Computer Vision
Lab Assignment Report - Condensation Tracker

ETH Zürich, 263-5902-00L Computer Vision HS2022
Prof. Marc Pollefeys, Prof. Siyu Tang, Dr. Fisher Yu

Anisha Mohamed Sahabdeen

December 9, 2022

1 Implementation

1.1 Color histogram

In order to obtain a CONDENSATION tracker based on color histograms, it is first required to implement the function that computes the color histogram for a given particle, i.e. bounding box, being considered. The `numpy` library function `numpy.histogram` is used to get a histogram for each of the RGB channels of the frame. The three histograms are first concatenated into a single vector and then normalised with respect to their sum.

1.2 Derivation of matrix A

The *prediction* step of the CONDENSATION algorithm requires to predict the *a priori* density distribution $p(x_t|Z_{t-1})$ given the sample set S_{t-1} computed in the previous iteration step. Here, the dynamics of the model are governed by a linear stochastic differential equation, so that each new sample may be generated as:

$$s_t'^{(n)} = As_{t-1}^{(n)} + w_{t-1}^{n} \tag{1}$$

where A defines the deterministic component of the system model and w_{t-1}^{n} corresponds to noise and is here assumed to be normally distributed.

Let us now consider two system models: (i) no motion (only noise), and (ii) constant velocity motion model.

(i) In the no-motion model, the state s is corresponds the position of the particles in the image, i.e. $s = [x\ y]^T$. Hence, the system model is described by the following:

$$\begin{cases} x_t^{(n)} = x_{t-1}'^{(n)} + \sigma_{P}{}_{t-1}^{(n)} \\ y_t^{(n)} = y_{t-1}'^{(n)} + \sigma_{P}{}_{t-1}^{(n)} \end{cases} .$$

The model can be thus be written in matrix form as:

$$\begin{bmatrix} x_t^{(n)} \\ y_t^{(n)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1}'^{(n)} \\ y_{t-1}'^{(n)} \end{bmatrix} + \begin{bmatrix} \sigma_{P}{}_{t-1}^{(n)} \\ \sigma_{P}{}_{t-1}^{(n)} \end{bmatrix} = A \begin{bmatrix} x_{t-1}'^{(n)} \\ y_{t-1}'^{(n)} \end{bmatrix} + \begin{bmatrix} \sigma_{P}{}_{t-1}^{(n)} \\ \sigma_{P}{}_{t-1}^{(n)} \end{bmatrix} ,$$

such that

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

(ii) In the constant-velocity model, the state s is described by the position of the particles and their velocities in the two image dimensions, i.e. $s = [x\ y\ \dot{x}\ \dot{y}]^T$. Hence, the system model is described by the following:

$$\begin{cases} x_t^{(n)} = {x'}_{t-1}^{(n)} + \dot{x}'^{(n)}_{t-1}\delta t + \sigma_P{}_{t-1}^{(n)} \\ y_t^{(n)} = {y'}_{t-1}^{(n)} + \dot{y}'^{(n)}_{t-1}\delta t + \sigma_P{}_{t-1}^{(n)} \end{cases} .$$

The model can be thus be written in matrix form as:

$$\begin{bmatrix} x_t^{(n)} \\ y_t^{(n)} \\ \dot{x}_t^{(n)} \\ \dot{y}_t^{(n)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1}^{(n)} \\ y_{t-1}^{(n)} \end{bmatrix} + \begin{bmatrix} \sigma_P{}_{t-1}^{(n)} \\ \sigma_P{}_{t-1}^{(n)} \end{bmatrix} = A \begin{bmatrix} {x'}_{t-1}^{(n)} \\ {y'}_{t-1}^{(n)} \\ \dot{x}'^{(n)}_{t-1} \\ \dot{y}'^{(n)}_{t-1} \end{bmatrix} + \begin{bmatrix} \sigma_P{}_{t-1}^{(n)} \\ \sigma_P{}_{t-1}^{(n)} \\ \sigma_V{}_{t-1}^{(n)} \\ \sigma_V{}_{t-1}^{(n)} \end{bmatrix} ,$$

such that

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

where δt is set to 1 as the propagation process must be set out in terms of discrete time t.

1.3 Propagation

Propagation of state density over time is performed via the `propagate` function, performing the calculation expressed in (1) to generate the samples of the new sample set S_t, which approximates the *a priori* density distribution $p(x_t|Z_{t-1})$. In order for the new particles to lie within the image frame, out-of-bound coordinate values are clipped.

This concludes the prediction step.

1.4 Observation

After the prediction step, the update step is performed to obtain an approximation of the *a posteriori* density distribution $p(x_t|Z_{t-1})$. This is done by reweighing all samples of the sample set S_t according to new measurements z_t. These are collected by means of the function `observe` which, after computing the color histogram of each particle of the bounding-box, calculates the weights from the chi-squared distance between the particles and targets color histograms, as written below:

$$\pi^{(n)} = p(z_t|{s'}_t^{(n)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{\mathcal{X}^2(CH_{s^{(n)}}, CH_{target_i})^2}{2\sigma^2}\Big) ,$$

where σ is the standard deviation of the observation noise. Note that in order for the weights to be equal to probabilities, they are normalized with respect to their sum.

1.5 Estimation

The mean state of the system is given by the weighted sum of the states of all particles. Note that the mean state color histogram is used to update the target color histogram at each iteration.

1.6 Resampling

The particles are randomly sampled with replacement by means of the `numpy` library function `numpy.random.choice`. Alternatively, low-variance sampling methods could be employed instead.

2 Experiments

The performance of the given implementation of the CONDENSATION algorithm was evaluated on three test videos. Experimental results are described below.

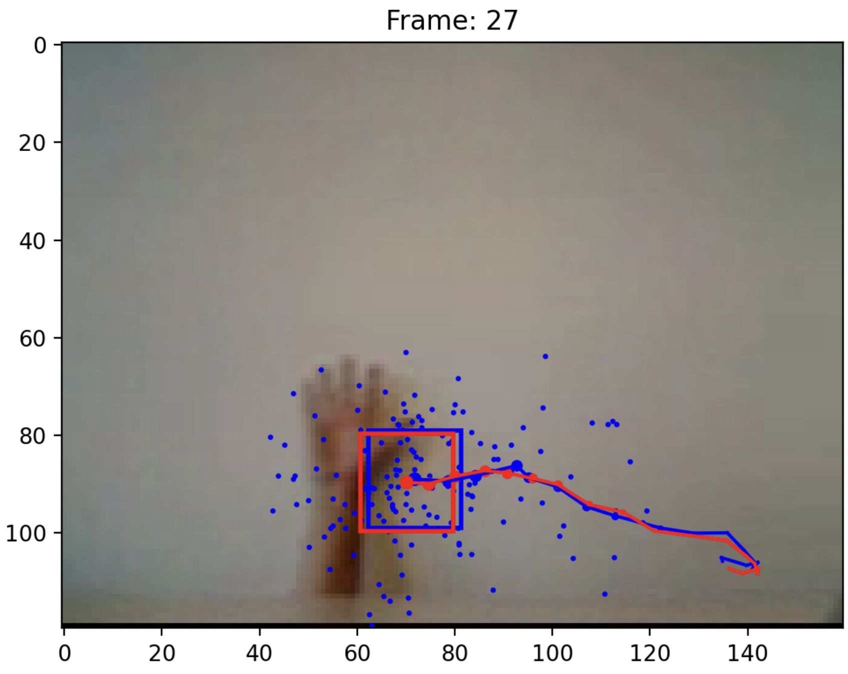


Figure 1: Test Video 1

2.1 Video 1

Video 1 pictures a hand moving from right to left on a uniformly-colored white background.

The tracking was performed with sufficiently good results both adopting a no-motion and constant-velocity model. Although after the first half of the trajectory, the tracker begins to follow the wrist instead of the hand due to changes in lighting and shading which make the wrist more similar to the original frame than the hand itself (Figure 1).

This issue can be mitigated by updating the target color histogram at each iteration so that it becomes more sensitive to changes in lighting. This is done by appropriately setting to a non-zero value the parameter α in the formula $CH_{target} = (1-\alpha)CH_{target} + \alpha CH_{mean-state}$, as shown in Figure 2.1, 2.2 and 2.3.

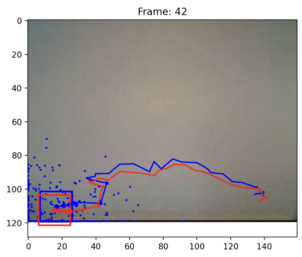


Figure 2.1: $\alpha = 0.25$

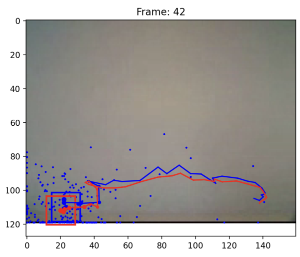


Figure 2.2: $\alpha = 0.50$

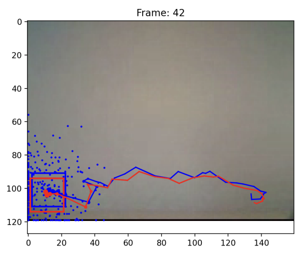


Figure 2.3: $\alpha = 0.75$

2.2 Video 2

Video 2 portrays a moving hand on a non-uniform background, and it additionally presents some occlusion during the hand's motion.

As shown in Figure 3, the no-motion model fails to track the object when the noise parameter σ_P on the particles position is set to a low value, i.e. $\sigma_P = 1$. However, when increasing σ_P, more particles are drawn from the non-occluded area, allowing to successfully track the hand (Figure 3.1 and 3.2).

On the other hand, when using the constant-velocity model, the systems parameters need to be carefully tuned for the tracker to succeed in the task. Note that for the following results, since little vertical motion is observed, the initial velocity is set to [10 1].

In order to overcome occlusion, the system noise parameter needed to be set at high values, so that the sample set was spread enough to recognize the hand when it became visible again after the obstacle.

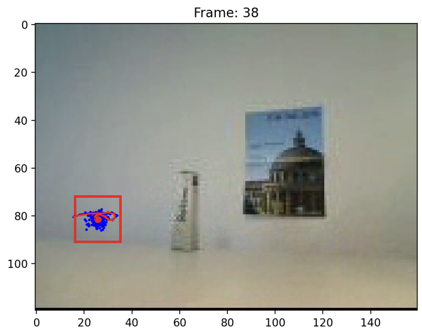


Figure 3.1: Test Video 2 with no-motion model, $\sigma_P = 1$.

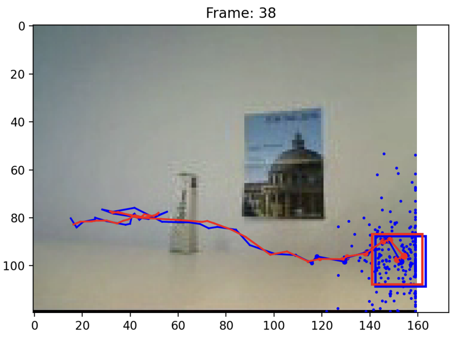
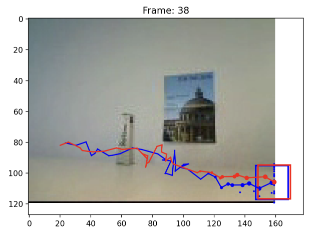


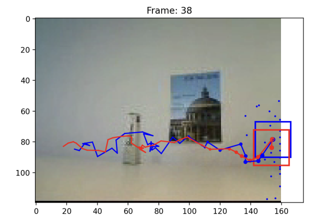
Figure 3.2: Test Video 2 with no-motion model, $\sigma_P = 10$.

However, if σ_P is set to a overly high value, tracking fails as the sample set is too noisy.

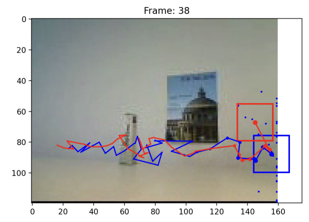
Furthermore, the number of particles needed to be high enough for the tracker to be sufficiently accurate, at the expense of a computational overload. As shown in Figure 4, better results were obtained by increasing the number of particles for each value of σ_P. Nonetheless, setting number of particles is too high with a high position noise could potentially lead the tracker to fail since particles will be too spread out.



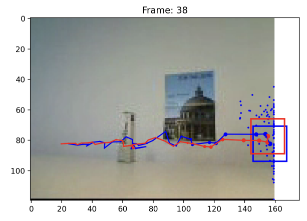
$num\_particles = 30,\ \sigma_P = 8$



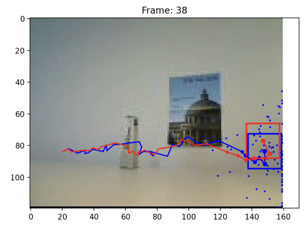
$num\_particles = 30,\ \sigma_P = 10$



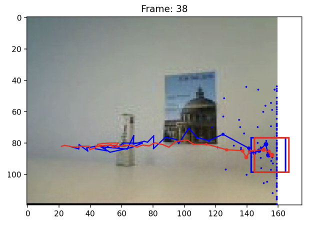
$num\_particles = 30,\ \sigma_P = 15$



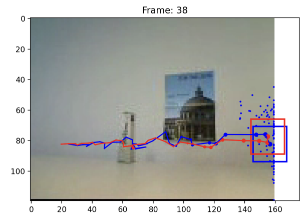
$num\_particles = 100,\ \sigma_P = 8$



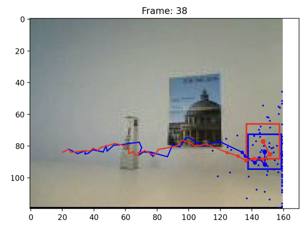
$num\_particles = 100,\ \sigma_P = 10$



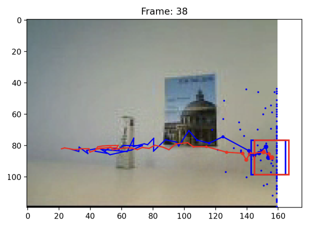
$num\_particles = 100,\ \sigma_P = 15$



$num\_particles = 150,\ \sigma_P = 8$



$num\_particles = 150,\ \sigma_P = 10$



$num\_particles = 150,\ \sigma_P = 15$

Figure 4

The measurement noise must also be measured carefully. If $\sigma_{observe}$ is set to be too small, the weights of the particles will be too small for them to represents good candidates to follow and therefore being stuck in occlusion, whereas for $\sigma_{observe} = 1$, all particles have high weights and thus represent good candidates, resulting in the tracker not following the hand at all.

2.3 Video 3

Video 3 features a ball bouncing on a wall. By applying the parameters of the optimal constant-velocity model for Video 2 ($\sigma_{position} = 8$, $num_particles = 150, \sigma_{observe} = 0.1$), the tracker manages to follow the ball until it hits the wall, but fails to track it when it bounces back (Figure 5.1). The poor results are given by the fact that the velocity is not constant in the ball's motion as it experiences a change in direction. Thus, a more accurate result is obtained by setting the velocity noise parameter to a higher value (Figure 5.2).

Alternatively, the no-motion model could be employed by setting a high value for position noise, as shown in Figure 5.3.

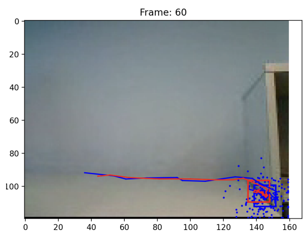


Figure 5.1

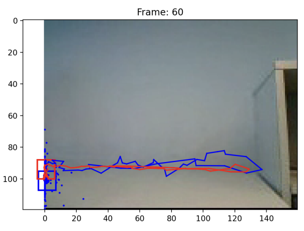


Figure 5.2: $\sigma_{velocity} = 3$

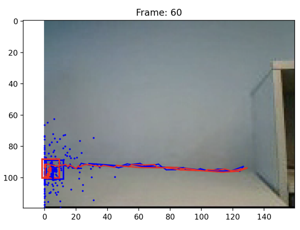


Figure 5.3: No-motion model

2.4 Parameter impact

- **Number of particles.** In general using a higher number of particles results in higher tracking accuracy, however if it is set too high coupled with high noise parameters, it might lead to the tracker following diverse paths.
- **Histogram bins.** Increasing the number of bins of the color histograms results in a more precise representation of a bounding-box. However, overly-increasing its value might lead to a non-negligible increase in chi-squared difference between similar bounding boxes. Instead, by setting it to a lower value, robustness to noise increased at the expense of a decrease in representation accuracy.
- **Appearence model updating.** If $\sigma_{observe} = 0$, the target histogram is never updated, and thus is always equal to the initially selected bounding box. In order to increase robustness to changes in lighting, its value must be increased appropriately. However, if it set too high, if the target is lost by the tracker for a small number of frames, it won't be retraceable in the future, leading to high risk of fail in occlusion instances.