**CSE 107: Lab 04: Histogram Equalization.**

**<Shane Clanton>**

**LAB: T 10:30-1:20pm**

**Yuxin Tian**

**November 26, 2022**

**Abstract:**

In a few sentences, describe the purpose of this lab. (Do not mention specific Python functions. This description should be at a very high level.) The purpose of this lab was to implement a function that would perform on histogram equalization on a greyscale image. It would then return the mean and standard deviation pixel values of the light & dark images as well as the equalized mean values of those images.

**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 | 126.597103 | 73.821584 |



Figure 1. The dark image.

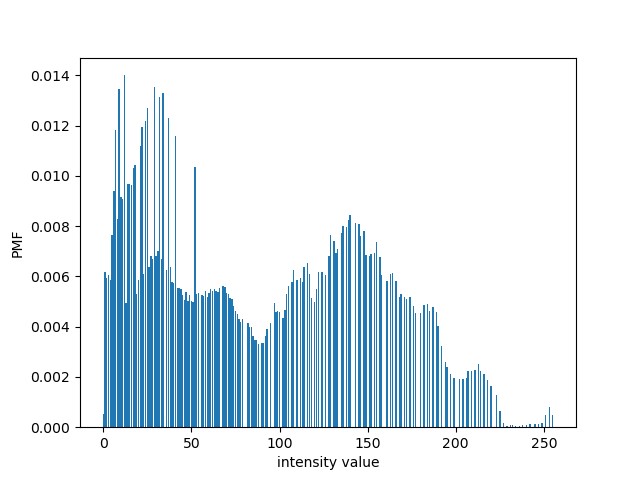


Figure 2. The histogram of the dark image.



Figure 3. The equalized version of the dark image.

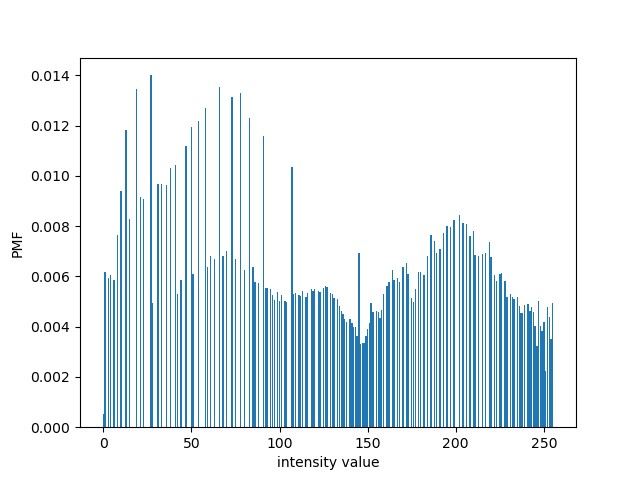


Figure 4. The histogram of the equalized version of the dark image.

<repeat the above for the light image>

|  |  |  |
| --- | --- | --- |
|  | Mean pixel value | Standard deviation of the pixel values |
| Light image | 182.023697 | 39.150688 |
| Equalized light image | 126.436432 | 73.390858 |



Figure 1. The light image.

Chart, histogram

Description automatically generated

Figure 2. The histogram of the light image.



Figure 3. The equalized version of the light image.

Chart, histogram

Description automatically generated

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

**Questions:**

<Your answers to the assignment questions>

Questions:

1. Discuss if you think the histogram equalized versions are visual improvements over the dark and light images.
   1. **I do believe that the histogram equalized versions of the dark and light images are visual improvements.**
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.
   1. **For the dark image histogram, the intensity values are closer to 0. And the light image histogram, are closer to 255. While the equalized versions of the light and dark images are more balanced on either side of the intensity axis.**
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.
   1. **How the equalized versions of the light and dark images are an improvement over the original images are that they are almost the exact same numbers unlike that of the original images that can clearly be seen that they are not the same.**
4. What was the most difficult part of this assignment?
   1. **The most difficult part was to properly get the equalized images as well as their histograms to properly display.**

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.

###############################################################################

test\_HistogramEqualization.py

# 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

# MyHEFunctions.py

# Import numpy

import numpy as np

def compute\_histogram( image\_pixels ):

#<your function header>

# compute\_histgram computing the normalized histogram.

#

# Syntax:

# compute\_histogram( image\_pixels )

#

# Input:

# image\_pixels = the pixels of the input image.

#

# Output:

# hist = numpy array of length 256 representing the normalized histogram.

#

# History:

# S. Clanton 10/26/2022 created

#<your implementation>

hist = np.zeros(shape=(1, 256))

for row in image\_pixels:

for pixel in row:

hist[0][int(pixel)] += 1

hist = hist[0] / (image\_pixels.shape[0]\*image\_pixels.shape[1])

return hist

def equalize( in\_image\_pixels ):

#<your function header>

# equalize equalizes the normalized histogram and uses it to return the equalized image

#

# Syntax:

# equalize( in\_image\_pixels )

#

# Input:

# in\_image\_pixels = equalized pixels of the input image

#

# Output:

# out\_image\_pixels = numpy array representing the equalized histogram

#

# History:

# S. Clanton 10/28/2022 created

#<your implementation>

hist = compute\_histogram(in\_image\_pixels)

eq\_hist = np.zeros(shape=(1, 256))

for i, entry in enumerate(hist):

pixel\_sum = 0

for j in range(i):

pixel\_sum += hist[j]

pixel\_sum \*= (256-1)

eq\_hist[0][i] = pixel\_sum

out\_image\_pixels = np.zeros(shape = in\_image\_pixels.shape)

for i, row in enumerate(in\_image\_pixels):

for j, pixel in enumerate(row):

out\_image\_pixels[i][j] = eq\_hist[0][int(pixel)]

return out\_image\_pixels

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()