CS5680/6680 – Fall Semester 2016
Assignment 1 – Matlab Warm-up Exercises
Due: 11:59 p.m. Saturday, September 10, 2016
Total Points: 30 points

General Assignment Instructions:

1. Be sure to place semicolons wherever appropriate, to suppress unnecessary console output, such as when loading images into memory, or operating on them.
2. Please include comments (e.g., **your name and assignment number**) at the top of each m-file. **In your main script, place a message "-----Finish Solving Problem X-----" followed by a pause command (i.e., wait for a key to be pressed before continuing) at the end of each solution, where X is the question number (i.e., 1, 2, 3, etc.).** For this assignment, you should have three .m files (main script, FindInfo.m, and BlurImage.m).
3. **You should submit your zipped m-files via the Canvas system. Please do not send any image!**
4. Discussion of the assignment is encouraged, but **you may not share code**.
5. Use Matlab "doc" or "help" to learn how to use a function. For example, if you want to learn how to use function imread, you can type "doc imread" or "help imread" on the command line to get more detailed instructions on how to use this function.

Problems:

1. **[1 point]**
Load the image ***peppers.bmp*** into a variable ***A***.

Display the loaded image ***A*** on figure 1 with the message "RGB Original Image" as the figure title.
{*Think: What is the data type of A? What is the size of A?*}

Matlab hints: imread, figure, imshow, title, disp, pause

2. **[5 points]**
Convert image ***A*** into a grayscale image and store it as ***B***.

Transpose image ***B*** as ***TB***.

Vertically flip image ***B*** as ***VB*** so that the left half of ***B*** becomes the right half of ***VB*** and the right half of ***B*** becomes the left half of ***VB***.

Flip columns of image ***B*** in the left/right direction as ***FB***.

For example,

$$\boldsymbol{B} = \begin{matrix} 64 & 2 & 3 & 61 \\ 9 & 55 & 54 & 12 \\ 17 & 47 & 46 & 20 \\ 40 & 26 & 27 & 37 \end{matrix} \quad \boldsymbol{VB} = \begin{matrix} 3 & 61 & 64 & 2 \\ 54 & 12 & 9 & 55 \\ 46 & 20 & 17 & 47 \\ 27 & 37 & 40 & 26 \end{matrix} \quad \boldsymbol{FB} = \begin{matrix} 61 & 3 & 2 & 64 \\ 12 & 54 & 55 & 9 \\ 20 & 46 & 47 & 17 \\ 37 & 27 & 26 & 40 \end{matrix}$$

Display images ***B***, ***TB***, ***VB,*** and ***FB*** on figure 2 with ***B*** located at the upper left, ***TB*** located at the upper right, ***VB*** located at the lower left, and ***FB*** located at the lower right. Label each image with its corresponding matrix name (e.g., ***B***, ***TB***, ***VB***, and ***FB***).

Matlab hints: rgb2gray, transpose (or '), subplot, fliplr, flipud, flipdim

3. **[10 points]**
Save the maximum, minimum, mean, and median intensity values of ***B*** in appropriate variables by calling appropriate Matlab built-in functions, respectively.

Write a Matlab function **FindInfo** to calculate the maximum, minimum, mean, and median intensity value of a grayscale input image. Inside FindInfo function, you are not allowed to call max, min, mean, and median functions. In other words, you have to write your own solutions to get different statistics computed. The prototype of **FindInfo** function is as follows:
function [maxVale, minValue, meanValue, medianValue] = FindInfo(oriIm); where oriIm is the original grayscale image.

Call FindInfo function in your main script to compute maximum, minimum, mean, and median intensity values of ***B*** and save the computed intensity values in appropriate variables, respectively.

Compare your computed four statistics with the four statistics obtained from the Matlab built-in functions using a series of "if" or "if … else" statements and print the comparison results on the Matlab console.

Note: The results computed from your function should be the same as the results computed from calling Matlab built-in functions. If they are not the same, either your implementation is wrong or you called Matlab built-in functions in a wrong way. Please fix the problems before submitting your assignment.

Matlab hints: max, min, mean, median

4. **[4 points]**
Normalize image ***B*** to ***C***, whose data type is **double** and whose values fall in the range of [0, 1] (i.e., the maximum intensity value of the image should be normalized to 1). Display image ***C*** on figure 3 with the message "Normalized Grayscale Image" as the figure title. (Note: Image ***C*** should appear the same as the image ***B***.)

Raise each pixel in the right quarter columns of image ***C*** to the power of 1.5 and raise each pixel in the left quarter columns of image ***C*** to the power of 0.5. Keep the middle two quarter columns of image ***C*** unchanged. Store the result as an image (matrix) ***D***. Display images ***D*** on figure 4 with the message "Processed Grayscale Image" as the figure title. **You are not allowed to use loops to accomplish the task.**

Save image ***D*** in jpeg format to a file called "X_D.jpg" where X should be your first name. Open it using a standard image viewing program to verify that the image is saved properly.

Matlab Hint: double, /, ./, ^, .^, imwrite, display, disp, :

5. **[5 points]**
Perform binary thresholding on the original normalized grayscale image ***C***. A threshold 0.3 is chosen and all values in ***C*** greater than the threshold are set to 1, otherwise set to 0. Find **two efficient solutions** to obtain the thresholded binary image and save it in ***bw1*** and ***bw2***. **Both solutions should not use any loop structure, should not call Matlab built-in function im2bw or imbinarize, and should be distinct in nature.**

Use the Matlab built-in function *im2bw* or *imbinarize* to do the same task and save its thresholded binary image in ***bw3***.

Compare your results ***bw1*** and ***bw2*** with the Matlab's result ***bw3***. If they are equal, display the message "Both of my methods worked"; otherwise, display the appropriate message such as "My method 1 worked but not my method 2", or "My method 2 worked but not my method 1", or "Both of my two methods did not work". Of course, the first message should be displayed if your solutions are correct.

Display ***bw1, bw2,*** and ***bw3*** side-by-side on figure 5 and label the three images with "my first method", "my second method", and "Matlab method", respectively.

Matlab Hint: find, >=, zeros, ones, &, &&

6. **[5 points]**
Write a Matlab function **BlurImage** to replace each of 16 pixels in a non-overlapping 4×4 block of any input image with the average intensity value of these 16 pixels. The prototype of **BlurImage** function is as follows:
function [blurredIm] = BlurImage(oriIm); where oriIm is the original image and blurredIm is the blurred image. Note: blurredIm and oriIm should have the same data type and the same size.

Call **BlurImage** function in your main script to blur image ***A***, and save the blurred image to variable ***BA***.
Call **BlurImage** function in your main script to blur image ***B***, and save the blurred image to variable ***BB***.

Display images ***A***, ***B***, ***BA*** and ***BB*** in the raster-scan order (left to right and top to bottom) with the appropriate title on figure 6.

7. Close all figures and clear all variables.
Matlab Hint: close, clear