**CSC784M – MP3 – 2T1819**

***(Due 19-Mar-2019, Tue, 6:00 PM)***

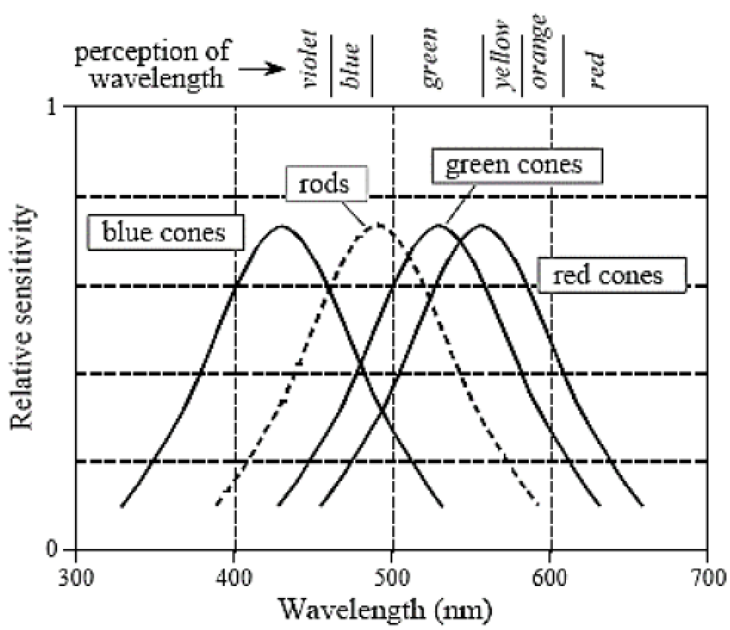
**INSTRUCTION:** This activity is to be done with your partner. Submit your documentation (in *.pdf file format) and your M-file(s) online thru Canvas on or before the deadline. Documents must contain codes and explanations, answer to questions, as well as screenshots of your results (if necessary). Submit a printed copy of your document before the start of the class on 19-Mar-2019.

**1.) Digital Image Structure**

1.1.) Download the image file '*fruits_rgb.bmp*' from our course files. Read the image and store it in a variable in MATLAB. Using commands such as '*image*', '*imagesc*', or '*imshow*', display the RGB image in Figure 1. In three separate figures (i.e., Figures 2, 3 and 4), display each color plane or channel of the original RGB image. Let Figure 2 represent the red color plane, Figure 3 represent the green color plane, and Figure 4 represent the blue color plane. Use '*colormap(gray)*' in displaying each color plane.

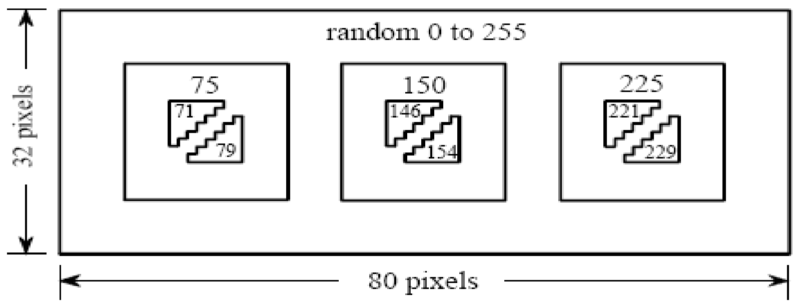
*Q*: Compared with the original image (Figure 1), what have you generally observed from the brightness or intensity patterns in Figures 2, 3 and 4?

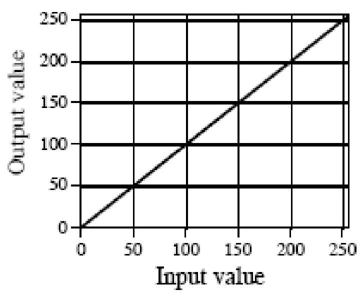
1.2.) Without using 'rgb2gray' or other similar functions in MATLAB, write your own M-file function/script that converts the input RGB image into an output grayscale (intensity) image by using a linear combination of the R, G, and B channels. Use the graph below in deciding the respective weights for the R, G, and B channels (then briefly explain your method).

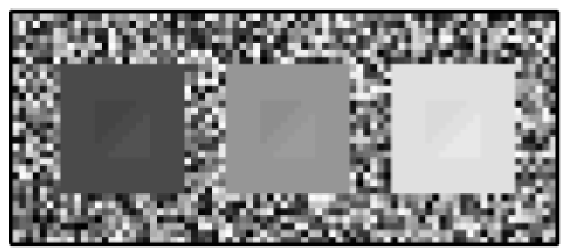


**2.) Brightness and Contrast**

2.1.) Write an M-file script that generates and displays the test image (as described in our last lecture) with the following specifications:

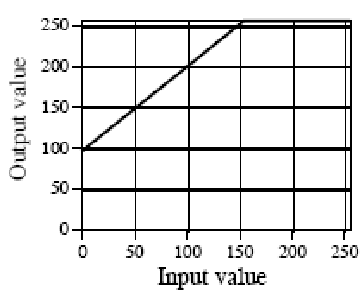


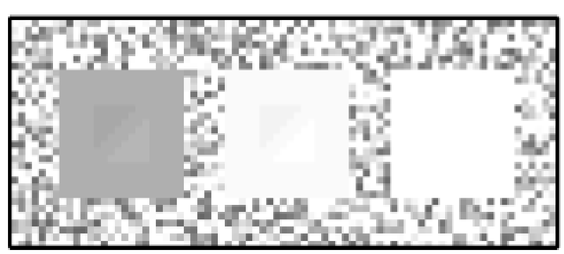



2.2.) Write an M-file script that generates and displays the following images by implementing the respective output transform curves shown at the left side:

a.)



b.)


Output value
250
200
150
100
50
0
0 50 100 150 200 250
Input value

c.)


Output value
250
200
150
100
50
0
0 50 100 150 200 250
Input value

d.)


Output value
250
200
150
100
50
0
0 50 100 150 200 250
Input value

e.)


Output value
250
200
150
100
50
0
0 50 100 150 200 250
Input value

**3.) Histogram Equalization**

3.1.) Without using the built-in functions 'histeq', 'adapthisteq', and the likes, write your own M-file that implements histogram equalization on an input grayscale image. Explain your main algorithm and discuss how you set parameters (if any). Test your M-file by using the input image titled '*ir_cam_im.bmp*'. Plot/display the input and the output images side-by-side such as shown below. In another window, plot the histograms of the input and the output images side-by-side for comparison.

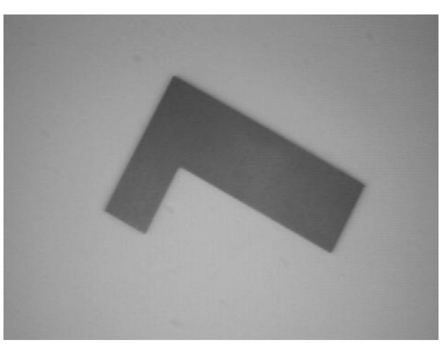
a. Original IR image

b. With grayscale transform

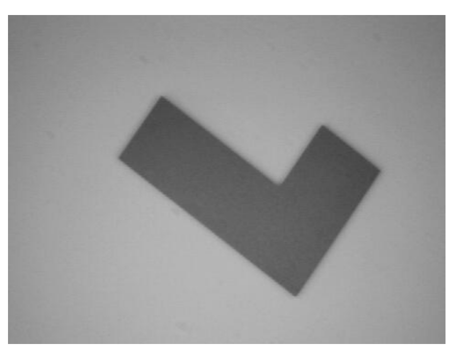
3.2.) Without using 'im2bw' and similar built-in functions in MATLAB, write your own script that converts an input grayscale image into a binary image by thresholding. Using the same input image ('*ir_cam_im.bmp*'), perform a combination of histogram equalization, thresholding, and image binarization such that the human being in the image could be possibly be segmented. Display the input and the output images side-by-side.

*Q*: How did you set your threshold? (Discuss your answer).

**4.) Geometric Transforms** (download the test images '*L_shaped1.bmp*' and '*L_shaped2.bmp*')



'L_shaped1.bmp'



'L_shaped2.bmp'

4.1.) Without using the built-in functions 'imtranslate' and the likes, write your own script that implements image translation by ($\beta x$, $\beta y$). Translate the L-shaped object in each of the test images such that its rightmost and its lowest corners respectively just touch (without clipping) the right and the bottom boundaries of the image. Display the original and the translated images side-by-side for comparison.
*Q*: What are the values of your $\beta x$ and $\beta y$ for each test image case? How did you obtain these values? Explain briefly.

4.2.) Without using the built-in functions 'imrotate' and the likes, write your own script that implements image rotation by an angle $\theta$ about an axis at (x0, y0). Using your M-file, rotate the L-shaped object in the each of the test images such that the 'L' is now in the upright (standing) position. Display the original and the rotated images side-by-side for comparison.
*Q*: What are the values of your $\theta$ and (x0, y0) for each test image case? How did you obtain these values? Explain briefly.

4.3.) Using a combination of histogram equalization, thresholding (and possibly rotation and translation), write a script that determines the approximate difference in size (in terms of number of pixels) of the two L-shaped objects. Briefly explain your algorithm. Include a screen shot of the command window showing the output of your M-file.