CSE 107 Lab 04: Histogram Equalization

Cyrus Arellano
Lab: Th 7:30-10:20pm
Haolin Liang
November 14, 2022

Abstract:

The purpose of this lab is to perform histogram equalization on grayscale images. These pixel values will range from 0 to 255. This lab will provide histograms, visuals mean, and standard deviation of its pixel values. This lab is also to help improve the equality of each image to our liking.

Quantitative Results:

	Mean pixel value	**Standard deviation of the pixel values**
Dark Image	82.816910	60.353374
Equalized Dark Image	126.128494	73.790344
Light Image	182.023697	39.150688
Equalized Light Image	125.940477	73.392597

Figure 1: Mean and Standard Deviation Pixel Values

Figure 2: The Dark Image

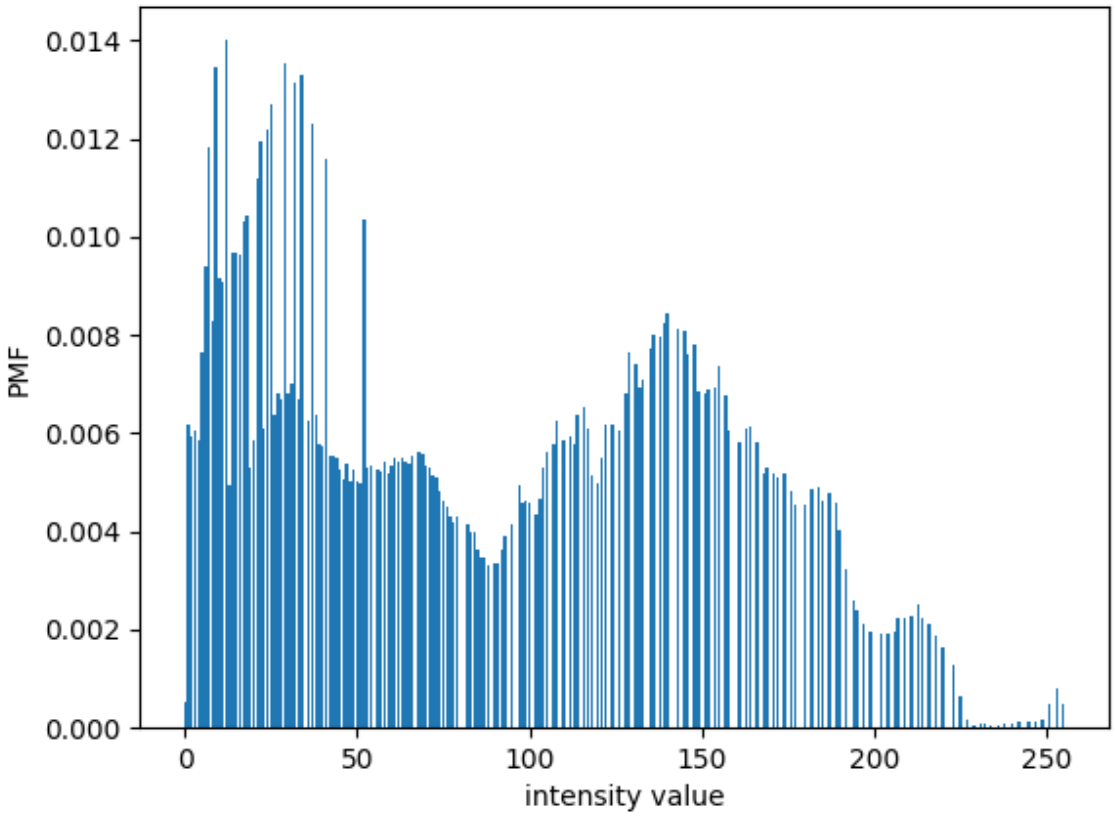

Figure 3: The Histogram of the Dark Image

Figure 4: The Equalized Dark Image

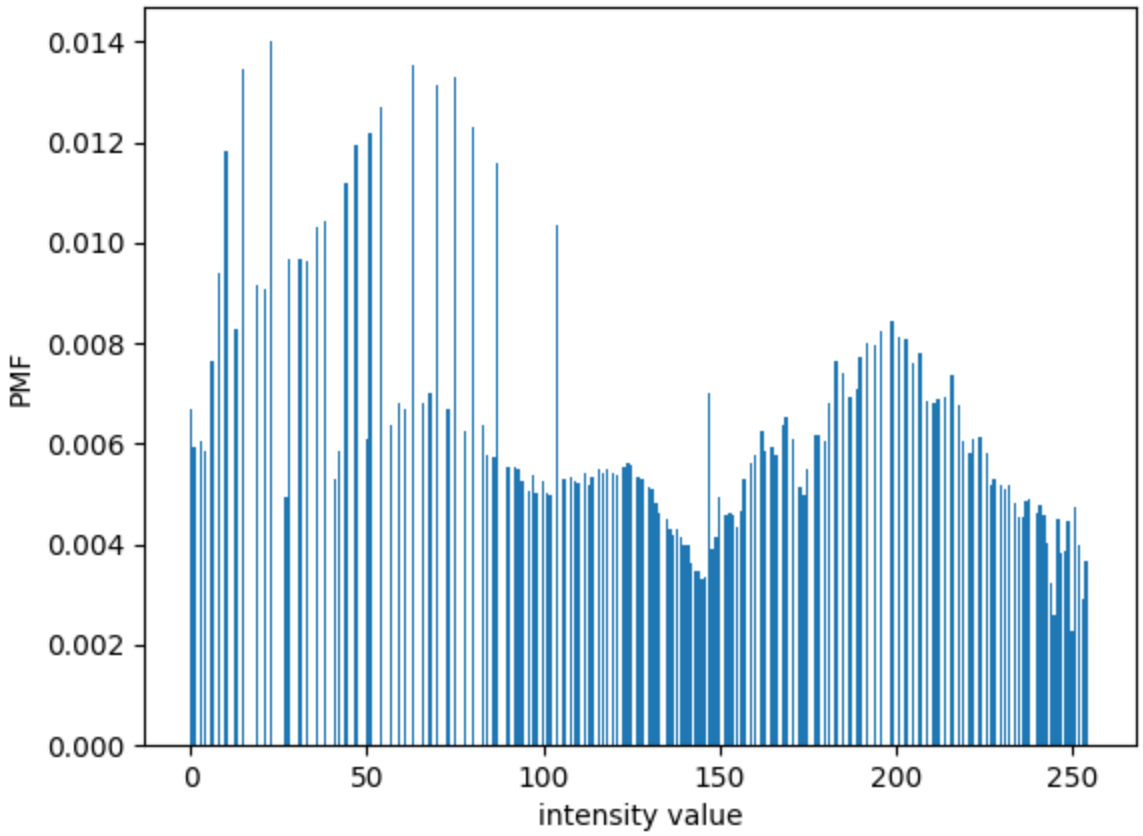

Figure 5: The Histogram of the Equalized Dark Image

Figure 6: The Light Image

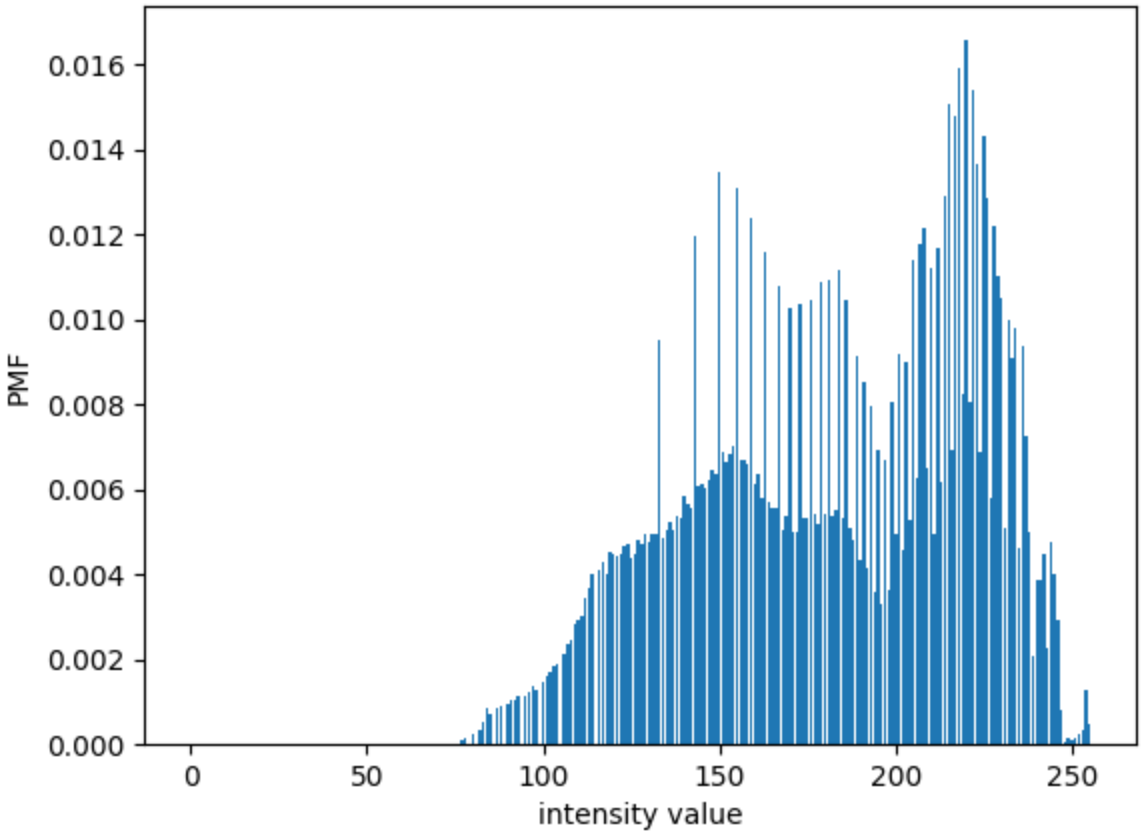

Figure 7: The Histogram of the Light Image

Figure 8: The Equalized Light Image

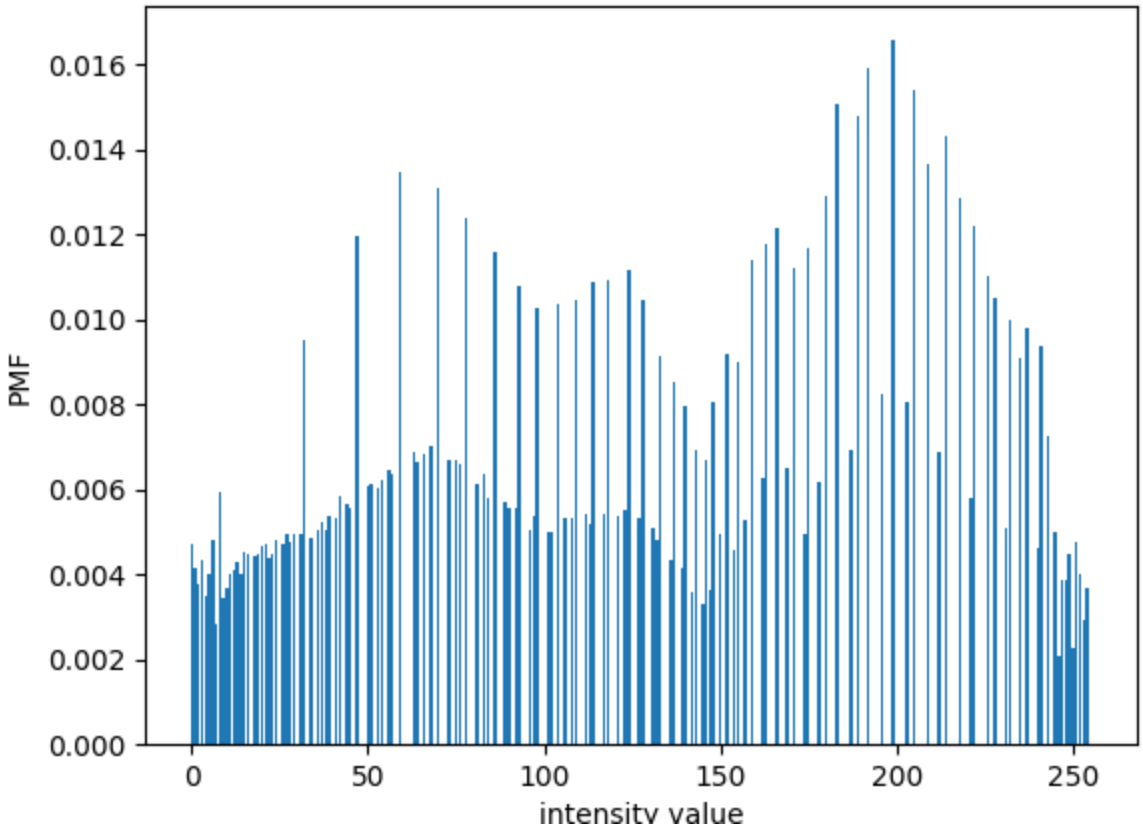

Figure 9: The Histogram of the Equalized Light Image

Questions:

1. Discuss if you think the histogram equalized versions are visual improvements over the dark and light images.

Visually, it looks like the histogram equalized versions are improved over the dark and light images. The darker equalized version looks clearer and less darker while the light equalized version looks clearer and less lighter. Both equalized versions look the same.

2. Discuss how this improvement is reflected in the differences between the histograms of the dark/light images and the histograms of their equalized versions.

In the equalized versions, the data looks more spread out than the ones that are not equalized. This reflects the images because the ones that are not equalized, the data is more towards one side while the data for the equalized version is more spread out.

3. Discuss how this improvement is reflected in differences between the mean and standard deviations of the pixel values in the dark/light images and the mean and standard deviations of the pixel values in their equalized versions.

Because the average is 82.818910 for the dark image, that means more pixels are towards the lower end of pixels which is darker. Same with the light image which has a mean of 182.023697 which helps reflect that this image is lighter mainly because of the high pixel values. When the equalized image has an average of 126.128494 and 125.940477, this means that the pixel values are more spread out, which mainly helps the improvement of the image.

4. What was the most difficult part of this assignment?

The most difficult part of this assignment was trying to figure out the equalize function and trying to get it to work.

Code:

My_HE_functions.py

```
import numpy as np

def compute_histogram( image_pixels ):
    #Gets image dimensions
    r,c = image_pixels.shape
    arr= [0] * 256
    q = r*c
    x = 0
    h= np.zeros(shape=(256))

    #Goes through value pixels and records for histogram
    for row in range(r):
        for col in range(c):
            value = int(image_pixels[row][col])

            for x in range(len(arr)):
                if(value == x):
                    arr[x]+=1

    #Inputs values into histogram
    for x in range(len(arr)):
        t = arr[x]
        value = t / q
        h[x] = value
```

```
    return h



def equalize( in_image_pixels ):
    #Gets pixels
    im = in_image_pixels
    rows, cols = im.shape


    #Creates new array
    n = np.empty((rows, cols), int)

    #Vector
    v= [0]*256
    h= compute_histogram(im)

    #Helps calculate and compute equalization
    L = 256
    for i in range(L):
        for j in range(i):
            v[i] += h[j]

        v[i] = (L-1) * v[i]

    #Puts into new array
    for x in range(rows):
        for y in range(cols):

            q = int(in_image_pixels[x][y])
            p = int(v[q])
            n[x][y] = p

    return n
```

```
def plot_histogram( hist ):
    # plot_histgram  Plots the length 256 numpy vector representing the
normalized
    # histogram of a grayscale image.
    #
    # Syntax:
    #   plot_histogram( hist )
    #
    # Input:
    #   hist = The length 256 histogram vector..
    #
    # Output:
    #   none
    #
    # History:
    #   S. Newsam     10/23/2022   created

    # Import plotting functions from matplotlib.
    import matplotlib.pyplot as plt

    plt.bar( range(256), hist )

    plt.xlabel('intensity value');

    plt.ylabel('PMF');

    plt.show()
```

test_HistogramEqualization.py

```
# Import pillow
from PIL import Image, ImageOps

# Import numpy
import numpy as np
```

```
from numpy import asarray

##############################################################################
#####
# Perform histogram equalization on the dark image.
##############################################################################
#####

# Read the dark image from file.
dark_im = Image.open('Lab_04_image1_dark.tif')

# Show the image.
dark_im.show()

# Create numpy matrix to access the pixel values.
# NOTE THAT WE WE ARE CREATING A FLOAT32 ARRAY SINCE WE WILL BE DOING
# FLOATING POINT OPERATIONS IN THIS LAB.
dark_im_pixels = asarray(dark_im, dtype=np.float32)

# Import compute_histogram from My_HE_functions.
from My_HE_functions import compute_histogram

# Compute the histogram of the dark image.
dark_hist = compute_histogram( dark_im_pixels )

# Import plot_histogram from My_HE_functions.
from My_HE_functions import plot_histogram

# Plot the histogram for the dark image.
plot_histogram( dark_hist )

print('Dark image has mean = %f and standard deviation = %f' % \
    (np.mean(dark_im_pixels), np.std(dark_im_pixels)))

# Import equalize from My_HE_functions.
from My_HE_functions import equalize

# Apply histogram equalization to the dark image.
equalized_dark_im_pixels = equalize( dark_im_pixels );
```

```
# Create an image from numpy matrix equalized_dark_image_pixels.
equalized_dark_image =
Image.fromarray(np.uint8(equalized_dark_im_pixels.round()))

# Show the equalized image.
equalized_dark_image.show()

# Save the equalized image.
equalized_dark_image.save('equalized_dark_image.tif');

# Compute the histogram of the equalized dark image.
equalized_dark_hist = compute_histogram( equalized_dark_im_pixels )

# Plot the histogram for the equalized dark image.
plot_histogram( equalized_dark_hist )

print('Equalized dark image has mean = %f and standard deviation = %f' % \
    (np.mean(equalized_dark_im_pixels), np.std(equalized_dark_im_pixels)))

##########################################################################
#####
# Perform histogram equalization on the light image.
##########################################################################
#####

# Read the light image from file.
light_im = Image.open('Lab_04_image2_light.tif')

# Show the image.
light_im.show()

# Create numpy matrix to access the pixel values.
# NOTE THAT WE WE ARE CREATING A FLOAT32 ARRAY SINCE WE WILL BE DOING
# FLOATING POINT OPERATIONS IN THIS LAB.
light_im_pixels = asarray(light_im, dtype=np.float32)

# Compute the histogram of the light image.
light_hist = compute_histogram( light_im_pixels )

# Plot the histogram for the light image.
```

```
plot_histogram( light_hist )

print('\nLight image has mean = %f and standard deviation = %f' % \
    (np.mean(light_im_pixels), np.std(light_im_pixels)))

# Apply histogram equalization to the light image.
equalized_light_im_pixels = equalize( light_im_pixels );

# Create an image from numpy matrix equalized_light_image_pixels.
equalized_light_image =
Image.fromarray(np.uint8(equalized_light_im_pixels.round()))

# Show the equalized image.
equalized_light_image.show()

# Save the equalized image.
equalized_light_image.save('equalized_light_image.tif');

# Compute the histogram of the equalized light image.
equalized_light_hist = compute_histogram( equalized_light_im_pixels )

# Plot the histogram for the equalized light image.
plot_histogram( equalized_light_hist )

print('Equalized light image has mean = %f and standard deviation = %f' %
\
    (np.mean(equalized_light_im_pixels),
np.std(equalized_light_im_pixels)))
```