CMSC 426 Homework 4 Due March 30 Spring 05

This homework counts for 1.25 times the other three so far.

Some of you have claimed difficulty in printing out files due to cost and other issues. IF this applies to you, please send me email.

1) Color-spaces: Read the chapter on color in the text. (40 points total)
   i) Read the i image called `flowers.tif` using `imread('flowers.tif')`.
   Display the red, green and blue components of this image as 3 gray level images on the same figure (using `subplot`) and print the result. (10 points)
   ii) . Transform the flowers image to the *HSV* color space using the Matlab function `hsv2rgb`. Display and print a single figure showing the *H*, *S*, and *V* components of this image as 3 gray level images. Use `subplot` to print all three on the same figure. (10 points)
   iii) Transform the flowers image to *YIQ* color space using the transformation

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

   Again using `subplot`, display and print a single figure showing the Y, I, Q components of this image as 3 gray level images. (10 points)
   iv) Transform the *RGB* flowers image into a gray level image using the function `rgb2gray`. Compare with the images constructed above. What method was used by `rgb2gray`? (10 points)

2) Programming RANSAC. (85 points)
   a. Write a function of the form:
   `[a, b, c] = gen_line(pt1, pt2)`
   This function should take two points as input, and return the parameters of a line that fit these points. The line should be
   ax + by + c = 0, where sqrt(*a*a + b*b) = 1*
   Test this function using the pairs of points:
   i) (1,1) (2,1)
   ii) (1,1) (1,2);
   iii) (1,1) (3,2).
   Turn in your code, and show calls to the routine and responses for these inputs. [10 points]
   b. Write a function of the form:
   `d = line_dist(L, p)`
   L represents a line (perhaps as a vector of [*a*,*b*,*c*]). *p* represents a point. *d* is the distance from the point to the line. Test your code with
   i) The line formed by the points: (0,1) (2,1) and the point (5,2).
   ii) The line formed by the points (1,1) and (3,2) and the point (7,4).

Turn in your code and show calls to the routine and responses for these inputs. [10 points]

c. Compute the correct answers to the problems in (b) by hand. Show your work and the results. [10 points]

d. Write a function of the form:

```
k = num_samples(n, m, p)
```

If there are *n* points in an image, and *m* of them come from a single line, then if we randomly sample *k* pairs of points we should have a probability greater than *p* of picking a pair of points that come from the line. *k* should be the smallest integer for which this is true. Test this with *n* = 20, *m* = 8, *p =0 .9*. Turn in your code, and show calls to the routine and responses for these inputs. [10 points]

e. Write a function of the form:

```
pts_cor = verify(p1, p2, pts, eps)
```

p1 and *p2* are points that will be used to form a line. pts is a list of different pts. pts_cor should contain a list of all points in pts whose distance to the line formed by p1 and p2 is less than eps. Test this routine with p1 = (1,1), p2 = (2,2), pts = [(3,4), (5,7), (3,8)], eps = 2. Turn in your code, and show calls to the routine and responses for these inputs. [15 points]

f. Write a function to generate test data, of the form: pts = test(n). This should generate points in the square [0,50], [0, 50]. The first 8 points should come from the horizontal line y = 25, with unit variance Gaussian noise added to them (use randn). n noise points should come from a uniform random distribution across the whole image (use the function rand). Test this by running it with n = 30 and plotting the results. Plot points from the line in one color, and noise points in a different color. Turn in the plot and code. [15 points]

g. Write the function [L, pts, c] = ransac(n). Generate n + 8 points using test, from 1f. Sample enough random pairs of points so there is probability greater than .95 that two will come from the line, using num_samples from 1d. For each pair of points, use verify from 1e) to find all points that are near the line, using eps = 2. Select the line that is close to the largest number of points. Return this line (L), the list of points (pts), and the number of these that actually came from the line (c, ie, the number that were among the first 8 returned by test). Run your code for n = [0, 10, 20, 50, 100, 200]. Plot c versus n for these results. Turn in your code and the plot. [15 points]