CSE 107: Lab 04: Histogram Equalization.

Valentinno Cruz
LAB: T 10:30-1:20pm
Yuxin Tian
November 6, 2022

Abstract:

Histogram Equalization is uses probability theory to simply create an image higher in contrast. Images are discrete or non-continuous, one can find the histogram with ease by finding all the probabilities of pixels from the range 0- 255 because it is a grey scale image. Upon finding these probabilities they need to be distributed equally in order to create a new histogram of the equalized image. This technique will work for dark tone and light tones images alike. The computations to make this technique work are mathematically not taxing whatsoever, this makes it very efficient. However, to increase efficiency even further, a lookup table of the probabilities can be used instead of having to calculate each and every pixel individually.

Results:

Figure 1 contains the dark image and figure 2 contains the histogram of the dark image. Figure 3 contains the equalized dark image and figure 4 contains the histogram of the equalized dark image. The following table contains the mean and standard deviations of the dark image and the equalized version of the dark image.

	Mean pixel value	Standard deviation of the pixel values
Dark image	82.816910	60.353374
Equalized dark image	128.685124	73.485598

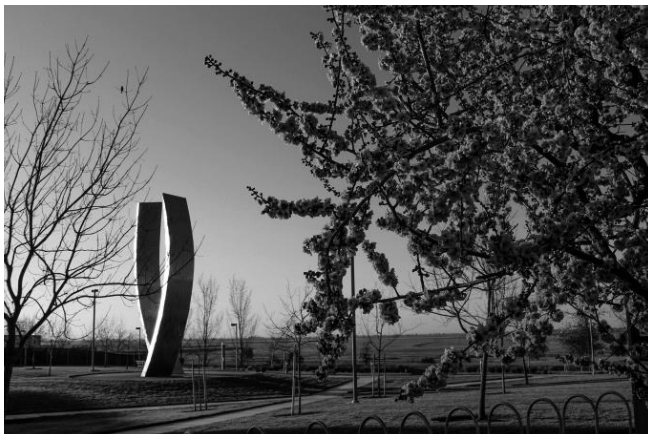


Figure 1. The dark image.

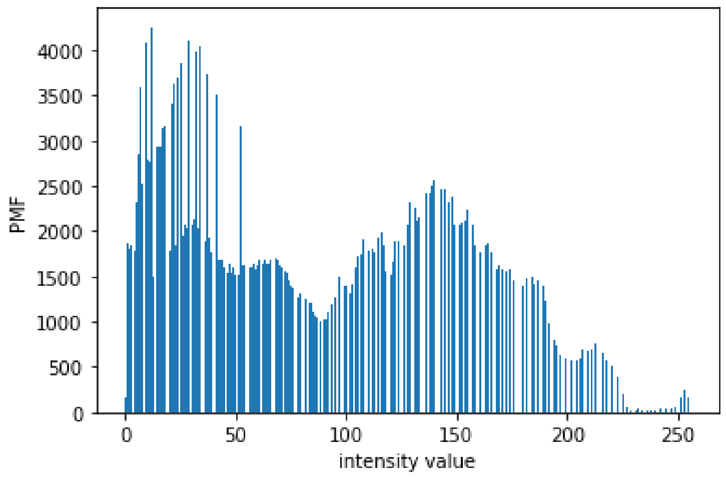


Figure 2. The histogram of the dark image.

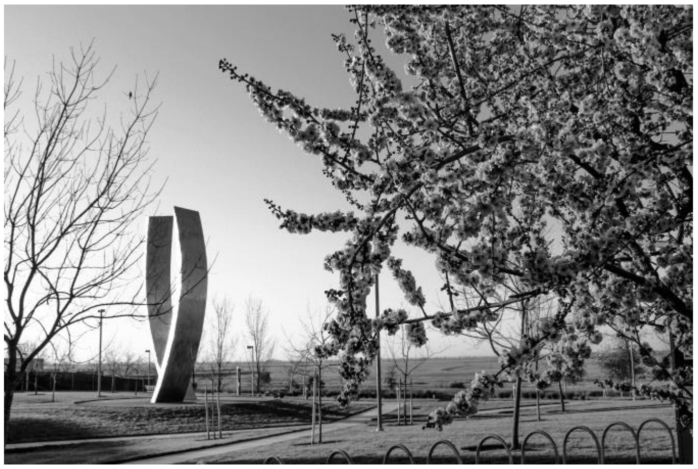


Figure 3. The equalized version of the dark image.

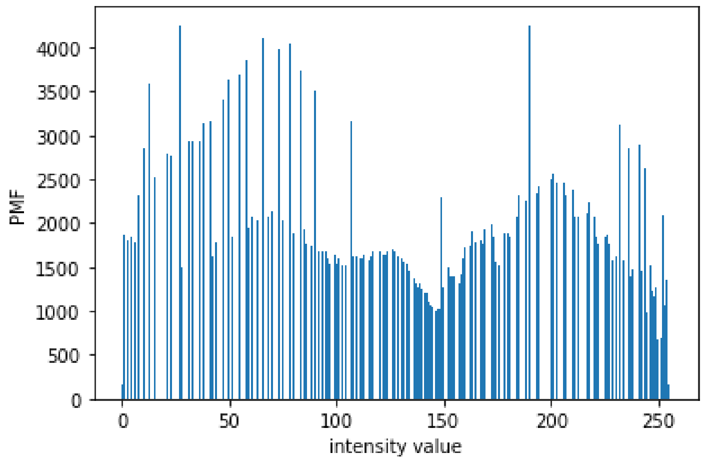


Figure 4. The histogram of the equalized version of the dark image.

Results:

Figure 1 contains the light image and figure 2 contains the histogram of the light image. Figure 3 contains the equalized light image and figure 4 contains the histogram of the equalized light image. The following table contains the mean and standard deviations of the light image and the equalized version of the light image.

	Mean pixel value	Standard deviation of the pixel values
Light image	182.023697	39.150688
Equalized light image	129.015983	73.940505

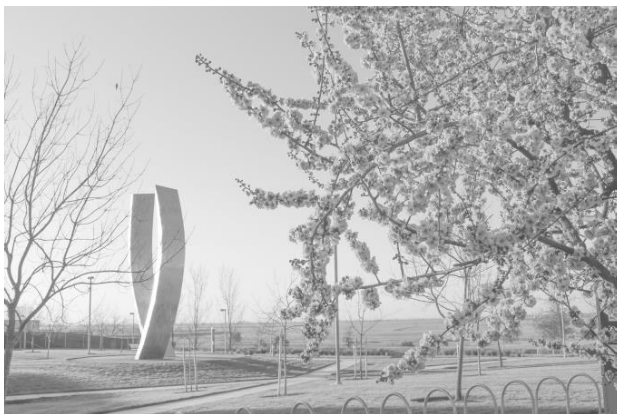


Figure 1. The light image.

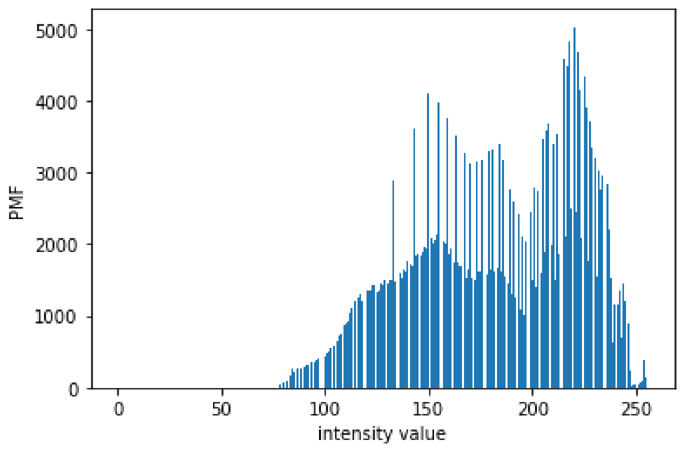


Figure 2. The histogram of the light image.

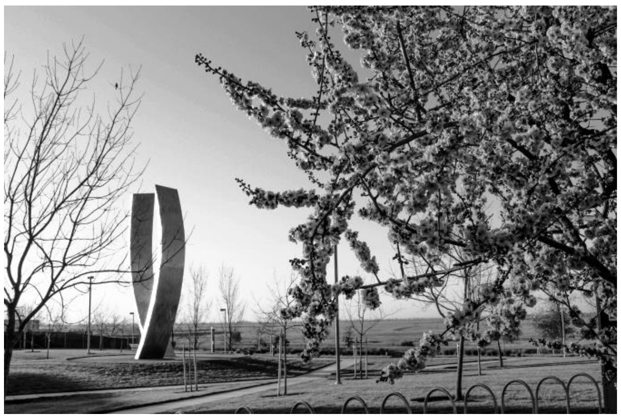


Figure 3. The equalized version of the light image.

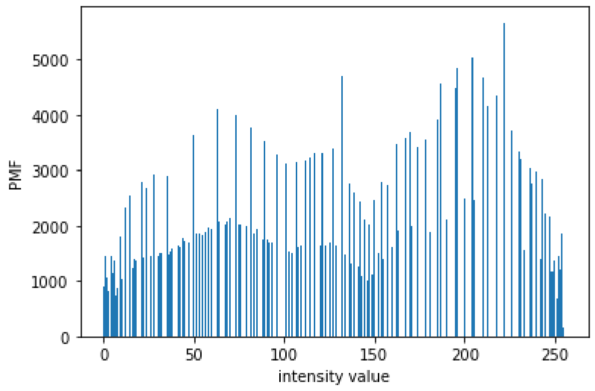


Figure 4. The histogram of the equalized version of the light image.

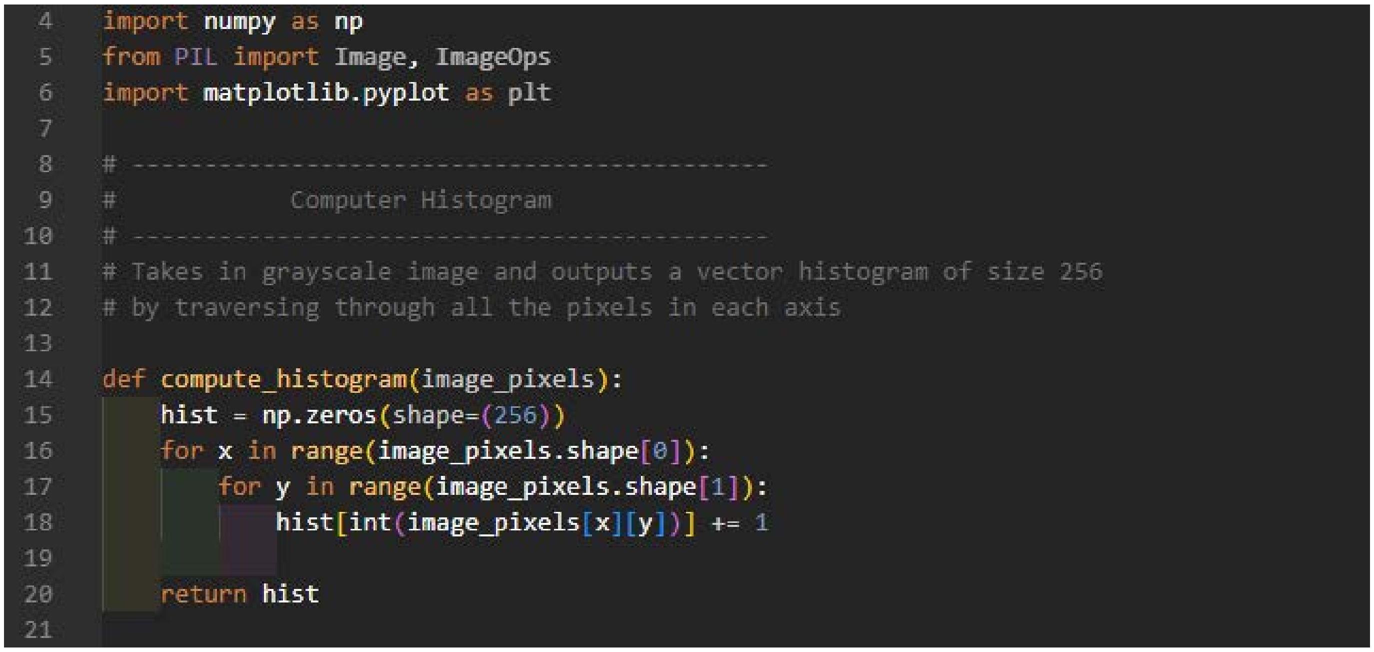
Questions:

1. Discuss if you think the histogram equalized versions are visual improvements over the dark and light images.
    a. As compared to dark and light image there is much improved visuals of equalized image.
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.
    a. Both dark, bright and original histogram equalized images seems like they have more detail than their original forms. Of course, there was no addition of detail but instead the details that were hidden are more defined. This higher contrast essentially shows how histogram equalization has very good applications for enhancing images.
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.
    a. Quantitatively the results were rather interesting. For the dark image in particular there was an understanding that the mean would originally be lower than the histogram of the equalized version. However, when it came to the light version the results were polar opposites. first the mean was high than it lowered. The standard deviation seemed to always be the same regardless of what was done, or in the least bit be very similar to the others. This means that the results show that the deviation itself is controlled by the actual size of the image, or the number of pixel values picked from an image.
4. What was the most difficult part of this assignment?
    a. The equalization portion took me quite some time to understand.

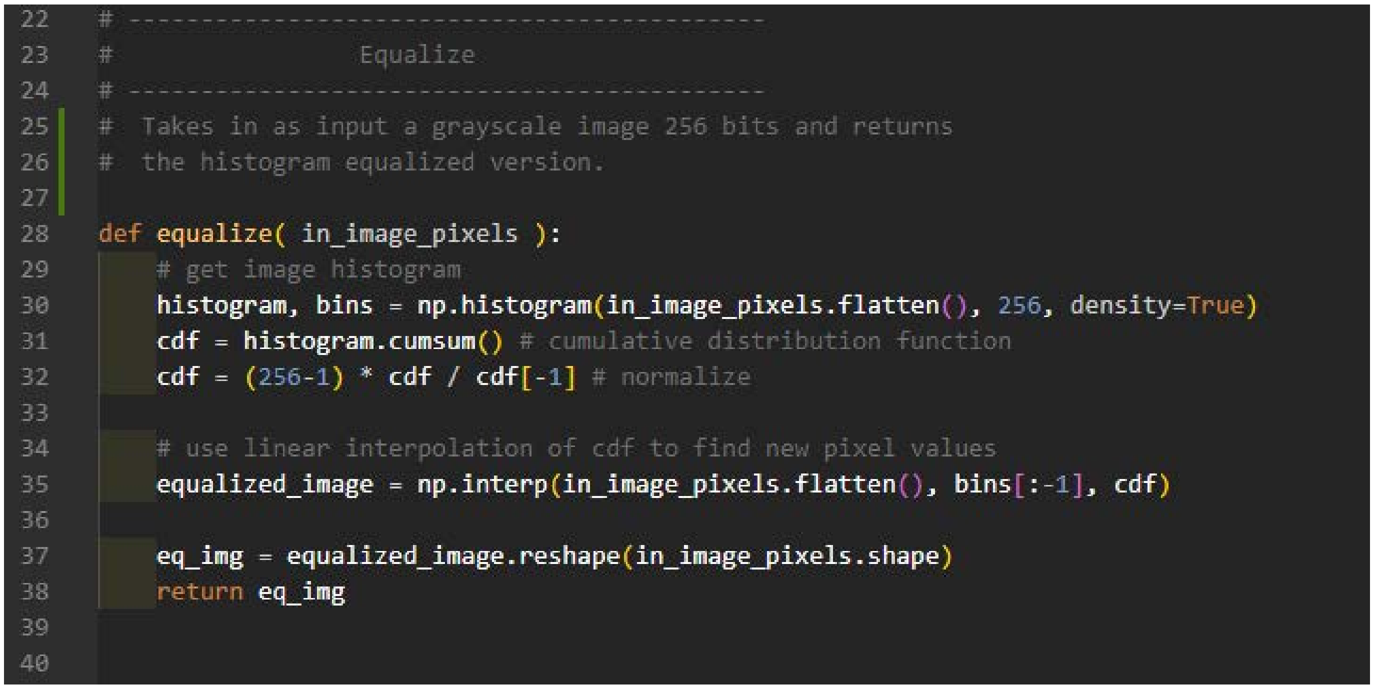
```
# Import pillow from PIL import
Image, ImageOps

# 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)))
```

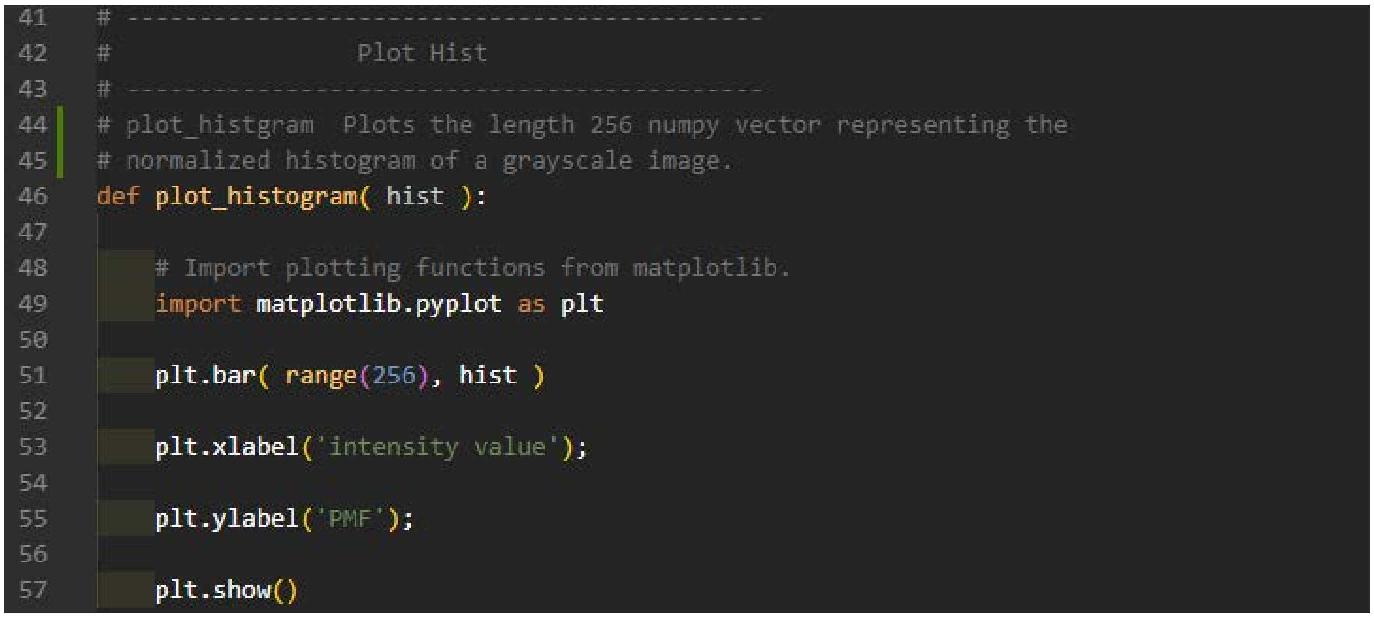
My_HE_functions.py



```python
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

# --------------------------------------------
#             Computer Histogram
# --------------------------------------------
# Takes in grayscale image and outputs a vector histogram of size 256
# by traversing through all the pixels in each axis

def compute_histogram(image_pixels):
    hist = np.zeros(shape=(256))
    for x in range(image_pixels.shape[0]):
        for y in range(image_pixels.shape[1]):
            hist[int(image_pixels[x][y])] += 1

    return hist

```



```python
# --------------------------------------------
#                 Equalize
# --------------------------------------------
#  Takes in as input a grayscale image 256 bits and returns
#  the histogram equalized version.

def equalize( in_image_pixels ):
    # get image histogram
    histogram, bins = np.histogram(in_image_pixels.flatten(), 256, density=True)
    cdf = histogram.cumsum() # cumulative distribution function
    cdf = (256-1) * cdf / cdf[-1] # normalize

    # use linear interpolation of cdf to find new pixel values
    equalized_image = np.interp(in_image_pixels.flatten(), bins[:-1], cdf)

    eq_img = equalized_image.reshape(in_image_pixels.shape)
    return eq_img


```



```python
# --------------------------------------------
#                 Plot Hist
# --------------------------------------------
# plot_histgram  Plots the length 256 numpy vector representing the
# normalized histogram of a grayscale image.
def plot_histogram( hist ):

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

    plt.bar( range(256), hist )

    plt.xlabel('intensity value');

    plt.ylabel('PMF');

    plt.show()
```