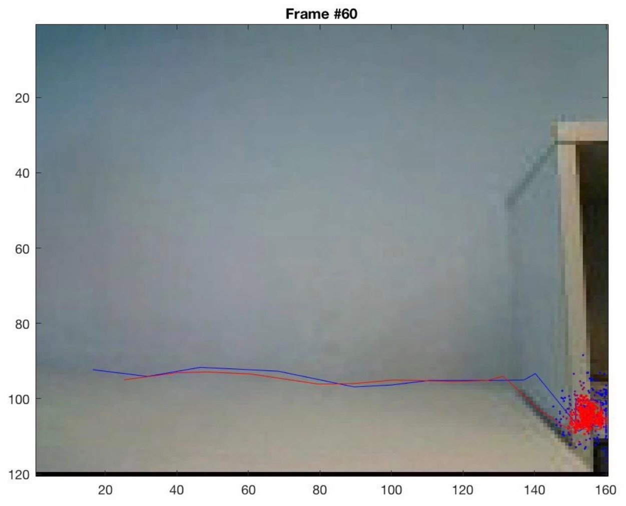


Figure 8: Tracking of a bouncing ball with constant velocity model and speed to the right.

In order to fix this, one idea would be to increase the speed noise $\sigma_{velocity}$, which would allow the particles' velocity to change. The result can be seen in Fig.9. In this case it worked, as in the particles follow the ball as it bounces back, but the speed changes are too random and are therefore sometimes very sudden.

Another idea is to keep $\sigma_{velocity}$ low and instead increase $\sigma_{position}$. This would enlarge the surface where the particles propagate, allowing some particles to still be on the ball after it bounces. The result looks much better (Fig.10).

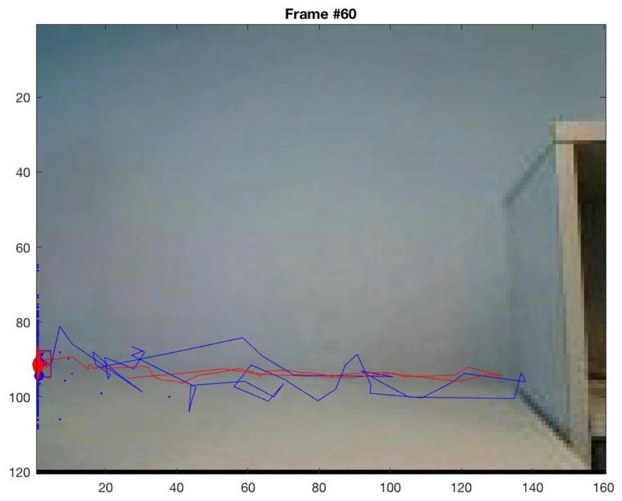


Figure 9: High speed noise: $\sigma_{velocity} = 7$, low position noise: $\sigma_{position} = 1$

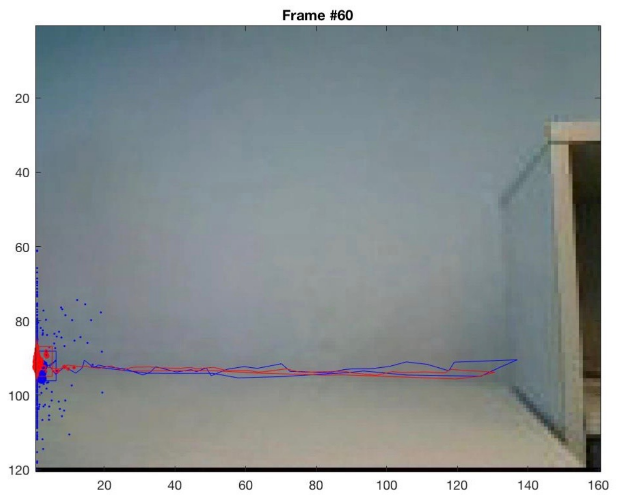


Figure 10: Low speed noise: $\sigma_{velocity} = 1$, high position noise: $\sigma_{position} = 10$

The most robust way to track the bouncing ball is actually simply using the no motion model (see Fig.11). Indeed, the particles don't try to predict the next state and don't get tricked by the sudden change of velocity direction of the ball.

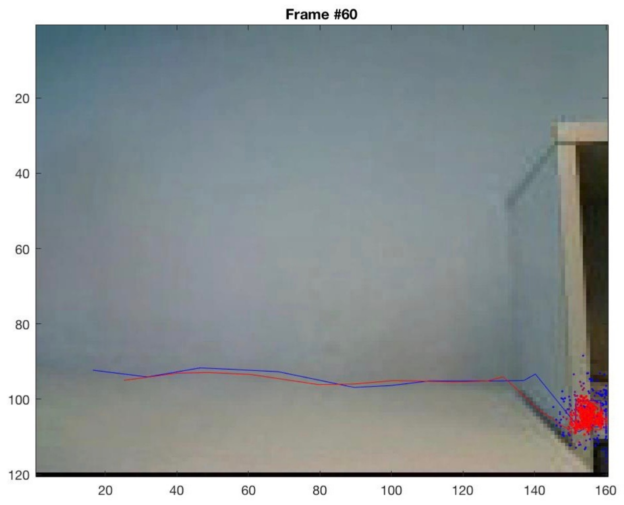


Figure 11: Tracking of a bouncing ball with no motion model

2.4 Study of the effect of other parameters

Firstly, I studied the effect of varying the number of particles. With a lot of particles (1000 particles), the tracking is very robust but the computation time a little longer (Fig.12). As we decrease the number of particles, the behavior is less predictable. For example, in video 2, we aren't sure that there will be enough particles to be able to find the hand after it is being occluded. This is confirmed when we try using too few particles: the tracking fails after the occlusion happens. In Fig.13, 20 particles are used, and we can see that the a priori mean states' evolution is quite jerky, while the a posteriori mean state tracks the hand more stably.

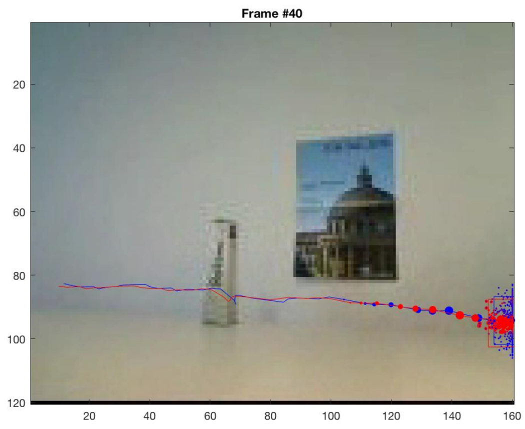


Figure 12: 500 particles

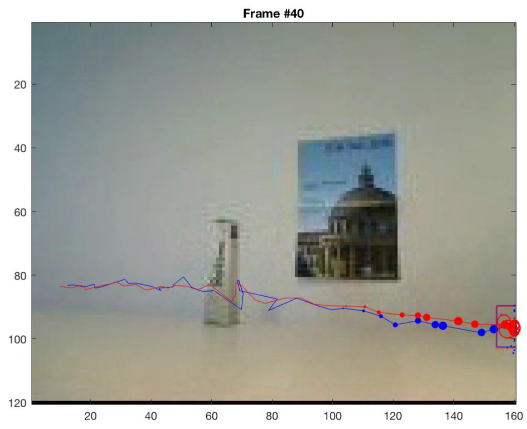


Figure 13: 20 particles

Then, I tried to vary the number of bins of the histogram. With a larger number of bins, the histograms are more precise and enable larger variety. However, too many bins could create a too big gap between 2 histograms that might be very close in reality. A smaller number of bins also has its advantages, as it makes it more robust to noise for example. But with too few bins, the histograms would be too similar and the tracking much less accurate. More precisely, the number of histograms affects how sensitive we are to the colors in the object we're tracking.
For instance, with only 2 bins in video 2, when the hand goes in front of the poster, because the background is very noisy the histograms are not precise at all and they can't determine which particles are important (Fig.14):

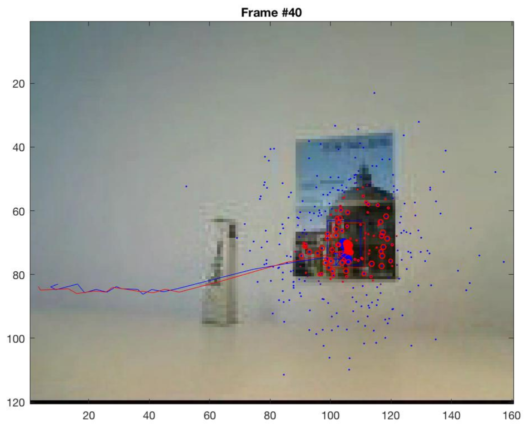


Figure 14: Only 2 bins for the histograms

Finally, I changed the parameter α to allow appearance model updating. In the first video, we already saw that when $\alpha = 0$, the particles track the wrist instead of the hand due to the changes of shadows and lightning. By allowing model updating, the target histogram will not always be the same one as the initial one, instead it adapts to the changes. The result of this has already been presented previously, in Fig.3.

3. Bonus Video

In order to apply what has been seen in this assignment to a more complex environment, I tried using a video I took at the Shibuya crossing in Japan in order to track someone crossing a road in a crowded environment. To do so, I used the constant velocity model with a velocity in the overall direction of the man, and a small system noise was used because the man is quite small in comparison to the size of the frame, and his walking pace is rather constant.
The man I decided to track is wearing a white shirt (Fig.15, bottom left), which is already in itself a challenge since the man constantly cross the white stripes of the crossing, and the particles could easily lose track of the back of the man's shirt. However, this wasn't an issue (see Fig.16).

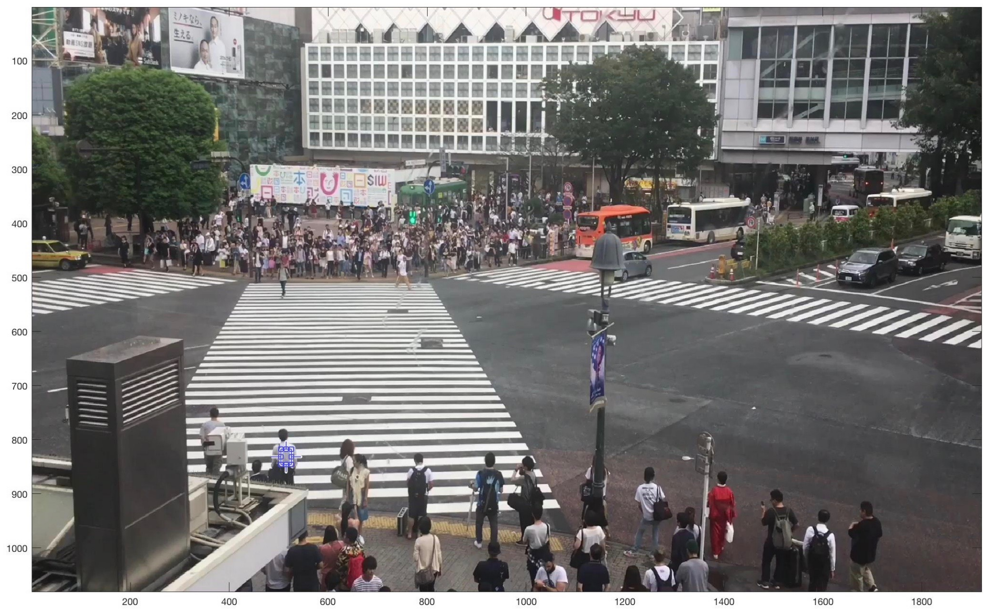


Figure 15: Initial tracking box

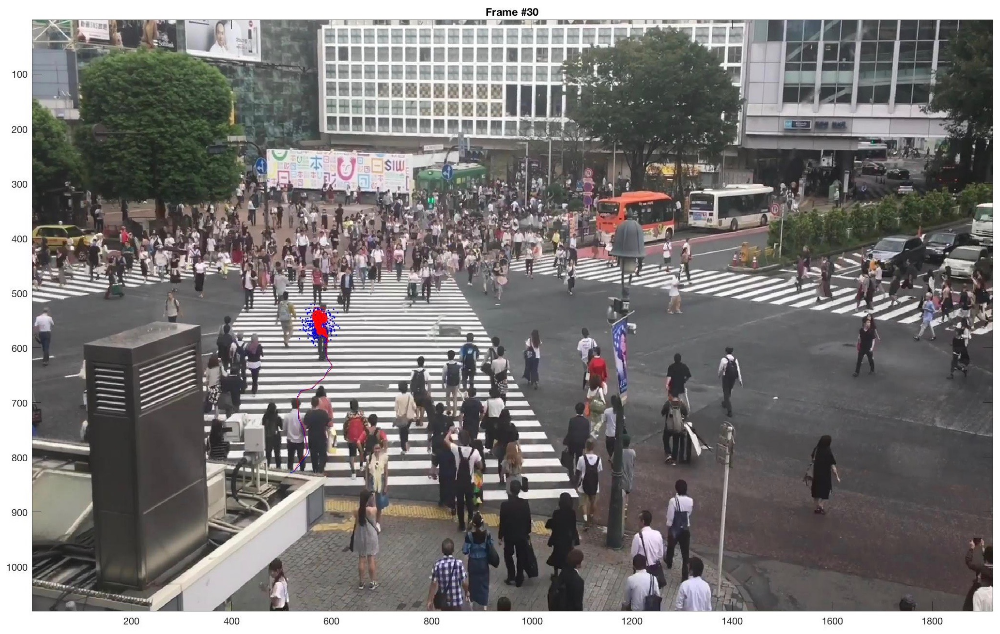


Figure 16: Halfway through crossing

The biggest issue of this problem is the moment where the man gets far away from the camera and begins crossing the paths of other people (around frame 47). As he gets further away, he gets smaller and therefore harder to track, and he crosses other people wearing light-colored shirts. By keeping α equal to zero, the tracker would usually end up getting confused and the particles would start tracking another man in a white shirt getting back towards the camera (see Fig.17 left). A small appearance model was therefore used to allow a little more flexibility: not too small (since the size of the man changes as it gets further away from the camera), but not too big either so that the particles don't get stuck too easily on other elements (such as the crossing stripes or other people's shirt). I therefire set it to 0.5. The result is a little bit more satisfying, as we see that the particles don't come back to us directly but try to keep track of the man, even though they still end up loosing him (Fig.17 right).

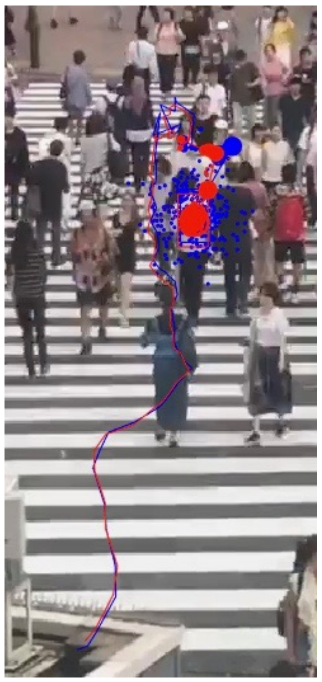
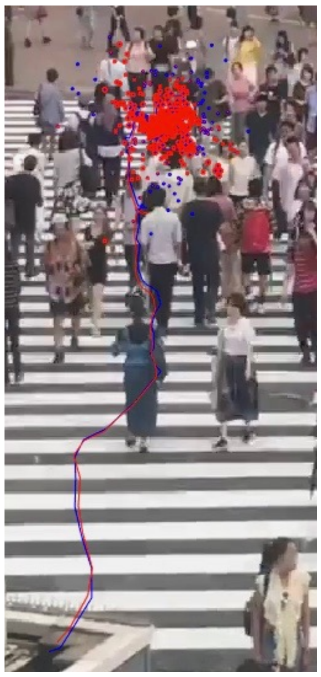



Figure 17

The overall result is quite satisfying. The difficulties faced at the end of the video were expected due to the heavy occlusions and bad quality of the video, but they could be overcome with a more robust algorithm. Tracking the color of the man's shirt isn't enough in order to obtain accurate results. It would be interesting to take into account the shape of what we're tracking, or its motion pattern. Human tracking in a crowded environment is a very complex task, and tracking based on a simple color histogram isn't good enough. As proposed in [1], combining color with motion model could be an interesting idea. The results obtained by this paper show that adding motion models outperforms the simple color model.

4. References

[1] Jiang Zhengqiang, Huynh Du, Moran William, Challa Subhash, Spadaccini N. (2010). Multiple Pedestrian Tracking Using Colour and Motion Models. 328-334. 10.1109/DICTA.2010.63.