Problem Set 1
CMSC 426
Due Sept. 13

Programming Assignment

The goal of this assignment is to give you some hands-on experience with histograms, and to get us started programming with images in Matlab. For the first three problems we will show you the results that your programs should produce on a test image, and then ask you to run your program on a different image.

Your solutions should be in a zip file and include a document containing the results of your program and all code you have written. Email this in a single zip file to Matthew at: 201208426submit@gmail.com. You should also hand in a hardcopy of the your writeup to Matthew or me.

1. **10 Points:** Implement a function, `myHist(I)` to determine the histogram of an image. myHist should take one argument, an image I, create a figure, and display the histogram of I, divided into bins that each contain a range of five intensity values. That is, the first bin should count the number of pixels with intensities 0-4. Include the figure showing this histogram for the image 'desert1bw.jpg' in your writeup. On the web page, we include a Walker Evans photograph, along with the histogram that we have computed for it.

   To do this, you can make use of the matlab commands *hist, figure, imshow, colormap(gray),* You might also find it helpful to look at the results of executing the commands: *a = randi(4,3) a(:)* (this is a hint). Note that once a figure has been created, the figure has a button that allows you to save it to a file in an image format.

2. **10 Points:** Implement a function, `myHistEq(I)`. This function takes one argument, an image *I*, and creates a new image which is *I* with an equalized histogram. This new image, along with its histogram, should be displayed in figures. You may find it helpful to use the Matlab function *histeq*. Include in your writeup the figures that are created when you apply histogram equalization to desert1bw from problem 1. The web page contains sample results when this routine is applied to the Walker Evans image from problem 1.

3. **10 Points:** Implement a function, `myHistSpec(I,J)`. This takes two images as input, *I* and *J*. *I* should be transformed by your function so that it has a histogram that is approximately the same as the histogram of J. It should display the transformed version of *I* in a figure, and also display its histogram. Apply this

function so that the image Johny Depp 2 is transformed to have the same histogram as the image Johny Depp 1 (in which he is Capt. Jack Sparrow).

4. **20 Points: Histogram comparison of color images**. In this problem, you will write code to generate a 3-D histogram of a color image, and then to compare two histograms to judge their similarity.

    - To create the histogram, you will divide the R, G and B channels into 8 intervals of equal size. This will produce a total of 8^3=256 buckets.

    - Do this by creating the function: `h = myColorHist(I)`, which takes an rgb image as input, and returns a histogram. `h` can be a vector of 256 values (or it could be a 3-D, 8x8x8 array if you prefer).

    - Next, create a function `d = histDist(h1, h2).` This function takes two histograms (as created by myColorHist) as input and returns a distance, which indicates the SSD between the two histograms. Keep in mind that before computing the sum of square distances between two histograms, they must be normalized; otherwise large images will always be considered to have histograms that are very different from small images.

    - Finally, use these two functions to compute a little 6x6 table, showing the distance between all pairs of the images: desert1, desert2, desert3, forest1, forest2, forest3. These are six, more or less random images I downloaded after googling "forest" and "desert". Turn in this 6x6 table, your code, and a one paragraph write-up explaining what you think these results indicate about the potential for using color histograms to classify images by the type of scene they depict.

5. **Challenge Problem (optional), 10 Points:** The purpose of this problem is to experiment more deeply with image comparison from problem 4. You can decide how to do this. For example, you can implement other approaches to image comparison, such as the Chi-squared or cosine distance. Try using different numbers of buckets. You may also consider applying these methods using different sets of images that you have downloaded. Write up your results, illustrating them with tables and figures. Do you find that making different choices, or using different data sets, can improve the usefulness of histogram-based image comparison for classifying images? For full credit, you must run some interesting experiments and draw solid conclusions. Some credit will be given for any reasonable effort.