影像處理概論 HW1

題目：Histogram Equalization

409410095 王子奕

Data due: April 17, 2023

Data handed in:

hw1.py、Lena.bmp、Peppers.bmp and this pdf all in

409410095.zip

- Technical description:

(兩張 test images: “Lena.bmp” 、 “Pepper.bmp” 需與 hw1.py 存在同個資料夾下)。

1. 首先，將兩張 test images 分別讀入兩變數 img1、img2。(如下圖)

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img1 = cv2.imread('Lena.bmp', cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('Peppers.bmp', cv2.IMREAD_GRAYSCALE)
```

2. 將兩 test images 原圖存入輸出視窗 “Figure1” 內。(如下圖)

```
# 放原圖 in figure 1
plt.subplot(321), plt.imshow(img1, cmap='gray')
plt.title('Original Image 1'), plt.xticks([]), plt.yticks([])
plt.subplot(322), plt.imshow(img2, cmap='gray')
plt.title('Original Image 2'), plt.xticks([]), plt.yticks([])
```

3. 定義 histogram equalization 的函式後，對兩原圖進行 global histogram equalization 的處理，(如下圖)

```
# histogram equalization函式
def hist_eq(img):
    # 計算影像histogram
    hist, bins = np.histogram(img.flatten(), 256, [0,256])
    # 計算累積分佈函式
    cdf = hist.cumsum()
    # normalize累積分佈函式
    cdf_normalized = cdf * hist.max() / cdf.max()
    # 計算histogram equalize後的像素值
    img_eq = np.interp(img.flatten(), bins[:-1], cdf_normalized).astype(np.uint8)
    img_eq = img_eq.reshape(img.shape)
    return img_eq

# 對每個圖進行global histogram equalization
img1_eq = hist_eq(img1)
img2_eq = hist_eq(img2)
```

4. 將處理過後的圖也存入輸出視窗“Figure1”內。(如下圖)

```
# 放global histogram equalization的圖 in figure 1
plt.subplot(323), plt.imshow(img1_eq, cmap='gray')
plt.title('Global Histogram Equalization 1'), plt.xticks([]), plt.yticks([])
plt.subplot(324), plt.imshow(img2_eq, cmap='gray')
plt.title('Global Histogram Equalization 2'), plt.xticks([]), plt.yticks([])
```

5. 定義一個將影像分割成 16 個相等 blocks 的函式(如下圖)

```
# 影像分割函式
def divide_image(img):
    h, w = img.shape
    # 分成16個相等的blocks
    block_h = h // 4
    block_w = w // 4
    blocks = []
    for i in range(4):
        for j in range(4):
            block = img[i*block_h:(i+1)*block_h, j*block_w:(j+1)*block_w]
            blocks.append(block)
    return blocks
```

6. 定義 local histogram equalization 的函式(如下圖)

```
# local histogram equalization函式
def block_hist_eq(img_blocks):
    img_blocks_eq = []
    for block in img_blocks:
        block_eq = hist_eq(block)
        img_blocks_eq.append(block_eq)
    return img_blocks_eq
```

7. 對原圖進行 local histogram equalization 的處理，並將處過後的圖也存入輸出視窗“Figure1”內(如下圖)

```
# 對每個影像進行local histogram equalization
img1_blocks = divide_image(img1)
img2_blocks = divide_image(img2)
img1_blocks_eq = block_hist_eq(img1_blocks)
img2_blocks_eq = block_hist_eq(img2_blocks)

# 將16個blocks還原為原始影像大小
img1_local_eq = np.vstack([np.hstack(img1_blocks_eq[i:i+4]) for i in range(0,16,4)])
img2_local_eq = np.vstack([np.hstack(img2_blocks_eq[i:i+4]) for i in range(0,16,4)])

# 放local histogram equalization的影像 in figure 1
plt.subplot(325), plt.imshow(img1_local_eq, cmap='gray')
plt.title('Local Histogram Equalization 1'), plt.xticks([]), plt.yticks([])
plt.subplot(326), plt.imshow(img2_local_eq, cmap='gray')
plt.title('Local Histogram Equalization 2'), plt.xticks([]), plt.yticks([])
```

8. 將兩原圖以及他們的 histogram 存入輸出視窗 “Figure2” 內（如下圖）

```
# 放原圖的histogram in figure 2
plt.figure(figsize=(10, 5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspace=0.3, hspace=0.3)
plt.subplot(221), plt.imshow(img1, cmap='gray')
plt.title('Original Image 1'), plt.xticks([]), plt.yticks([])
plt.subplot(222), plt.hist(img1.flatten(), 256, [0,256], color='r')
plt.title('Histogram 1'), plt.xlim([0,256])
plt.subplot(223), plt.imshow(img2, cmap='gray')
plt.title('Original Image 2'), plt.xticks([]), plt.yticks([])
plt.subplot(224), plt.hist(img2.flatten(), 256, [0,256], color='r')
plt.title('Histogram 2'), plt.xlim([0,256])
plt.subplots_adjust(bottom=0.05)
```

9. 將兩進行過 global histogram equalization 處理的圖以及他們的 histogram 存入輸出視窗 “Figure3” 內(如下圖)

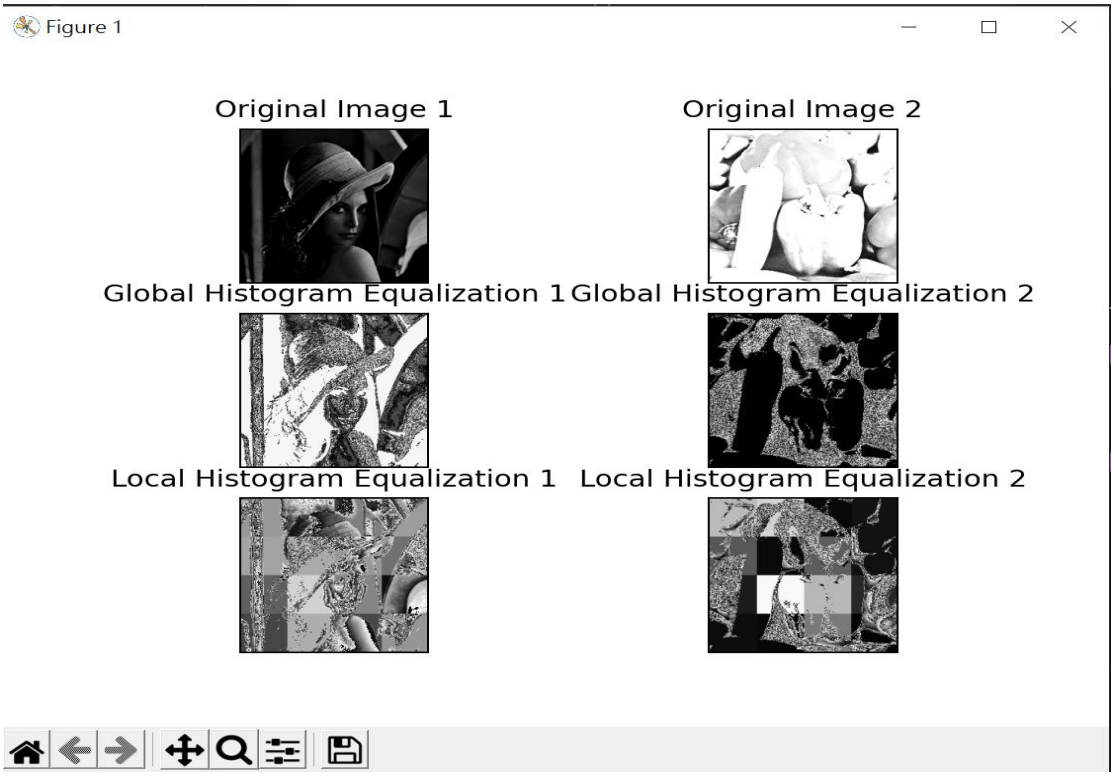
```
# 放global approach後的圖的histogram in figure 3
plt.figure(figsize=(10, 5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspace=0.3, hspace=0.3)
plt.subplot(221), plt.imshow(img1_eq, cmap='gray')
plt.title('Global Histogram Equalization 1'), plt.xticks([]), plt.yticks([])
plt.subplot(222), plt.hist(img1_eq.flatten(), 256, [0,256], color='r')
plt.title('Histogram 1'), plt.xlim([0,256])
plt.subplot(223), plt.imshow(img2_eq, cmap='gray')
plt.title('Global Histogram Equalization 2'), plt.xticks([]), plt.yticks([])
plt.subplot(224), plt.hist(img2_eq.flatten(), 256, [0,256], color='r')
plt.title('Histogram 2'), plt.xlim([0,256])
plt.subplots_adjust(bottom=0.05)
```

10. 將兩進行過 local histogram equalization 處理的圖以及他們的 histogram 存入輸出視窗 “Figure4” 內，並輸出所有輸出視窗(Figure1~Figure4)(如下圖)

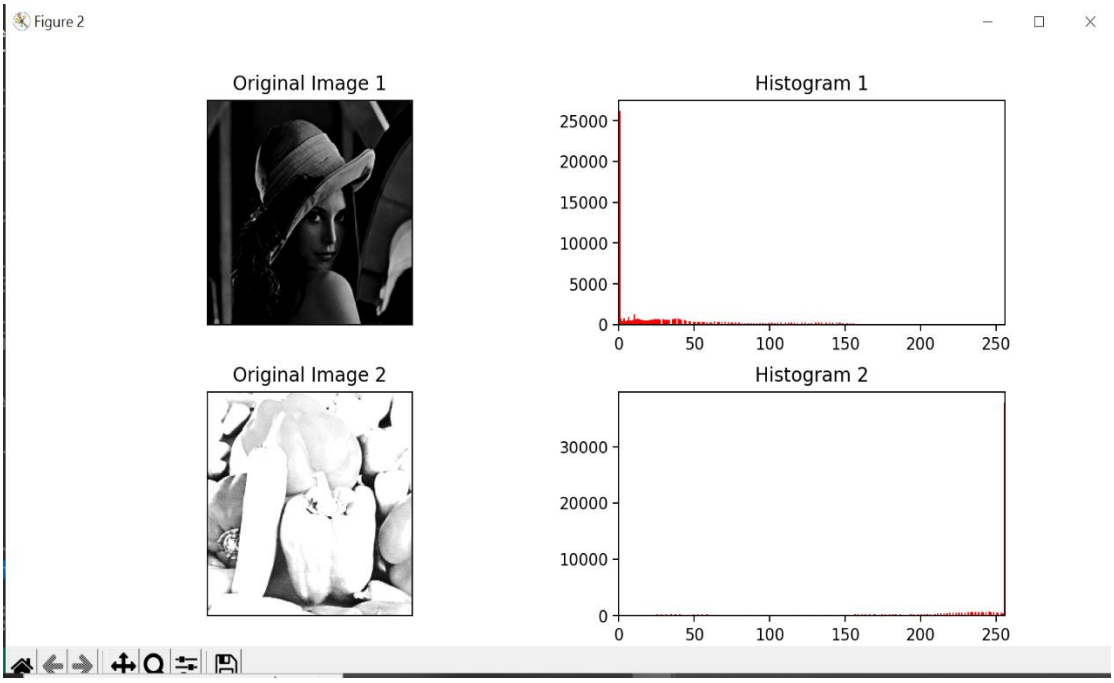
```
# 放local approach後的圖的histogram in figure 4
plt.figure(figsize=(10, 5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspace=0.3, hspace=0.3)
plt.subplot(221), plt.imshow(img1_local_eq, cmap='gray')
plt.title('Local Histogram Equalization 1'), plt.xticks([]), plt.yticks([])
plt.subplot(222), plt.hist(img1_local_eq.flatten(), 256, [0,256], color='r')
plt.title('Histogram 1'), plt.xlim([0,256])
plt.subplot(223), plt.imshow(img2_local_eq, cmap='gray')
plt.title('Local Histogram Equalization 2'), plt.xticks([]), plt.yticks([])
plt.subplot(224), plt.hist(img2_local_eq.flatten(), 256, [0,256], color='r')
plt.title('Histogram 2'), plt.xlim([0,256])
plt.subplots_adjust(bottom=0.05)
# 顯示所有輸出結果(所有Figure)
plt.show()
```

- Experimental results:

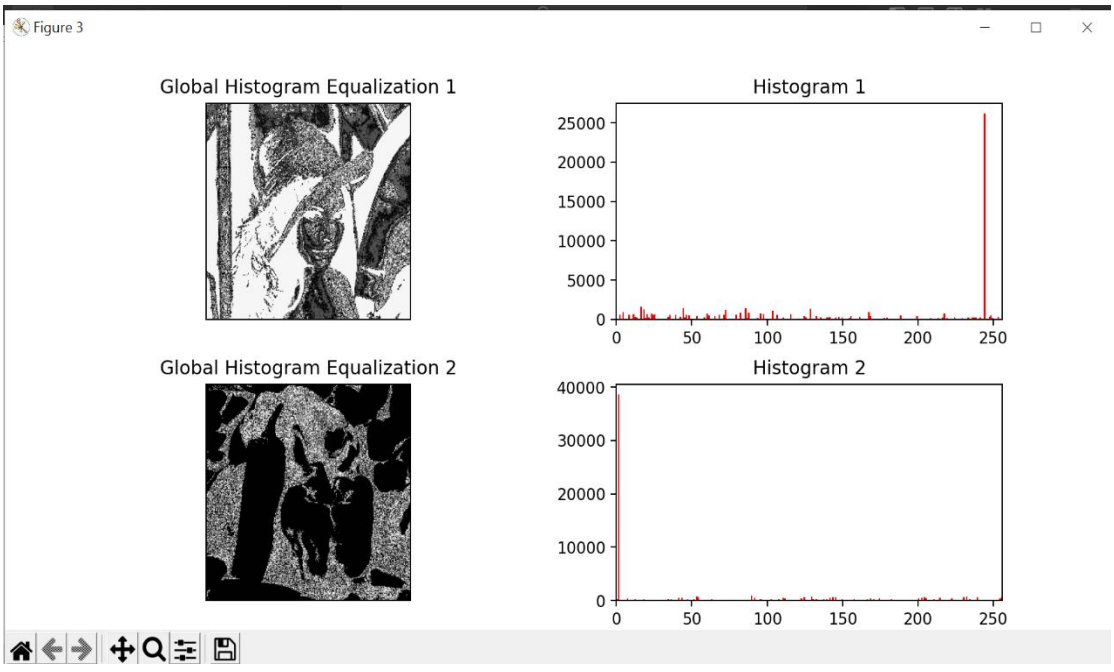
1. 該程式的執行結果為四個輸出視窗(Figure1~Figure4)，Figure1 存放原圖以及做過處理的所有圖。(如下圖)



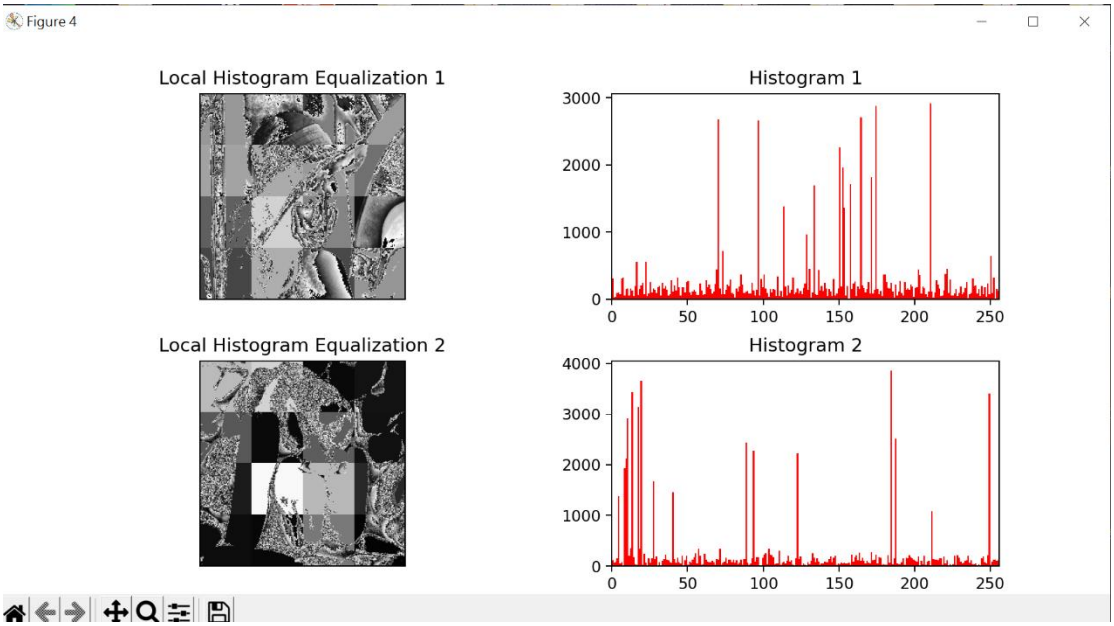
2. Figure2 存放原圖以及原圖的 histogram。(如下圖)



3.Figure3 存放 global histogram equalization 處理的圖以及處理後圖的 histogram (如下圖)



4.Figure4 存放 local histogram equalization 處理的圖以及處理後圖的 histogram（如下圖）



- Discussions:

global histogram equalization 是一種直接應用於整個圖像的方法。在這種方法中，整個圖像的 histogram 被平均分配到整個像素值範圍內，從而增強圖像的對比度。這種方法非常簡單，可以快速實現，但它有一些缺點。當圖像中存在非常明亮或非常暗的區域時，global histogram equalization 可能會產生過度增強對比度的效果，從而導致圖像細節的丟失。

local histogram equalization 是一種更加複雜的方法，可以解決 global histogram equalization 中存在的一些問題。在這種方法中，圖像被分成若干個子圖像，每個子圖像的

histogram 被 equalized，從而增強局部區域的對比度。這種方法可以在保留圖像細節的同時增強對比度。global histogram equalization 的 histogram 平均分配在整個像素值範圍內，導致 histogram 上的像素分布較為均勻。而 local histogram equalization 的 histogram 則分布在不同的區域內，導致 histogram 上出現多個峰值。

- References and Appendix:

--references:

1. https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html

2. https://python-course.eu/numerical-programming/image-processing-techniques-with-python-and-matplotlib.php

3. https://www.youtube.com/watch?v=uBWpt9IkPFA

4. https://www.sciencedirect.com/science/article/abs/pii/S1077314298907238

5. https://gist.github.com/cindylin190011/218907ed7f59a4e8ddd2a4058ed64996