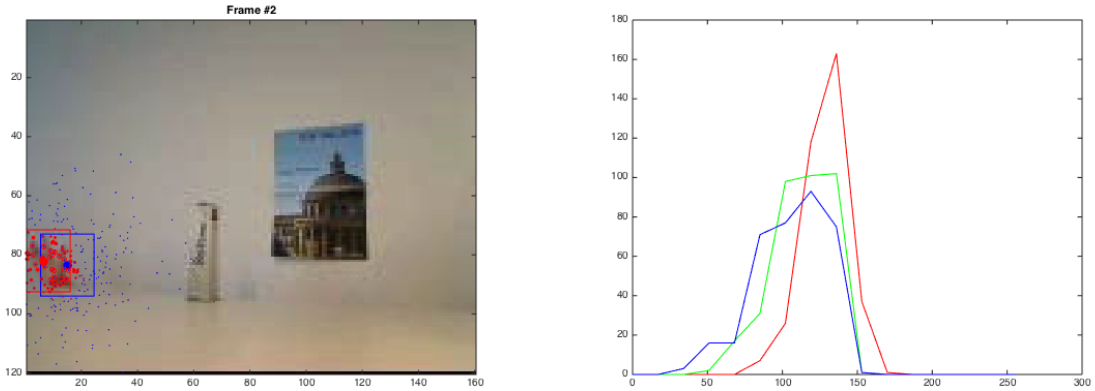
Computer Vision, 8th assignment

Mathis Lamarre

CONDENSATION Tracker Based On Color Histograms

Color histograms

In order to get the normalized histogram, first, we make sure that the bounding box is within the frame. Then, we split the part of the image defined by the bounding box into three matrices, one for the Red, one for the Green and one for the Blue channel. Then, using the matlab function `imhist` with the `hist_bin` parameter, we compute the histogram segmented in `hist_bin` bins. Finally we concatenate them in a vector and normalize it. On Fig.1, we can see an example of a color histogram.



A B

Fig. 1: Example of a color histogram. A shows the image and the bounding box. B shows the color histogram.

Derive matrix A

The dynamic matrix A transforms the old state s_{t-1} into the new state s_t following the linear stochastic differential equation : $s_t = As'_{t-1} + w_{t-1}$. There are two models which have different sizes. In the one with no motion at all, the samples only have two attributes: their x- and y- position. As they are not supposed to move, the A matrix is the identity:

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

For the model with constant velocity, the samples have four attributes: the first two are their position and the last two their velocity. The speed is constant so the lower right part of the matrix is the identity, but the position changes so the first two lines multiply both the position and the speed. The new position is the old one plus the time elapsed times the speed in the corresponding direction.

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

In our case, we set dt to one so our velocities will be in $frames^{-1}$.

Propagation

We propagate particles using the linear stochastic differential equation $s_t = As'_{t-1} + w_{t-1}$. The A matrices were derived earlier and the stochastic element w is the system noise from normal distribution with $\sigma_{position}$ and $\sigma_{velocity}$ as standard deviations. After updating the particle's position, we make sure it is still within the frame and set it to the nearest point on the edge if it doesn't.

Observation

In order to give a weight to each particle, we compute its histogram CH_{part} in a bounding box centered around its position with a given *height* and *width*. Then, we compute the weight π with the following formula: $\pi = \frac{1}{\sqrt{2\pi}\sigma_{obs}} e^{-\frac{\chi^2(CH_{part}, CH_{target})^2}{2\sigma_{obs}^2}}$ where CH_{target} is the histogram of the target we stored. The χ^2 distance is computed with a provided function. Finally, we normalize the weights.

Estimation

To estimate the mean weight, we simply compute the dot product of the particles and their weights. This will multiply each particle's attributes by their corresponding weight. We don't have to divide by the sum of the weights as they are normalized.

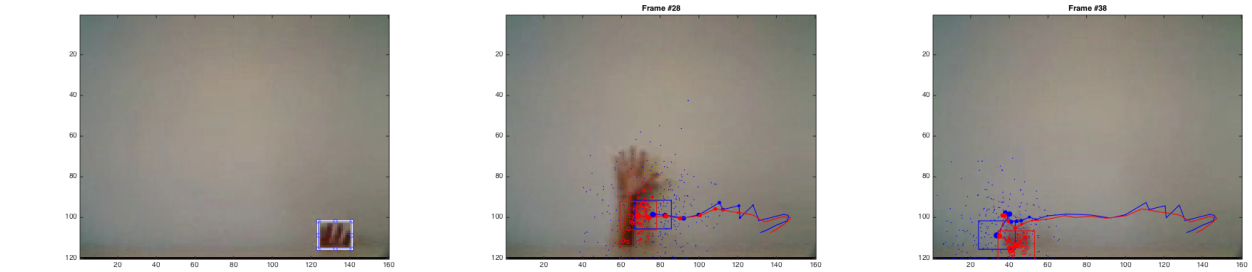
Resampling

To do the resampling, we use the same technique (low variance) than we used for the particle filter in another assignment. We adjusted some parts of the algorithm to our 2D-problem.

Experiments

video1

First, we track a moving hand on a uniform background.



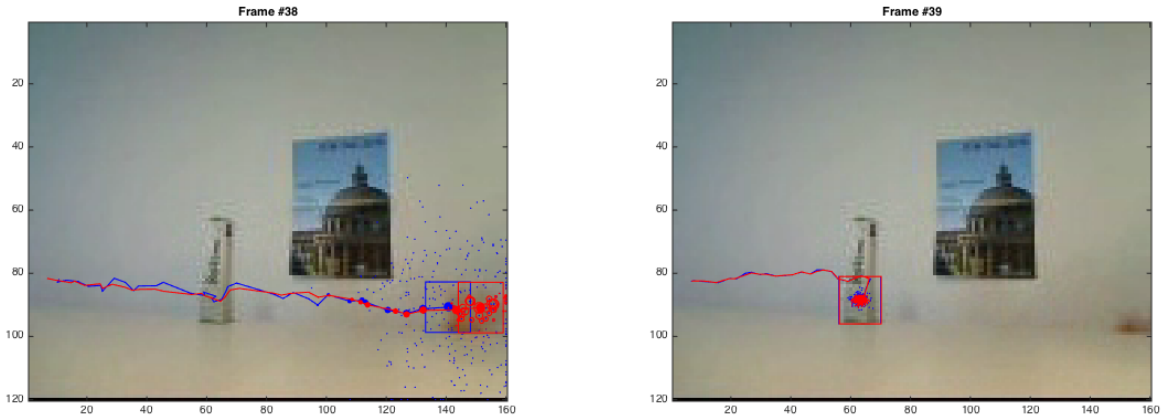
A B C

Fig. 2: Tracking on a uniform background. The blue points are the a priori particle and the blue line their mean state. The red points are the a posteriori particles and the red line their mean state. A shows the initial target box. B shows a frame in the middle of the movement. C shows the end result of the tracking.

As we can see on figure 2, the tracking is relatively good as the particles follow the arm. However, they are not always precisely on the hand but rather on the forearm. This is due to the fact that on the first frame, the fingers have a very high contrast but when the arm moves, due to the lighting and shadows, the big contrast is more on the forearm. We could improve this by changing the α in the update of the color histogram of the target: $CH_{target} = (1-\alpha) \cdot CH_{target} + \alpha \cdot CH_{E[S_t]}$. However the results are not perfect.

video2

Now, we track a moving hand with some clutter and occlusions. We first use the model of no motion.



A B

Fig. 3: Tracking with clutter and occlusions with the no motion model. A shows results with a big system noise: $\sigma_{position} = 15$. B shows results with a small system noise: $\sigma_{position} = 1$.

Even though the hand is moving with a constant velocity and is occluded by an object at one point, the no motion model still manages to track through the entire sequence. This is due to the system noise. During the update of the particles, the stochastic term is relatively big so the particles are spread on a big surface. This enables some particles to still be on the hand when it "reappears" from behind the obstacle and therefore the tracking can continue (see Fig.3-A). However, when the system noise is decreased, the particles are much closer to each other, and the tracking does not succeed with the no motion model: they get stuck when the object is hidden (see Fig.3-B).

To solve this problem, we use the constant velocity model.

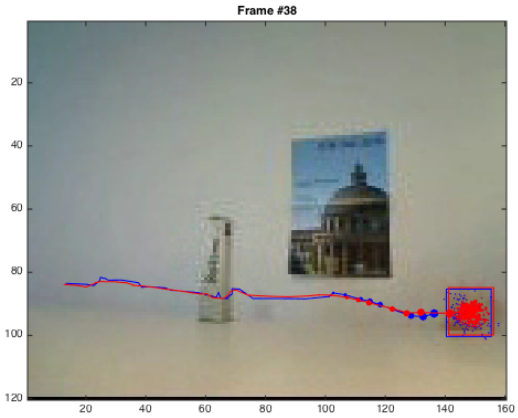
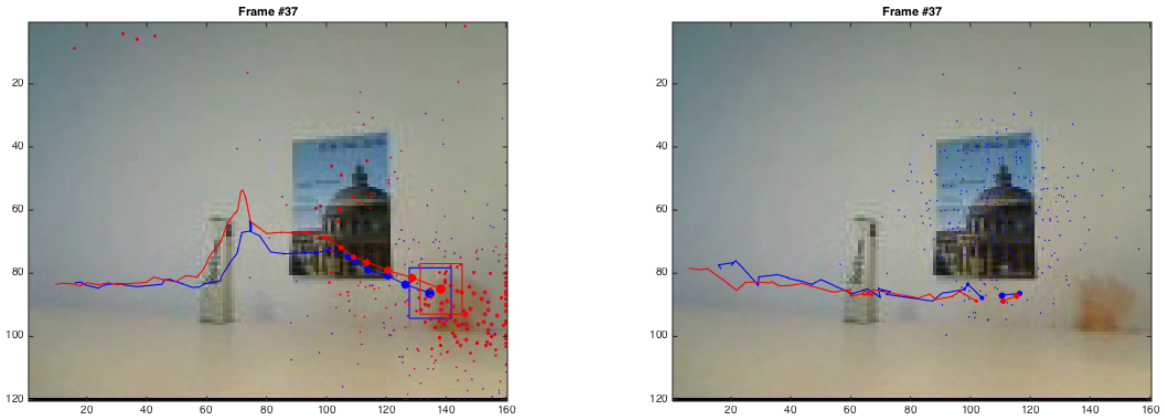


Fig. 4: Tracking with clutter and occlusions with the constant velocity model. The blue points are the a priori particle and the blue line their mean state. The red points are the a posteriori particles and the red line their mean state. The velocity is $[8, 0]^T$ and $\sigma_{position} = 1$.

We used a velocity only in the x-axis as this corresponds to the motion of the hand. As we can see on Fig.4, this model is very accurate even with the small system noise. The predicted particles are even a bit ahead of the hand sometimes when they are updated. As the hand's speed is not perfectly constant, they "wait" for the hand every now and then, notably when it is behind the obstacle.

We test the effect of the measurement noise.



A B

Fig. 5: Tracking with clutter and occlusions with different measurement noise. A shows results with a big measurement noise: $\sigma_{observe} = 0.6$. B shows results with a small measurement noise: $\sigma_{observe} = 0.008$.

On Fig.5-A, we see the effect of a large measurement noise. The weights are very wide spread so importance is given even to not very relevant particles. For example, when the hand is hidden, the particle start to track something else and move up. However, the system is able to recover and recognizes the hand when it reappears. It has no problem tracking when the hand when it is in front of the poster (different background). We notice that even in the hand there still are some particles in irrelevant places of the frame like the top left. On Fig.5-B, we study a small measurement noise. The weights distribution is very narrow so particles that are just a bit different will a very small weight. The system is very strict. This is what we see as small changes completely mess the overall tracking. When the hand comes in front of the poster, the background change makes the particles' histogram too different from the target histogram for them to have a relevant score and it fails.

video3

We now try to track a bouncing ball with the same parameters as before. The main difference is that its direction changes.

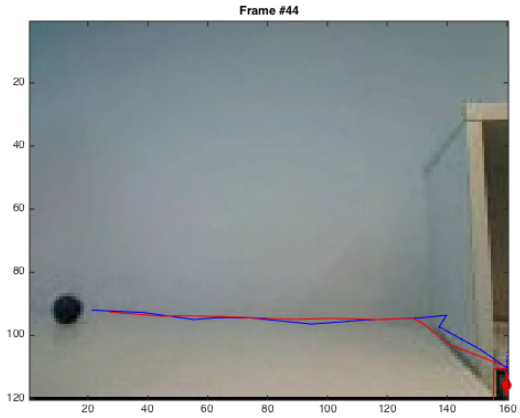


Fig. 6: Tracking of a bouncing ball with same parameters as before. It uses constant velocity model with a speed to the right.

As we can see on Fig.6, the same parameters as before (constant speed to the right) do not work with the bouncing ball. Its direction changes drastically and the particles are all updated to its right and never find it back. In order to fix that, we can change several parameters.

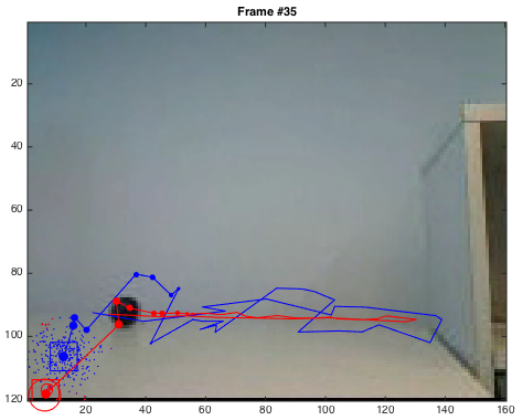


Fig. 7: Tracking of a bouncing ball with big speed system noise: $\sigma_{velocity} = 5$.

In Fig.7, we can see the effect of a big $\sigma_{velocity}$ allowing the particles' velocity to change. In this case it worked but the results are not perfect as the velocity change is sometimes too drastic and too random. Another strategy is to increase the $\sigma_{position}$ so that there still are some particles in the ball region even when it bounces.

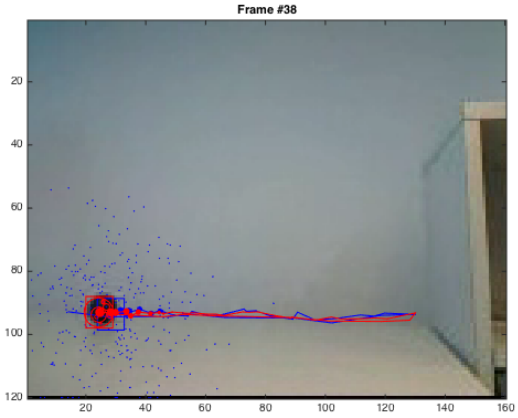
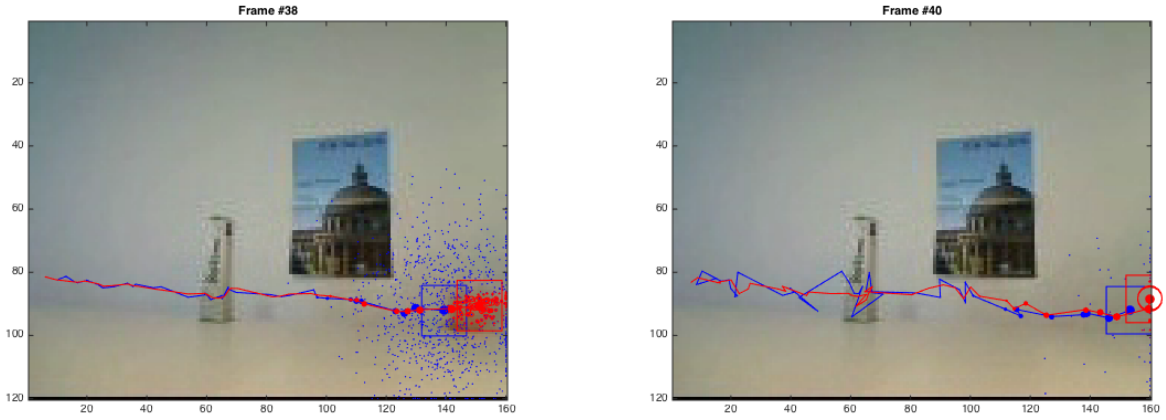


Fig. 8: Tracking of a bouncing ball with no motion model.

The most robust version to track the bouncing ball is to use the no motion model (see Fig.8). This way the particles don't try to predict the next state and don't get tricked by the sudden change of velocity.

Then, we studied other parameters. First, we varied the number of particles. With a more particles (see Fig.9-A), the tracking is even more stable, but the computation is longer. On the other hand, with fewer particles (see Fig.9-B), the behavior is less predictable and reproducible. We can notice that the a priori mean state's evolution is very jerky. However, after weighting them, the a posteriori mean state follows the trajectory in a more stable fashion.



A B

Fig. 9: Tracking with different of number of particles. A uses 1000 particles. B uses 50 particles.

Then, we changed the number of bins in the histogram. With more bins, the histograms are more precise so they allow for a bigger variety. Too many histograms might cause a distance too big betweens two histograms that might be not very different in reality though. A smaller number of bins allows for a bit more flexibility (resistance to noise for example). With too few bins however, all the histograms are quite similar so the tracking is not accurate at all (see Fig.10). When the background is really noisy, like the poster, the histograms are not precise at all and can't produce a relevant weight.

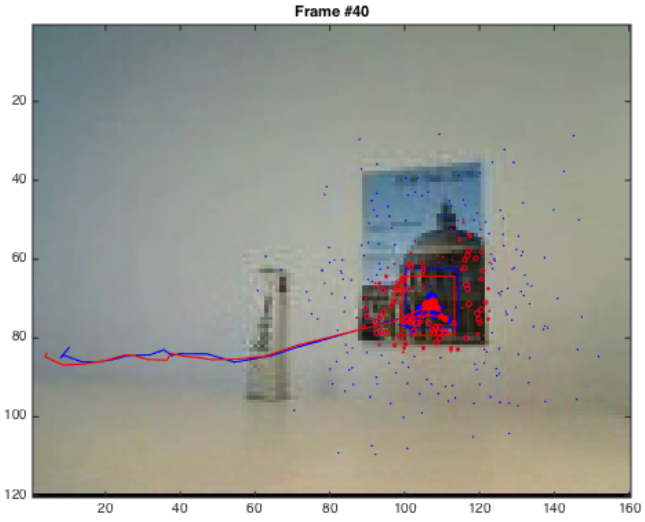


Fig. 10: Tracking with only 2 bins.

Finally we changed the α parameter to allow for model updating. With the first video, we noticed that the particles stopped tracking the hand and moved to the forearm because of the lightning change. When allowing model updating, the target histogram will not always be the one of the beginning so it adapts to the smooth changes in shadowing. As we can see on Fig.11, the tracking goes a bit up, which is the more representative of the actual motion of the hand.

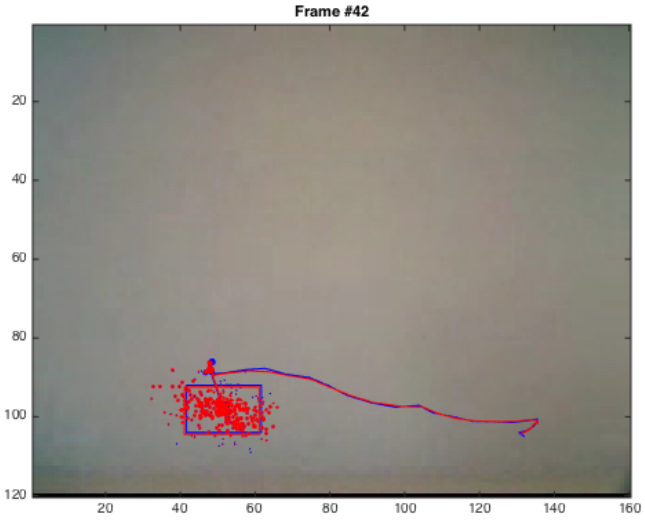


Fig. 11: Tracking with appearance model updating. $\alpha = 0.5$.

Try your own video

In order to face new problems, we tracked an object in front of a resembling background that is dynamic. In this video a player kicks a penalty so the white ball travels in front of moving player whose shoes and jerseys are also white. To do so, the constant velocity model was used with a velocity in the overall direction of the ball, and small system noise were used because the ball is quite small and its velocity rather constant. A small appearance model update was used to allow for flexibility (the ball's shape might change as it approaches the

camera) but not too big so that the particles don't get stuck on a white element (shoes, jerseys or lines on the field). The overall result is very satisfying.

Fig. 12: Tracking with appearance model updating. $\alpha = 0.5$.

To improve the algorithm's performance, we could develop a way to follow objects with speed. A way to do it would be to use the constant velocity model but select the particles whose weight are higher at each step and assign their speed to all the resampled particles, like a natural selection.