Computer Vision Assignment 6

Saurabh Vaishampayan
svaishampaya@student.ethz.ch

December 2021

1 Implementation details

1.1 color_histogram function

Here we compute the histogram for each of the R,G,B channels for a specified bounding box in the frame. Before computing the histogram, we ensure that all the points of the bounding box lie in the video frame. If the location of the bounding box is such that some points lie outside, we clip the locations of the box to stay within the frame. However, this can reduce the points in the histogram computed. So before returning the histogram (computed using np.histogram()), we normalise it by the total counts.

1.2 Derivation of the A matrix

Given the state of the system s_{t-1} at time $t-1$, we want to find the transformation to get the state of the system $s_t$at time t. The transformation depends on the motion model we are assuming. If the evolution is described by a linear stochastic differential equation, with noise at time t given by w_t, then we have:

$$s_t = As_{t-1} + w_{t-1} \tag{1}$$

In the assignment we are considering 2 models: No motion and motion with constant velocity.

1. No motion model:

   - Here the state of the system is given only by its location as we are not considering the effects of velocity.
   - State of system described by $s = [x, y]^T$ since we are in a 2D setting.
   - Since we have no motion assumption, we will have:

     $$x_t = x_{t-1} + w_{t-1}$$

     $$y_t = y_{t-1} + w_{t-1}$$

     In matrix form this transformation can by modelled as:

     $$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \end{pmatrix} + w_{t-1}$$

   - Hence A is given by:

     $$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2}$$

2. Constant velocity model:

   - Here the velocity is assumed to be constant (upto noise).
   - State of system is described both by location of the particle as well as its velocity. $s = [x, y, v_x, v_y]^T$.

- Once we have the velocity, we can model the evolution of location over time by:

$$x_{new} = x_{old} + v_x \delta t$$

$$y_{new} = y_{old} + v_x \delta t$$

If we assume that between adjacent frames of the video, 1 unit of time has passed we get $\delta t = 1$. If we have a frame rate then simply replace δt with it (and suitable scaling also needs to be made for velocity estimates). Here in our experiment we assume $\delta t = 1$ and derive A accordingly.

$$\begin{pmatrix} x_t \\ y_t \\ v_{x,t} \\ vy, t \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ v_{x,t-1} \\ vy, t-1 \end{pmatrix} + w_{t-1}$$

- Hence A is given b:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

1.3 Propagation

Here we have to compute the state of each particle at new time instant given the data at previous time instants. This is done using the linear stochastic equation (1). Depending on the model, we calculate and use the A matrix. The noise w_{t-1} is normally distributed with specified variance. This is obtained by scaling the standard normal random variable by appropriate standard deviations (which are different for position and velocity uncertainty).

2 Observation

Here we have to calculate the weights for each particle depending on how good/bad the descriptor of the particle is compared to the target descriptor. The descriptor for each particle is given by its color histogram within a bounding box with the particle at its centre. This is computed using the color_histogram function (which also takes care of boundary effects as mentioned earlier). The weights for particle i are given by:

$$\pi_i = \frac{1}{\sqrt{2\pi}\sigma_o} e^{-\frac{\chi^2(CH_i, CHtarget)}{\sigma_0^2}} \tag{4}$$

Here σ_o is the observation noise standard deviation. The χ^2 cost is computed using the already given function. It computes how similar/dissimilar the color histograms of the target and the color histogram of the particle i are. Afterwards, before returning the weights, we normalize them so that the sum of probabilities for particles is 1.

2.1 Estimation

We have to compute the mean state of the system, which is given by the weighted sum of states of individual particles. We simply multiply the state of each particle by its weight and sum along the first axis to get the weighted mean state. We have already normalized the weights so we do not need to divide by the sum of weights.

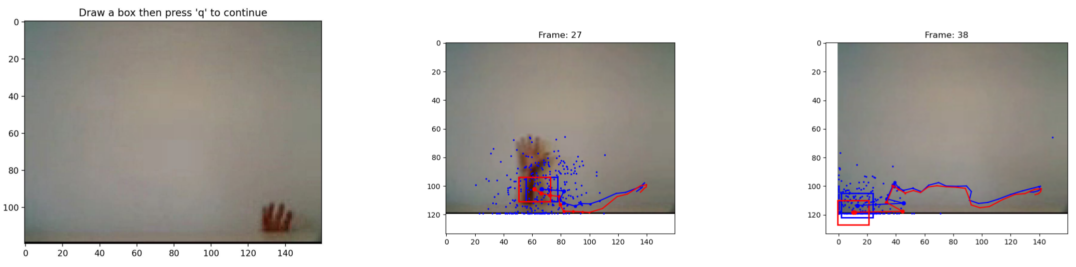
2.2 Resampling

For implementing resampling we used low variance resampling. The algorithm for this was implemented from Page 110 in the book "Probabilistic Robotics" by W Burgard et al.

3 Experiments

3.1 Video 1

We use the no motion model for tracking the hand. This video depicts a moving hand in a uniform white background. We obtain the following results for tracking: We can see that the



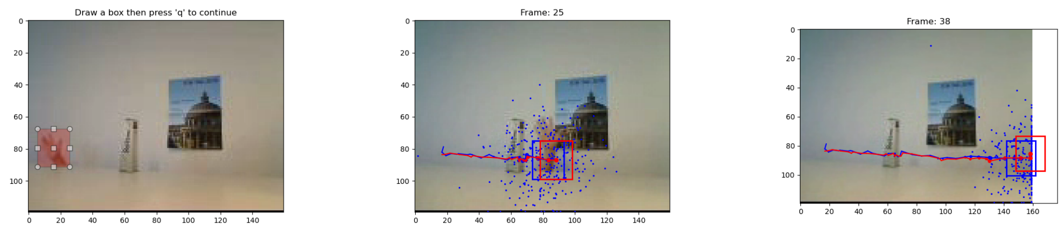
(a) Initialisation (b) Intermediate Frame (c) Intermediate Frame

Figure 1: Video 1 Tracking with No Motion Model

tracking is quite good, as the hand is being followed. However, we can see that even if we set the initial target box to the tip of the fingers, the tracker ends up following the wrist or the upper part of the forearm. This is because in the initial frame, only the fingers are visible and they are darker than while during the intermediate frames. During the intermediate frames, the whole palm is visible and is lighter in appearance and also has more contrast. In the intermediate frames, the wrist and the forearm have a color distribution similar to the fingers in the initial frame and hence the tracker follows these. The effect of this can be overcome by increasing the appearance update parameter.

3.2 Video 2

This video also contains a hand moving. However the background is not uniform and also the hand is occluded in an intermediate part of the video. We now vary the model and parameters and study the effects.



(a) Initialisation (b) Intermediate Frame (c) Intermediate Frame

Figure 2: Video 2 Tracking with No Motion Model with high measurement noise

First we try the no motion model. We study the effects of small and large system noise. The results are given in Figure 2 (for large system noise). We can see that even though the hand is being occluded by an object at some point, the no motion model still manages to track the hand throughout the entire video. This can be explained by the quite high system noise ($\sigma_{position} = 15$): when the hand is occluded, since the stochastic term is quite big, the particles are spread on a large surface. Therefore, when the hand reappears after the occlusion, some particles will still be on the hand and this will allow the tracking to continue. When $\sigma_{position}$ is reduced to 1 (low system noise), we see that the tracking fails, as in Fig 3. Now we study the effects of using

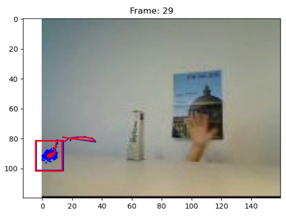


Figure 3: Video 2 tracking No motion model with low measurement noise

constant velocity model. We keep the $\sigma_{position} = 1$ as in the previous failed no motion model. Since the motion is primarily along x axis, we set the initial velocity = [5,0]. We can see in Fig 4 that the constant velocity model allows the model to perform better even if it has low position noise as in the previous failed no motion model.

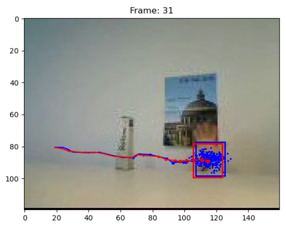
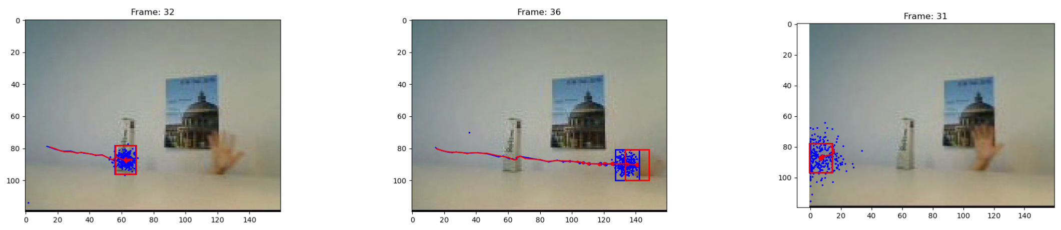


Figure 4: Video 2 tracking constant velocity model

Now we change the measurement noise and see the effects. This is controlled by $\sigma_{observation}$ parameter. We show the effects for very low measurement noise, very high measurement noise and optimal measurement noise:



(a) Low measurement noise $\sigma_{obs} = 0.01$

(b) Optimal measurement noise $\sigma_{obs} = 0.1$

(c) High measurement noise $\sigma_{obs} = 1$

Figure 5: Video 2 Effects of measurement noise

In the case of a large measurement noise, the weights are widely spread so importance is given even to particles that shouldn't be relevant. All the particles are good candidates. The particles stay stuck in the uniform wall.
In the case of low measurement noise, the weight distribution is very narrow and particles that have a slightly different colour histogram will have small weight. Small changes therefore mess up the tracking as can be seen in Fig5a. If the measurement noise is very small, none of the particles are good candidates resulting in sum of weights being 0.
We summarise the optimal parameters for Video 2 which are used for tracking:

Parameter	Value
Measurement noise	0.1
System Noise(Position)	3
Initial velocity	[5,0]
Appearance update	0.3
System noise(velocity)	1

3.3 Video 3

Now we first use the same optimal parameters as used for Video 2. The results are in Fig 6. As we can see, the result isn't as good as with video 2: the moment the ball bounces, the particles keep being updated to the right and they never find the ball again.

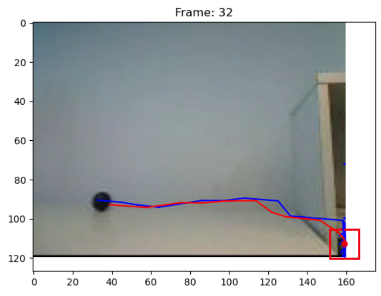


Figure 6: Video 3 tracking with constant velocity model and Video 2 optimal parameters

Constant velocity is not a good assumption for this model because the ball bounces back, thereby incurring a change in velocity direction. Also the collision is not elastic and the speed also changes during collision. We now try to increase the system noise for velocity σ_v from 1 to 3 to allow for changes in velocity. The result is in Fig 7.

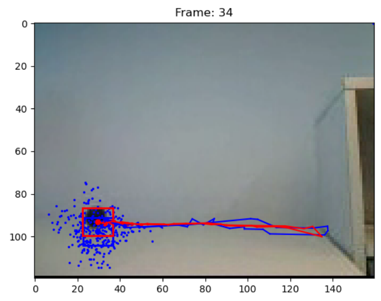


Figure 7: Video 3 tracking with constant velocity model and higher velocity variance

However an even better approach would be to try the no motion model with a larger position noise. The particles have more degrees of freedom and and are not restricted to have constant velocity, allowing more particles to find the ball resulting in better tracking. The results of this (plotted in Figure 8) are better than the constant velocity model with higher velocity variance from Fig 7(previous experiment).

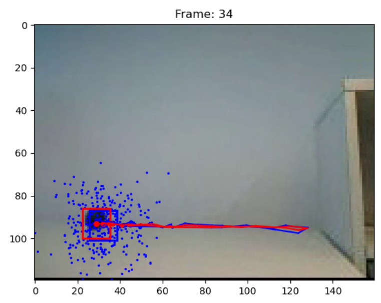


Figure 8: Video 3 tracking with no motion model and higher position variance

4 Effects of parameters

1. Appearance model updating: By allowing appearance model to update we get robustness against lighting changes and constrast changes. If this is set to zero, the target histogram is always the histogram of the initially selected bounding box. The initialisation may not be a good descriptor and may fail in lighting/contrast changes. However if this parameter is kept to high, then the target histogram has a high contribution of just the previous data. If there are some errors, these will get compunded because the reference histogram (previous value) is wrong and all subsequent comparisons will be compared to evolution of this reference histogram leading the particles astray. The optimal parameter will allow a combination of initial histogram and model update.

2. Histogram bins: If histogram bins are high, we get better resolution. However too many bins can result in large difference between 2 histograms that might be closer in reality. If the number of bins is kept small, we get robustness to noise. But if number of bins are too small, it becomes tough to distinguish two patches that might be very different in reality.

3. Number of particles: Higher number of particles results in better performance and robustness at the expense of more computational load. If we have too few particles, they may not be able to track movements in presence of occlusions. This is because we are trying to represent a continuous multimodal probability distribution by a finite set of particles in Monte Carlo. If total particles are too small, there is smaller probability that the particles will be able to cover the intended state set.