Assignment10-Histogram Equalization

Md Al Shahria

Roll: 1810676121

What is a Histogram of An Image?

A histogram of an image is the graphical interpretation of the image's pixel intensity values. It can be interpreted as the data structure that stores the frequencies of all the pixel intensity levels in the image. As we can see in the

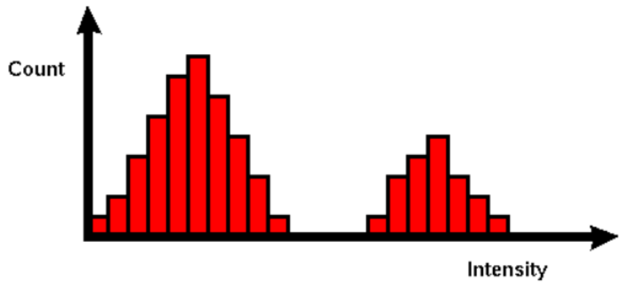

Figure 1: Histogram

image above, the X-axis represents the pixel intensity levels of the image. The intensity level usually ranges from 0 to 255. For a gray-scale image, there is only one histogram, whereas an RGB colored image will have three 2-D histograms — one for each color. The Y-axis of the histogram indicates the frequency or the number of pixels that have specific intensity values.

What is Histogram Equalization?

Histogram Equalization is an image processing technique that adjusts the contrast of an image by using its histogram. To enhance the image's contrast, it spreads out the most frequent pixel intensity values or stretches out the intensity range of the image. By accomplishing this, histogram equalization allows the image's areas with lower contrast to gain a higher contrast.

Why Do You Use Histogram Equalization?

Histogram Equalization can be used when you have images that look washed out because they do not have sufficient contrast. In such photographs, the light and dark areas blend together creating a flatter image that lacks highlights and shadows. Let's take a look at an example -

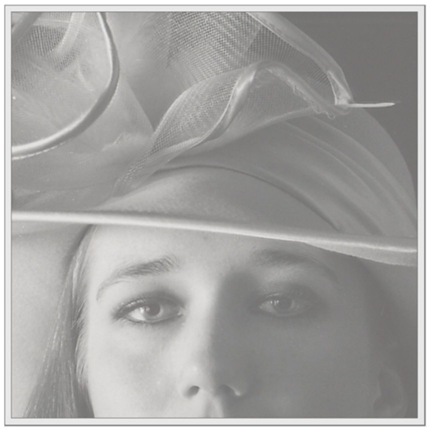

Figure 2: low contrast image

How to Use Histogram Equalization

Before we get started, we need to import the OpenCV-Python package, a Python library that is designed to solve computer vision problems. In addition to OpenCV-Python, we will also import NumPy and Matplotlib to demonstrate the histogram equalization.

```python
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

Next, we will assign a variable to the location of an image and utilize .imread() method to read the image. And we'll use OpenCV RGB2GRAY method to convert this image into grayscale which isn't mandatory.

```python
img_path = '.../eyes.jpg'
img = plt.imread(img_path)
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

Now that our test image has been read, we can use the following code to view its histogram.

```python
plt.subplots(figsize=(20,20))
plt.subplot(2,2,1)
plt.imshow(img,cmap='gray')

plt.subplot(2,2,2)
original_hist = plt.hist(gray.ravel(),256,[0,256])

plt.subplot(2,2,3)
plt.imshow(equalized,cmap='gray')

plt.subplot(2,2,4)
equalized_hist = plt.hist(equalized.ravel(),256,[0,256])
```

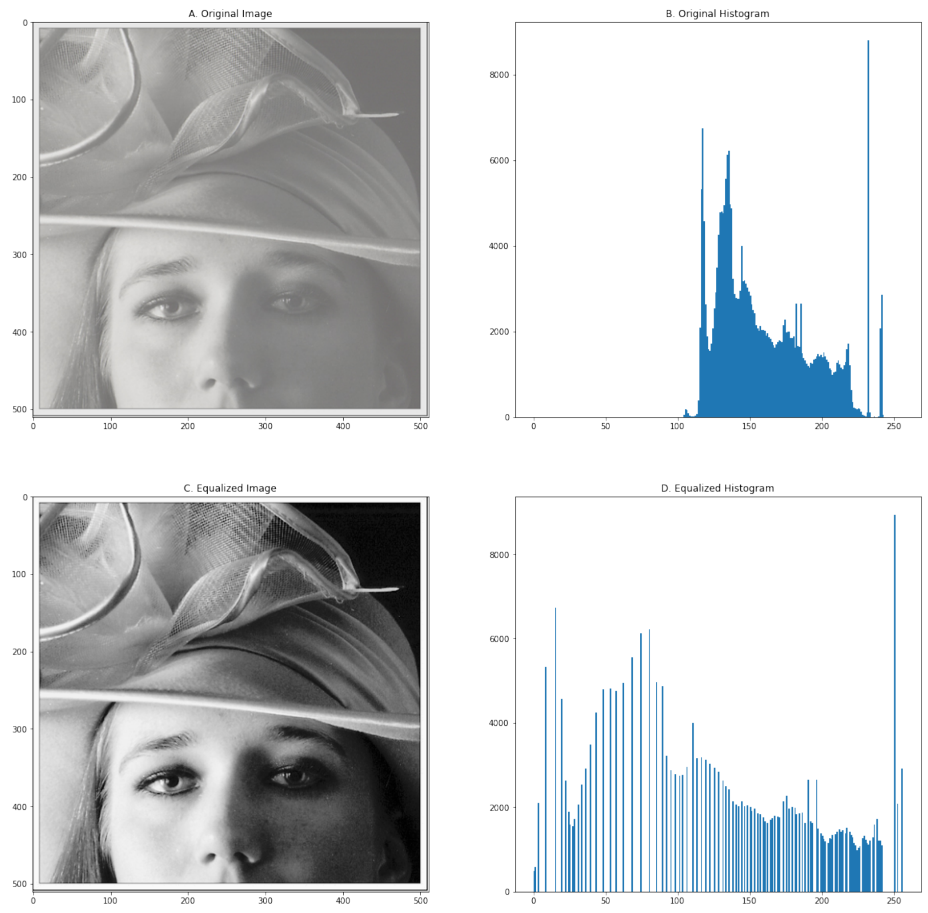

Figure 3: Histogram of Original and Output Image

As displayed in the figure 3(B) above, the majority of the pixel intensity ranges between 110 and 235, peaking around at 240. However, you can also see that the far left and right areas do not have any pixel intensity values. This reveals that our test image has poor contrast.

we utilized OpenCV-Python's .equalizeHist() method to spreads out the pixel intensity values. Unlike the original histogram, in equalized histogram fig.3(D) the pixel intensity values now range from 0 to 250 on the X-axis.

If we compare the two images above, we will find that the histogram equalized image has better contrast. It has areas that are darker as well as brighter than the original image. Another example,

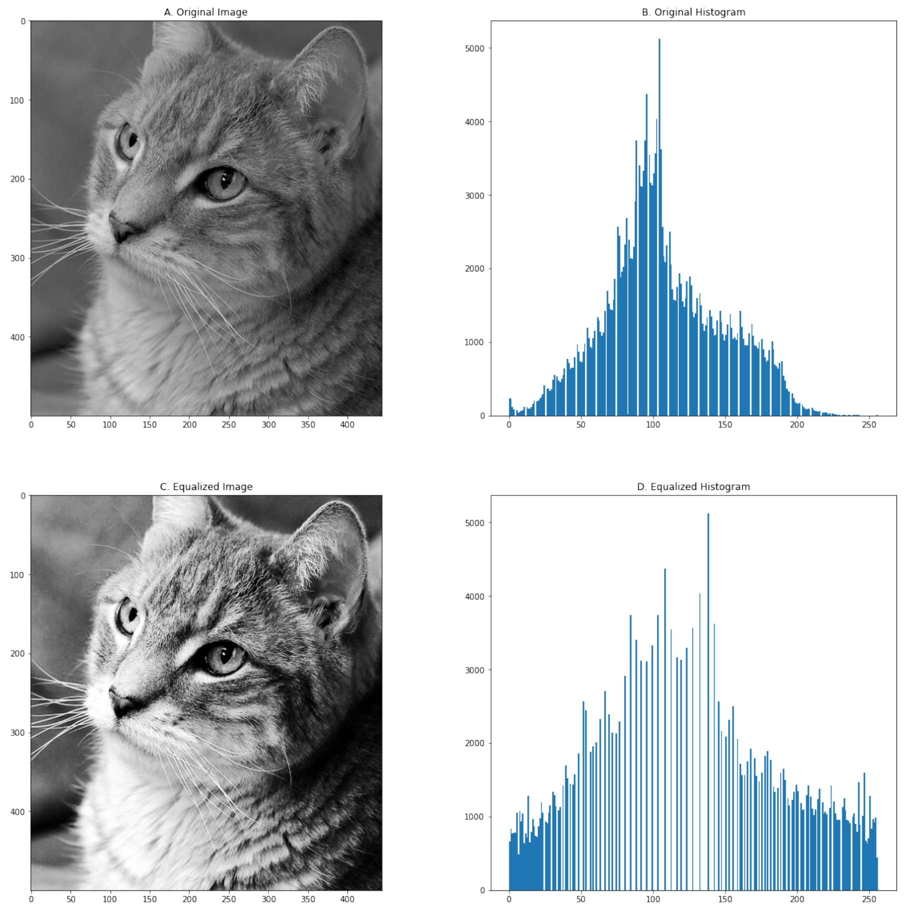

Figure 4: Histogram of Original and Output Image

Histogram Equalization can also balance high contrast image. An Example,

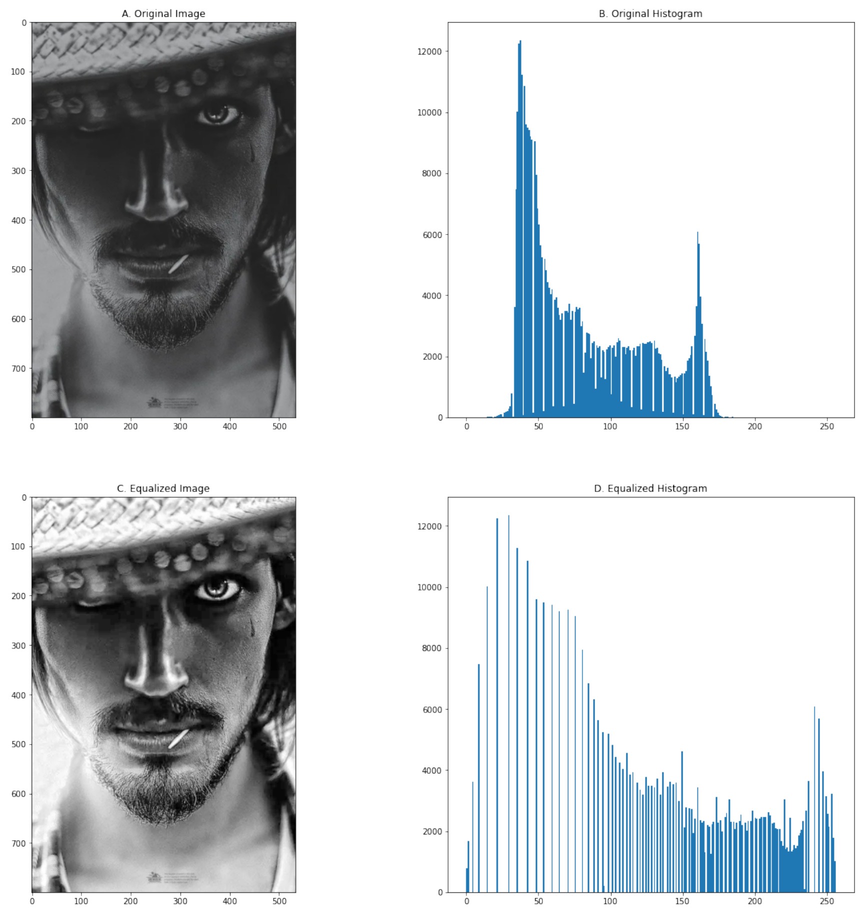

Figure 5: Histogram of Original and Output Image