Visual Tracking: Particle Filtering.

1 Introduction

This practical course deals with particle filtering algorithms applied to Visual Tracking.

2 Particle Filter

Particle Filter is a stochastic sequential algorithm that estimates the posterior distribution at each time of a state sequence, using the previous state distribution, a prediction function and a likelihood function (observation). *SIR* (*Sequential Importance Resampling*) filters, the most popular in computer vision, are based on an Importance Sampling step.

We can define an approximation of $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})$, the state posterior probability function at time $t-1$ using N discrete weighted samples $\{\mathbf{X}_{t-1}^n, \pi_{t-1}^n\}_{n=1}^N$:

$$p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) \approx \sum_{n=1}^{N} \pi_{t-1}^n \delta(\mathbf{X}_{t-1} - \mathbf{X}_{t-1}^n), \tag{1}$$

where δ is the Kronecker function and π_{t-1}^n is the weight associated to the n^{th} sample, $n \in 1...N$, such as $\sum_{n=1}^{N} \pi_{t-1}^n = 1$. The discrete approximation of Chapman Kolmogorov equation is given by:

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t-1}) \approx \sum_{n=1}^{N} \pi_{t-1}^n p(\mathbf{X}_t|\mathbf{X}_{t-1}^n), \tag{2}$$

where $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ is a probability law that describes the dynamics of the system. The equation (2) defines a set of N weighted particles $p(\mathbf{X}_t|\mathbf{X}_{t-1}^n)$, with weigths π_{t-1}^n. This function produces the probability function of the current state, given the previous observation. It updates the previous posterior probability with a dynamic model. The sequential Bayesian Filter can be written as follow:

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx C^{-1} p(\mathbf{Z}_t|\mathbf{X}_t) \sum_{n=1}^{N} \pi_{t-1}^n p(\mathbf{X}_t|\mathbf{X}_{t-1}^n). \tag{3}$$

Some samples have large weights whereas other one have ones close to zero. In order to focus around interesting areas of the state space, a new sampling has to be done. We use an importance sampling algorithm, combined to the dynamic step that uses the proposal law:

$$\mathbf{X}_t^n \sim q(\mathbf{X}_t) = \sum_{n=1}^{N} \pi_{t-1}^n p(\mathbf{X}_t|\mathbf{X}_{t-1}^n) \tag{4}$$

This equation produces, at each time, a new set of unweighted particles. For each sample, $\mathbf{X}_t^n$ The likelihood function is applied: $\pi_t^n = P(\mathbf{Z}_t|\mathbf{X}_t^n)$. A set of weighted samples are then

generated $\{\mathbf{X}_t^n, \pi_t^n\}_{n=1}^N$, that approximates $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$, the posterior probability function of the state at time t.

A visual representation of the *SIR* algorithm is given into figure 1. It is divided in three main steps:

- (a) Importance sampling: particles are drawn according to their weight from the previous set of particles at $t-1$. This step copies particles with a high weight and removes particles with a low weight. The output of this step is a set of unweighted particles (all particles are associated with the same weight) that approximates the same distribution than the input set.
- (b) Dynamic prediction: move each particle according to a proposal function (that describes the dynamics of the system to track) $p(\mathbf{X}^*|\mathbf{X})$. When no information is available on the dynamic, a random step can be applied $p(\mathbf{X}^*|\mathbf{X}) = \mathcal{N}(0, \sigma)$, where the value of σ has to be carefully designed.
- (c) Estimation of a weight for each new particle, using the likelihood function of the observation $\mathbf{Z}_t$ at time t, given the sample $\mathbf{X}_t^n$: $\pi_t^n = P(\mathbf{Z}_t|\mathbf{X}_t^n)$

In this practical, we want to track a manually selected coloured area. The state vector is composed by $\mathbf{X} \doteq (x, y, w, h)$ with x, y, the position of the upper left point of the ROI, w, the width of the roi and h, the height of the bounding box. The following of this course should be done using the file `track_pf.py`

3 Questions

1. The Importance sampling algorithm is defined into the function `IR(part,w)` with *part*, an $(nbp \times 4)$ array, nbp the number of particles and w a nbp size vector of associated weights. This function generates Nbp output particles (with $w = 1/Nbp$). You have to analyse it and explain the principle of importance resampling.
2. The function `predict(part,im,bx=3,by=3,bw=0,bh=0)` applies the dynamics to a set of particles. In our case, we want to apply a random step. You have to fill the function using the following tips:
   - `np.random.randn(n1,n2)` returns a $n1 \times n2$ array of random numbers that follows a normal law ($\mathcal{N}(0,1)$ with mean=0 anf std=1)
   - `toto.shape` returns an nd-vector providing the shapes of a nd-array.
   - `np.clip(x,xmin,xmax)` returns, for each element of the array x, the max between x and $xmin$ and the min between x and $xmax$. It can be used to be sure that no particle is generated outside the image.

   You have to complete the function in order to apply an additive noise to each particle. (bx, by, bw, bh) are respectively the standard deviation of the noise for each components of the state vector.
3. The function `featureHist(pa,im,nbin=16)` computes a $3 \times nbin$ feature vector corresponding to the concatenation of RGB histograms of the ROI image associated to particle *pa*. Explain why color histogram can be a good description for objects visual tracking?

4. the function `likelihood(part,im,model,la = 20)` computes the likelihood values (weights) for a set of particles. It uses the Battacharrya distance (defined to measure similarities between two mass functions). What is the mathematical link between the Battacharrya distance and the weight? What append is the parameter λ (la) increases?

5. the `main` function described the particle filter algorithm applied to a video sequence. The display can be done in two ways: 1) using opencv (fast but not easy to show plots or histograms) or 2) using matplotlib (slow but can be tuned easily). See the file `my_func.py` to have details. You have to complete the file into the `While` loop to implement the three stages of the particle filter: 1) sampling, 2) prediction and 3) likelihood function.

6. You have to display, for several standard deviations (1, 10 et 50) of the random walk, the marginal distribution of the state (x or y) and discuss the results. You should swap to `matplotlib` display with the flag `DISPLAY_OPENCV = False` to do that.

7. You have to display, for several sizes of the filter (5, 10, 50, 100 particles) the output of the filter.

8. Remove the Importance Resampling step and discuss the resulting behaviour of the tacking process.

9. The variable `step` is a sample time factor applied to the video sequence. try to increase this factor and discuss the consequence on the tracking.

10. According to the time, try to modify the model in order to change the RGB histogram representation by a HS (hue, saturation) one.
    The opencv function `cv2.cvtColor(image, CV.XXX)` will help you.

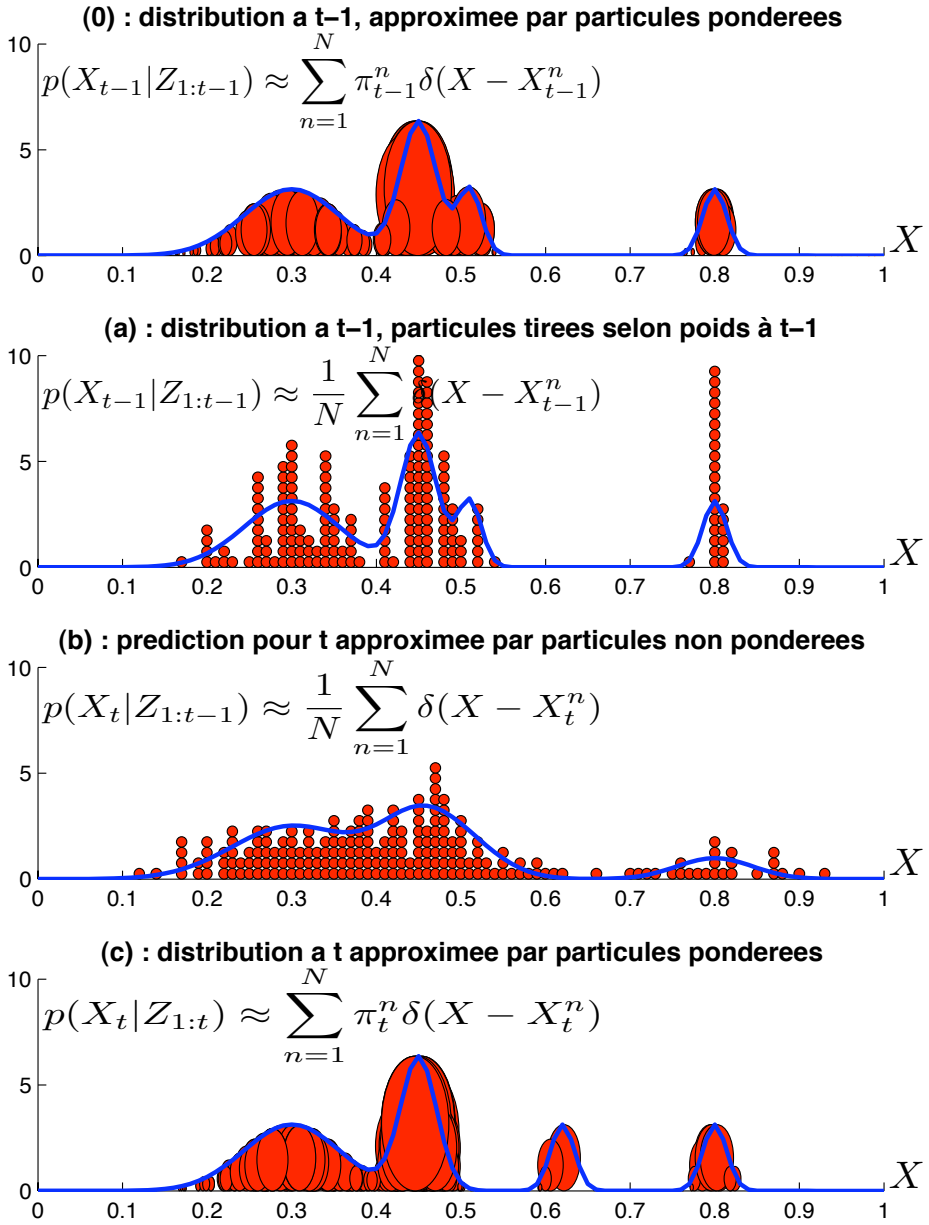

Figure 1: Illustration of the SIR particle filter algorithm: (a) $t-1$ posterior distribution approximated by a set of weighted particles, (b) $t-1$ posterior distribution approximated by a set of unweighted particles (after the Importance sampling step), and c), t current posterior distribution approximated by a set of weighted particles by the likelihood function.