# CSE 107: Lab 04: Histogram Equalization

**Angelo Fatali**
**LAB: R 7:30-10:20pm**
**Liang, Haolin**
**November 14, 2022**

**Abstract:**

The point of this lab is to implement a function that will perform histogram equalization on any grayscale image. The two grayscale images we are using are the same but one is lighter and the other is darker. Once these functions are implemented, we can compare the original images and original histograms to the new equalized versions and see if there is an improvement.

**Results:**

| | Mean pixel value | Standard deviation of the pixel values |
|---|---|---|
| **Dark image** | 82.816910 | 60.353374 |
| **Equalized dark image** | 128.402897 | 73.39699 |
| **Light image** | 182.023697 | 39.150688 |
| **Equalized light image** | 128.563568 | 73.829833 |

Table 1: Contains the mean and standard deviations of the dark/light image and the equalized version of the dark/light image.

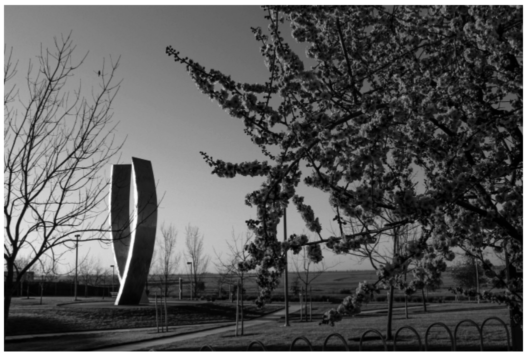

Figure 1: The dark image.

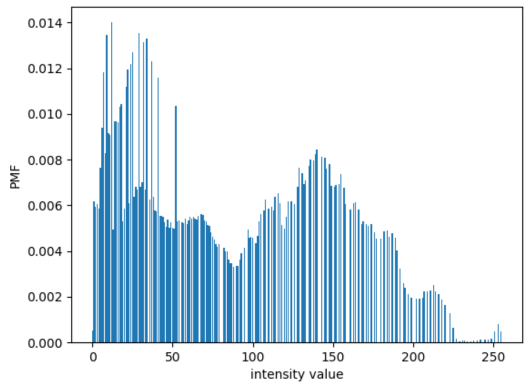

Figure 2: The histogram of the dark image.

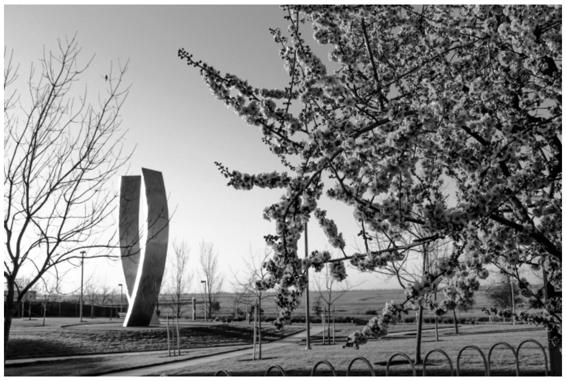

Figure 3: The equalized version of the dark image.

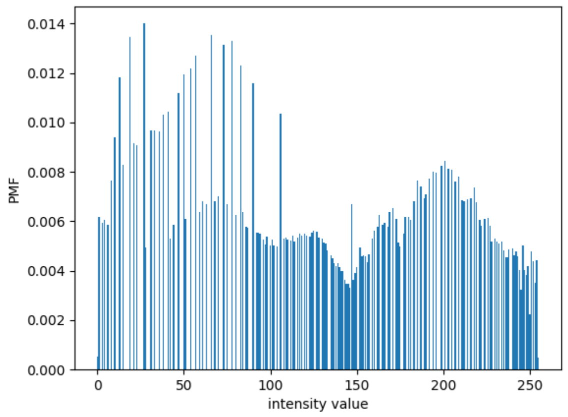

Figure 4: The histogram of the equalized version of the dark image.

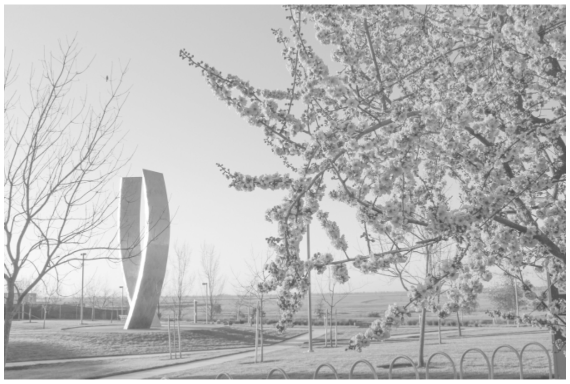
Figure 5: The light image.

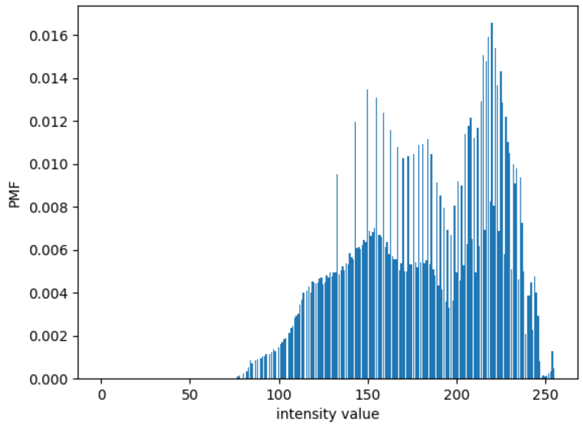

Figure 6: The histogram of the light image.

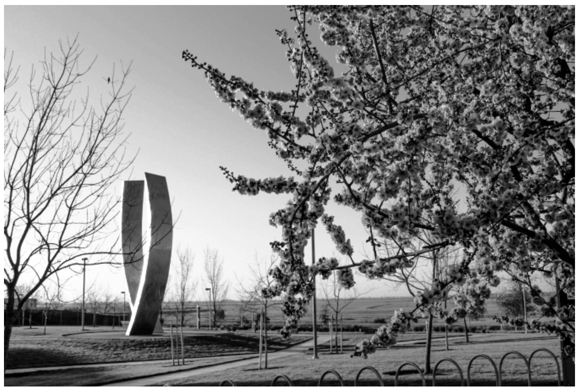
Figure 7: The equalized version of the light image.

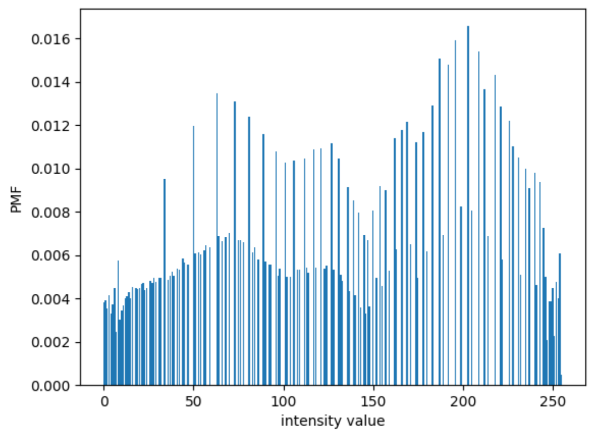

Figure 8: The histogram of the equalized version of the light image.

**Questions:**

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

- I think the equalized light image is a major improvement to the original light image. However, due to personal preference, I don't think the equalized dark image is that big of an improvement to the original dark image. Sometimes a dark image is someone a photographer is trying to achieve, however, it is an improvement depending on preference.

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.

- The dark histogram has most of the values on the left, meaning a lower intensity value, while the light histogram has the opposite. The equalized version of both light a dark have a much broader range of values and not more values on one side, however it seems like the light equalized has a fe that have a higher PMF than the dark equalized histogram.

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.

- The mean of the dark image is about 82 and deviation is about 60 and once equalized went to about 128 and 73, respectively. When compared the the light image, its about a mean of 182 and deviation is about 39 and once equalized went to about 128 and 73, respectively. The equalized versions are about the same values for both the mean and deviation.

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

- Nothing was super difficult other than figuring out the correct documentation for np built in functions

**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)))
```

**<u>My_HE_functions.py</u>**

```python
# MyHEFunctions.py

# Import numpy
import numpy as np

# Write the functioncompute_histogram() that takes a numpy matrix
# representing a grayscale image as input and outputs a length 256
# numpy vector representing the normalized histogram of the image.
def compute_histogram( image_pixels ):
    # compute_histogram  Computes the normalized histogram of a
    # grayscale image.
    #
    # Syntax:
    #   hist = compute_histogram( image_pixels )
    #
    # Input:
    #   image_pixels = The grayscale image as a numpy matrix.
    #
    # Output:
    #   hist = The 256 numpy vector representing a normalized
    #   histogram.
    #
    # History:
    #   F. Angelo     11/13/2022   created

    # Create a length 256 vector of zeros.
    hist = np.zeros(shape=(256))

    # Loop through all the pixels in the image.
    for x in range(image_pixels.shape[0]):
        for y in range(image_pixels.shape[1]):
            # Increment the histogram bin corresponding to the pixel value.
            hist[int(image_pixels[x][y])] += 1

    # Normalize the histogram.
    hist = hist / np.sum(hist)

    return hist




# Write the function equalize() that takes a numpy matrix
# representing a grayscale image as input and outputs a numpy
# matrix representing the histogram equalized version of the
# image.
def equalize( in_image_pixels ):
    # equalize  Takes in as input a grayscale image 256 bits and returns
    # the histogram equalized version.
    #
    # Syntax:
    #   eq_pixels = equalize( in_image_pixels )
    #
    # Input:
    #   in_image_pixels = The grayscale image as a numpy matrix.
    #
    # Output:
    #   eq_img = The equalized grayscale image as a numpy matrix.
    #
    # History:
    #   F. Angelo     11/13/2022   created

    # Get the histogram of the input image.
    hist = compute_histogram( in_image_pixels )

    # Compute the cumulative distribution function.
    cdf = np.cumsum( hist )

    # Normalize the cdf.
    cdf = (256-1) * cdf / cdf[-1]

    # Use linear interpolation of cdf to find new pixel values.
    equalized_image = np.interp(in_image_pixels.flatten(), range(256), cdf)

    # Reshape the equalized image.
    eq_img = equalized_image.reshape(in_image_pixels.shape)

    return eq_img


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()
```