CS5680/6680 – Fall Semester 2024
Assignment 1 –Warm-up Exercises
Due: 11:59 p.m. Sunday, September 8, 2024
Total Points: 25 points

General Assignment Instructions:

1. Discussion of the assignment is encouraged, but **you may not share code**.

Problems:

1. **[1 point]**
Load the image ***peppers.bmp*** into a variable ***pepperIm***. Load the image ***Lena.jpg*** into a variable ***lenaIm***.

Display the loaded image ***pepperIm*** and ***lenaIm*** side-by-side on figure 1 with the message "Original Images" as the figure title.

Matlab hints: imread, figure, imshow, subplot, title
Python hints: imread, figure, imshow, subplot, suptitle

2. **[4 points]**
Convert image ***pepperIm*** into a grayscale image and store it as ***pepperGrayIm***.

Transpose image ***pepperGrayIm*** as ***pepperGrayImT***.

Vertically flip image ***pepperGrayIm*** as ***pepperGrayImV*** so that the left half of ***pepperGrayIm*** becomes the right half of ***pepperGrayImV*** and the right half of ***pepperGrayIm*** becomes the left half of ***pepperGrayImV***.

Flip rows of image ***pepperGrayIm*** in the up/down direction as ***pepperGrayImF*** so that the first row of ***pepperGrayIm*** becomes the last row of ***pepperGrayImF*** and the second row of ***pepperGrayIm*** becomes the second to the last row of ***pepperGrayImF***, etc.

For example,

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

Display images ***pepperGrayIm***, ***pepperGrayImT***, ***pepperGrayImV,*** and ***pepperGrayImF*** on figure 2 in the raster-scan order (left to right and top to bottom). In other words, ***pepperGrayIm*** is located at the upper left, ***pepperGrayImT*** is located at the upper right, ***pepperGrayImV*** is located at the lower left, and ***pepperGrayImF*** is located at the lower right. Label each image with its corresponding matrix name (***pepperGrayIm***, ***pepperGrayImT***, ***pepperGrayImV***, and ***pepperGrayImF***).

Matlab hints: rgb2gray, transpose (or '), fliplr, flipud, flipdim
Python hints: cvtColor, transpose, flip, copy

3. **[8 points]**
Save the maximum, minimum, mean, and median intensity values of ***lenaIm*** in appropriate variables by calling appropriate built-in functions, respectively.

Write a function **FindInfo** to calculate the maximum, minimum, mean, and median intensity value of a grayscale input image. Inside FindInfo function, you must write your own solutions to get different statistics computed and you are allowed to use built-in functions to find the dimension of the image.

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

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

Note: If the results computed from your function are not the same as the results computed from calling built-in functions, either your implementation was wrong or you used the wrong built-in functions or called the built-in functions in the wrong way. Please fix the problems before submitting your assignment.

Matlab hints: size, max, min, mean, median
Python hints: shape, max, min, mean, median

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

Raise each pixel in the rows of the second quarter of image ***normalizedLenaIm*** to the power of 1.25 and raise each pixel in the rows of the third quarter of image ***normalizedLenaIm*** to the power of 0.25. Keep the first and the fourth quarter rows of image ***normalizedLenaIm*** unchanged. Store the result as an image (matrix) ***processedNormalizedLenaIm*** and display it 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 ***processedNormalizedLenaIm*** in jpeg format to a file called "X_ processedNormalizedLenaIm.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, :
*Python Hint: astype, /, **, imwrite, :*

5. **[5 points]**
Perform binary thresholding on the normalized grayscale image ***pepperGrayImN***. A threshold is chosen as the absolute value of the difference between the mean and the standard deviation of all values in ***pepperGrayImN.*** Display the chosen threshold value on the console.

Set all values in ***pepperGrayImN*** greater than the threshold to 1. Set all values in ***pepperGrayImN*** less than the threshold to 0. Find **two efficient solutions** to obtain the thresholded binary image and save it in ***bw1*** and ***bw2***, respectively. **Both solutions should not use any loop structure, should not call any built-in functions, and should be distinct.**

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

Compare your results ***bw1*** and ***bw2*** with ***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 methods did not work”. Of course, the first message should be displayed if your solutions are correct. **Make sure that you consider the four aforementioned conditions in your coding.**

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

Matlab Hint: std, find, >, zeros, ones, &, &&, isequal, ~
Python Hint: stdev, >, threshold, and, ==, ~, all

6. **[4 points]**
Write a function **GenerateBlurImage** to replace each of n^2 pixels in a non-overlapping n×n block of any input image (grayscale image or color image) with the average intensity value of these n^2 pixels. For simplicity of coding, you can assume that the number of rows and the number of columns of an input image are divisible by 4 and n is divisible by 4. The output is a blurred color or grayscale image. **The implementation of GenerateBlurImage function should contain one section of code to handle both color and grayscale images (similar to the polymorphism concept).**

Call **GenerateBlurImage** function to blur the color image ***pepperIm*** using a block size of 8 and save the blurred image to variable ***pepperImBlur***.

Call **GenerateBlurImage** function to blur the grayscale image ***lenaIm*** using a block size of 16 and save the blurred image to variable ***lenaImBlur***.

Display images ***pepperIm***, ***lenaIm***, ***pepperImBlur,*** and ***lenaImBlur*** in the raster-scan order (left to right and top to bottom) with the appropriate title on figure 6.