Name: ________________________________

Score: /17

CSE 5524 Computer Vision for HCI

Homework Assignment #6

Covariance Tracking:

1) Use the <u>covariance matching technique</u> to find the correct match in the color image given on the WWW site (target.jpg). The model covariance matrix (of <x,y,R,G,B> features) is given below (notice x,y vs. row,col!).

```
modelCovMatrix = [47.917          0      -146.636     -141.572     -123.269;
                       0     408.250     68.487       69.828       53.479;
                  -146.636    68.487      2654.285     2621.672     2440.381;
                  -141.572    69.828      2621.672     2597.818     2435.368;
                  -123.269    53.479      2440.381     2435.368     2404.923];
```

Test <u>all possible</u> 1-pixel overlapping windows (each of size 70 rows by 24 columns, with the upperleft-corner as the window origin) in the image with the given model. Save the match distance for each box location in the image at each pixel location (for the origin of the window). Plot/display the match-distance-image. Provide the **location of the best match** distance for the best candidate. Note that the above given covariance matrix is <u>biased</u> (normalized with 1/(M*N)), and Matlab's cov function is <u>unbiased</u> by default using 1/(M*N-1), so call `cov( X, 1 )` to make it consistent (biased). Leave the image with colors ranging 0-255 (do not scale/normalize the colors). **NOTE: make sure not to take a *log()* of zero at any time!** [5 pts]

Mean-Shift:

2) Create a function to extract a feature vector for each pixel in a circular neighborhood (< radius) around (*x*,*y*):

```
[ X ]=circularNeighbors(img, x, y, radius);
```

For each pixel, use the same format to return as used above (<x_i,y_i,R,G,B>). That is, *X* should be a *K*x5 matrix, where each row is for one of the pixels in the neighborhood. Assume that the (x,y) passed into the function are <u>real</u> (non-integer) values, and do <u>NOT</u> round them in the function for computation of the neighborhood. [2 pts]

[Next Page]

3) Create a function to build a color histogram from a neighborhood of points:

```
[ hist ]=colorHistogram(X, bins, x, y, h);
```

The histogram (*hist*) should be a *bins* x *bins* x *bins* color cube (RxGxB). The bins should be evenly spaced. For example, if *bins*=4 then the pixel-value limits for each bin will be {0-63, 64-127, 128-191, 192-255}. Be sure to test your code on pixels with possible RGB values of 0 and 255. Weight the construction of the histogram using an Epanechnikov kernel centered at real-valued (*x*, *y*) and with bandwidth *h*. Normalize the histogram/cube so it sums to 1. (This function will be used to make your model histogram "q_model" and to make the candidate test histogram "p_test") [3 pts]

4) Create a function to calculate a vector of the mean-shift weight**s** (w), where there is a weight w_i for each pixel *i* in the neighborhood: [2 pts]

```
[ w ]=meanshiftWeights(X, q_model, p_test, bins);
```

5) Load the images img1.jpg and img2.jpg (from the website), and use the functions above to perform mean-shift tracking.

Build a model from img1 using a circular neighborhood with a radius of 25 pixels centered at the 150th column and 175th row (in Matlab, this would mean (x_0,y_0) = (150.0, 175.0); whereas in python this would mean (x_0,y_0) = (149.0, 174.0)) and a color histogram of size 16x16x16 (cube). Build the weighted cube histogram using an Epanechnikov kernel with bandwidth *h* = 25 (same as the earlier radius).

Run **25 iterations** of mean-shift tracking on img2. DO NOT ROUND coordinates or values at any time!

Report the final (*x*, *y*) location (DO NOT ROUND) and Euclidean distance between the last two iterations (see Step 4 on the Algorithm slide). [3 pts]

6) As usual, turn in and upload your material.