Assignment 4

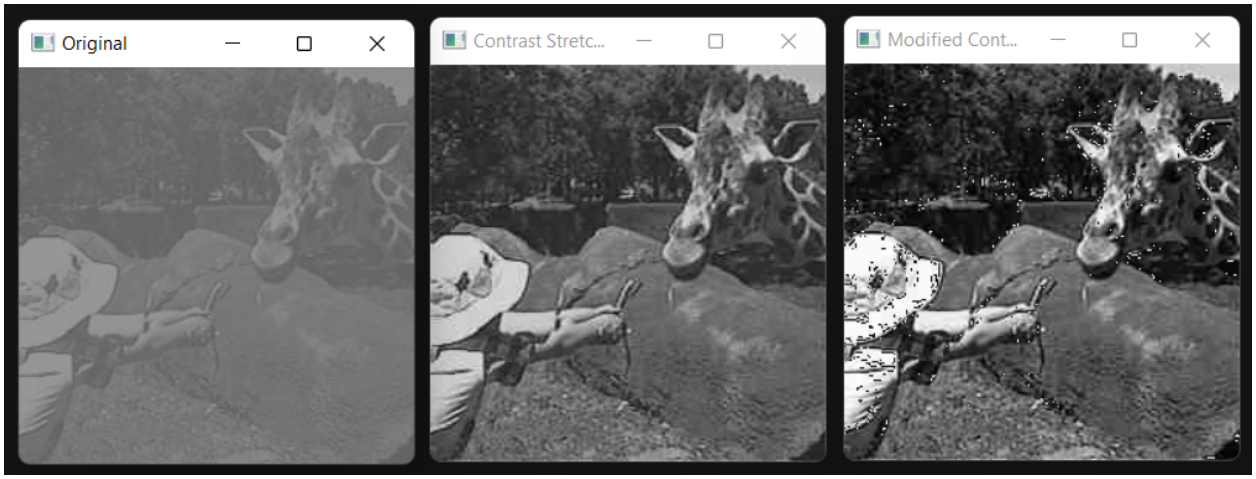
1. **Create python functions for contrast stretching and modified contrast stretching respectively**
2. **Apply your functions with an image and observe change of the image histogram**

The code below shows the contrast stretching and modified contrast stretching methods (lower at 1% and upper at 99%). Moreover, I provide you with histograms of both processings.

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

img = cv.imread('giraffe.jpg',0)
cv.imshow("Original",img)

row = img.shape[0]
col = img.shape[1]
# zeros array stores the stretched image
cont = np.zeros((row,col),dtype = 'uint8')
minimg = np.min(img)
maximg = np.max(img)

# contrast stretching
for i in range(row):
    for j in range(col):
        cont[i,j] = 255*(img[i,j]-minimg)/(maximg-minimg)

# Modified contrast stretching
# percentile at 1% and 99%
mod_cont = np.zeros((row,col),dtype = 'uint8')
minimg = np.percentile(img,1)
maximg = np.percentile(img,99)
for i in range(row):
    for j in range(col):
        mod_cont[i,j] = 255*(img[i,j]-minimg)/(maximg-minimg)

```

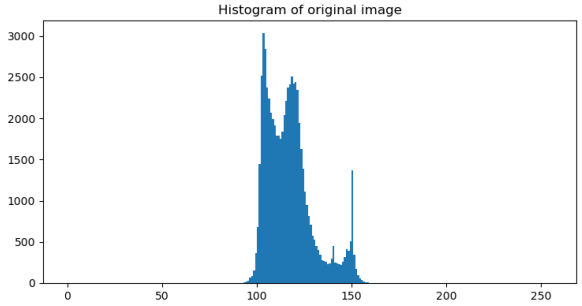
```
29
30  # Displat the stretched image
31  cv.imshow('Contrast Stretching',cont)
32  cv.imshow('Modified Contrast Stretching',mod_cont)
33  cv.waitKey(0)
34
35  plt.subplot(221)
36  plt.hist(img.ravel(),256,[0,256])
37  plt.title("Histogram of original image")
38
39  plt.subplot(222)
40  plt.hist(cont.ravel(),256,[0,256])
41  plt.title("Histogram from contrast stretching")
42
43  plt.subplot(223)
44  plt.hist(mod_cont.ravel(),256,[0,256])
45  plt.title("Histogram from modified contrast stretching")
46
47  plt.subplot(224)
48  plt.hist(img.ravel(),256,[0,256],color="g")
49  plt.hist(cont.ravel(),256,[0,256],color="black")
50  plt.hist(mod_cont.ravel(),256,[0,256],color="r")
51  plt.title("Comparision before and after")
52  plt.legend(["original",'contrast','modified'])
53
54  plt.show()
```

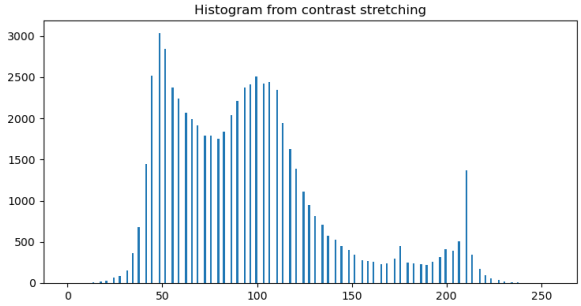
Image results:

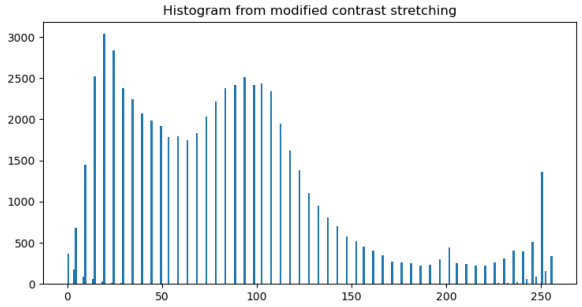


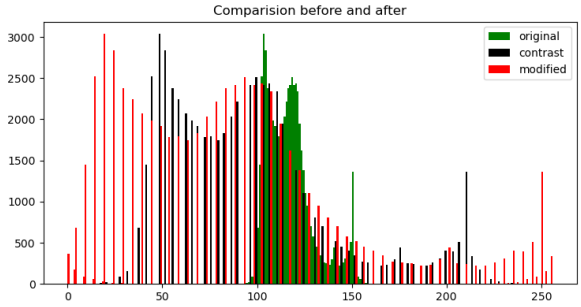
Savinee Khraithong EGBI 6213467

Histogram results:







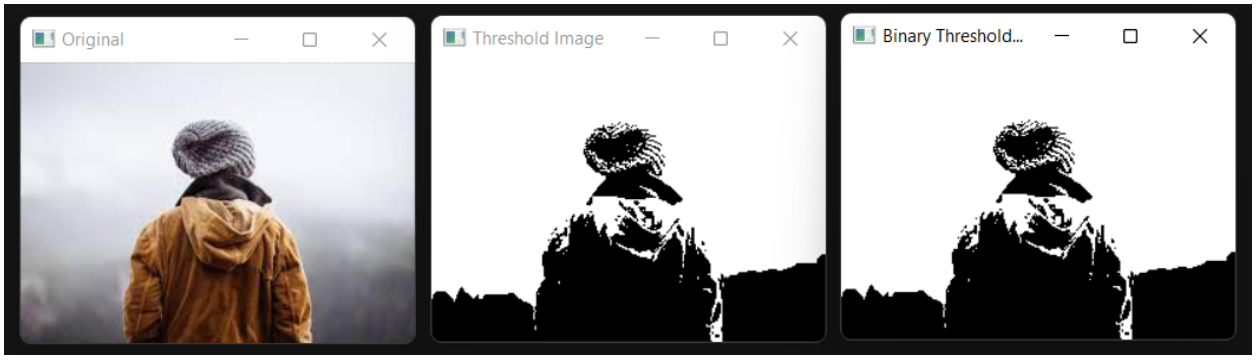


3. Create python function for thresholding an image and apply to an image

The code shows the way to define the threshold by coding as conditions and using a library from openCV.

```python
import cv2 as cv
import numpy as np

img = cv.imread('images.jpg')
cv.imshow("Original",img)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

threshold = 132
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        if img[i,j] > threshold:
            img[i,j] = 255
        else:
            img[i,j] = 0

cv.imshow('Threshold Image',img)

ret, thresh1 = cv.threshold(img, threshold, 255, cv.THRESH_BINARY)
cv.imshow('Binary Threshold from openCV', thresh1)

cv.waitKey(0)
```

Image results:



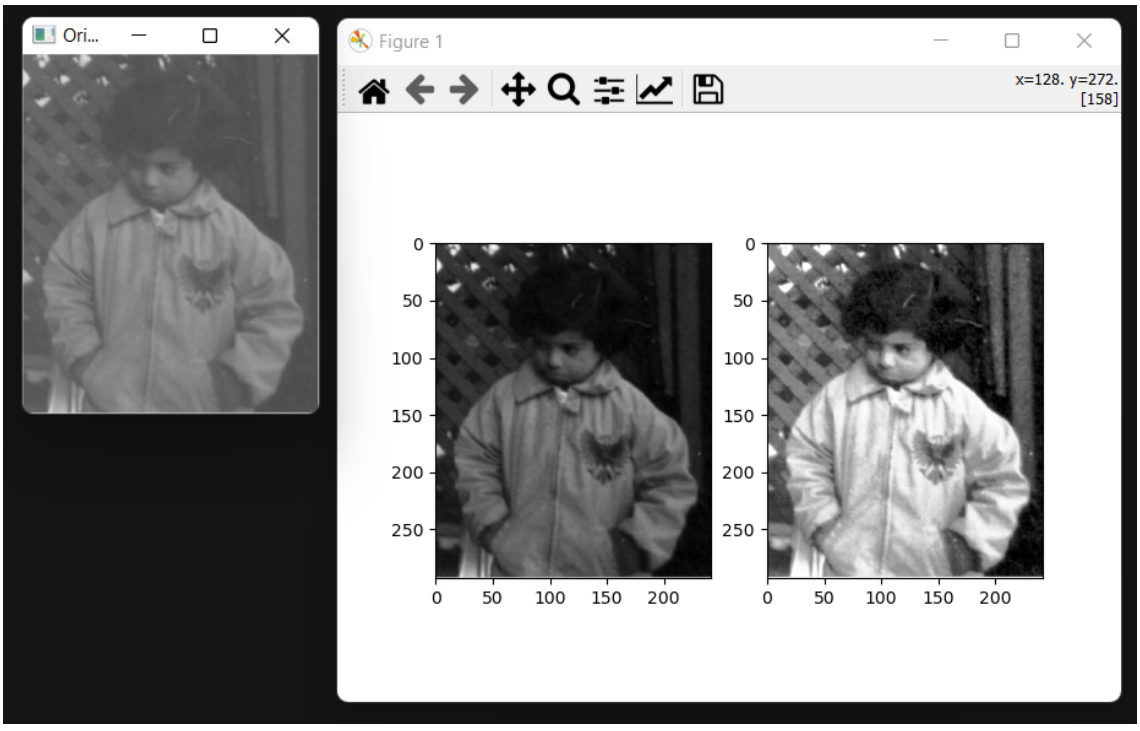
4. Write python function for histogram equalization and apply to an image

Coding the histogram equalization by using cumulative function and then, comparing the results. Moreover, there is a library from openCV which can be used as well.

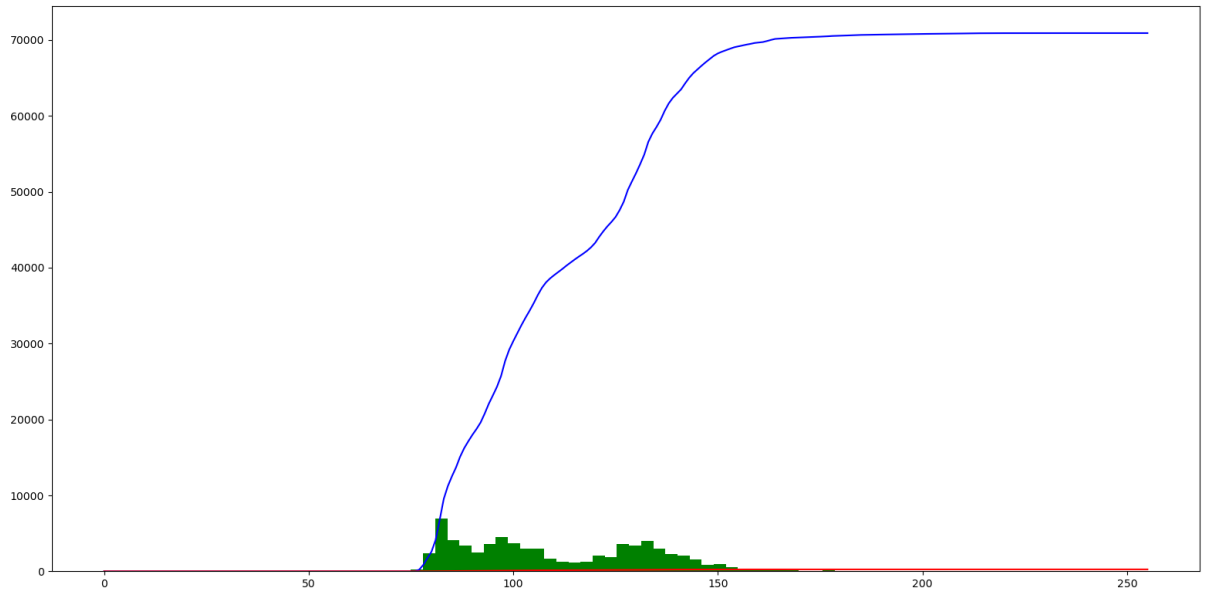
```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img = cv.imread('boy.png',0)
cv.imshow("Original",img)

# pixels in 1D array
flat = img.flatten()
plt.hist(flat, bins=50,color="g")

def histo(image_array, bins):
    histo = np.zeros(bins)
    for i in image_array:
        histo[i] += 1
    return histo

hist = histo(flat, 256)

# cumulative function
def cumu(var):
    cumvar = np.zeros(var.shape)
    cumvar[0] = var[0]
    for i in range(1,256):
        cumvar[i] = cumvar[i-1] + var[i]
    return np.array(cumvar)

cumu_sum = cumu(hist)
plt.plot(cumu_sum,color="b")
```

```
31    # normalization to the image
32    data = (cumu_sum - cumu_sum.min()) * 255
33    n = cumu_sum.max() - cumu_sum.min()
34    cumu_sum = (data / n).astype('uint8')
35    plt.plot(cumu_sum,"r")
36
37    # apply cumulative to each index
38    new_img = cumu_sum[flat]
39    new_img = np.reshape(new_img, img.shape)
40
41
42    plt.subplot(121)
43    plt.imshow(img, cmap='gray')
44
45    plt.subplot(122)
46    plt.imshow(new_img, cmap='gray')
47
48    plt.show()
```

Image results: due to the matplotlib which cannot plot the exact grayscale, the left image in Figure1 is quite different from the real original image. The right image is new image after doing equalization



Histogram result: blue line is cumulative graph of image, green line is image's histogram, and red line is normalization



OpenCV coding:

```python
import cv2 as cv
import numpy as np

img = cv.imread('boy.png',0)
equ = cv.equalizeHist(img)
equ_img = np.hstack((img,equ)) #stacking images side-by-side
cv.imwrite('equ.png ',equ_img )
```

Image results: The right image is the result

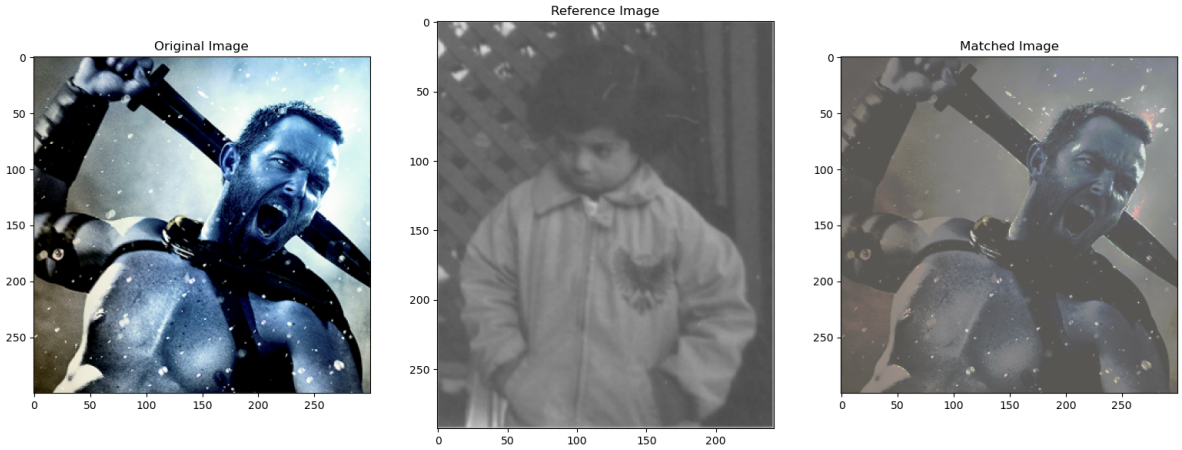
5. Study histogram specification and develop a function, test your function by applying an image and a targeted image

Developed coding of histogram specification as below, then plotting the histogram to compare the result of matching the image.

```python
import matplotlib.pyplot as plt
from skimage import exposure
from skimage.exposure import match_histograms
import cv2 as cv

# main image
img = cv.imread("300.png")

# reference image
ref = cv.imread("boy.png")

matched = match_histograms(img, ref, multichannel=True)

plt.subplot(131)
plt.title("Original Image")
plt.imshow(img)

plt.subplot(132)
plt.title("Reference Image")
plt.imshow(ref)

plt.subplot(133)
plt.title("Matched Image")
plt.imshow(matched)

plt.show()
```

```python
for i, img in enumerate((img, ref, matched)):
    img_hist, bins = exposure.histogram(img[..., i])
    plt.subplot(131)
    plt.plot(bins, img_hist / img_hist.max())
    plt.legend(["original","reference","matched"])
    plt.title("Histogram")

    img_cdf, bins = exposure.cumulative_distribution(img[..., i])
    plt.subplot(132)
    plt.plot(bins, img_cdf)
    plt.legend(["original","reference","matched"])
    plt.title("Cumulative graph")


plt.show()

```

Image results:



Histogram results: