CS5680/6680 – Fall Semester 2024
Assignment 2 – Image Enhancement in the Spatial Domain
Due: 11:59 p.m. Sunday, September 22, 2024
Total Points: 35 points

Problems:

Read in the image (***Food.jpg***) and save it in an array ***foodIm*.**

1. [7 points]
Implement a **LinearScaling** function to **linearly** rescale (transform) the intensity values of the grayscale input image to new intensity values selected by the user. The **LinearScaling** function has two input parameters, **inputIm** and **range**, where **inputIm** is the original grayscale image and **range** is a two-element vector containing the new minimum and maximum intensity of the rescaled (transformed) image (e.g., **scaledIm**). The **LinearScaling** function has two output parameters, where **scaledIm** is the transformed image and **transFunc** is the transform function, which is an *n*-element vector that stores the newly transformed intensity for each of the possible grayscale intensities (i.e., the first and last elements of **transFunc** are **range**(1) and **range**(2), respectively). Make sure that your function shows an appropriate error message if the input is incorrect. Note: Both input and output images of the **LinearScaling** function should be an array with the same size and the same data type uint8.

Design four **LinearScaling** function calls to rescale the image ***foodIm*** to a new image ***scaledFoodIm***. The first three of these function calls will have invalid data in **range** so appropriate error messages such that **range** contains non-integer values, negative values, and a larger first element than the second element will be displayed. The last function call will rescale the image ***foodIm*** to a new image ***scaledFoodIm*** with an appropriate range [newMin newMax] so ***scaledFoodIm*** has a good quality. Plot **transFunc** in Figure 1 with appropriate titles on both the *x* and *y* axes.

2. [7 points]
Implement a **CalHist** function to calculate the normalized histogram of the grayscale input image. Inside **CalHist** function, you must write your own solutions and you are only allowed to use built-in functions to find the dimension of the image.

Call the **CalHist** function to calculate the normalized histogram **nH1** of the image ***foodIm*.**

Call the **CalHist** function to calculate the normalized histogram **nH2** of the image ***scaledFoodIm***.

Implement a function to compute the Chi-square distance, as shown below, to measure the similarity between the two normalized histograms ***x*** and ***y*** of the same size, where *n* is the number of elements in each histogram.

$$\chi = \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i - y_i)^2}{x_i + y_i}$$

Implement a function to compute the histogram intersection, as shown below, to measure the similarity between the two normalized histograms ***x*** and ***y*** of the same size.

$$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sum_{i=1}^{n}\min(x_i, y_i)$$

Display the two normalized histograms **nH1** and **nH2** side-by-side in Figure 2 with appropriate titles on both the *x* and *y* axes. On the console, display the two similarity scores and write down your interpretation regarding the similarity level of the two normalized histograms based on the two similarity metrics, namely, Chi-square distance and histogram intersection.

3. [8 points]
Implement a **HistEqualization** function to perform histogram equalization on a grayscale input image to achieve the maximum gray level of 256 **by using the four steps explained in class**. The **HistEqualization** function has one input parameter **inputIm** representing the original grayscale image and two output parameters, **enhancedIm** and **transFunc**, where **enhancedIm** is the histogram equalization result (e.g., histogram equalized image) and **transFunc** is the histogram equalization transform function, which is a 256-element vector that stores the newly transformed intensity with the first and last elements being intensities 0 and 255, respectively. Note: Both input and output images of the **HistEqualization** function should be an array with the same size and the same data type uint8.

Call the **HistEqualization** function to produce the enhanced image ***equalizedFoodIm*** of the original image ***foodIm*** and the corresponding transform function. On the console, display the running time of **HistEqualization** in seconds to accomplish the task.

4. [2 points]
Call an appropriate built-in function to perform histogram equalization on the original grayscale image ***foodIm*** to achieve the maximum gray level of 256 and return the corresponding transform function. On the console, display the running time of the **built-in function** in seconds to accomplish the task.

Note: If the built-in function does not return a transform function, you can simply call the built-in function to return the histogram-equalized image.

5. [11 points]
Please carefully read the paper titled "Contrast Enhancement Using Brightness Preserving Bi-Histogram Equalization" to understand the basic idea of Brightness Preserving Bi-Histogram Equalization (BBHE). Implement a **BBHE** function to perform a contrast enhancement operation as proposed in the paper and return the transform function, which is a 256-element vector that stores the newly transformed intensity with the first and last elements being intensities 0 and 255, respectively. On the console, display the running time of **BBHE** in seconds to accomplish the task. Make sure to use comments to describe basic steps or ideas of the technique.

Call the **BBHE** function to generate the enhanced image ***BBHEFoodIm*** of the original image ***foodIm***.

Display the enhanced images generated in Problems 3, 4, and 5 side-by-side in Figure 3 with appropriate titles.

Plot the transform functions obtained in Problems 3, 4, and 5 side-by-side in Figure 4 with appropriate titles on both the *x* and *y* axes. If the built-in function does not return a transform function in Problem 4, you plot the transform functions obtained in Problems 3 and 5 side-by-side in Figure 4 with appropriate titles on both the *x* and *y* axes.

Write your implementation to compute and display the Peak Signal to Noise Ratio (PSNR) values to compare ***equalizedFoodIm*** (obtained in Problem 3) with the original image ***foodIm*** and compare ***BBHEFoodIm*** (obtained in Problem 5) with the original image ***foodIm***, respectively.