Nicolas Pfitzer 5/12/2023

Exercise 6 Report: Condensation tracking

1) Implementation

a. Color Histogram

The data structure used for the color histogram is a 3-dimensional array of size *hist_bin^3*, where each dimension represents one of the three RGB colors. The 3D color space is discretized along every dimension by to the number of bins described by the *hist_bin* variable. The *color_histogram* function computes for each of pixel within a boundary box its histogram coordinates and increments the corresponding bin by 1, thus allowing for efficient description of the color information within the boundary box.

b. Dynamic matrix A

In this exercise, we were asked to consider two prediction models:

i. No motion (just noise)

In this system, the model has no deterministic component. The prediction on the next object state depends solely on the noise vector w_{t-1} that models the uncertainty. This disturbance vector follows a gaussian distribution, centered at the position of the particle at time *t-1* with standard deviation described by the *sigma_position* parameter. Furthermore, the distribution must be truncated to not exceed the borders of the image frame.

ii. Constant velocity

The prediction model of the system with constant velocity has the same stochastic component as in the “no motion” model (extended to also include velocity noise), however we must now also include a deterministic matrix A.

$$A = \begin{matrix} 1+\frac{v_{t-1,x}}{s_{t-1,x}} & 0 & 0 & 0 \\ 0 & 1+\frac{v_{t-1,y}}{s_{t-1,y}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

c. Propagation

The *propagate* function receives as input, the *a posteriori* set of samples S_{t-1} from the density $p(x_{t-1}|Z_{t-1})$ and propagates the particles (samples) using the models described above to generate the *a priori* set S'_{t-1} from the density $p(x_t|Z_{t-1})$.

d. Observation

The *observe* function computes the new weights of the propagated particles. We start by calculating the *chi2 distance* between the color histogram of each new particle bounding box and the reference histogram. The distances are then used to assign the weights, which follow a gaussian distribution, centered around 0 and with standard deviation *sigma_observe*.

e. Estimation

The *estimate* function computes the *a posteriori* mean state by calculating the weighted sum of all observed particles. To achieve this, the weights are fist normalized by dividing them by their sum. The mean state is then used to draw the red bounding box, which indicates the current best estimate on the position of the object.

f. Resampling

The resampling is the final step in the condensation tracking algorithm. The new samples are drawn with replacement from the updated/weighted set of particles, with higher weights having a higher probability of being selected (potentially multiple times). The samples are drawn using the *choice* function from the python *random* module.

2) Experiments and Results

a. Video 1 – Hand with no obstacles or obstructions

After tweaking some of the tracking parameters, the following parameters seemed to produce the most consistent results:

Model	0
Number of particles	100
Sigma observe	10
Alpha	0.2
Sigma position	15
Histogram bin size	8

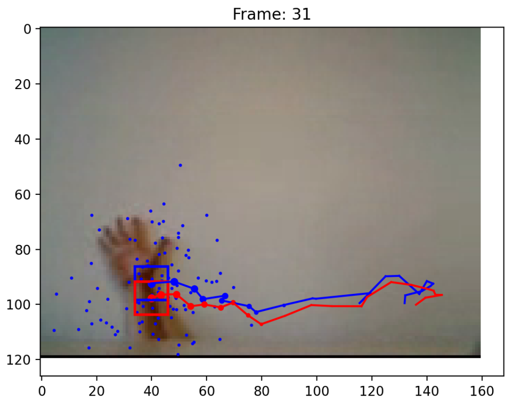


Figure 1 Hand with no obstacles or obstructions – No motion model

Overall, the results are satisfying, however the tracker tended to shift to the upper wrist, instead of tracking the fingertips.

If the parameters shown above are not chosen properly, the tracker will fail, as shown in Figure n°2:

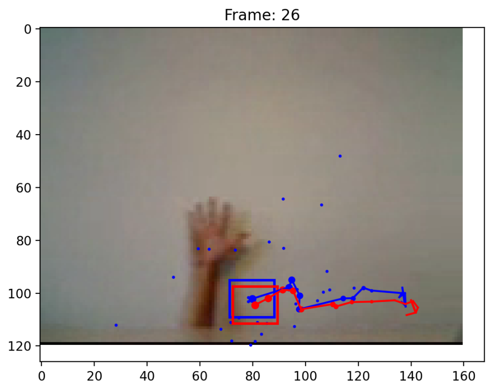


Figure 2 Example of a fail case, where the "sigma_observe" parameter is too high (more on that later)

b. Video 2 – Hand with obstructions

Using the same parameters as before, the tracking algorithm was able to perform quite well on the second video with obstructed views.

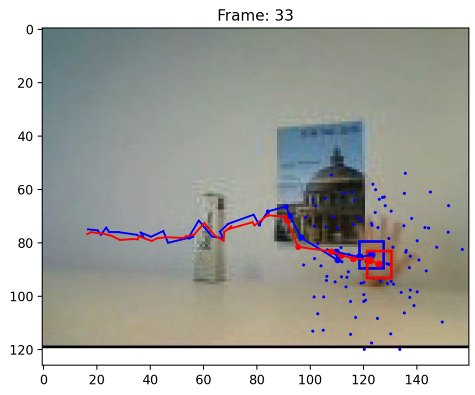


Figure 3 Hand with obstructions - No motion model

Constant velocity motion model:

Since the hand has a reasonably linear trajectory, the constant velocity motion model can also be implemented. However, tests showed that the second model did not provide significant improvements to the accuracy of the model when the number of particles was higher than 30. It was only by drastically lowering *num_particles* to approx. 10, that the

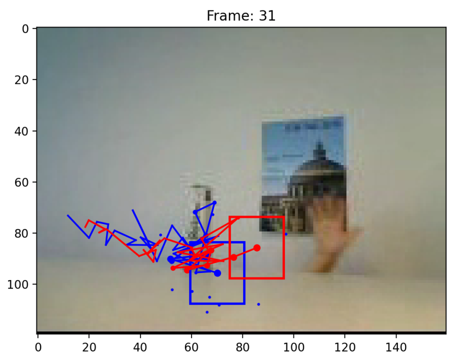


Figure 5 Failed tracking with no motion model and 10 particles

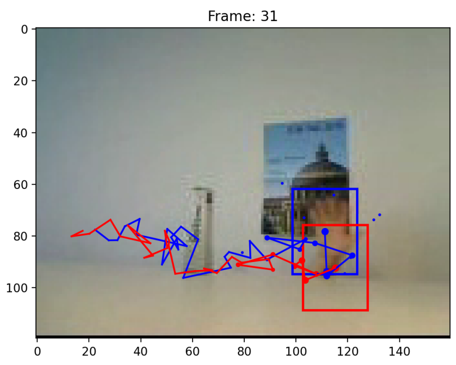
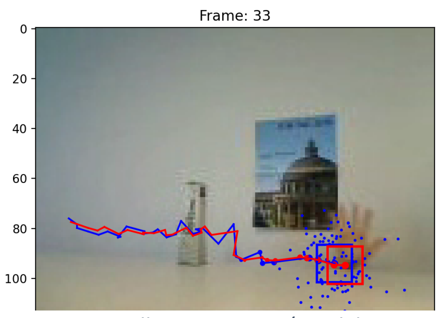


Figure 4 Constant velocity model (vx = 10) , 10 particles

constant velocity motion model showed clear advantages.

Effects of system noise:

The system noise plays a crucial role in finding the *a priori* position of the object between two subsequent frames. Depending on the distance traveled by the object between the two images, the condensation algorithm might have to propagate particles further to keep tracking properly. A small standard deviation will keep the particle propagations close together, whereas a large sigma, will drive them apart.



₁*Figure 7 Small system noise / model = 0*

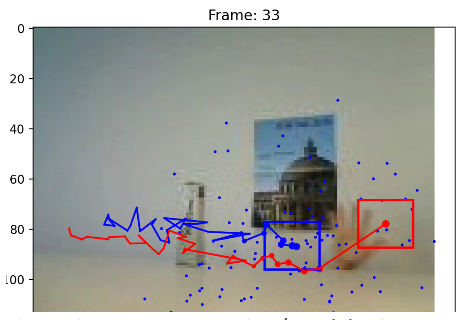


Figure 6 Large system noise / model = 0

Effects of measurement noise:
The measurement noise or observation noise is responsible for assigning the weights to the particles. If the sigma value is too low, the probability of each sample will be too low, making it difficult for the algorithm to estimate the next state. Inversely, if the sigma value is too high, the samples will be weighted too high, thus increasing the odds of misidentifying the next pose.

c. Video3 – Rolling ball with bounce.

Once again using the same parameters as the first video, the following results were achieved:

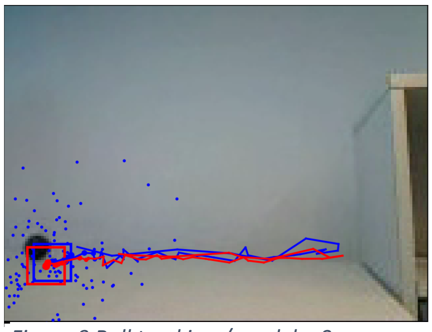


Figure 9 Ball tracking / model = 0

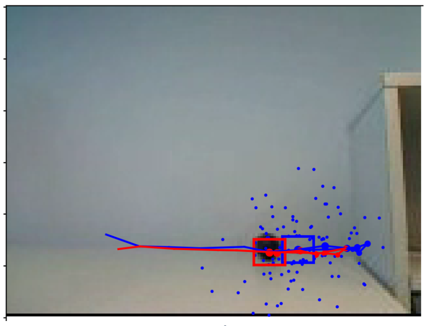


Figure 8 Ball tracking / model = 1

Since the ball changes direction, it made little sense to use the constant velocity motion model. As we can see from Fig. 9, the *a priori* mean state trails back, when the ball starts rolling in the opposite direction, because the model still assumes a constant velocity. The first, no motion, model produced better results.

Furthermore, the velocity of the ball in this video is higher. Therefore, it was important to favor a larger system noise. By setting the *sigma_position* parameter to 20, the tracker was able to successfully follow the ball with as little as 30 particles!