Homework #3 (50p)

Your third homework involves writing a histogram-based tracking in gray-level and color images. There are four stages to this homework:

1. (5p) Using the Matlab programs provided with this homework to compute edges in gray-level images.
   You should use the programs provided with this description to gain experience and some intuition with edge detection. You should apply these programs to several different images. Those programs are BiSmooth.m (binomial smoothing), Gradient.m (gradient computation), NMS.m (non-maxima suppression), and Hysteresis.m (hysteresis thresholding as in Canny edge detector). These programs work for gray-level images only. You should read the programs to become familiar with some simple Matlab functions used in them. In addition you should use Matlab function quiver to display image gradients. Here is an example:

   ```matlab
   > A = imread('fr05.tif');
   > A1 = BiSmooth(A,8);
   > [Gx,Gy] = Gradient(A1);
   > m1=NMS2(Gx,Gy,3);
   > m1 = Hysteresis(m1,3);
   > [I,J] = find(m1>0); % row and column indexes
   > In = find(m1>0); % indexes in a vector form
   > [M,N] = size(m1);
   > quiver(J,M-I+1,Gx(In),Gy(In),1)
   ```

   Create a web page and post the original and gradient images (vector fields)—email the URL to me at zduriccs.gmu.edu. You can save your figures using print command in tiff, jpeg, or eps formats. You should post the results for at least three different images.

2. (5p) Writing a Matlab program for computing edges in color images.
   Implement the structure matrix method discussed in class. You can compute eigenvalues and eigenvectors using eig command. You can combine the eigenvector and the edge strength to obtain color gradients. You can then use NMS.m and Hysteresis.m to find edges and quiver to display them. Apply your method to several images given here. Post your code and the results on your web page. You can start by using function ColorEdges.m. Note: You will need to compute edge orientation correctly.

3. (20p) Writing a Matlab program for computing color and edge histograms in images.
   Color histogram should be 4096-bin: \([r/16] * 256 + (g/16) * 16 + b/16\) will convert a color value into an index. Edge histograms should be 36-bin: you can turn a gradient angle into an index by dividing the orientation angle by 36 and rounding to the nearest integer. Test your color and edge histograms on images. Post the results on your web page.
Write two functions for histograms comparison: histogram intersection and chi-squared measure. Test your functions on images.

a) Histogram intersection. Given two color histograms $H_1(\cdot)$ and $H_2(\cdot)$ their intersection is given by

$$H_1 \cap H_2 = \sum_i \frac{\min\{H_1(i), H_2(i)\}}{\max\{H_1(i), H_2(i)\}}.$$ 

Large values correspond to high similarity.

b) Chi-squared measure: $\chi^2$. Given two histograms $H_1$ and $H_2$ the $\chi^2$ measure of their similarity is given by

$$\chi^2(H_1, H_2) = \sum_{i: H_1(i) + H_2(i) > 0} \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}.$$ 

Small values correspond to high similarity.

4. (20p) **Using color and edge histograms for tracking.**

Write a program for tracking objects in image sequence provided for the second homework. An additional color image sequence will be added so that you can test your method on color images. Start by choosing a region in the first image and find its best match in the second image and so on. You can choose regions larger than $10 \times 10$ to obtain more meaningful histograms.

**Submitting your homework**

You will hand in a report that will include the following:

i. description of all algorithms used in your program, ii. your Matlab code, iii. examples of your tracking results—i.e., example images with overlayed boxes around tracked regions, or $(x, y)$ positions of the centers of the tracked region(s).

Alternatively, you can post your results on a web page. In that case you can post a video sequence showing your tracking results.