Lab Assignment #3

Due Monday, October 23

(due in class as a hardcopy)

Title: Histogram Equalization

In this lab, you will implement a function that performs histogram equalization on a grayscale image. You can assume all images are grayscale and have values from 0 to 255.

a) Write the function `compute_histogram` that takes a grayscale image as input and returns a length 256 vector `h` which is the normalized histogram of the values in the image. `h` should have values from 0 to 1 and its components should sum to 1.

b) Write the function `plot_histogram` that takes as input a histogram (as returned from `compute_histogram`) and plots it similar to the plots shown below. The x-axis should range from 0 to 255 and be labeled "intensity value". The y-axis should range from 0 to the maximum value of the histogram and should be labeled "PMF" (for probability mass function).

c) Write the function `histogram_transform` which takes as input a histogram (as returned from `compute_histogram`) and returns a length 256 vector `T` which represents the transformation function for histogram equalization. That is, when you index into `T` using a pixel value from an input image (offset by 1), you get the value of the pixel in the output image. Thus `T` is a look-up-table for the transformation. `T` should be computed using equation 3.3-8 (3rd edition) or equation 3-15 (4th edition) in the text.

d) Write the function `equalize` which takes as input a grayscale image and returns the grayscale image which is the histogram equalized version of the input. This function should:

- Call `compute_histogram` to compute the histogram of the input image.
- Call `plot_histogram` to plot the histogram of the input image. Consider using the `bar` function. You can change the range of the axes using the `axis` function. You can change the labels of the axes using the `xlabel` and `ylabel` functions.
- Call `histogram_transform` to compute the transformation function `T`.

- Use the transformation function `T` to produce a histogram equalized version of the input image.
- Call `compute_histogram` to compute the histogram of the equalized image.
- Call `plot_histogram` to plot the histogram of the equalized image.
- The function `equalize` should also print to the console the mean and standard deviations of the input and equalized images. You can use the built in Matlab functions `mean` and `std` to compute these directly from the images but first you will need to transform you image matrices into vectors using the Matlab construct `your_image_matrix(:)` (and possibly convert your images to type `double`).

Apply your `equalize` to the following two images available at the course website (on the schedule page):

- Lab_03_image1_dark.tif
- Lab_03_image2_light.tif

For each image, turn in:

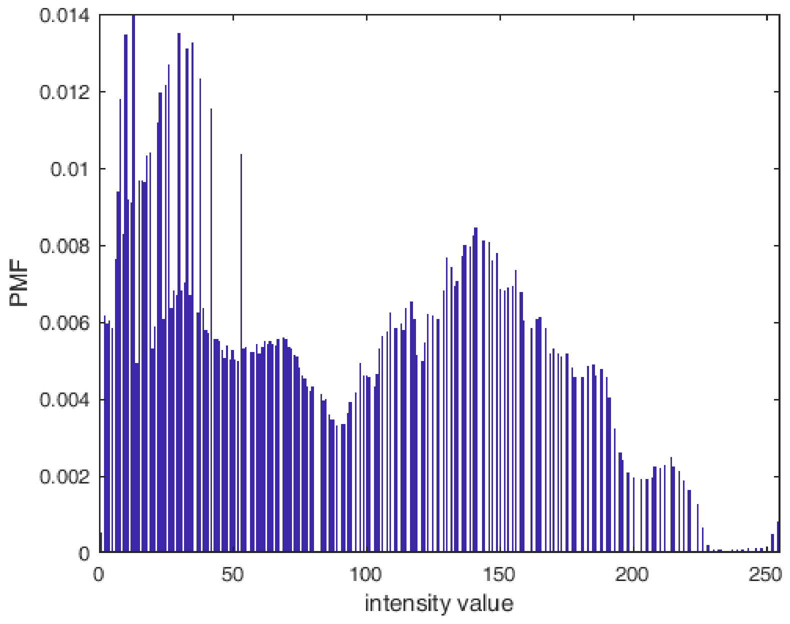
- The plot of the histogram of the original image.
- The histogram equalized version of the image.
- The plot of the histogram of the equalized image.
- The mean and standard deviation of the original and equalized versions.

Turn in your code for the following:

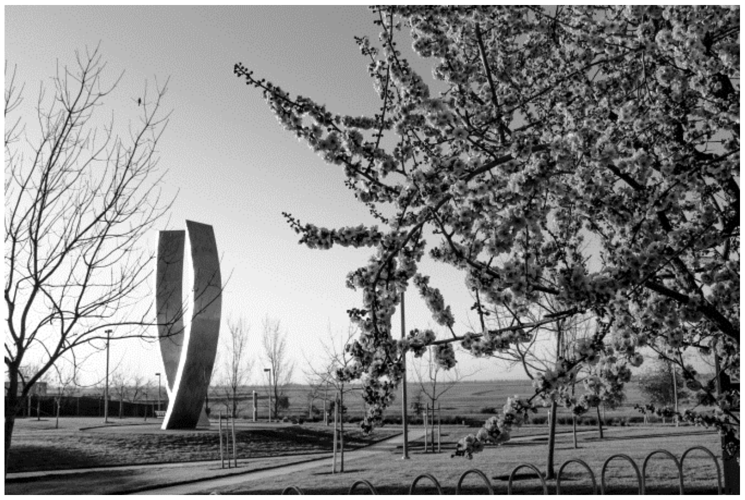
- Your test script.
- Your `compute_histogram` function.
- Your `plot_histogram` function.
- Your `histogram_transform` function.
- Your `equalize` function.

In your results section, simply discuss if you think the histogram equalized versions are improvements over the light and dark images. Also discuss whether the equalized versions look as good as the original image (also available on the course website) and why or why not.

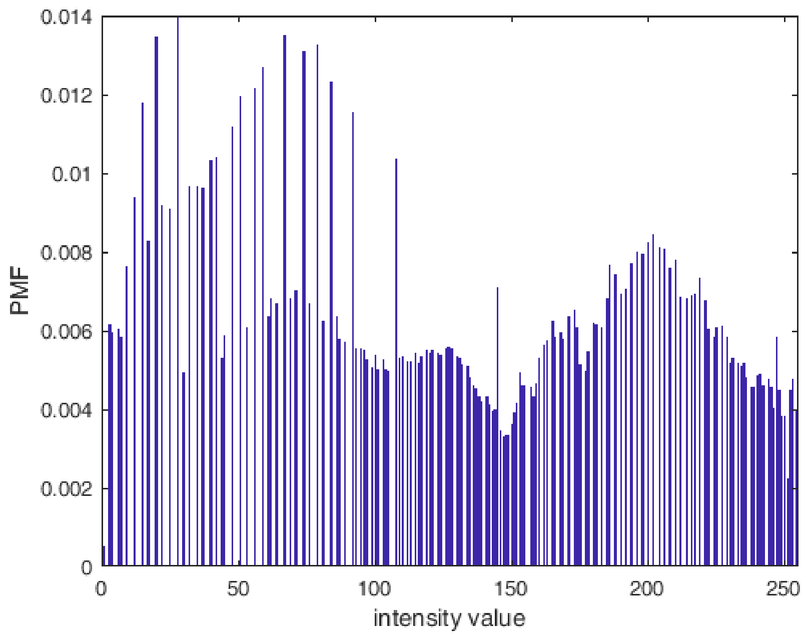
For example, here are my results for the first image Lab_03_image1_dark.tif:



Histogram of input image



Equalized image



Histogram of equalized image

Dark image: mean= 82.82, standard deviation= 60.35

Equalized image: mean= 128.40, standard deviation= 73.40

(note that your results may not be exactly the same as mine)