Problem Set 1
CMSC 426
Due Feb. 5

Programming Assignment

The goal of this assignment is to get you started programming with Matlab, especially using images in Matlab. You'll also make some histograms; we'll discuss these a bit more on Monday. I'm including a Matlab function called ps1. You can edit this file to add your own functions. ps1 also contains some comments that might be helpful.

The problem set will make use of six images, located at
http://www.cs.umd.edu/~djacobs/CMSC426/desert1.jpg,
http://www.cs.umd.edu/~djacobs/CMSC426/desert2.jpg,
http://www.cs.umd.edu/~djacobs/CMSC426/desert3.jpg,
http://www.cs.umd.edu/~djacobs/CMSC426/forest1.jpg,
http://www.cs.umd.edu/~djacobs/CMSC426/forest2.jpg,
http://www.cs.umd.edu/~djacobs/CMSC426/forest3.jpg.

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 Hao at: assignmentcmsc426@gmail.com. You should also hand in a hardcopy of the your writeup to Hao or me.

1. **10 Points:** This is just a basic problem, to get you started. Write a function "fib" that computes the Fibonacci number. Remember that's the series 1, 1, 2, 3, 5, 8, 13, 21, 34, …. Use your function to compute fib(32), and turn in the result. Using the template, this is run by: ps1(1,32).

2. **10 Points:** Implement a function, "matrix_replace". Executing matrix_replace(A, i, j) takes a matrix, A, finds all elements that are equal to i, and replaces them with j. **You are not allowed to use any loops in implementing this**. Hint: take a look at the Matlab function "find".

3. **10 Points:** Implement a function, "myhist", that will compute the histogram of an image. That is, myhist will take an image as input, and return a vector, h, as output. h(i) just tells us how many pixels in the image have an intensity of I (for information on histograms, check the notes for Feb. 3). Write your function so that it can take a grayscale or RGB image as input. If the input is an RGB image, convert it to grayscale using rgb2gray. The template contains code to display the histogram you compute in a figure. It also has timing code. Since everyone uses different machines, absolute timing isn't that meaningful, so I'm comparing the time required to compute the histogram to the time required to sum up all the pixel values in the image. Try to make this ratio as small as possible. In my

code, the histogram computation takes about 15 times as long as the summation. Extra credit for anyone who can get this ratio below 5 (note, timings can vary, so to be sure, you have to repeat this many times and take the average).

Test your code on the desert 1 image. Turn in a copy of the histogram figure, along with your code (note that figures come with menus that allow you to save them).

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=512 buckets. For example, one bin of the histogram will contain all pixels with an R value between 0 and 31, a G value between 32 and 63, and a B value between 0 and 31.

- Do this by creating the function: `h = myColorHist(I)`, which takes an rgb image as input, and returns a histogram. h should be a vector of 512 values. Do not loop through all the pixels, or this routine could be quite slow.

- 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 sum of square distances (SSD) between the two histograms. To compute the SSD between two vectors, subtract one vector from the other, square the differences, and add these up. Before computing the sum of square distances between two histograms, they must be normalized. That is, divide all entries in the histogram but the sum of all the entries, so that the values in the normalized histogram add up to one. 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 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.