ECS 189G: Intro to Computer Vision, Spring 2015
Problem Set 0

Instructor: Yong Jae Lee (yjlee@cs.ucdavis.edu)
TA: Ahsan Abdullah (aabdullah@ucdavis.edu)
TA: Vivek Dubey (vvkdubey@ucdavis.edu)

Due: Friday, April 10th, 11:59 PM

Instructions

1. Answer sheets must be submitted on SmartSite. Hard copies will not be accepted.
2. Please submit your answer sheet containing the written answers in a file named: FirstName_LastName_PS0.pdf.
3. Please submit your code and input/output images in a zip file named: FirstName_LastName_PS0.zip. Please do not create subdirectories within the main directory.
4. You may collaborate with other students. However, you need to write and implement your own solutions. Please list the names of students you discussed the assignment with.
5. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
6. If plots are required, you must include them in your answer sheet (pdf) and your code must display them when run. Points will be deducted for not following this protocol.

The goal of this warm-up problem set is to become familiar with basic Matlab commands, practice manipulating vectors and matrices, and try out basic image display and plotting functions. If you are unsure what a function does, at the command line, type 'help' and then the command name.

I. Using Matlab [60 points]

1. Read over the provided Matlab introduction code and its comments: http://web.cs.ucdavis.edu/~yjlee/teaching/ecs189g-spring2015/psets/matlab.pdf Open an interactive session in Matlab and test the commands by typing them at the prompt. (Skip this step if you are already familiar with Matlab.)
2. Describe (in words where appropriate) the result of each of the following Matlab commands. Use the `help` command as needed, but try to determine the output without entering the commands into Matlab. Do **not** submit a screenshot of the result of typing these commands. [10 points]

a.
```
>>  x = randperm(1000);
```

b.
```
>>  a = [1,2,3; 4 5 6; 7 8 9];
>>  b = a(2,:);
```

c.
```
>>  a = [1,2,3; 4 5 6; 7 8 9];
>>  b = a(:);
```

d.
```
>>  f = randn(5,1);
>>  g = f(find(f > 0));
```

e.
```
>>  x = zeros(1,10)+0.5;
>>  y = 0.5.*ones(1,length(x));
>>  z = x + y;
```

f.
```
>>  a = [1:100];
>>  b = a([end:-1:1]);
```

3. Write a few lines of code to do each of the following. Copy and paste your code into the answer sheet. [20 points]

   a. Use `rand` to write a function that returns the roll of a six-sided die.

   b. Let `y` be the vector: `y = [1 2 3 4 5 6]'`. Use the `reshape` command to form a new matrix **Z** that looks like this: $\boldsymbol{Z} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$

   c. Use the `max` and `find` functions to set `x` to the maximum value that occurs in **Z** (above), and set `r` to the row it occurs in and `c` to the column it occurs in.

   d. Let `v` be the vector: `v = [1 8 8 2 1 3 9 8]`. Set a new variable `x` to be the number of 1's in the vector `v`.

4. Create any 100 x 100 matrix **A** (not all constant). Save **A** in a .mat file called PS0_A.mat and submit it. Write a script which loads PS0_A.mat and performs each of the following actions on **A**. Name it PS0_Q1.m and submit it. Try to avoid using for loops. [30 points]

   a. Plot all the intensities in **A**, sorted in decreasing value. Provide the plot in your answer sheet. (Note, in this case we don't care about the 2D structure of **A**, we only want to sort all the intensities in one list.)

   b. Display a histogram of **A**'s intensities with 20 bins. Provide the plot in your answer sheet.

   c. Create a new matrix **Z** that consists of the bottom left quadrant of **A**. Display **Z** as an image in your answer sheet using `imagesc`.

   d. Generate a new image **W**, which is the same as **A**, but with **A**'s mean intensity value subtracted from each pixel. Display **W** as an image in your answer sheet using `imagesc`.

e. Create and display a new matrix **Y** that represents a color image with the same size as **A**, but with 3 channels to represent R G and B values. Set the values of **Y** to be green (i.e., R = 0, G = 255, B = 0) wherever the intensity in **A** is greater than a threshold `t` = the mean intensity of **A**, and black (i.e., R = 0, G = 0, B = 0) everywhere else. Plot **Y** using `imagesc`.

II. Short programming example [40 points]

Write functions to do each of the following to an input color image, and then write a script that loads an image, applies each transformation to the original image, and displays the results in a figure using the Matlab `subplot` function in a 3x2 grid (3 rows and 2 columns). Label each subplot with an appropriate `title`. Name the script PS0_Q2.m.

Apply the script to a color image of your choosing, and show the results.

a) Convert the input color image into a grayscale image.

b) Convert the grayscale image to its "negative image", in which the lightest values appear dark and vice versa (i.e., 0 is mapped to 255, 255 is mapped to 0, etc.)

c) Map the input color image to its "mirror image", i.e., flipping it left to right.

d) Swap the red and green color channels of the input color image.

e) Average the input color image with its mirror image (use typecasting!)

f) Add or subtract a random value between [0,255] to every pixel in the grayscale image, then clip the resulting image to have a minimum value of 0 and a maximum value of 255.

Matlab tips: Do the necessary typecasting (uint8 and double) when working with or displaying the images. Some useful functions: `title, subplot, imshow, mean, imread, rgb2gray` (converts a color image to grayscale). Again, try to avoid using loops. [1]

This assignment is adapted from the PS0 assignments of Kristen Grauman's CS 376: Computer Vision at UT Austin and Devi Parikh's ECE 5554: Computer Vision at Virginia Tech.