CSx495 — Computer Vision
Problem Set 6: Particle Tracking

Here you are going to experiment with a particle filter tracker such as was described in class. Recall that we need a variety of elements: (1) A *model* - this is the "thing" that is actually being tracked. Maybe it's a patch, a contour, or some other description of an entity; (2) a representation of *state* x_t that describes the state of the model at time t; (3) a *dynamics model* $p(x_t|x_{t-1})$ that describes the distribution of the state at time t given the state at $t - 1$; (4) a *measurement* z_t that some how captures the data of the current image; and finally (5) a *sensor model* $p(z_t|x_t)$ that gives the likelihood of a measurement given the state. For Bayesian-based tracking, we assume that at time $t - 1$ we have a belief about the state represented as a density $p(x_{t-1})$, and that given some measurements z_t at time t we update our *Belief* by computing the posterior density $Bel(x_t) \propto p(z_t|x_t)p(x_t|u_t, x_{t-1})Bel(x_{t-1})$.

Kalman filter provided an analytic method for doing this under the assumption that density representing the belief at time t, the noise component of the dynamics and the sensor model were all simple Gaussian distributions. Particle filters provide a sample-based method of representing densities that removes that restriction and also is tolerant of occasional large deviations from the well-behaved model. In this assignment, you will be implementing a particle filter to track entities in video.

As a reminder as to what you hand in: A Zip file that has

1. Images (either as JPG or PNG or some other easy to recognize format) clearly labeled using the convention PS ⟨number⟩–⟨question number⟩–⟨question sub⟩–counter.jpg

2. Code you used for each question. It should be clear how to run your code to generate the results. Code should be in different folders for each main part with names like PS1-1-code. For some parts – especially if using Matlab – the entire code might be just a few lines.

3. Finally a PDF file that shows all the results you need for the problem set. This will include the images appropriately labeled so it is clear which section they are for and the small number of written responses necessary to answer some of the questions. Also, for each main section, if it is not obvious how to run your code please provide brief but clear instructions. **If there is no Zip file you will lose at least 50%!**

This project uses files stored in the directory `http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/ProblemSets/PS6` There are three video (.avi) files there: pres_debate.avi, which shows two candidates in a 2012 town hall debate, noisy_debate.avi, which is the same video overlaid with fluctuating Gaussian noise, and pedestrians.avi, which shows a group of people crossing a street in London. For each video file, there is a text file that contains the object bounding box at the first frame that you can use for initialization. The format of the text file is: the top-left coordinate (not center) and the size (width & height) of the bounding box. You can define your own initialization setting (for example using Matlab function imrect() and wait()).

1 Particle Filter Tracking

In class we discussed both the theory of and the implementation specifics behind particle filtering. The algorithm sketch is provided in the notes and it describes a single update of the particle filter. What is left up to the implementor are several details including what is the model, what is the state, the number of particles, the dynamics/control function $p(x_t|x_{t-1}, u_t)$, the measurement function $p(z_t|x_t)$, and the initial distribution of particles.

For this question, you will to need track an image patch template (a face) taken from the first frame of the video. For this assignment the model is simply going to be the image patch, and the state will be only the 2D center location of the patch. Thus each particle will be a (u,v) pixel location representing a proposed location for the center of the template window. We will be using a basic function to estimate the dissimilarity between the image patch and a window of equal size in the current image, the Mean Squared Error:

$$MSE(u_p, v_p) = \frac{1}{mn} \sum_{u=1}^{m} \sum_{v=1}^{n} (Template(u, v) - Image^t(u + u_p - m/2, v + v_p - n/2))^2$$

The funny indexing is just because (u, v) are indexed from 1 to M (or N) but the state $\langle u_p, v_p \rangle$ is the location of the center. MSE, however, only indicates how dissimilar the image patch is whereas we need a similarity measure so that more similar patches are more likely. Thus, we will use a squared exponential equation (Gaussian) to convert this into a usable measurement function:

$$p(z_t|x_t) \propto \exp(-\frac{MSE}{2\sigma_{MSE}^2})$$

To start out, you might use a value of $\sigma_{MSE} = 10$ but you are welcome to change this value.

For the dynamics we're going to use normally distributed noise since head movements can be unpredictable and often current velocity isn't indicative of future motion; so our dynamics model is merely

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \delta_t, \text{ where } \delta_t \sim N(0, \Sigma_d), \Sigma_p = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix}$$

which is just a fancy way to say you add a little bit of Gaussian noise to both u and v independently.

The number of particles and initial distribution are up to you to figure out. You may need to tune your parameters for these to get better tracking/performance.

In order to visualize the tracker's behavior you will need to overlay each successive frame with the following visualizations:

1. Every particle's (u,v) location in the distribution should be plotted by drawing a colored dot point on the image. Remember that this should be the center of the window, not the corner.

2. Draw the rectangle of the tracking window associated with the Bayesian estimate for the current location which is simply the *weighted* mean of the (u,v) of the particles.

3. Finally we need to get some sense of the standard deviation or spread of the distribution. First, find the distance of every particle to the weighted mean. Next, take the weighted sum of these distances and plot a circle centered at the weighted mean with this radius.

You will have to produce these visuals for select frames but you will not have to submit the entire video.

For reading the frames Matlab users should look at *VideoReader* and OpenCV users should look at the class *VideoCapture*. For sampling, Matlab folks might look at *randsample* whereas Python users, I might suggest looking at the function *numpy.random.multinomial*. For C++ users, I suggest getting a life. And then looking at the function *boost::random::discrete_distribution* or *std::discrete_distribution*.

1.1 Implement the particle filter and run it on the pres_debate.avi clip. You should begin by attempting to track Romney's face. Tweak the parameters including window size until you can get the tracker to follow his face faithfully (5-15 pixels) up until he turns his face significantly. Run the tracker and save the video frames 28, 84, and 144 with the visualizations overlaid.

Output: The code, the 3 image frames with overlaid visualizations, and the image patch used for tracking.

1.2 Experiment with different dimensions for the window image patch you are trying to track. Decrease the window size until the performance of the tracker degrades significantly. Try significantly larger windows than what worked in 1.1. Discuss the trade-offs of window size and what makes some image patches work better than others for tracking.

Output: Discussion in the pdf. Indicate 2-3 advantages of larger window size and 2-3 advantages of smaller window size.

1.3 Adjust the σ_{MSE} parameter to higher and lower values and run the tracker. Discuss how changing this parameter alters the results and attempt to explain why.

Output: Discussion in the pdf.

1.4 Try and optimize the number of particles needed to track the target. Discuss the trade-offs of using a larger number of particles to represent the distribution.

Output: Optimized particle number and discussion in the pdf.

1.5 Run your tracker on noisy_debate.avi and see what happens. Tune your parameters so that the cluster is able to latch back onto his face after the noise disappears. Include varying σ_{MSE}. Report how the particles respond to increasing and decreasing noise. Save the video frames 14, 32, and 46 with the visualizations overlaid.

Output: The code, the 3 image frames with overlaid visualizations, and discussion in the pdf.

2 Appearance Model Update

Let's say we'd now like to track Romney's left hand (the one not holding the mic). You might find that it's difficult to keep up using the naïve tracker you wrote in question 1. The issue is that while making rapid hand gestures, the appearance of the hand significantly changes as it rotates and changes perspective. However, if we make the assumption that the appearance changes smoothly over time, we can update our appearance model over time.

Modify your existing tracker to include a step which uses the history to update the tracking window model. We can accomplish this using what's called an Infinite Impulse Response (IIR) filter. The concept is simple: we first find the best tracking window for the current particle distribution as displayed in the visualizations. Then we just update the current window model to be a weighted sum of the last model and the current best estimate.

$$Template(t) = \alpha \, Best(t) + (1 - \alpha)Template(t - 1)$$

where $Best(t)$ is the patch of the best estimate or mean estimate. It's easy to see that by recursively updating this sum, the window implements an exponentially decaying weighted sum of (all) the past windows.

2.1 Implement the appearance model update. Run the tracker on pres_debate.avi and adjust parameters until you can track Romney's hand up to frame 140. Run the tracker and save the video frames 15, 50, and 140 with the visualizations overlaid.

Output: The code, the 3 image frames with overlaid visualizations, and the image patch used for tracking.

2.2 Try running the tracker on noisy_debate.avi. Adjust the parameters until you are able to track the hand all the way to frame 140. Indicate what parameters you had to change to get this to work on the noisy video and discuss why this would be the case.

Output: The code, the 3 image frames with overlaid visualizations, the discussion in the pdf, and the image patch used for tracking.

3 Incorporating More Dynamics

For this question we will work with a much more difficult video to perform tracking with, pedestrians.avi. For this problem, we'd like to be able to track the blond-haired woman as she crosses the road. If you try applying your adaptive tracker to this video, you will probably find that you will have difficulty dealing simultaneously with occlusion and the perspective shift as the woman walks away from the camera. Thus, we need some way of relying less on updating our appearance model from previous frames and more on a more sophisticated model of the dynamics of the figure we want to track.

For this problem, expand your appearance model to include window size as another parameter. This will change the representation of your particles (how?). You are highly recommended to use the *imresize* method in matlab for your implementation.

3.1 Run the tracker and save the video frames 40, 100, and 240 with the visualizations overlaid. You will receive partial credit if you can show the tracking size estimate (illustrate this with the rectangle outline) up to frame 100. You will receive full credit if you can reliably track all the way to the end of the street and deal gracefully with the occlusions (reasonable tracking at frame 240).

Output: The code, the 3 image frames with overlaid visualizations, and the image patch used for tracking.

3.2 Try and optimize the number of particles needed to track the model in this video. Compare that to the number you found in problem 1.4. Why is this number different?

Output: The number of particles you found to be optimal and discussion in the pdf.