National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE 433: Digital Image Processing

Name: Amal Saqib

CMS: 282496

Class: BSCS 9C

Lab 6: Histogram Equalization

Date: 25th October 2021

Time: 2.00Pm to 5.00Pm

Instructor: Dr. Imran Malik

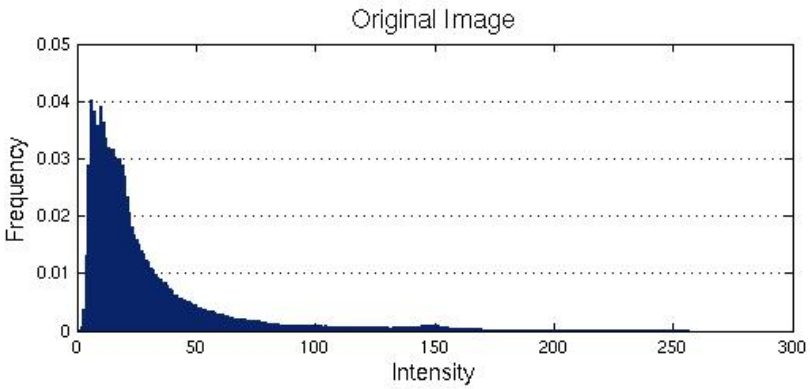
Lab 6

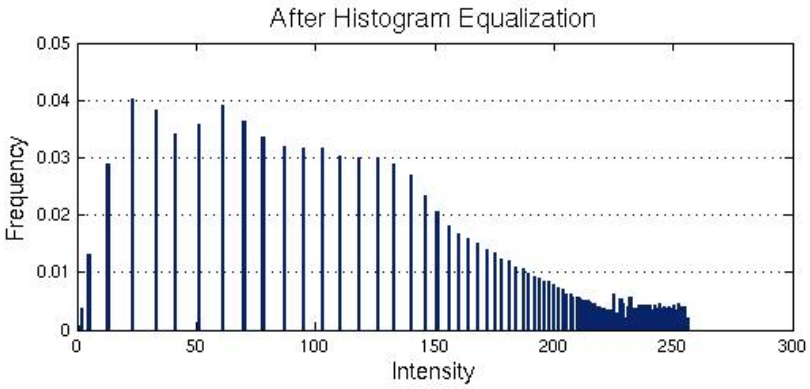
Histogram Equalization

Introduction

In this lab you will try to improve the contrast of an image by doing histogram equalization.

Consider the example below.





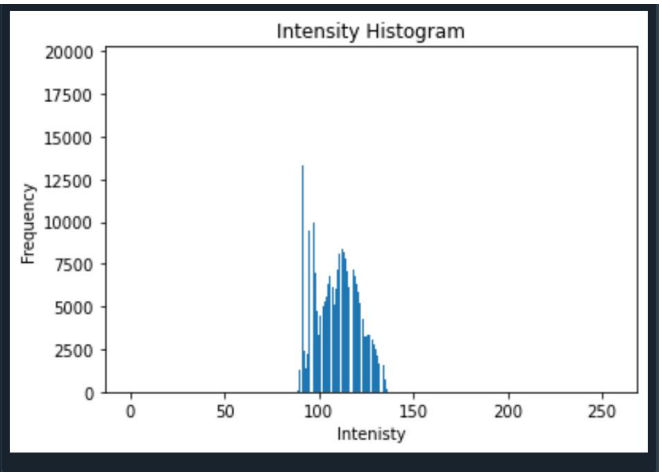
Tools/Software Requirement

Python 3.X

Description

a) Write a program that shows the histogram for a given image. For displaying the histogram, you may use Matplotlib. The following link might be helpful in this regard. https://bespokeblog.wordpress.com/2011/07/11/basic-data-plotting-with-matplotlib-part-3-histograms/

```
from PIL import Image
from matplotlib import pyplot as plt
import numpy as np

def populateCountArray(img):
    countArray = [0]*256

    pix = img.load()
    M, N = img.size

    for x in range(M):
        for y in range(N):
            countArray[pix[x, y]] += 1

    return countArray

def showHistogram(bars, array):
    plt.title("Intensity Histogram")
    plt.xlabel("Intenisty")
    plt.ylabel("Frequency")
    plt.bar(bars, array)
    plt.show()

image = Image.open("image_lab6.jpg").convert('L')
histArray = populateCountArray(image)

bars = np.arange(len(histArray))
showHistogram(bars, histArray)
```



b) Write a program that equalizes the histogram of a given image. Consider the formula,

$$s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j) = \frac{(L-1)}{MN} \sum_{j=0}^{k} n_j$$

where

$$k = 0,1,2,\ldots,L-1$$

CODE

```
from PIL import Image
from matplotlib import pyplot as plt

L = 256

# returns the width and height of the image
def imageProperties(img):
   return img.size

# returns the number of pixels of each intensity level
def populateCountArray(img):
   countArray = [0] * 256
```

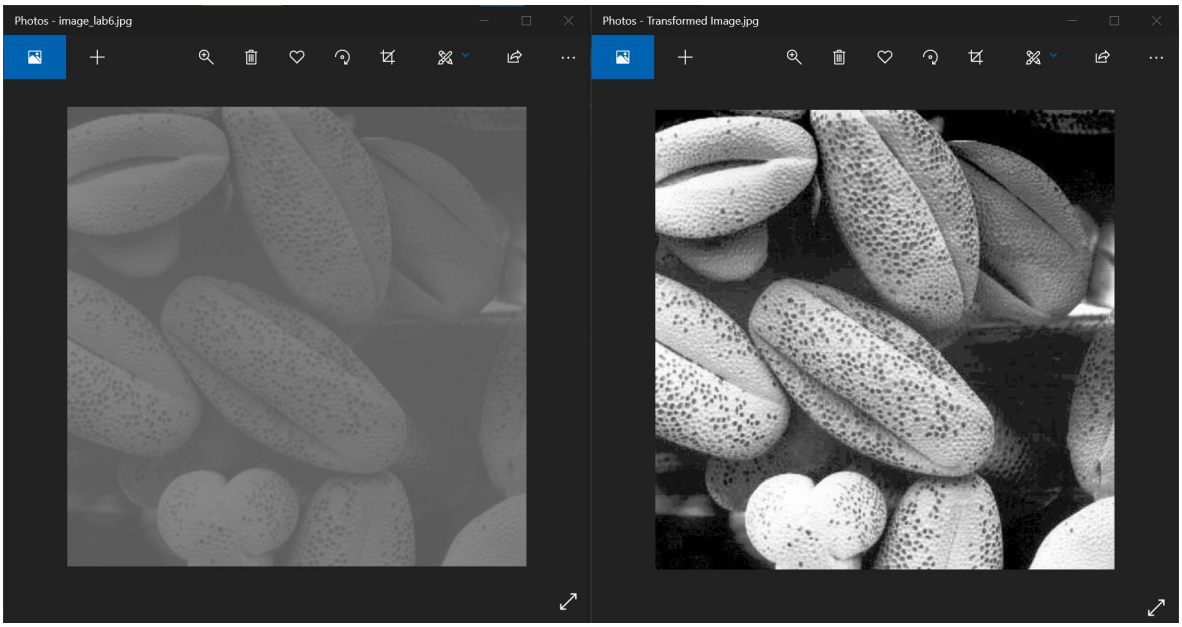
```
    # load the pixels
    pix = img.load()

    # M is the width and N is the height of the image
    M, N = imageProperties(img)

    # iterate through the pixels
    for x in range(M):
        for y in range(N):
            # add 1 to the countArray whenever a particular intensity pixel is found
            countArray[pix[x, y]] += 1

    return countArray

# probability distribution function
def pdf(array, img):
    pdfArray = [0] * 256
    M, N = imageProperties(img)

    # pdf = frequency / total no. of pixels
    for i in range(len(array)):
        # M * N is the total no. of pixels
        pdfArray[i] = array[i] / (M * N)

    return pdfArray
# cummulative frequency distribution function
def cdf(array):
    cdfArray = [0] * 256

    # at every index, find the sum of current and all previous pdfs
    for i in range(len(array)):
        for j in range(i):
            cdfArray[i] += array[j]
    return cdfArray

# map cdf to intensity values
def transformation(array):
    transformed = [0] * 256
    for i in range(len(array)):
        transformed[i] = round(array[i] * (L - 1))
    return transformed

# plot the histogram
def showHistogram(array):
    plt.title("Intensity Histogram")
    plt.xlabel("Intenisty")
    plt.ylabel("Probability distribution function")
    plt.bar([i for i in range(256)], array)
    plt.show()
```

```
# apply histogram equalization to the input image
def outputImage(array, img):
   pix = img.load()
   M, N = imageProperties(img)

   # iterate the pixels
   for x in range(M):
      for y in range(N):
         # change the intensity of the pixel to the transformed one
         pix[x, y] = array[pix[x, y]]

   return img


# open image
image = Image.open("image_lab6.jpg").convert('L')
histArray = populateCountArray(image)
pdfArray = pdf(histArray, image)
cdfArray = cdf(pdfArray)
transformed = transformation(cdfArray)
# show the histogram for the input image
showHistogram(histArray)
# get output image
output = outputImage(transformed, image)
histOutput = populateCountArray(image)
# show the histogram for the input image
showHistogram(histOutput)
# save output image
output.save("Transformed Image.jpg")
```

```python
from PIL import Image
from matplotlib import pyplot as plt

L = 256

# returns the width and height of the image
def imageProperties(img):
    return img.size

# returns the number of pixels of each intensity level
def populateCountArray(img):
    countArray = [0] * 256

    # load the pixels
    pix = img.load()

    # M is the width and N is the height of the image
    M, N = imageProperties(img)

    # iterate through the pixels
    for x in range(M):
        for y in range(N):
            # add 1 to the countArray whenever a particular intensity pixel is found
            countArray[pix[x, y]] += 1

    return countArray

# probability distribution function
def pdf(array, img):
    pdfArray = [0] * 256
    M, N = imageProperties(img)

    # pdf = frequency / total no. of pixels
    for i in range(len(array)):
        # M * N is the total no. of pixels
        pdfArray[i] = array[i] / (M * N)

    return pdfArray
# cummulative frequency distribution function
def cdf(array):
    cdfArray = [0] * 256

    # at every index, find the sum of current and all previous pdfs
    for i in range(len(array)):
        for j in range(i):
            cdfArray[i] += array[j]
    return cdfArray

# map cdf to intensity values
def transformation(array):
    transformed = [0] * 256
    for i in range(len(array)):
        transformed[i] = round(array[i] * (L - 1))
    return transformed
```

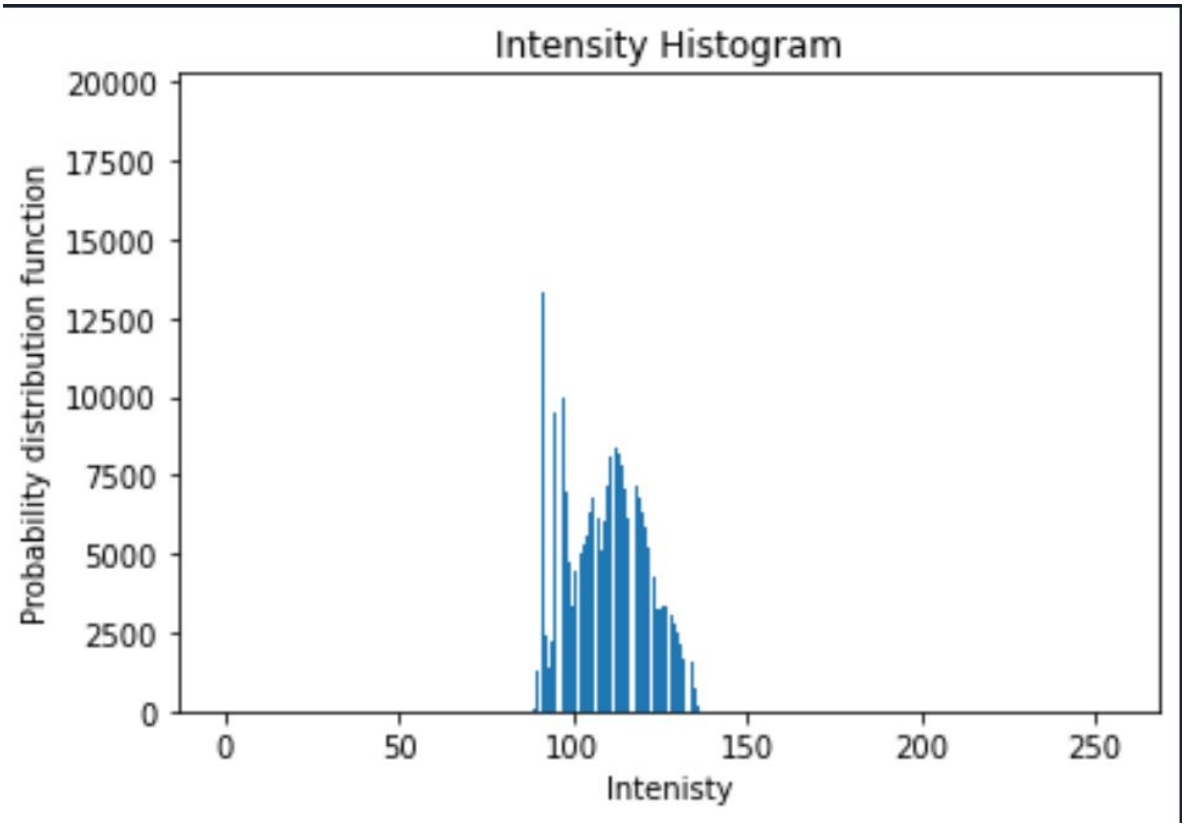
```
56  # plot the histogram
57  def showHistogram(array):
58      plt.title("Intensity Histogram")
59      plt.xlabel("Intenisty")
60      plt.ylabel("Probability distribution function")
61      plt.bar([i for i in range(256)], array)
62      plt.show()
63
64  # apply histogram equalization to the input image
65  def outputImage(array, img):
66      pix = img.load()
67      M, N = imageProperties(img)
68
69      # iterate the pixels
70      for x in range(M):
71          for y in range(N):
72              # change the intensity of the pixel to the transformed one
73              pix[x, y] = array[pix[x, y]]
74
75      return img
76
77
78  # open image
79  image = Image.open("image_lab6.jpg").convert('L')
80  histArray = populateCountArray(image)
81  pdfArray = pdf(histArray, image)
82  cdfArray = cdf(pdfArray)
83  transformed = transformation(cdfArray)
84  # show the histogram for the input image
85  showHistogram(histArray)
86  # get output image
87  output = outputImage(transformed, image)
88  histOutput = populateCountArray(image)
89  # show the histogram for the input image
90  showHistogram(histOutput)
91  # save output image
92  output.save("Transformed Image.jpg")
```

OUTPUT

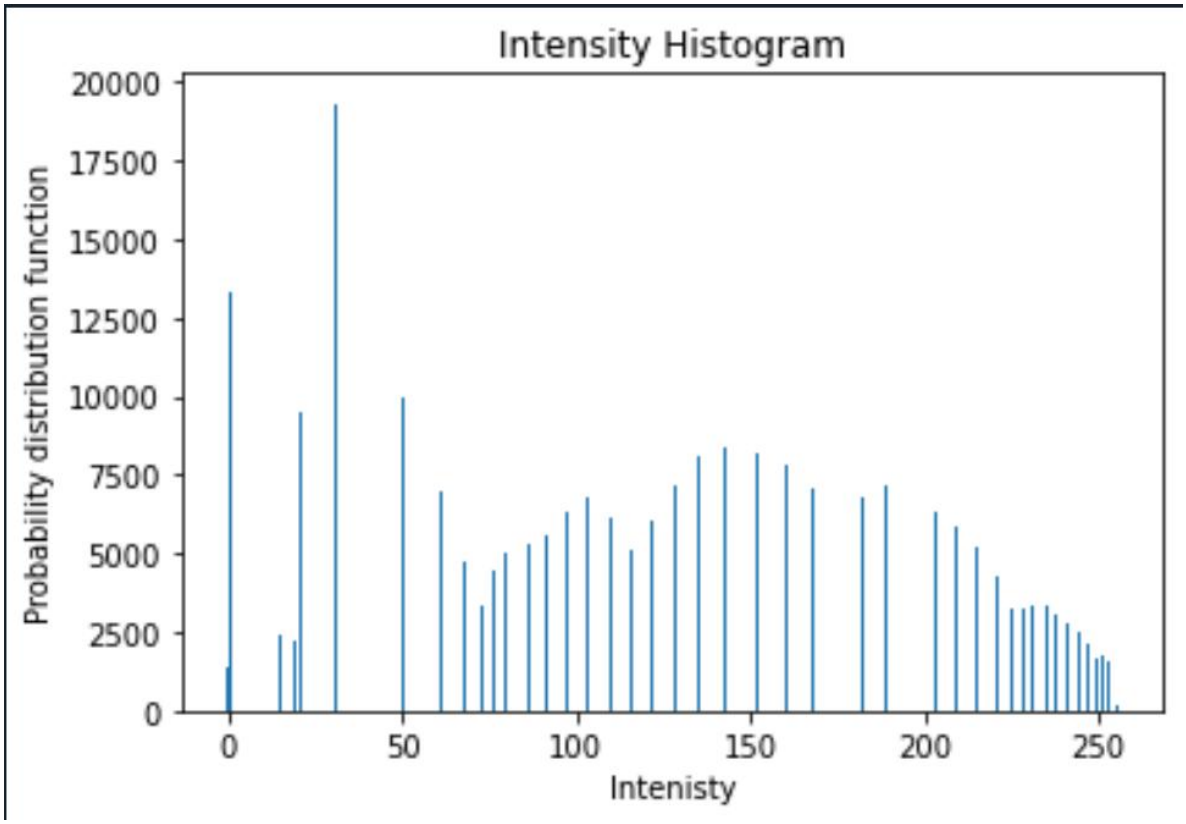


c) Show the histograms before and after equalization. Again consult Matplotlib. Opencv can also be used for some parts.

Before:



After:



d) Does the equalized histogram has a uniform distribution?
No. At low intensity values, the histogram is not uniform. At middle/high intensity the histogram is almost uniform.

Some important points about the exercise.

1. You should apply histogram equalization on Greyscale image (given in lab folder).
2. You should not use the builtin histogram equalization method available in matplotlib or opencv. You can use these packages only for displaying histograms.
3. You should implement the formula mentioned in the exercise sheet to implement histogram equalization.

Deliverable

Please upload the report with code and screenshots of output.