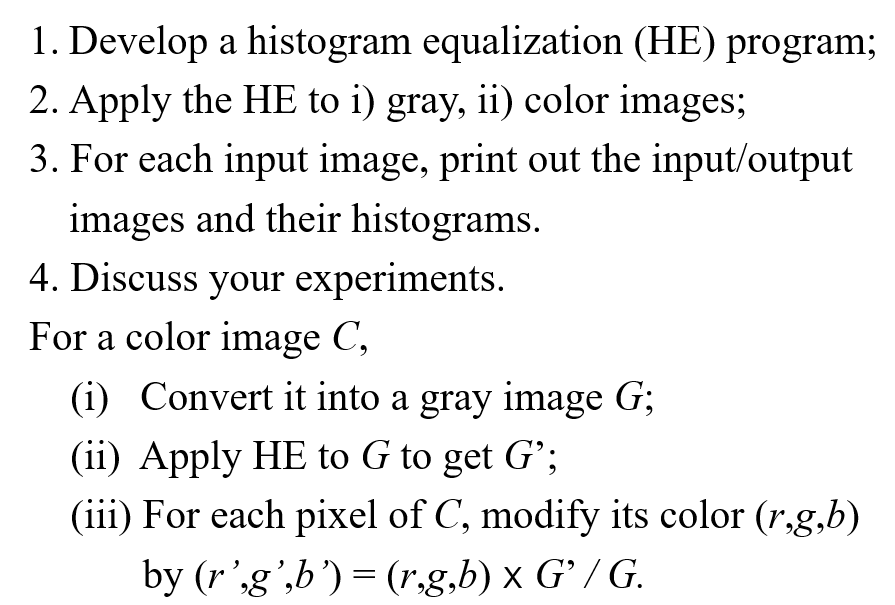
**40847041S朱自宇 hw3**

**Problem statement:**



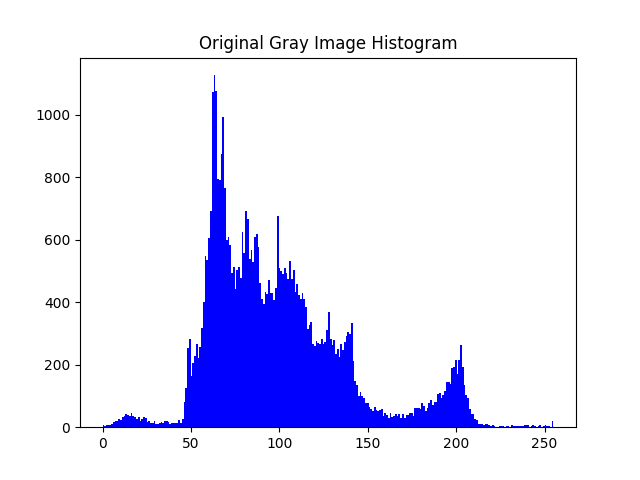
**Experimental results:**

* **Input/Output images:**

**Input grayscale image:**

****

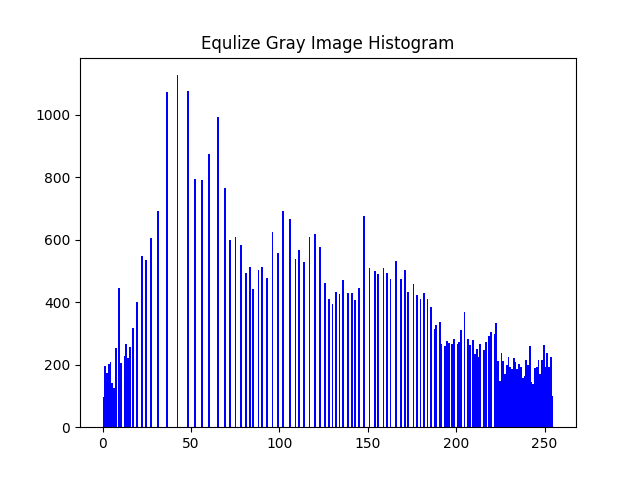
**Histogram of the image:**



**Output HE grayscale image:**



**Output HE grayscale image histogram:**



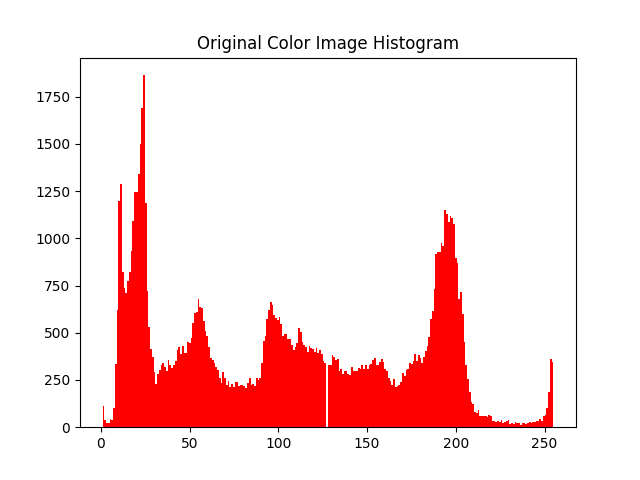
**Input color image:**

****

**Turn the color image to gray image:**

****

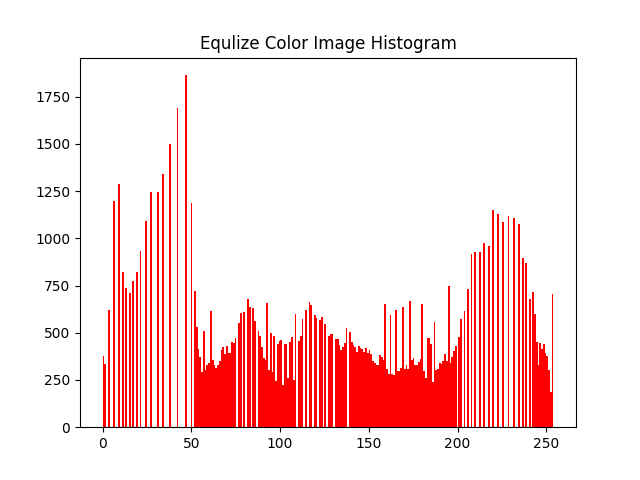
**Histogram of the gray color image:**



**Output HE gray color image:**

****

**Output HE gray color image histogram:**



**Output HE color image:**

****

* **Source code:**

"""

110-2 NTNU CSIE

ImageProcessing

Assignment3

by 40847041S 朱自宇

"""

import numpy as np

import cv2

import matplotlib.pyplot as plt

# 1. Develop a HE program

# read the gray image

grayimg = cv2.imread("Ukraine.png", cv2.IMREAD\_GRAYSCALE)

# read the color image and convert it into gray image

colorimg = cv2.imread("me.png")

graycolorimg = cv2.cvtColor(colorimg, cv2.COLOR\_BGR2GRAY)

# image size

(grayH, grayW)= grayimg.shape

(colorH, colorW, colorS) = colorimg.shape

# resize the color image, since I think it's too big :(

colorimg = cv2.resize(colorimg, (int(colorW/2), int(colorH/2)),interpolation=cv2.INTER\_AREA )

graycolorimg = cv2.resize(graycolorimg, (int(colorW/2), int(colorH/2)), interpolation=cv2.INTER\_AREA )

(colorH, colorW) = graycolorimg.shape

# show the original grya image

cv2.imshow("Input Gray image", grayimg)

# show the color image and its gray version

cv2.imshow("Input Color image", colorimg)

cv2.imshow("Gray Color image", graycolorimg)

"""

Histogram Equalization:

gray image

"""

# some variables

grayBase = np.unique(grayimg.flatten()) # grayscale value

grayCounter = [] # count of each value

Gcount = 0 # CDF, total value

# histogram equlization

for i in grayBase:

Gcount += np.sum(grayimg==i)\*255 / grayimg.size

grayCounter.append( Gcount )

for i in range(len(grayCounter)):

grayCounter[i] = round(grayCounter[i])

# make a new image of the HE result

grayoutput = np.zeros(grayimg.size,dtype=np.uint8)

count = 0

for i in grayimg.flatten():

grayoutput[count] = int(grayCounter[np.where(grayBase==i)[0][0]])

count += 1

# show the result

cv2.imshow("Output Gray image", grayoutput.reshape(-1,grayW))

"""

Histogram Equalization:

color image

"""

# some variables

colorBase = np.unique(graycolorimg.flatten()) # grayscale value

colorCounter = [] # count of each value

Ccount = 0 # CDF, total value

# histogram equalization

for j in colorBase:

Ccount += np.sum(graycolorimg==j)\*255 / graycolorimg.size

colorCounter.append( Ccount )

for j in colorCounter:

j = round(j)

# make a new image of the HE result

coloroutput = np.zeros(graycolorimg.size,dtype=np.uint8)

count = 0

for j in graycolorimg.flatten():

coloroutput[count] = int(colorCounter[np.where(colorBase==j)[0][0]])

count += 1

coloroutput = coloroutput.reshape(-1,colorW)

# turn the HE result back to color image

(B,G,R) = cv2.split(colorimg)

count = 0

for i in range(graycolorimg.shape[0]):

for j in range(graycolorimg.shape[1]):

if graycolorimg[i][j] == 0:

graycolorimg[i][j] = 1

# use int to calculate, avoiding overflow problems

graycolor = np.array(graycolorimg,dtype=np.int)

coloroutput2 = np.array(coloroutput,dtype=np.int)

b = np.array(B,dtype=np.int)

g = np.array(G,dtype=np.int)

r = np.array(R,dtype=np.int)

# do the calculate

for j in coloroutput2:

b[count] = b[count]\*j/graycolorimg[count]

g[count] = g[count]\*j/graycolorimg[count]

r[count] = r[count]\*j/graycolorimg[count]

count += 1

# limit the boundary

for i in range(b.shape[0]):

for j in range(b.shape[1]):

if b[i][j] > 255:

b[i][j] = 255

if g[i][j] > 255:

g[i][j] = 255

if r[i][j] > 255:

r[i][j] = 255

# make the output color image array

B = np.array(b,dtype=np.uint8)

G = np.array(g,dtype=np.uint8)

R = np.array(r,dtype=np.uint8)

coloroutput2 = cv2.merge([B,G,R])

# show the result

cv2.imshow("Output gray\_color image", coloroutput)

cv2.imshow("Output color image", coloroutput2 )

# show the histogram of the two works

plt.figure(1)

plt.title("Original Gray Image Histogram")

plt.hist(grayimg.flatten(),bins=256,color='blue')

plt.figure(2)

plt.title("Equlize Gray Image Histogram")

plt.hist(grayoutput,bins=256,color='blue')

plt.figure(3)

plt.title("Original Color Image Histogram")

plt.hist(graycolorimg.flatten(),bins=256,color='red')

plt.figure(4)

plt.title("Equlize Color Image Histogram")

plt.hist(coloroutput.flatten(),bins=256,color='red')

plt.show()

# wait until the user press any button to close the img window

cv2.waitKey(0)

cv2.destroyAllWindows()

* **Comments:**

I made a few mistakes, so it took longer than I expect ☹