CSC487/2503 – Computational Vision – Fall 2008

Assignment 4: Particle Filtering

Due: at the beginning of the lecture, 1:10 pm., Thurs., Dec. 4
This assignment is worth 15 marks towards your grade in this course.

This assignment involves the implementation and evaluation of a simple particle filter.

Download `pigeonTrack.zip` from the course web page. If you are running Matlab on CDF that is all you will need to download, since the image sequence is at `/u/jepson/pub/pigeon/`. Do not copy or download the image sequence to your CDF directory, since you do not have enough disk space. If you are running Matlab at home, you need to download the image sequence in `pigeon.zip`.

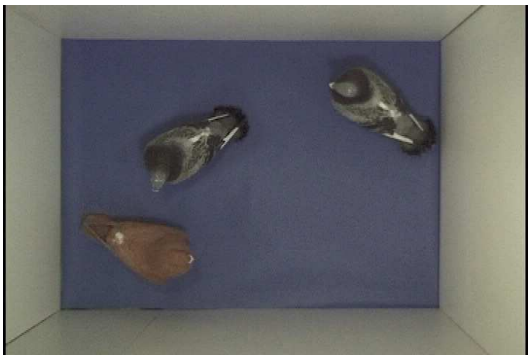

Figure 1: The first frame of the sequence showing the "grey" pigeon and its clone, along with the "red" pigeon.

What to hand in. Write a short report addressing each of the itemized questions below (hand-written reports are fine). You can assume that the marker knows the context of the questions, so do not spend time repeating material in the hand-out or in class notes. Include print outs of the output from your program in your report. Also, please email `mstefan@cs.toronto.edu` each of the Matlab files that you altered or created.

Diving in. There is nothing you need to hand in for this section. See the README file in the handout code for a detailed description of the process of building an image observation model and the associated likelihood.

To compute the image likelihood for a given image position $\vec{x}$ and orientation θ, we first place a coarse grid on the image centered on $\vec{x}$ and oriented according to θ (see Fig. 2). Within each cell of this grid we then compute a histogram of image gradients. The histogram bins range over both the amplitude and the orientation of the gradients. Moreover, the image gradient orientations are rotated into the coordinates of the grid pattern (i.e. according to θ). Due to this rotation we obtain some degree of pose invariance for the HOG descriptor. In particular, if we simply translate and rotate an image patch, and place the grid on top of this patch with the same translation and rotation, then the extracted gradient histograms should be the same (modulo interpolation and gradient estimation

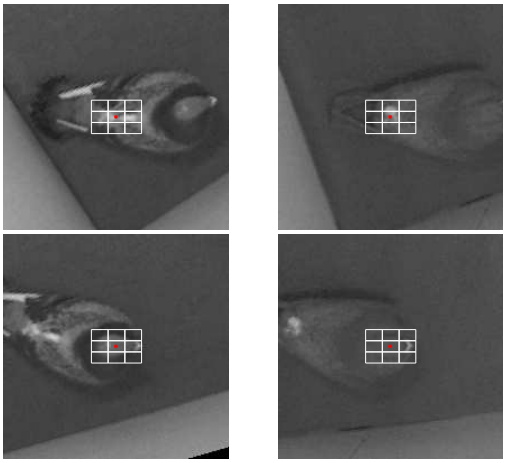

Figure 2: Target HOG grids for the grey pigeon and its clone (left) and the red pigeon (right).

errors).

The set of histograms within these grid cells form what is called a Histogram of Oriented Gradients model, or HOG model (see Dalal and Triggs, Histograms of oriented gradients for human detection, CVPR, 2005). The general motivation for this model is that the individual histograms for each cell allow for some spatial distortion of the pattern of gradients within that cell. This is coupled with information from the variation of the histograms over the whole HOG grid, which maintains some information about the general spatial layout of the target.

The HOG implementation is provided in the handout code. The actual implementation differs from the above description in that, instead of rotating the grid pattern by θ, we rotate the images by $-\theta$ (actually, by a coarse discrete approximation of $-\theta$) and then compute the histograms with the HOG grid aligned with the x and y axes in the rotated image. This allows us to use integral images to quickly compute the histograms for all the cells. Note we actually vary θ over the entire range $[0, 2\pi)$ even though the smaller range $\theta \in [0, \pi/2)$ would still achieve an axis aligned grid. I chose not to use the smaller range primarily to save my programming time. (But this simplification increases the runtime, especially in question 3 below.)

The script file `grabHOG.m` extracts the HOG models for the canonical position and orientation for each of four different situations (see Fig. 2). We refer to these models as the "targets" we wish to track. The goal of this assignment is, when given appropriate initial guesses, to track the position and orientation of these targets through a short image sequence (76 frames).

During tracking suppose $I(\vec{x}, t)$ is the image at time t and we have a hypothesized target pose $\vec{a}_t = (\vec{x}_t^{\,T}, \theta_t)^{\,T}$. We wish to determine if the image patch in the neighbourhood of $\vec{x}_t$ with the orientation of θ_t is similar to the particular target selected. To do this we first extract the HOG model on the image $I(\vec{x}, t)$, with the grid of cells translated and rotated according to the pose $\vec{a}_t$. We then compare this newly extracted HOG model with the original HOG target. In particular, for each cell of the grid pattern, the histogram for the current image $I(\vec{x}, t)$ is compared with the corresponding histogram from the target. A histogram similarity score is computed for each corresponding pair, and the average similarity score $z_t(\vec{a}_t)$ is computed. This z_t is simply a number between zero and one (with the value one denoting a perfect match, zero the worst possible match).

Specific questions to be answered are listed below:

1. **[5 pts] Log odds.** First we consider how to map a score $z_t(\vec{a_t})$ to the log likelihood ratio

$$L(z) \;=\; \log\left[\frac{p(z|\text{ target present}, \vec{a})}{p(z|\text{ target absent}, \vec{a})}\right]. \tag{1}$$

Here the distribution $p(z|\text{ target present}, \vec{a})$ represents the probability that a score z is observed when the target is actually present and the hypothesized pose $\vec{a}$ is correct. Similarly, $p(z|\text{ target absent}, \vec{a})$ represents the probability of z given the target is not actually present at (or near) the pose $\vec{a}$. We refer to these two conditional distributions as $p_{On}(z)$ and $p_{Off}(z)$, respectively.

The reason we want to compute the log likelihood ratio $L(z)$ is that really we wish to compare the probability that the target is present (with the specified pose) to the probability that it is absent. This is discussed on p.32 of the tracking lecture notes.

In order to get a model for $L(z)$ we first need to gather some data. Read steps 1 and 2 in the README file in the handout code. After extracting a HOG model using `grabHOG.m`, run `trainHOG.m` to extract histograms of the similarity scores z_t. The script `trainHOG.m` asks you to mouse in two different types of image regions (including their orientation), one which only consists of negative examples, the other which only consists of positive examples. You will need to collect several such sets of examples, for both $p_{On}(z)$ and $p_{Off}(z)$. The script simply loops until you hit `Enter` without mousing in any points. As a rule of thumb, you should keep mousing in examples until the displayed histograms are reasonably stable from one step to the next. You typically get many more negative examples per mouse click than positive ones, so you will probably have to use more loop iterations to properly sample the positive example set. The performance of your tracker will depend on how representative your set of positive and negative examples are, so take your time.

The results of this training are written by `trainHOG.m` into the mat files `respHist*.mat` in the `./head` and ./back subdirectories.

Next, by running `fitLogOdds.m`, you can eyeball a fit of $\log(p_{On}(z)/p_{Off}(z))$ with a logistic model

$$L(z;\mu,s,\lambda) = \left[\frac{1-e^{-\lambda(z-\mu)}}{1+e^{-\lambda(z-\mu)}}\right] s. \tag{2}$$

Here s, μ, and λ are three positive scalar parameters. The corresponding Matlab variables are `mu`, `scl` and `lambda`, and these appear near the top of the script `fitLogOdds.m`. You need to roughly set the values of these three parameters to model the observed log likelihood ratio. Different parameters should be used for each of the targets. These parameters are saved by `fitLogOdds.m` in the `hog*.mat` files so that they can be used later.

You will notice that a single logistic model does not provide a fit to the observed behaviour of $\log(p_{On}(z)/p_{Off}(z))$ over the whole range $z \in [0,1]$. Instead, you need to decide which portion of the plot you wish to model with the logistic function $L(z)$.

In your write up, provide a short discussion of which portion of the $\log(p_{On}(z)/p_{Off}(z))$ curve you decided to fit with the logistic $L(z)$ and explain why. In your report include the plots of these logistic fits produced by `fitLogOdds.m`. Finally, include a brief explanation of how you can decide from just the plots of $p_{On}(z)$ and $p_{Off}(z)$ how easy a target will be to track.

At this stage you can run `tryHOG.m` to try out your detector on images selected randomly from the sequence. This will use the parameters you set for the logistic model (obtained from the modified

`hog*.mat` file). The script `tryHOG.m` asks you to mouse in an image patch, and an orientation, and then the detector is run over this whole patch at the specified orientation.

If the training was done correctly, you should observe a peak in the logistic response $L(z(\vec{a}(\vec{x},\theta)))$ for positions $\vec{x}$ and orientations θ in the neighbourhood of the target's pose. And this peak should appear much stronger in the likelihood image $e^{L(z)}$. The true target may not always be the global maximum, but it should often be. The presence of other local maxima in the responses is normal.

2. **[25 pts] Particle filter.** As discussed on pp.22-25 of the tracking lecture notes, we need two components to build a particle filter for tracking. First we need a measurement model. Here we will use the observation probability provided by the logistic model developed in question 1 above. Secondly, we need a model of the dynamics. In particular, we need to specify the probability distribution for the pose of the target $\vec{a}_t$ given the pose at the previous time $\vec{a}_{t-1}$.

   (a) **[5 pts] Model of dynamics**. For the dynamics model we will use a zero velocity random walk

   $$\vec{a}_t = \vec{a}_{t-1} + \vec{n}_t, \tag{3}$$

   where the process noise $\vec{n}_t$ is a mean zero, Normally distributed random variable, with a diagonal covariance C_D. The variances in the x and y image directions can be taken to be equal, but the variance in θ should be different, so $C_D = \text{diag}(\sigma_x^2, \sigma_x^2, \sigma_\theta^2)$ for some constants σ_x and σ_θ.

   Implement this dynamics model by completing the M-file `propagateSamples.m`. The job of `propagateSamples.m` is to take an unweighted set of samples representing the filtering distribution $p(\vec{x}_{t-1}|\vec{z}_{1:t-1})$, propagate these samples to the next frame using $p(\vec{x}_t|\vec{x}_{t-1})$ and return a new set of samples representing the proposal distribution $p(\vec{x}_t|\vec{z}_{1:t-1})$. (Hint: My completed Matlab method is less than 10 lines long.)

   (b) **[10 pts] Reweight particle set using likelihoods**. Your next task is to complete the M-file `updateWeights.m`. This function is given as input parameters: a) the sample set from the previous step; b) the current image; and c) the parameters of the likelihood model. The log likelihood $\log p(z_t|a_t)$ at all of the given samples for $\vec{a}_t$ is computed in the handout code with the call to `pigLogLike`.

   Given these log likelihoods your job is to compute weights for each of the samples. This is described in Step 3 on page 23 of the lecture notes. (This again only takes a few lines of Matlab code.)

   In addition to computing the weights, your `updateWeights.m` function should also compute the effective number of particles, as described on p.30. Simply print the effective number of particles on the console.

   (c) **[5 pts] Go!** Try the resulting particle filter on the four targets in Fig. 2. You need to adjust the covariances in your dynamics prior.

   In your write up describe how you set the covariances for the dynamics. Also include a few frames showing examples of the tracking at various points during the sequence. These figures can be obtained by running `showResults.m` after you have run the particle filter. You can set the loop over frames in `showResults.m` to be just one frame, so you can capture the desired figures.

   (d) **[5 pts] Evaluation of results**. In your report provide a detailed evaluation of the performance of the tracking algorithm. Explain, in general, what is required of the image sequence and the target for there to be a reasonable expectation that this algorithm will work. What

are the algorithm's strengths? What are the algorithm's weaknesses? What are the algorithm's failure modes (i.e., what are the categorically distinct ways in which the algorithm will fail)? Try to be clear and concise in your answers.

3. **[10 pts] Tracking two modes.** Since the particle filter maintains a representation of the filtering distribution, it seems plausible that the particle filter might be useful for simultaneosly tracking a couple of modes. In order to investigate this we can make use of the pigeon and its clone. These have identical appearances (even their "mother" cannot tell them apart).

   Modify `partFilt.m` so that the initial guess encompasses two sets of poses, one for the target on one of the clones, and one for the same target on the other clone. This initial sample set should be split roughly 50/50 between these two pigeons.

   The other changes you need to make to `partFilt.m` are as follows. First the radius of the cropped images needs to be increased. This radius is set by the variable `sqrCropRad` in `partFilt.m`. Reset `sqrCropRad` to be more than the maximum distance between the two pigeons (i.e., 400). Next you need to turn off the "safety net" by setting the variable `safetyNet` to false at the top of `partFilt.m`. (The safety net uses some coarse tracking results to center the feasible region for particles roughly in the neighbourhood of the target.) Finally, the last change required is to reset the flag `cropRotatedIm` at the top of `partFilt.m` to be false. This allows the image warping routine to rotate the image region without cropping any of the corners. The results displayed in Figure 201 by `partFilt.m` will now appear to change size. However that is just due to the change in the bounding box around the rotated image. The image features themselves are not being resized.

   Given the above changes, try simultaneously tracking the targets on the backs of the two grey pigeons. Then try tracking the heads of the two identical grey pigeons.

   In your write up clearly describe your results. Are their additional failure modes that were not present for the single target tracking case? If so, explain why these additional types of failures occur.