Homework 4

(1) Problem statement

1. Develop a histogram equalization (HE) program;
2. Apply the HE to i) gray, ii) color images;
3. For each input image, print out the input/output images and their histograms.
4. Discuss your experiments.

For a color image C,

(i) Convert it into a gray image G;

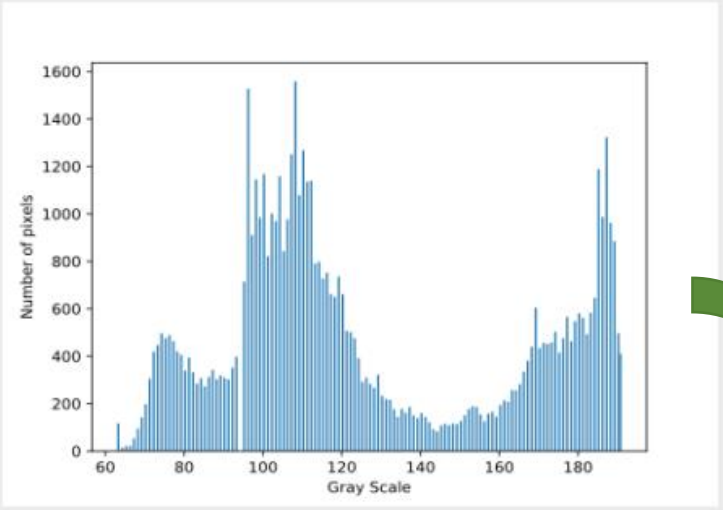
(ii) Apply HE to G to get G';

(iii) For each pixel of C, modify its color (r,g,b) by $(r',g',b') = (r,g,b) \times G' / G$.

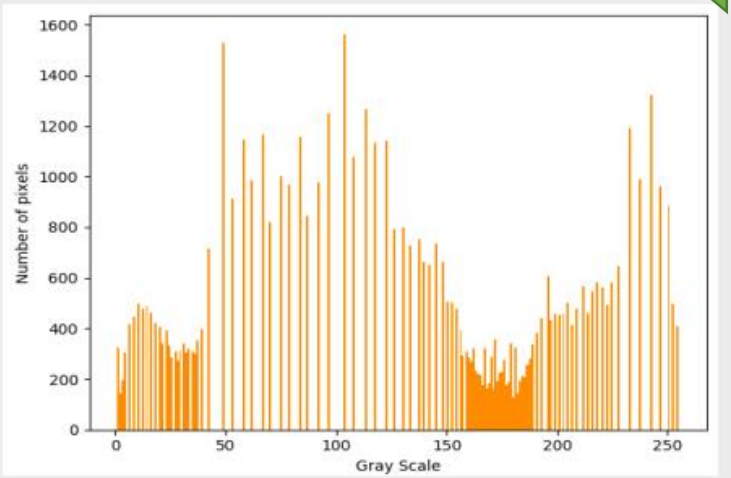
(2.1) Experimental results

(i) Histogram Equalization to gray images

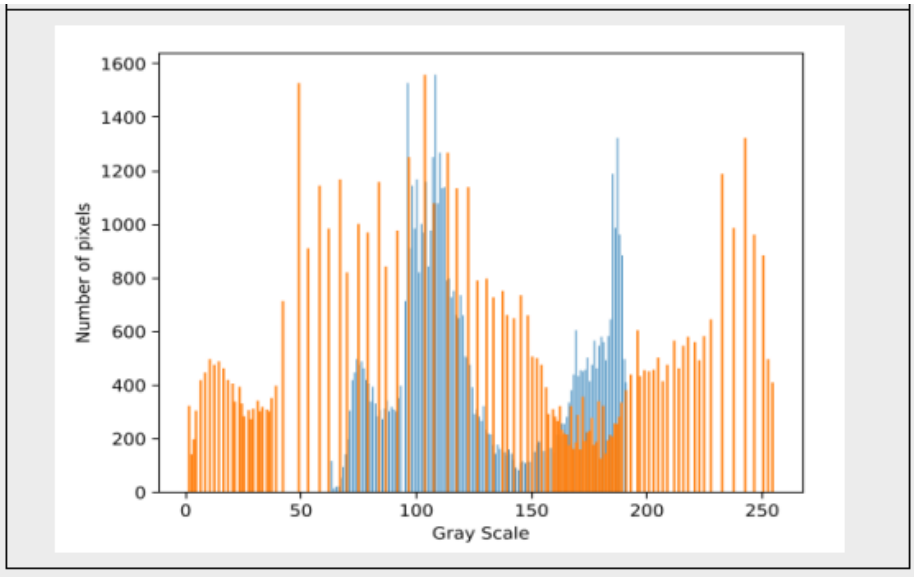
Input gray Image and histogram



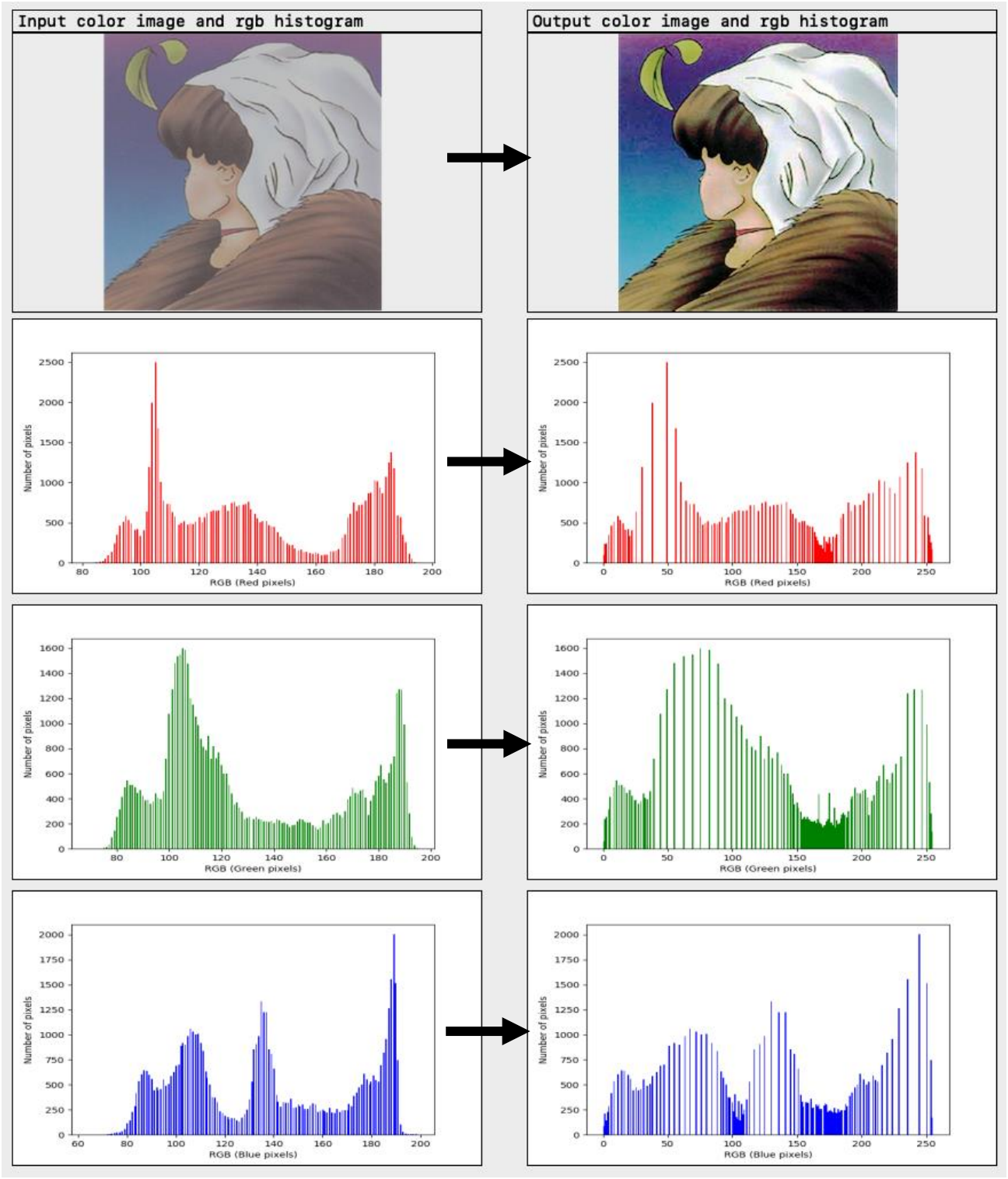
Output gray Image and histogram

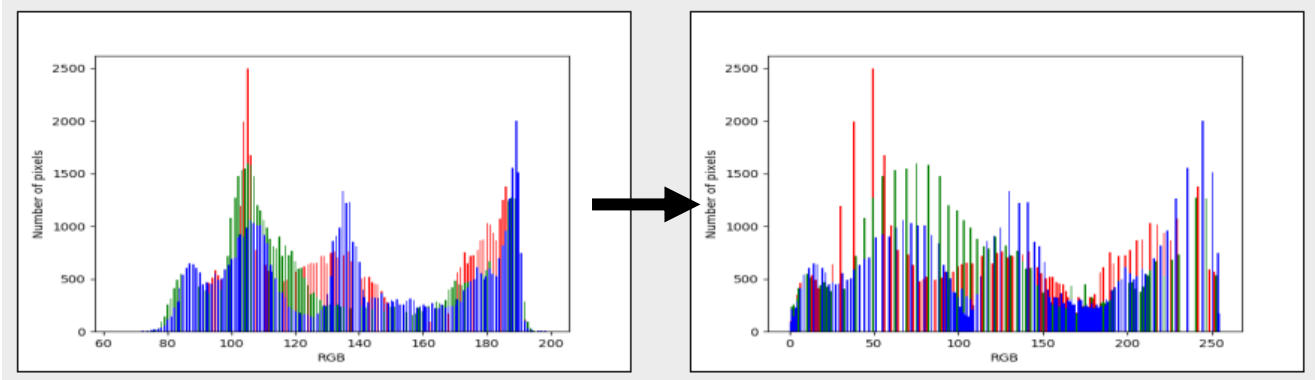


Comparison of poor contrast histogram with equalization histogram ：



(ii) Histogram Equalization to color images





(2.2) Source code

(i) Histogram Equalization to gray images

```python
# HW4-i (Implement Histogram Equalization program to gray image)

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

# Input a grayscale image
gray_img = Image.open('grayscale_low.png').convert("L")
L = 256  # 8 bits grayscale

# Define 8 bits grayscale value into an array
gray_scale = gray_img.histogram()   # nk : Number of pixels in each grayscale(n0~n255)

# Calculate total pixels
Width, Height = gray_img.size  # Size: 244 * 244
N = Width * Height             # N = Total pixels = 59536

# Save original gray_img histogram
a = np.array(gray_img)
plt.hist(a.ravel(), bins=L)  # ravel(): Return a contiguous flattened array , bins: number of pixels
plt.ylabel('Number of pixels')
plt.xlabel('Gray Scale')
plt.savefig('input_hist.svg')  # Save input_img histogram
plt.show()

# HISTOGRAM EQUALIZATION (HE) to gray
# Step1 : Calculate PMF and CDF
PMF = [0] * 256     # initial zero value in PMF[0] ~ PMF[256]
CDF = [0] * 256     # initial zero value in CDF[0] ~ CDF[256]
sum = 0             # initial cummulative_probability
for r in range(256):
    PMF[r] = gray_scale[r] / N      # Calculate Probability mass function(PMF)
    sum += PMF[r]
    CDF[r] = sum                    # Calculate Cumulative distribution function(CDF)

# Step2: s=T(r) , T(r): transformation function
for y in range(Height):
    for x in range(Width):
        ori_pixel = gray_img.getpixel((x, y))
        gray_img.putpixel((x, y), round((L-1) * CDF[ori_pixel]))

# Save output_img histogram
gray_img.save('output_img.png')
a = np.array(gray_img)
plt.hist(a.ravel(), bins=L, color='orange')
plt.ylabel('Number of pixels')
plt.xlabel('Gray Scale')
plt.savefig('output_hist.svg')  # Save output_img histogram
plt.show()

```

HE program

(ii) Histogram Equalization to color images

```python
# HW4-ii (Implement Histogram Equalization program to color image)

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

# Input a color image (RGB)
color_img = Image.open('color_low.png').convert('RGB')
L = 256

# Define RGB value into an array
color_scale = color_img.histogram()
color_scale_R = color_scale[0:256]      # nk : Number of pixels in each Red pixels(n0~n255)
color_scale_G = color_scale[256:513]    # nk : Number of pixels in each green pixels(n256~n512)
color_scale_B = color_scale[512:769]     # nk : Number of pixels in each blue pixels(n512~n768)

# Calculate total pixels
Width, Height = color_img.size  # Size: 244 * 244
N = Width * Height              # N = Total pixels = 59536

# Save original color_img histogram
a = np.array(color_img)

# Red Histogram
plt.hist(a[:, :, 0].ravel(), bins=256, color='red')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Red pixels)')
plt.savefig('input_red_hist.png')  # Save input_img histogram
plt.close()

# Green Histogram
plt.hist(a[:, :, 1].ravel(), bins=256, color='green')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Green pixels)')
plt.savefig('input_green_hist.png')  # Save input_img histogram
plt.close()

# Blue Histogram
plt.hist(a[:, :, 2].ravel(), bins=256, color='blue')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Blue pixels)')
plt.savefig('input_blue_hist.png')  # Save input_img histogram
plt.close()

# HISTOGRAM EQUALIZATION (HE) to color image
# initial zero value in PMF_RGB
PMF_R = [0] * 256
PMF_G = [0] * 256
PMF_B = [0] * 256
# initial zero value in CDF_RGB
CDF_R = [0] * 256
CDF_G = [0] * 256
CDF_B = [0] * 256
# initial cummulative_probability (RGB)
sum_R = 0
sum_G = 0
sum_B = 0

```

Show input image histogram

HE initial value

```python
# Step1 : Calculate PMF and CDF
for i in range(256):
    PMF_R[i] = color_scale_R[i] / N      # Calculate Probability mass function(PMF_R)
    sum_R += PMF_R[i]
    CDF_R[i] = sum_R                     # Calculate Cumulative distribution function(CDF_R)

for i in range(256):
    PMF_G[i] = color_scale_G[i] / N      # Calculate Probability mass function(PMF_G)
    sum_G += PMF_G[i]
    CDF_G[i] = sum_G                     # Calculate Cumulative distribution function(CDF_G)

for i in range(256):
    PMF_B[i] = color_scale_B[i] / N      # Calculate Probability mass function(PMF_R)
    sum_B += PMF_B[i]
    CDF_B[i] = sum_B                     # Calculate Cumulative distribution function(CDF_G)

# Step2: s=T(r) , T(r): transformation function
for y in range(Height):
    for x in range(Width):
        r, g, b = color_img.getpixel((x, y))
        color_img.putpixel((x, y), (round((L - 1) * CDF_R[r]), round((L - 1) * CDF_G[g]), round((L - 1) * CDF_B[b])))
```

" HE program "

```python
# Save output_img histogram
color_img.save('output_img.png')
a = np.array(color_img)

# Red Histogram
plt.hist(a[:, :, 0].ravel(), bins=256, color='red')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Red pixels)')
plt.savefig('output_red_hist.png')
plt.close()

# Green Histogram
plt.hist(a[:, :, 1].ravel(), bins=256, color='green')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Green pixels)')
plt.savefig('output_green_hist.png')
plt.close()

# Blue Histogram
plt.hist(a[:, :, 2].ravel(), bins=256, color='blue')
plt.ylabel('Number of pixels')
plt.xlabel('RGB (Blue pixels)')
plt.savefig('output_blue_hist.png')
plt.close()
```

Show output image histogram

(2.3) Comments

Input color image C

Convert into gray image G >>>

Image G (Gray image)

Apply HE to G to get G' >>>

Image G' (After equalization)