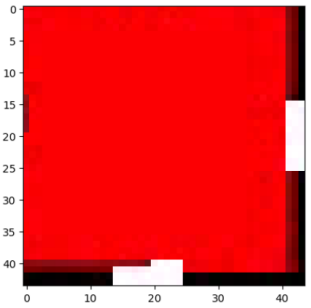
BACHELARD Maéva
GALLIEN Anthony

TP3/4 Object tracking

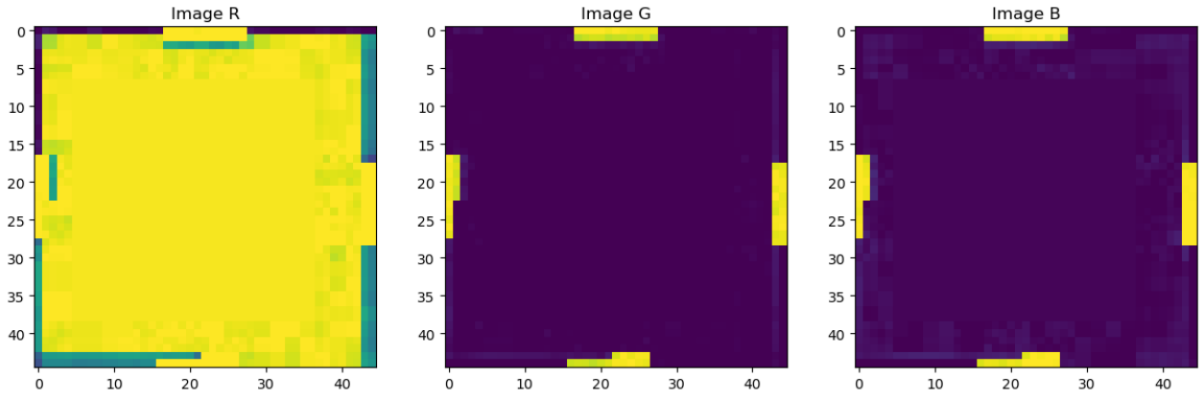
1- Tracking simple objects in videos

1. Particle Filters

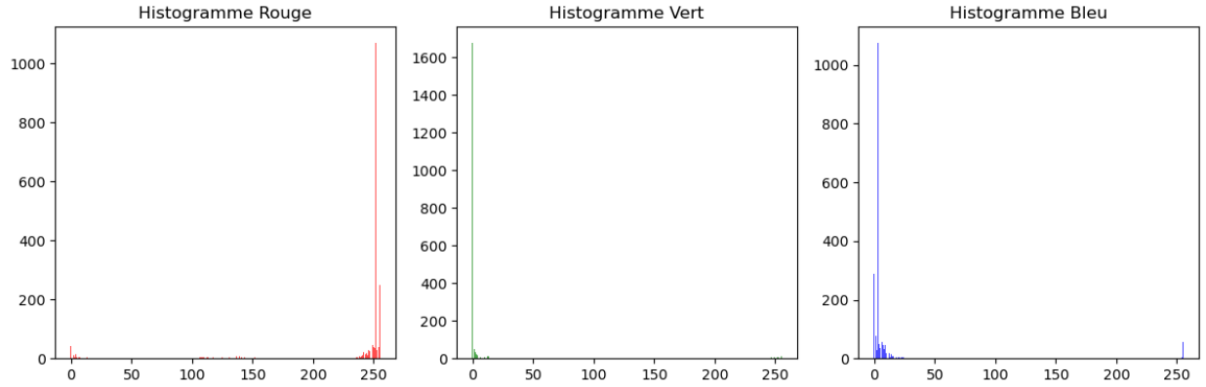


The video size is 480 by 640.
By looking at the first frame of the video, we manually find that the rectangle size is 45 by 45 and its center is at coordinates 242, 323.

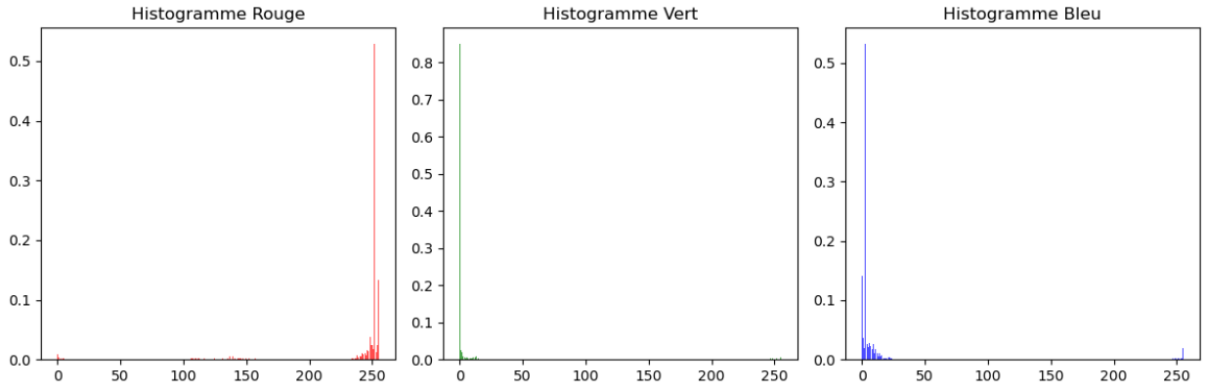
We can print what we will call an "observation", which is an image of size 45 * 45 around the supposed center of the object. This observation is in RGB, which gives us 3 images to observe and detail:

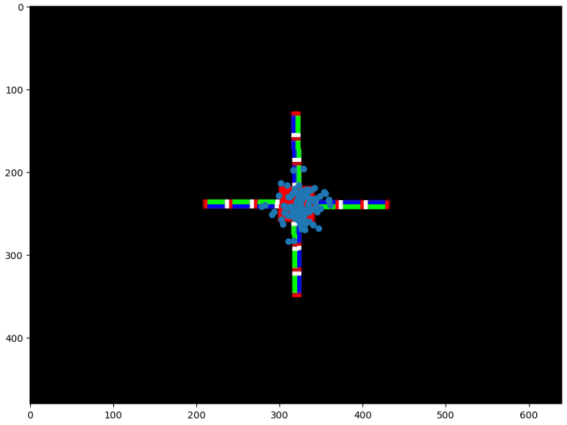


From these 3 images we can calculate 3 histograms, one for each color:



The principle is to compare the 3 histograms of the initial observation with the histograms of surrounding observations. A distance between the reference observation and the particle observation is calculated for each color. The formula takes into account the fact that histograms must be normalized (as it is below):

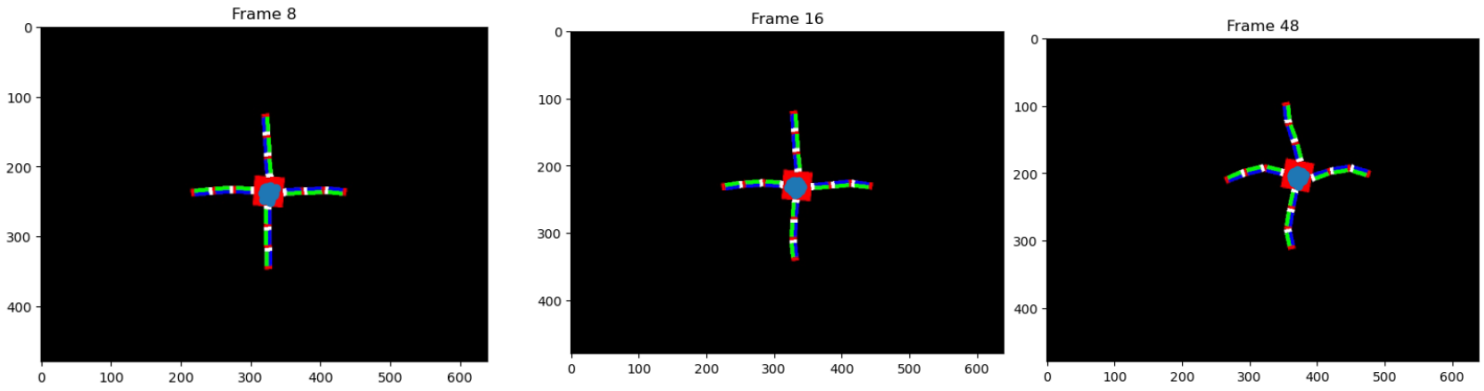




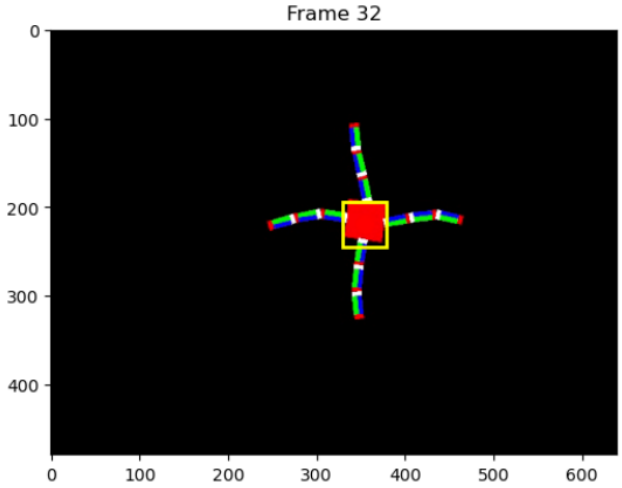
Next, we'll randomly create n particles whose center is close to the original center. As explained above, each of these particles will be linked to an observation of size 45 * 45, whose distance from the reference observation will be calculated. Using this distance, we'll modify the weights associated with each particle. This step is very important, as it will have an impact on resampling, the step that limits particle degeneration and enables us to track the object perfectly.

Our resampling function takes particles with updated weights and limits/deletes those with weights too low and, on the contrary, multiplies those with weights high enough. That way, all our particles follow the object.

Finally, we combine all the previous steps into a single function, resulting in a group of points that follows the center of the image almost perfectly:



Now we're going to average the particle positions according to the weights, in order to obtain the "new" center on each frame and deduce the object (the observation) we're tracking by framing it.

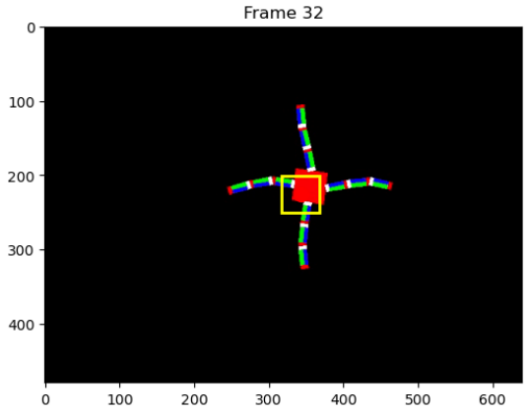


Now let's play with the parameters to test their effectiveness and importance:

- Transition model :

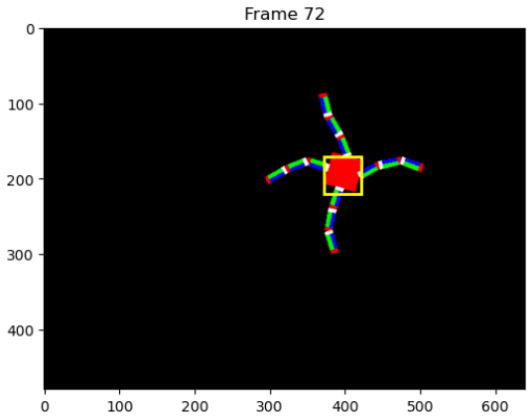
Here we can play with the square matrix of size 2, which "sends" the particles around the previous position.
For a 5*I matrix, the results are very satisfactory for the rest of the video frame by frame, when the movement is not too fast. Note that as you increase the values on the diagonal, the new positions can become further and further apart. For 5000*I, you can go up to 100 pixels further.
This parameter needs to be adapted to the object under study and its assumed speed (as well as the number of frames when cutting the video).
For the rest, we'll keep 5*I.

- Lambda :



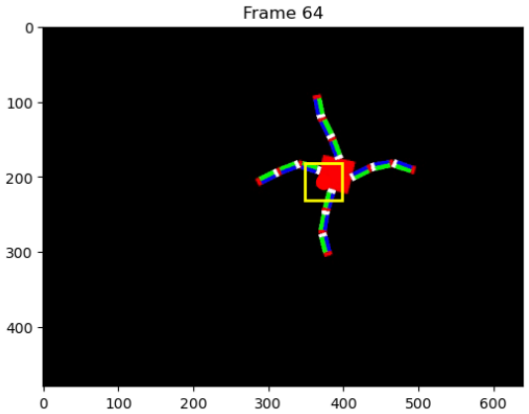
If the lambda is too small, then we cannot capture the movement of the image. (lambda = 0.1 on the image opposite), for very large lamba (lambda = 50) on the contrary, the image is tracked perfectly, but the calculations seem to take longer. In the future, we'll take a value greater than 1 but less than 5, which seems to give great results without taking up too much time.

- Color histogramme :



The more information you have, the more accurate the tracker becomes. Indeed, when tested on a single color (see test opposite), the tracker is still good, but slightly less accurate (the top corner protrudes, for example).
We assume that our image is particularly suited to the case where it is possible to study a single color (if all other parameters are well chosen). However, it is understandable that in some cases, where certain colors are less significant, it will be difficult to track the object correctly.

- Number of particles

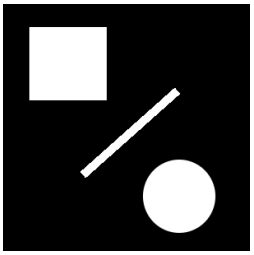


With 5 particles, it's impossible to track the object correctly (image opposite). With 15 or more particles, we can track the object correctly, even if the precision isn't great. From 50 particles upwards, it's possible to track the object perfectly. Above 50, accuracy doesn't increase significantly, and the calculation time becomes very long. In our case, using 50 particles gives really good results. We assume that the larger the zone studied, the more particles will be needed to assess the object's correct position.

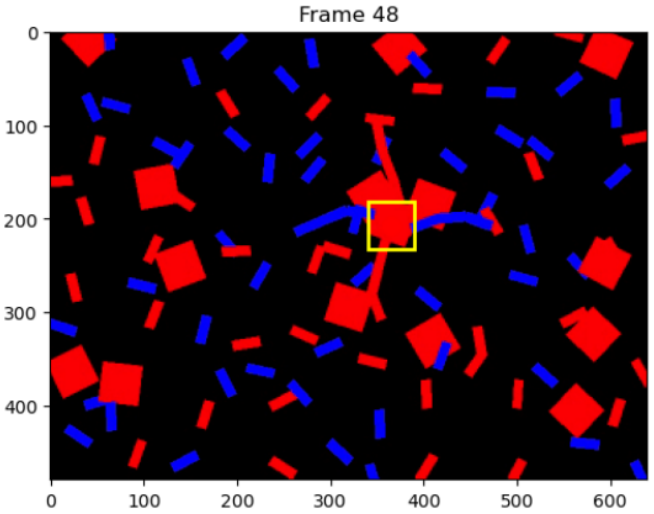
If we don't have the possibility of having frames as close to each other as in our example, then we'll have to play with the parameters we studied earlier, mainly the covariance matrix that "sends" the particles to previous positions, as well as the number of particles. This will enable us to better predict a more distant position of the object and more significant changes.

The likelihood model is a crucial component of the particle filter, as it determines how closely the generated particles match the observed data. As seen in previous lab work, there are other images we could study, such as :

- Contours
- Edges
- Magnitude and directions
- Image gradients

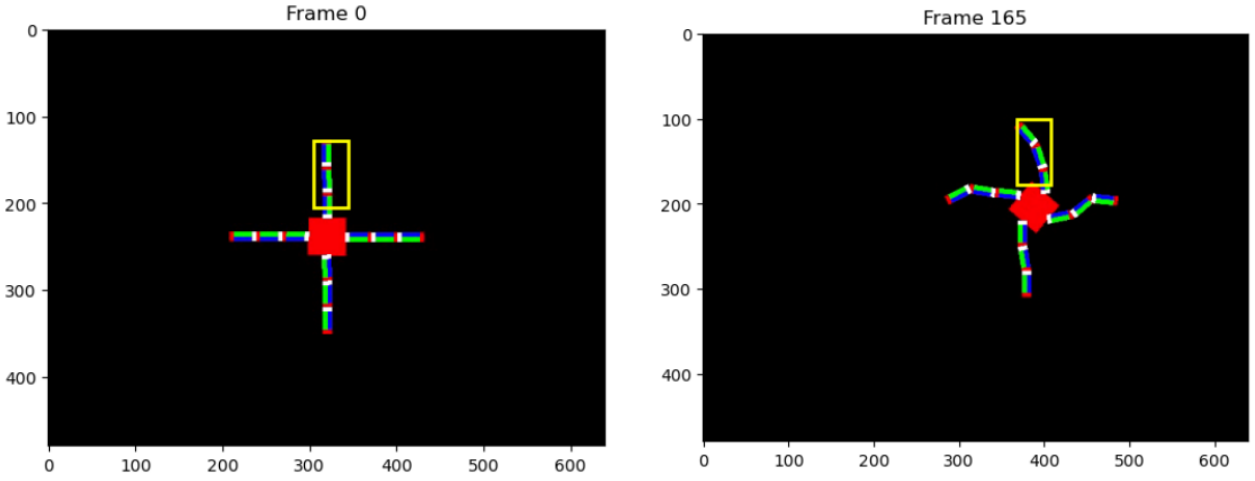


In our case, given the colors of the image, the method we use seems the best, but in the case of this image studied a few weeks earlier, a likelihood method using contours or gradients seems more appropriate.

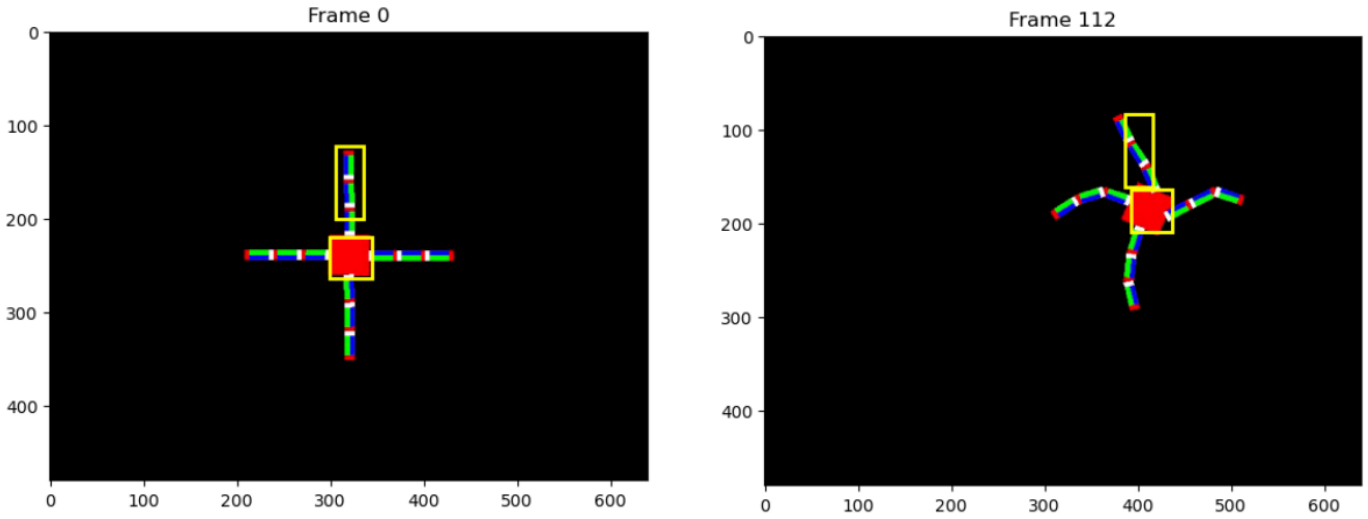


We finally test with the second video, the one where the object is surrounded by other objects, and once again, with the parameters well chosen, we track our object almost perfectly.

Up to now, we've been tracking a square object corresponding to the center of the image, but now we're enhancing our submatrix function so that we can start tracking rectangular objects. This means we can now track the object's legs:



Finally, we create a new function from the previous one that will allow us to track several objects simultaneously:



Finally, we performed the tracking on the central square and all the legs by initializing the centers of the various elements. This works very well but takes quite a long time to compute.