Computer Vision
CS 6476 , Spring 2018

PS 6

Professor :
Cedric Pradalier

Author :
Melisande Zonta

April 20, 2018

Contents

1 Particle Filter Tracking

1.1 First Tracking

After implementing the particle filter (see below), we use it to track Mitt Romney's face on the presidential debate video after extracting the template based on the accompanying text file (Figure 2).
We decided to use the standard deviation mentioned in the text for the MSE, and to add the same amount of noise to both the x and y position of the center.
The same value is used at the initialization and in the dynamics model, since it seemed to work best that way. Moreover, since the movement of the face starts quickly in the video, having some initial spreading of the particles appears useful. We get a pretty good result even after he turns his face (Figure 1). Although the patches content then does not match exactly the template, the appearance of his profile is not too far away, which explains the proper working of the tracker in this case.

Table 1: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[5. 5. 0.]
Number of particles	100

```
class TrackingPF():
    def __init__(self, template, initial_pose, initial_uncertainty, N):
        self.template = template
        self.initial_height = template.shape[0]
        self.initial_width  = template.shape[1]
        self.N = N

        # Initialisation of the particle cloud around the initial position
        # and size
        self.X = np.vstack([initial_pose[0,0], initial_pose[1,0], 1.0])
        self.particles = [self.clip_scale(self.X + self.draw_noise(
    initial_uncertainty)) for i in range(0,self.N)]

    def draw_noise(self, std):
        noise = np.zeros(std.shape)
        for i in range(std.shape[0]):
            noise[i,0] = np.random.normal(0, std[i,0], (1,1))
        return noise

    def clip_scale(self, particle):
        particle[2,0] = max(0.1, min(particle[2,0], 1.4))
        return particle

    def predict(self, precision):
        new_particles = []
        for p in self.particles:
            new_particles.append(self.clip_scale(p + self.draw_noise(precision)
    ))
        self.particles = new_particles

```

```
    def update(self, frame, std_mse):
        W = []
        for p in self.particles: # calculate weights
            rows = self.initial_height * p[2,0]
            cols = self.initial_width * p[2,0]
            minr = max(0, min(int(p[1,0]-floor(rows/2)), frame.shape[0]-1))
            maxr = max(0, min(int(p[1,0]+ceil(rows/2)), frame.shape[0]-1))
            minc = max(0, min(int(p[0,0]-floor(cols/2)), frame.shape[1]-1))
            maxc = max(0, min(int(p[0,0]+ceil(cols/2)), frame.shape[1]-1))
            slicer = slice(minr, maxr)
            slicec = slice(minc, maxc)
            patch = cv2.cvtColor(frame[slicer, slicec], cv2.COLOR_BGR2GRAY)
            #print(int(rows), minr, maxr, int(cols), minc, maxc)
            try:
                temp = cv2.resize(self.template, (patch.shape[1], patch.shape
[0]))
                MSE = np.sum(np.power(np.float64(temp) - np.float64(patch), 2))
                MSE /= patch.size
                w = np.exp(-MSE / (2 * std_mse * std_mse))
            except:
                w = 0
            W.append(w)

        W /= np.sum(W) # normalize

        Q = np.cumsum(W)

        new_particles = []
        for i in range(self.N):
            p = self.particles[bisect.bisect_left(Q, np.random.uniform(0,1))]
            new_particles.append(p)

        self.particles = new_particles

    def update_mean(self):
        X = np.zeros((3,1))
        for p in self.particles:
            X += p
        self.X = X / len(self.particles)
        return self.X
```

Figure 1: ps6-1-1-frames-28-84-144.png

Figure 2: ps6-1-1-template.png

1.2 Window Size Discussion

In this section, we tried to increase and to decrease window size to see the impact on the tracker's accuracy.

Table 2: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[5. 5. 0.]
Number of particles	100
Small scale	0.1.
Large scale	2.5.

We see in Figure 3 which used a large template (Figure 5) that increasing the 4 size of the template in this case does not have any real impact and our tracking is still good. However, greatly reducing the size of the template (Figure 6), down to a tenth of its original size, our tracker gets lost and ultimately considers one of the stars in the background as a very likely candidate (Figure 4). All in all, we could list some advantages of both approaches. Using one or the other will greatly depend on the specifics of the video and the target we wish to track.

- Small window:
  - useful if the target contains a small detail that is constantly visible and stands out from the rest of the video (e.g. eyes, QR code, etc) ;
  - possibly problematic if the target is moving fast, since the small size of the window would make it easier to loose.
- Large window:
  - interesting since it takes a larger portion of the image and might be more robust for example when tracking an object that is rotating and standing out from the background ;
  - pointless if the large window includes a sizable portion of the background and if this one is changing quickly (e.g. a pedestrian walking sideways in front of a succession of varying facades).

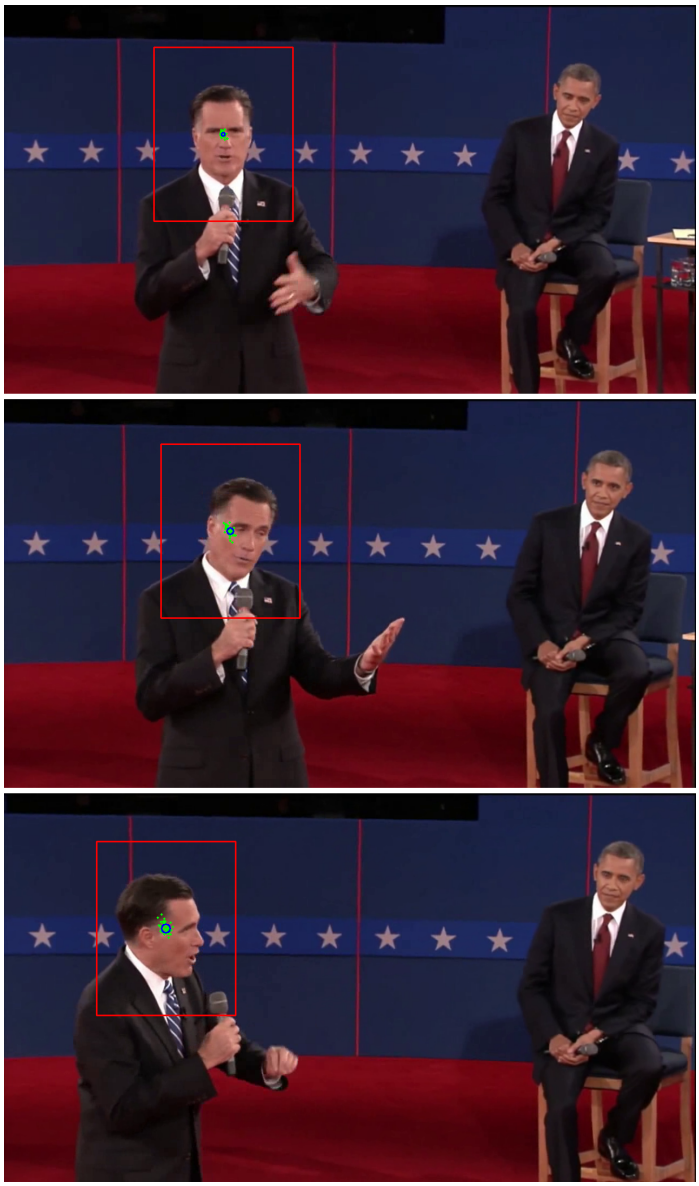

Figure 3: ps6-1-2-frames-large-28-84-144.png

Figure 4: ps6-12-template-large.png

Figure 5: ps6-12-frames-small-28-84-144.png

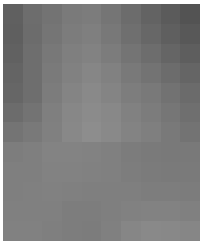

Figure 6: ps6-12-template-small.png

1.3 MSE Standard Deviation Discussion

We now try to see the impact of σ_{MSE} on the tracking.

Table 3: Parameters

MSE standard deviation (low)	2.0
MSE standard deviation (high)	20.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[5. 5. 0.]
Number of particles	100

Although the tracking works well in both cases (see Figure 7 and Figure 8), we notice that it is "smoother" with a small value of σ_{MSE} (particles are all tightly grouped), while a larger value, which spreads out the particles, gives a less clean result.
This is explained by the fact that a smaller σ_{MSE} implies that the exponential decay is stronger. Thus, the weights for particles with a large MSE gets small really fast and they have a low probability to get picked at resampling.

Figure 7: ps6-1-3-frames1-28-84-144.png

Figure 8: ps6-12-template-small.png

1.4 Particle Number Discussion

We now wish to optimize the number of particles. We try to reduce it as much as possible while preserving a good quality of tracking.

Table 4: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[5. 5. 0.]
Number of particles	100

Figure 9: ps6-1-4-frames-28-84-144.png

Our trials show us that reducing the number of particles down to just 10 still gives a proper tracking on the full video. However, a small number of particles makes the tracking very unstable so we settled on 30 particles, which gives us a reliable and quite stable tracking

(Figure 9).

1.5 Noisy Tracking on Romney

We now apply our tracker to a noisy version of the debate, where bursts of noise appear several time along the course of the video. We make some tests with three different values of σ_{MSE}.

Table 5: Parameters

MSE standard deviation (low)	5.0
MSE standard deviation (mid)	10.0
MSE standard deviation (high)	15.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[5. 5. 0.]
Number of particles	30

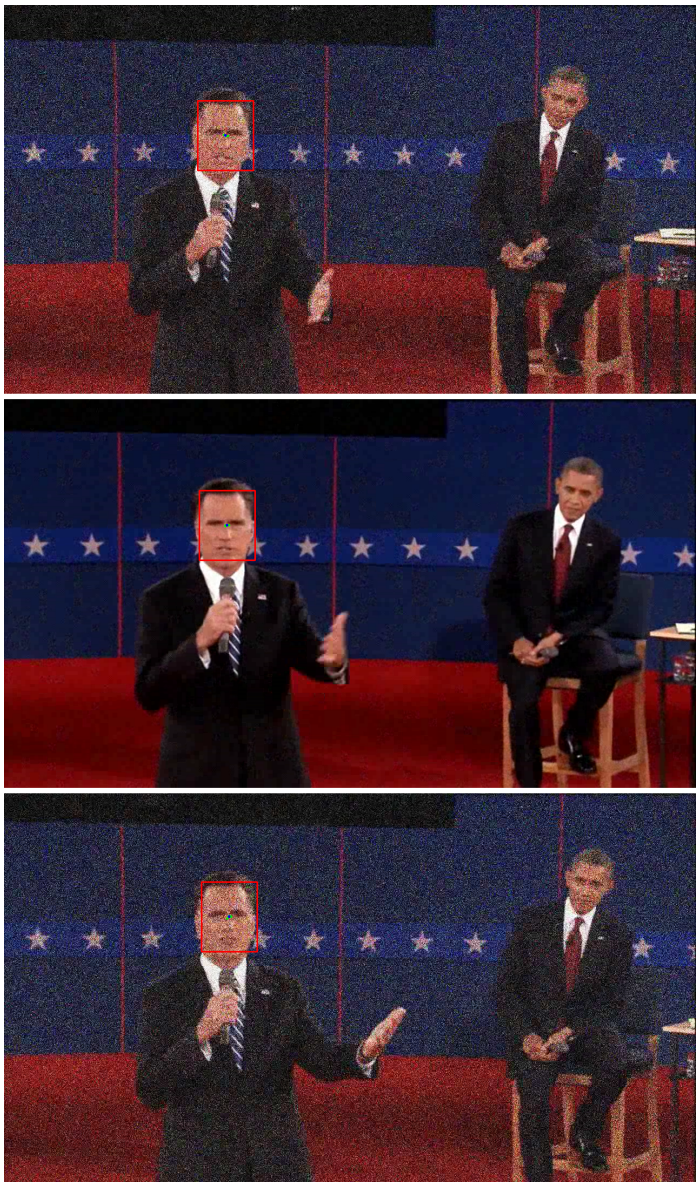

Figure 10: ps6-1-5-frames1-14-32-46.png

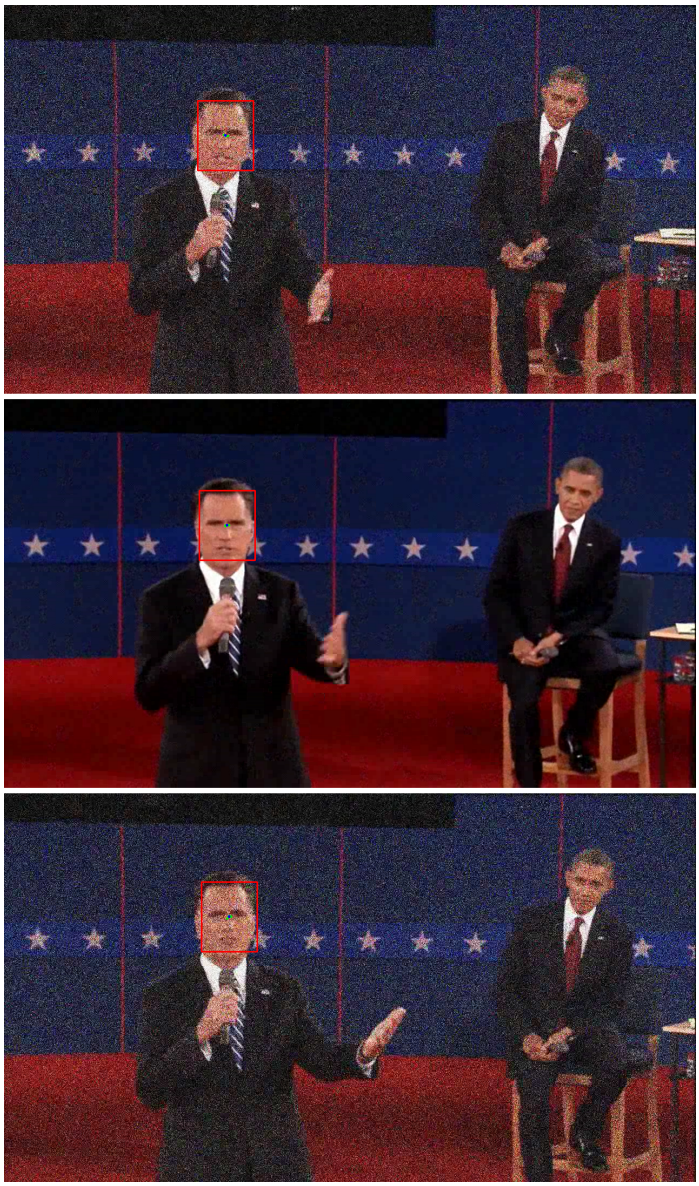

Figure 11: ps6-1-5-frames1-14-32-46.png

The three trackers do fairly well at tracking Romney's face, and all three offer the same response to an increasing level of noise. In this situation, the particles all tend to spread out around the mean value. The large value of σ_{MSE} (Figure 11), as explained above, gives a result that is not very smooth and particles get spread in noisy phases. The middle value gives a cleaner tracking but for this particular case we would probably choose the lowest (Figure 10) of the 3, because it is less sensitive to the varying noise (particles spread less easily). If the noise bursts were longer, a small σ_{MSE} would probably help in keeping a good track of Romney's face, while higher values would see their mean keep increasing.

2 Appearance Model Update

2.1 Appearance Model Update

We now wish to track the hand of Romney, which is see at different angles throughout the video. We thus need to update the template in order to make it evolve with the movement. This is done in the new function shown below.

We also decide to take a small initial template, which does not cover the full hand (Figure 13). This helps in focusing on the hand and discarding as much of the background as possible.

Table 6: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[20. 20. 0.]
Number of particles	100
Alpha	0.2

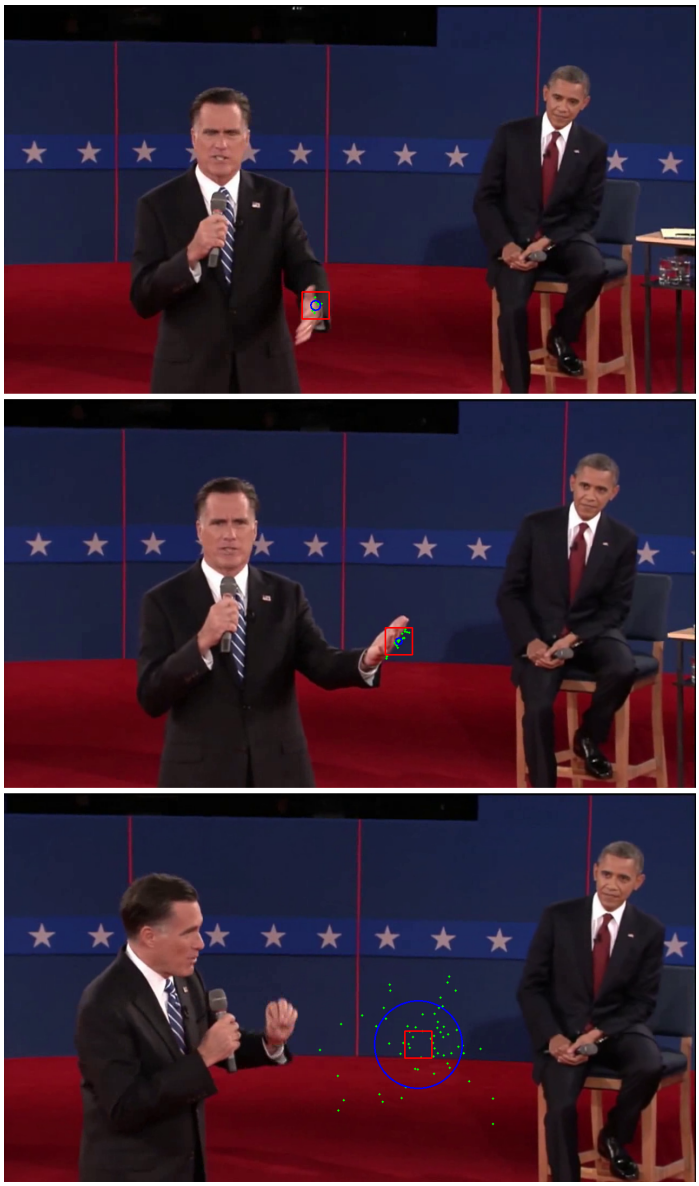

Figure 12: ps6-2-1-frames-15-50-140.png

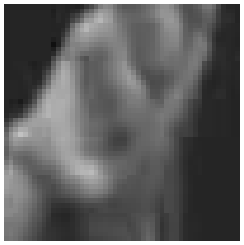

Figure 13: ps6-2-1-template.png

Moreover, we increase the standard deviation on the dynamics model and the number of particles so that we get a better chance at having some particles to follow the hand, especially since it moves much faster than the head.
Finally, the constant value α used in the template update step is chosen to be low, so that changes in appearance is taking into account slowly (history has more weight). Indeed since our template is limited to a small portion of the hand, we suppose it won?t change too fast in appearance.
The result is visible in Figure 12.

```
1
2     def update_template(self, frame, alpha=None):
3         rows = self.initial_height * self.X[2,0]
4         cols = self.initial_width * self.X[2,0]
5         minr = max(0, min(int(self.X[1,0]-floor(rows/2)), frame.shape[0]-1))
6         maxr = max(0, min(int(self.X[1,0]+ceil(rows/2)), frame.shape[0]-1))
7         minc = max(0, min(int(self.X[0,0]-floor(cols/2)), frame.shape[1]-1))
8         maxc = max(0, min(int(self.X[0,0]+ceil(cols/2)), frame.shape[1]-1))
9         slicer = slice(minr, maxr)
10        slicec = slice(minc, maxc)
11        patch = cv2.cvtColor(frame[slicer, slicec], cv2.COLOR_BGR2GRAY)
12        temp = cv2.resize(self.template, (patch.shape[1], patch.shape[0]))
13        if alpha is not None:
14            temp = alpha * patch + (1 - alpha) * temp
15        self.template = temp
```

2.2 Noisy Tracking on Romney's Hand

The same template is now tracked in the noisy version of the debate.

Table 7: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[5. 5. 0.]
Predict noise standard deviation	[20. 20. 0.]
Number of particles	200
Alpha	0.01

Figure 14: ps6-2-2-frames-15-50-140.png

We decide to increase the number of particles and reduce the value of α in order to be more robust to the noise.
Reducing alpha especially appears to be very important, since it gives a very large weight to the history in order to limit the impact of the noise on the template update.
Although on the first frames it appears that the mean of the tracker gets further away from the hand, some particles drag it back and we do not loose it in the long run.
We manage to get good results (Figure 14) but they are not necessarily consistent over several tests. However, we decide to keep it as such and limit the number of modified parameters in order to be able to compare the results with the normal version of the video.

3 Incorporating More Dynamics

3.1 Pedestrian Tracking

The next challenge is to deal with occlusion on a new video with pedestrians crossing the road, thus getting further away from the fixed camera.
We first need to adapt the template scale along the course of the video in order to account for the increasing distance to the target, which is done in the code below.
However, we no longer update the template according to the history, since the target orientation does not change in the first part of the video. This makes the content of the initial template (Figure 16) still relevant after many frames.

Table 8: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[0.1 0.1 0.1]
Predict noise standard deviation	[0.5 0.5 0.2]
Number of particles	200

We keep the default value for σ_{MSE} but force much lower values on the standard deviation for the x and y coordinates, since the position of the center does not move very much in the first hundred frames (only the scale). The standard deviation for the scale is then set to a reasonably high level, so that when scale tends to get lost during the occlusion it can be recovered efficiently afterwards.
The result is visible in Figure 15.

Figure 15: ps6-3-1-frames-40-100-240.png

Figure 16: ps6-3-1-template.png

3.2 Pedestrian Tracking Optimization

The last step, once more, is to find an optimal number of particles to track the pedestrian.

Table 9: Parameters

MSE standard deviation	10.0
Initial noise standard deviation	[0.1 0.1 0.1]
Predict noise standard deviation	[0.5 0.5 0.2]
Number of particles	100

Here again, we try to reduce the number of particles while keeping a relatively "stable" tracking. We settle on 100 particles, with a result visible in Figure 17. The scale parameter is still very unstable during the occlusion, but it gets back quickly on target while not "shaking" too much. Below this, we also increase the risk that the scale parameter is not properly recovered after the occlusions. This number is more than 3 times higher than what we found in question 1.4.

Figure 17: ps6-3-2-frames-40-100-240.png