Computer Vision

Assignment 11 : Particle Tracking

Autumn 2018

Nikita Rudin

December 26, 2018

In this lab we are implementing a tracker based on condensation filter. The user selects a part of the image, which will then be tracked by propagating and re-sampling particles according the color histogram around them.

1 Tracker implementation

In this section we will see the main steps of the tracker implementation

1.1 Colour histogram

Given a rectangular part of he image we want to calculate its colour histogram. We start by selecting the appropriate pixels from the image, making sure that pixels outside of the frame are not included. We then use the histcounts() function on all three RGB colours with fixed bins from 0 to 255. finally we combined the three histograms in one vector.

1.2 Dynamics matrix

For the propagation of particles we use a linear system. Each particle is multiplied by the matrix A and then noise is added. For the first model where the motion is purely due to noise A is simply the identity and noise is added to the position using randn() multiplied by $\sigma_{position}$. In the second model we assume constant velocity. This time the position is increased as follows:

$$r_{k+1} = r_k + dtv_k = Ar_k$$

with

$$r = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

As such

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

This time the noise is added to the velocity instead of the position using $\sigma_{veloctity}$

1.3 Particles propagation

The model described above is used to propagate each particle. For better performance this is done by multiplying the full matrix of particles by A at once. Noise is added to either the position or the velocity respectively. If a particle propagates outside of the frame it is put to the edge.

1.4 Observation

Once we have propagated our particles. We take the take the image inside a rectangle of the same size as the original one and centred on the particle. We use the colour histogram function to evaluate a histogram for each particle. Finally we compare these histograms with the original one using the χ^2 cost. The cost is used to set the weight of the particle. Finally, the weights are normalized such that their sum is 1.

1.5 Estimation

In order to estimate the state we simply take the weighted average of the particles. It is crucial that the sum of weights is 1 in order to have the right scale.

1.6 Re-sampling

The final step is to re-sample the particles according to their weight. We use the wheel re-sampling algorithm. Each new particle is drawn for the previous set with replacement. A particle has a probability of being chosen equal to its weight.

2 Experiments

2.1 Video 1

In the first video we see an arm moving in front of a uniform background. The tracking should be simple, but there are some issues like the quality of the video, the low contrast between the arm and the background and a change in illumination.

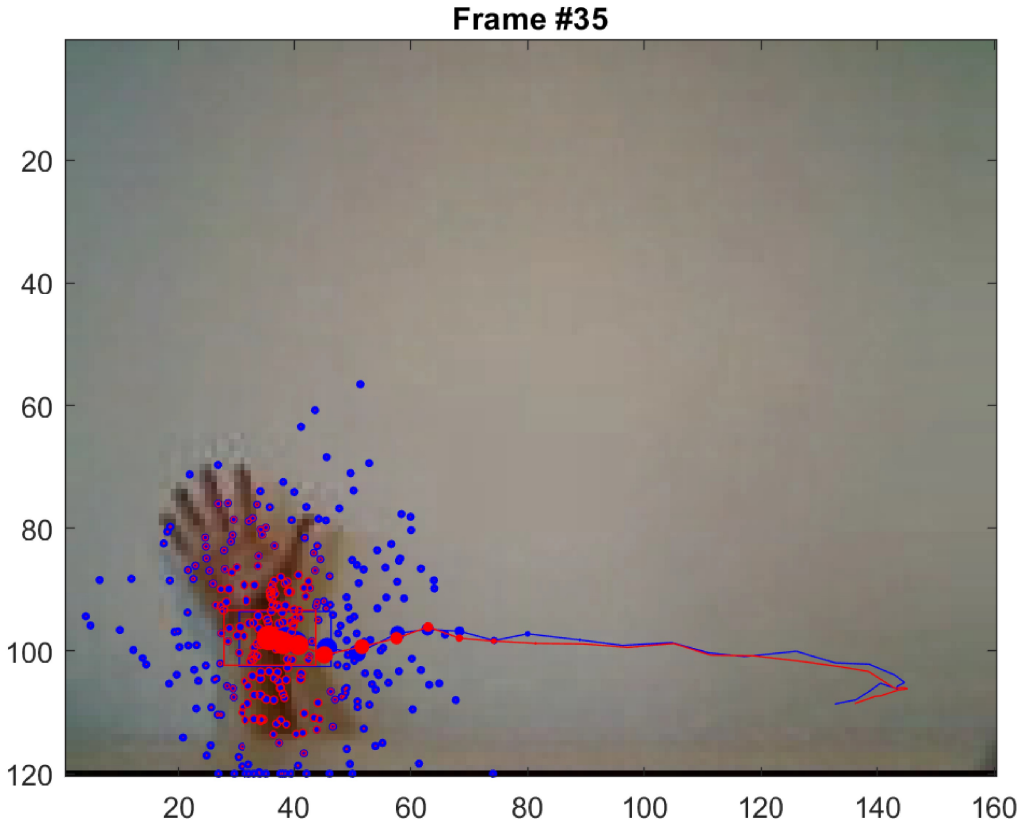
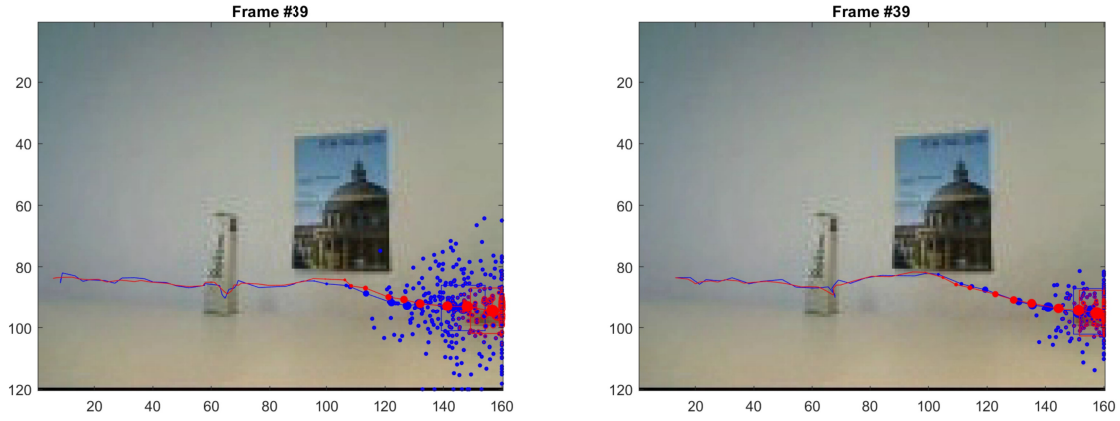


Figure 1: Resulting plot from the tracking of the arm. Mean before a priori estimate in blue and a posteriori in red.

We can see that the tracking works relatively well. The main issue is that the tracked area changes from the arm to the forearm. This is due to a change of illumination. The arm becomes brighter during the video and the forearm is then closer to the original histogram. Increasing α should allow us to solve this problem, but in practice it doesn't work well.

2.2 Video 2

In the second video the background contains other objects. The arm is even occluded during a few frames, making the tracking more difficult. Comparing the models (pure noise or constant velocity) we can see that the constant velocity results are slightly smoother. In theory they should handle the occlusion better, because the particles should continue moving with the right velocity. In practice this effect is mostly counteracted by the re-sampling done at each step, which makes the velocity very random.

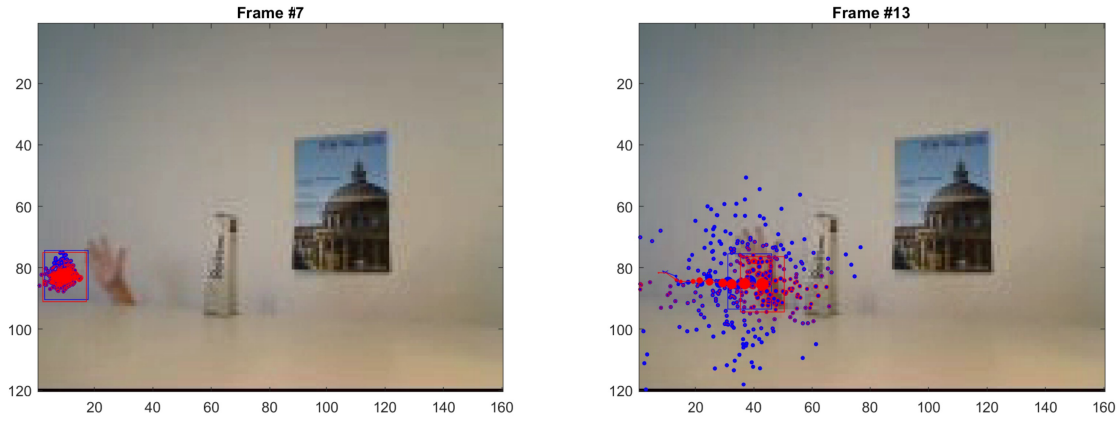


(a) Noisy position

(b) Noisy velocity

Figure 2: Comparison of the two models. We can see that the results end up being very similar.

The effect of changing $\sigma_{position}$ or $\sigma_{velocity}$ is mainly to increase the distribution of particles at each step. With a low σ the particles stay grouped together and may not catch the right position especially if the object is moving quickly, while with higher σ the distribution is larger and the precision of the estimate will be decreased because less particles will be close to the correct position.

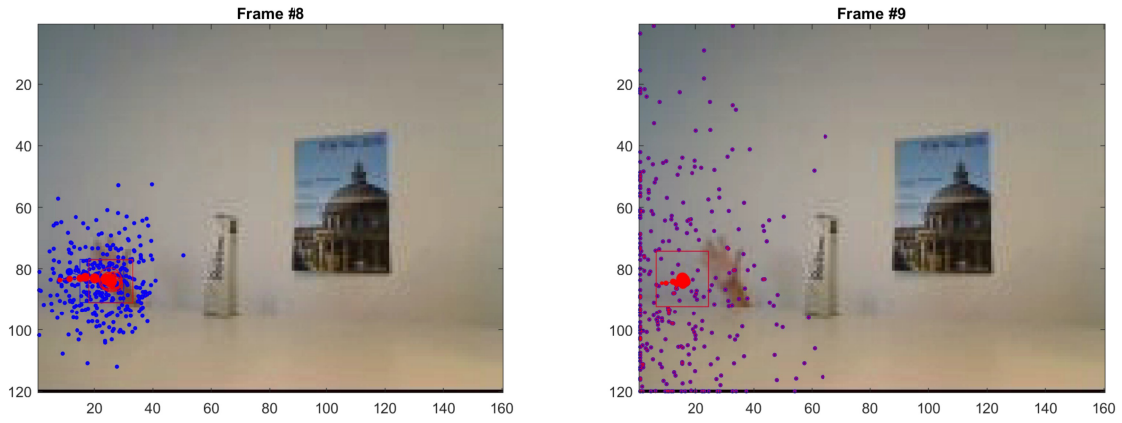


(a) Low σ

(b) High σ

Figure 3: Effect of $\sigma_{position}$ on the distribution of particles

Similarly $\sigma_{observation}$ affects the way particles are re-sampled. If it is large many particles are taken into account for the estimate. This means that we are using some particles that do not represent the right position. If it is small only a few particles are used, but the estimate is more noisy.



(a) Low σ (b) High σ

Figure 4: Effect of $\sigma_{observation}$ on the distribution of particles. The red circles represent the effect of the particles on the a posteriori estimate.

Finally the chosen parameters are:

$$\sigma_{position} = 10$$

$$\sigma_{velocity} = 5$$

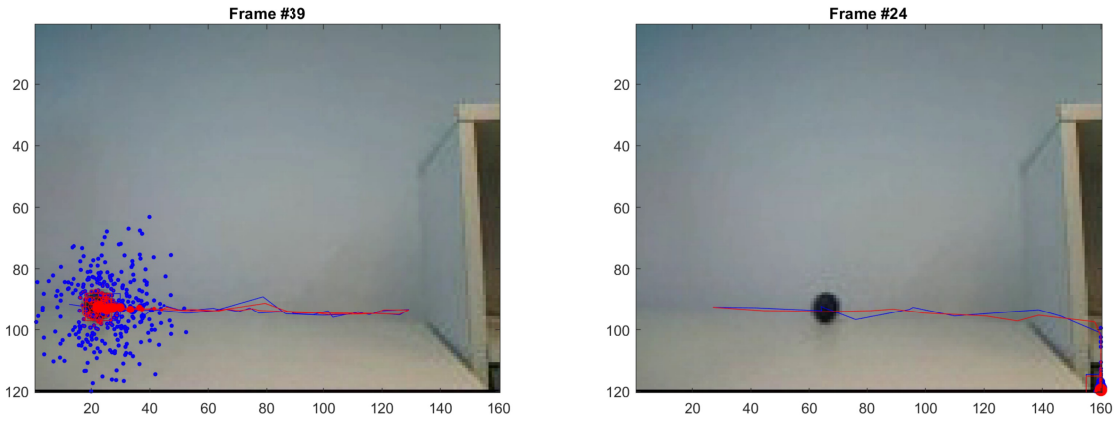
$$\sigma_{observation} = 1$$

$$\alpha = 0$$

2.3 Video 3

In this 3rd video we track a ball that bounces on a wall. The main challenge is fast motion of the ball for the 1st model and the fact that the ball bounces back for the constant velocity model. The 1st part of the tracking works better with the constant velocity model. This is logical because this model represents well the actual dynamics of the ball. The problem is that once the ball bounces back, this model expects to see it further and as such, most particles propagate in the previous direction. To make things worse, on the far right there are black spots with very similar colour histograms. The particles often end up tracking this corner instead of the ball. We can improve the results by increasing $\sigma_{velocity}$, but then we are back to the first model where the motion is purely random.

Using the pure noise model, the tracking is slightly worse during the 1st part, but it handles the bounce much better. The parameters from the previous video seem to work well for the 1st model.



(a) Pure noise model

(b) Constant velocity model

Figure 5: Comparison of the models for the 3rd video

2.4 Own video

I have tried the obtained filter with a personal video. We can see that with a higher frame rate (60 fps) and a much higher resolution the tracking works really well. It is important to note that the dynamics sigma should be increased according to the resolution, since they in units of pixels.

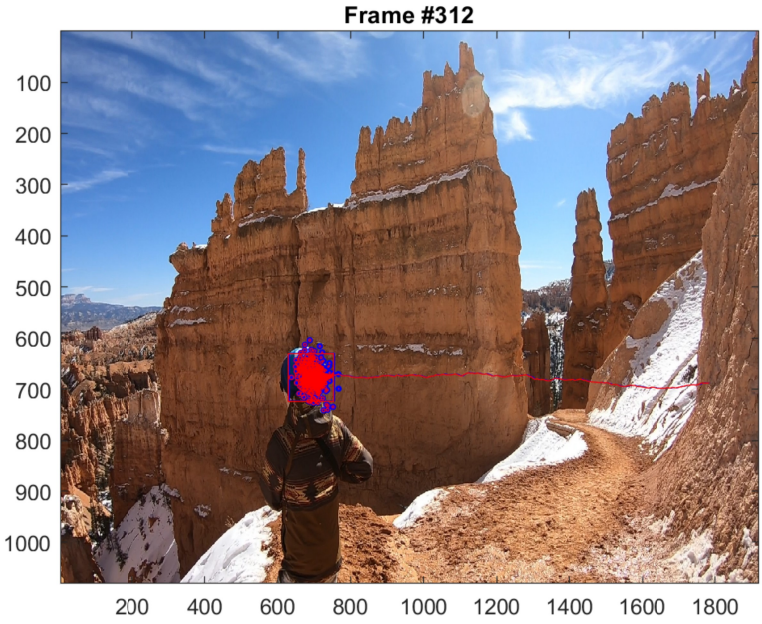


Figure 6: Tracking of a dark blue and white hat. The higher resolution and frame rate has a big impact on the quality of the tracking.

3 Discussion

The number of particles is a tradeoff between precision and computational efficiency of the algorithm. If we have more particles, there is a higher probability that one of them will be close to the actual position. The obvious trade-off is that more computational resources are required.
The number of bins is another trade-off. If it is too high we will include too much noise in the observations and a small change in illumination can lead too a high χ^2 cost. If it is too low we will loose too much information and different patches could look very similar.
The advantage of allowing the target measurement to change is a robustness to gradual changes such as illumination. The main disadvantage is that in case of a misalignment in the tracking, the appearance will progressively change and the errors will accumulate. After some iterations the tracked object will be completely different from the desired one. It seems that in our examples the disadvantages of this techniques are more important than the advantages and as such, the tracking works better without it.