多媒體系統與應用

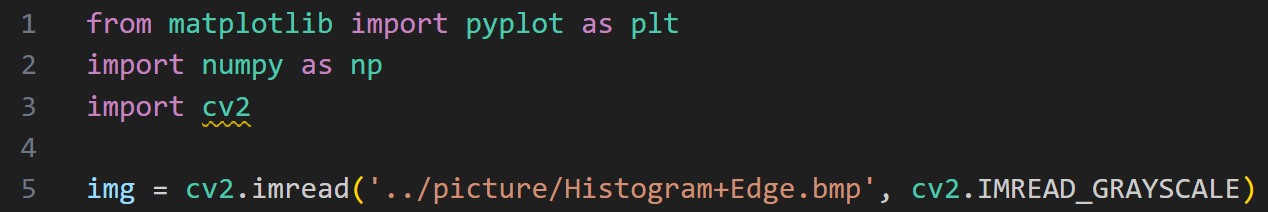
Project3-Histogram Equalization and Edge Detection

F74094017 李昆翰

一、程式環境和輸入影像預處理:

我於本次作業的運行環境選用的是 python 的 pipenv 中，且關於輸入的圖片是用 OpenCV 將它以灰階圖的方式做讀取。

具體使用的套件和圖片的引入方式為下面截圖所示:



二、Histogram equalization:

1. code:

def histogram\_equalize(img):

# make the histogram of original image by numpy

hist, bins = np.histogram(img.flatten(), 256, [0, 256])

# calculate the cumulative sum

cdf = hist.cumsum()

# normalize the cdf

cdf = cdf \* 255 / cdf[-1]

# then, equalize the image by the `cdf`` we get

equalized\_img = np.interp(img.flatten(), bins[:-1],

cdf).reshape(img.shape)

# make the histogram of equalized image by numpy

equalized\_hist, \_ = np.histogram(equalized\_img.flatten(), 256, [0,

256])

return hist, equalized\_hist, equalized\_img

hist, equalized\_hist, equalized\_img = histogram\_equalize(img)

cv2.imwrite('../figure/histogram\_equalization.png', equalized\_img)

plt.figure(figsize=(10, 7))

plt.plot(hist, color='black')

plt.title("Original Histogram")

plt.savefig('../figure/Original\_Histogram.png')

plt.figure(figsize=(10, 7))

plt.plot(equalized\_hist, color='black')

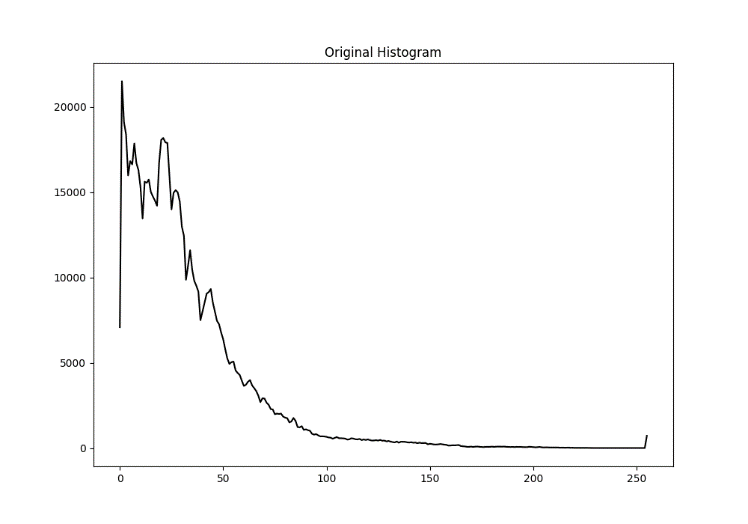
plt.title("Equalized Histogram")

plt.savefig('../figure/Equalized\_Histogram.png')

2. 說明:

基本上，這個 section 幾乎都是靠使用 numpy 套件所完成。

首先，我們會先將輸入進來的灰階圖，使用 np.histogram() 函式，來得到未經過 equalization 的 original histogram，以及該灰階圖的 bins，以計算 cumulative sums。該函式得到的 original histogram 為以下:

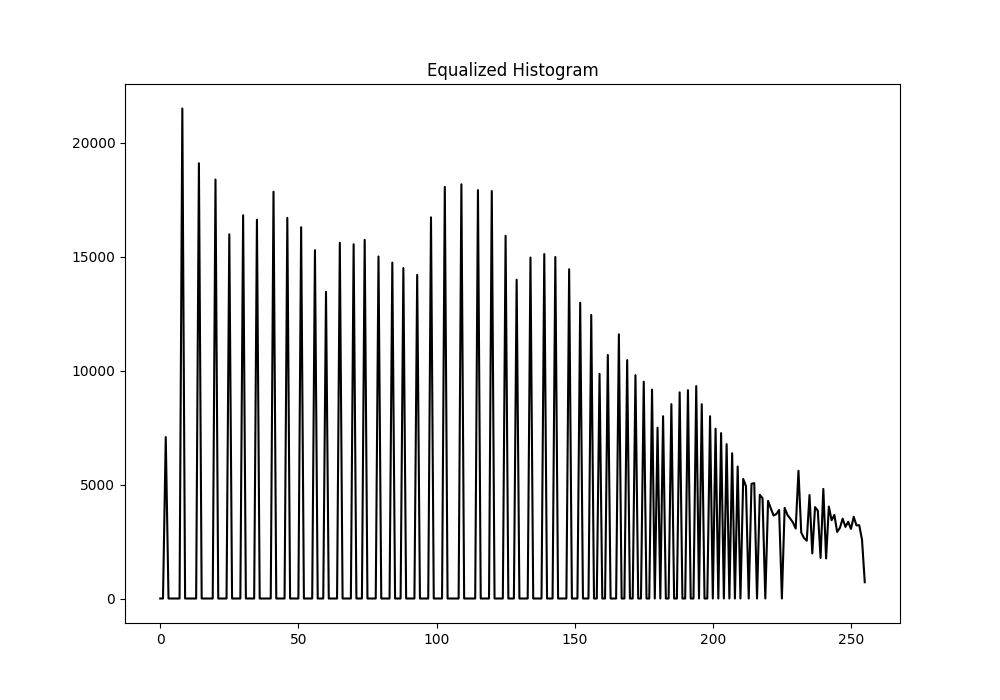


下一步，我們將得到的 original histogram 使用 np.cumsum() 得到 cumulative sum。之後，將整條 cumulative sum 針對灰階像素數和最大的 cumulative sum 做 normalize，得到 cumulative distribution function (cdf)。

最後，使用 np.intern()，將我們於上一部得到的 cdf 以線性插值的方式，把 original histogram 做 equalization，並將其 reshape 為灰階圖的陣列形式。因此，我們可得到 equalized image 為以下:



以及 equalized histogram 為以下:



三、Edge detection:

1. code:

def sobel\_operation(img):

# Init. the gx and gy edge (both are 2D-array)

gx = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

gy = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

# Init. the df(x,y)/dx and df(x,y)/dy for sobel operator

x, y = img.shape

sx = np.zeros((x, y))

sy = np.zeros((x, y))

# calculate the integral

# similar to the equation in P.59 of the ppt

for r in range(x):

for c in range(y):

for s in range(3):

for t in range(3):

if r + s < x and c + t < y: # prevent out of bound

sx[r][c] += gx[s][t] \* img[r + s][c + t]

sy[r][c] += gy[s][t] \* img[r + s][c + t]

# G = |df(x,y)/dx| + |df(x,y)/dy|

for r in range(x):

for c in range(y):

sx[r][c] = abs(sx[r][c])

sy[r][c] = abs(sy[r][c])

return sx + sy

sobel = sobel\_operation(img)

cv2.imwrite('../figure/sobel\_operator.png', sobel)

2. 說明:

在本次作業中的 edge detection 中，我只有實現了 sobel operator 的方式。

在一開始，我使用課程 ppt 裡面的方式，先初始化 gx 和 gy 兩個 edge 陣列。並且，我也依照圖片的二維像素數，先初始化了儲存 sobel operation integer 結果的兩個二維陣列。

之後，按照 gx 和 gy 的 3x3 陣列對輸入進來的圖片做掃描，其中也做了邊界檢查防止指向的陣列位置超出範圍。

最後，我將每一個經過處理後的像素點做絕對值，並回傳 sx 和 sy 相加的二維陣列作為 sobel operator 的邊界掃描結果。最終結果為下圖所示:

