Assignment: Backgrounding and Optical Flow.

April 6, 2010

1 Backgrounding

In this part of the assignment, you will develop a simple background subtraction program.

1. In this assignment, you are given two "videos". Each video is stored as a 4-dimensional matlab array. The first three dimensions represent rows, columns, and the red-green-blue layers of an image. The fourth dimension represents time. The videos are named train_data.mat and test_data.mat, and you can download them from the course web page. You may want to make reduced resolution versions of these videos by clipping them, sub-sampling them, or reducing the number of frames so that you can get your code working more easily. You can "play" the videos using the following sequence of matlab commands. Load these 4-dimensional arrays into matlab by just typing 'load train_data' and 'load test_data'. You should enter all the commands as a single line:
   `for i=1:100 imagesc(train_data(:,:,:,i)); drawnow; end`

2. Next, your job is to make a background model. The background model will have two parts. The first part is the average value of a pixel in a particular location over time. This should be the mean image of the training movie. The size of this mean image should be (241, 361, 3). I'll let you figure out how to create it, but I will tell you that you can do it in a single command!

3. The second part of the background model is the average distance that each pixel is from the mean value at that location. This is essentially a measure of the pixel variance. To compute this, for each pixel in a given position in each frame, compute the squared Euclidean distance to the mean pixel at that location (from the previous step). Then take the average value of this distance at each location in the image. Take the square root of each value in the average distance image. You can refer to this as you *standard deviation image*. Plot the standard deviation image.

4. Now, using your model derived from the training data, try to find moving objects in the test sequence. In particular, for some value alpha, find all of the pixels in each frame of the test sequence that are within alpha standard deviations of the mean value. For a single image, you can visualize your results by making a

black and white image in which the pixels which are “moving” are white, and the pixels which are not moving are black. You can visual your results for the whole test sequence by adding together all of the individual binary images for each frame.

5. Experiment with different values of alpha, and produce a summary image for each value of alpha. Turn in these summary images, along with your mean image and standard deviation image.

2 Optical Flow

In this part of the assignment, you will implement a version of the Lucas-Kanade optical flow algorithm.

In this assignment, you are given the first two frames of the "Flower Garden" video. You can see the whole video as an mpeg movie on the course web site under the "links" section. The frames provided on the course web site have been stored in as gray-scale images in a 3-dimensional array, where, as usual, the first two dimensions are rows and columns, and the third dimension indexes time (in this case just t=1 t=2). Load this .mat file into matlab.

Next, you will be implementing the Lukas-Kanade optical flow algorithm. The goal will be to estimate the optical flow at a certain set of points. Remember that for Lukas-Kanade, for each flow vector that you estimate, you will be choosing a region over which to analyze the two images. If we had more time, I would have you experiment with different sized regions, but since we are short on time, I'll tell you how large to make these regions. For this assignment, I want you to use regions that are 15 pixels by 15 pixels, and *non-overlapping*. That is, since the input images are 360 by 240, you should have an array of 24 by 16 optical flow vectors at the end of your procedure.

The Lukas-Kanade algorithm works by trying to find an optical flow vector with the components u and v for each region that minimizes the error of the following series of equations:

$$E_x^1 u + E_y^1 v = -E_t^1 \tag{1}$$

$$E_x^2 u + E_y^2 v = -E_t^2 \tag{2}$$

$$E_x^3 u + E_y^3 v = -E_t^3 \tag{3}$$

$$\ldots \tag{4}$$

$$E_x^{225} u + E_y^{225} v = -E_t^{225}. \tag{5}$$

Recall that E_x^1 is the estimate of the image derivative in the horizontal direction *at the first pixel in the image patch* (hence the superscript "1"). For the purposes of this assignment you can just take that to be the difference between that first pixel and the one immediately to its right. (NOTE: You will have to do something special for the image regions that are on the rightmost edge of the image, since those regions will not have a pixel immediately to the right of the rightmost pixel. Do whatever you like to deal with this. It won't have a major impact on the results.) The last index is 225 since there 225 pixels in the image region being analyzed.

This equation can be rewritten in matrix form as

$$A \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{b}, \tag{6}$$

where

$$A = \begin{bmatrix} E_x^1 & E_y^1 \\ E_x^2 & E_y^2 \\ E_x^3 & E_y^3 \\ \ldots\ldots & \\ E_x^{225} & E_y^{225} \end{bmatrix}$$

and

$$\mathbf{b} = \begin{bmatrix} -E_t^1 \\ -E_t^2 \\ -E_t^3 \\ \cdots \\ -E_t^{225} \end{bmatrix}.$$

Since you are trying to solve for two unknowns with 225 equations, there are in general no values of u and v that will satisfy these equations. To find the "best fit", i.e. the values of u and v that will generate a $\hat{\mathbf{b}} = A \begin{bmatrix} u \\ v \end{bmatrix}$, such that $\hat{\mathbf{b}} - \mathbf{b}$ has smallest magnitude, the idea is to use the so-called "pseudo-inverse" of A.

The derivation of the pseudo-inverse goes like this. Let $\mathbf{f}$ represent the flow vector with components u and v. Then we have:

$$A\mathbf{f} \approx \mathbf{b} \tag{7}$$

$$A^T A\mathbf{f} \approx A^T \mathbf{b} \tag{8}$$

$$(A^T A)^{-1} A^T A\mathbf{f} \approx (A^T A)^{-1} A^T \mathbf{b} \tag{9}$$

$$\mathbf{f} \approx (A^T A)^{-1} A^T \mathbf{b}. \tag{10}$$

The expression

$$(A^T A)^{-1} A^T$$

is called the *pseudo-inverse* of A, and you will be happy to know there is a function in matlab to compute the pseudo-inverse. The function is `pinv()`.

Your job is to for each region in the image:

1. Compute the matrix A and the vector $\mathbf{f}$.
2. Compute the pseudo-inverse of A.
3. Provide an estimate of u and v for each image region by calculating the vector $\mathbf{f}$.

Final part of assignment. When you have estimate for u and v of each patch, now it is time to display them. There is a command in matlab called `quiver()` which plots a set of two-dimensional vectors as arrows on the screen. Try to figure out how to use this to plot your optical flow results.

IMPORTANT: quiver is difficult to use, because it works with x,y coordinates rather than row,column coordinates. Also, a small value of y is at the bottom of the screen, whereas for a row coordinate, a small value is plotted at the top of the screen. This makes it quite confusing to use. I suggest using the version of quiver with 4 arguments, and providing explicit x and y coordinates for each flow vector that you are plotting. You can create the x and y coordinate matrices using a double loop.

Turn in your final plot of your optical flow vectors. Here is what mine looked like:

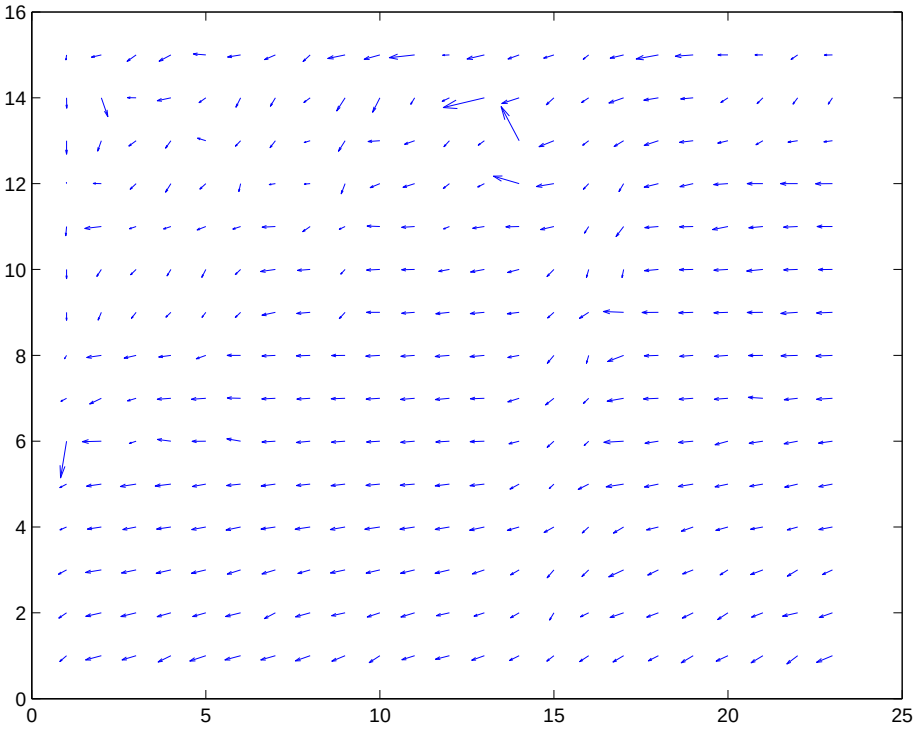

Figure 1: **Lucas-Kanade flow between first two images of garden sequence.**