電子總整_HW1_B11002220

OpenCV 直方圖等化

程式碼與註解

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

# 載入圖片
image = cv2.imread("hw1.jpeg")
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 顯示原圖片 & 直方圖
plt.figure()
plt.title("Before equalizeHist by opencv ")
plt.imshow(gray_image, cmap='gray')
plt.show()
hist = cv2.calcHist([gray_image], [0], None, [256], [0, 256])
hist.flatten()#扁平化 為了之後能直接顯示
plt.figure()
plt.title("hist by opencv ")
plt.xlabel("Bins")
plt.ylabel("numbers of piexls ")
plt.plot(hist)#畫圖
plt.show()#顯示

# 轉換成等化直方圖
eqHist = cv2.equalizeHist(gray_image)
# 顯示轉換後 & 直方圖
plt.figure()
plt.title("After equalizeHist by opencv")
plt.imshow(eqHist, cmap='gray')# cmap表示要顯示灰階 如果不做會被認為是RGB的輸入
plt.show()

hist = cv2.calcHist([eqHist], [0], None, [256], [0, 256])
hist.flatten()# 二維轉一維才能顯示
# print(hist)
plt.figure()
plt.title("hist by opencv")
plt.xlabel("Bins")
plt.ylabel("numbers of piexls ")
```

```
plt.plot(hist)
plt.show()
```

手刻法直方圖等化

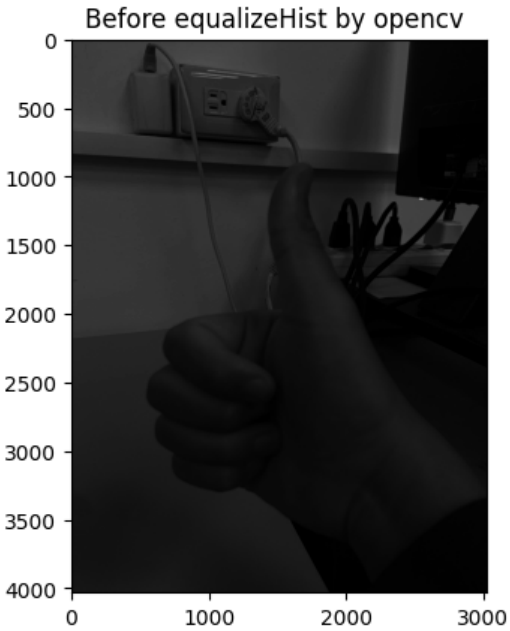
程式碼與註解

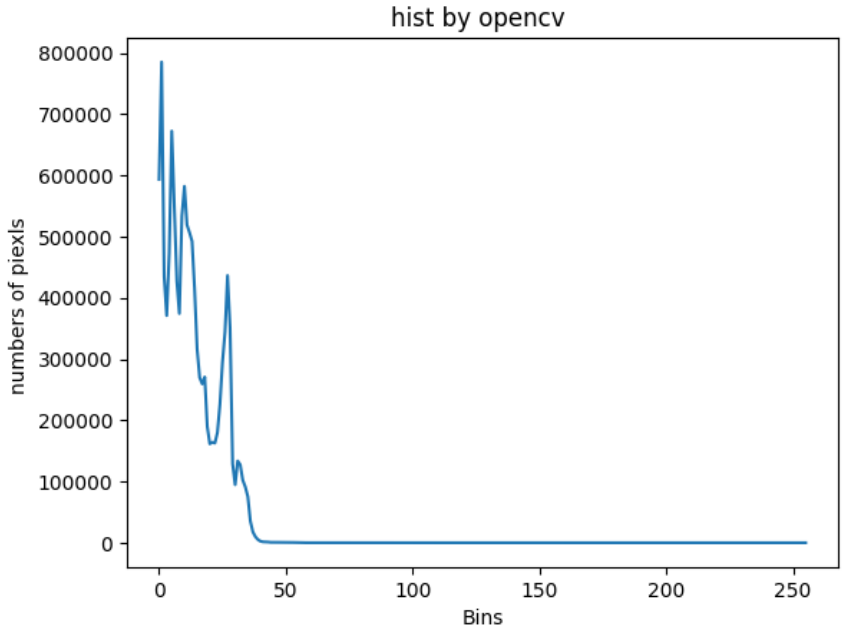
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
# 計算hist
def calchist(gray_image):
    hist = np.zeros(256) #空的陣列 0-255
    for h in range(gray_image.shape[0]):#高度
        for w in range(gray_image.shape[1]):#寬度
            hist[gray_image[h,w]] += 1 #gray_image[h,w] 當前亮度 0-255 這是輸入 因此整句話是指 把亮度作為輸入該個亮度點多了一個
    return hist

# 計算累積分佈函數
def cumsum(pmf):
    cdf = np.zeros_like(pmf)  # 創建一個與PMF相同大小的零數組，用於存儲CDF的值
    cumulative_sum = 0
    for i in range(len(pmf)):
        cumulative_sum += pmf[i] #累加
        cdf[i] = cumulative_sum
    return cdf

# 載入圖片
image = cv2.imread("hw1.jpeg")
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print(gray_image)
# 顯示原圖
plt.figure()
plt.title("Before equalizeHist by hand ")
plt.imshow(gray_image, cmap='gray')
plt.show()
# 把二維數據轉成一維直方圖
hist = calchist(gray_image )
# 顯示直方圖
plt.figure() #創建一個圖
plt.title("hist by hand ")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
```

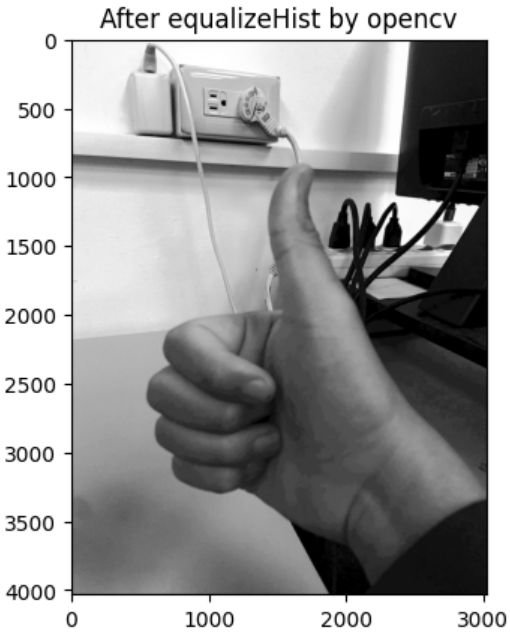
```
plt.plot(hist) #畫一維陣列的圖
plt.xlim([0, 256])#限制x範圍
plt.show()
# 計算並顯示概率質量函數（PMF）
pmf = hist / sum(hist)
cdf = cumsum(pmf)

# 換出轉移曲線 cdf[gray_image[height, width]] * (255-0)
# 找到換算的亮度 = 距離差  乘上 放大的比例
# 1. cdf - cdfmin 是指當下機率與最小的機率距離
# 2.  (255/ (cdf_max - cdf_min) )) 是指這個距離放大的倍數
# 3. pmf 是指該個亮度在整體的佔比
# 4. cdf 是指小於等於該個亮度在整體的佔比
# 正規化 CDF
cdf_min = cdf[0]
cdf_max = cdf[-1]
cdf_normalized = (cdf - cdf_min) *  255/ (cdf_max - cdf_min)
new_ = np.zeros_like(gray_image) # 創建一個新的二維陣列儲存圖片轉換結果
height, width = gray_image.shape

for w in range(width):
    for h in range(height):
        # 使用正規化後的 CDF 進行映射
        new_[h, w] = cdf_normalized[gray_image[h, w]]

#顯示圖片結果
plt.figure()
plt.title("After equalizeHist by hand")
plt.imshow(new_, cmap='gray',vmin=0, vmax=255)
plt.show()
# 計算並顯示處理後直方圖
hist = calchist(new_)
plt.figure()
plt.title("hist by hand")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
plt.plot(hist)
plt.xlim([0, 256])
plt.show()

```

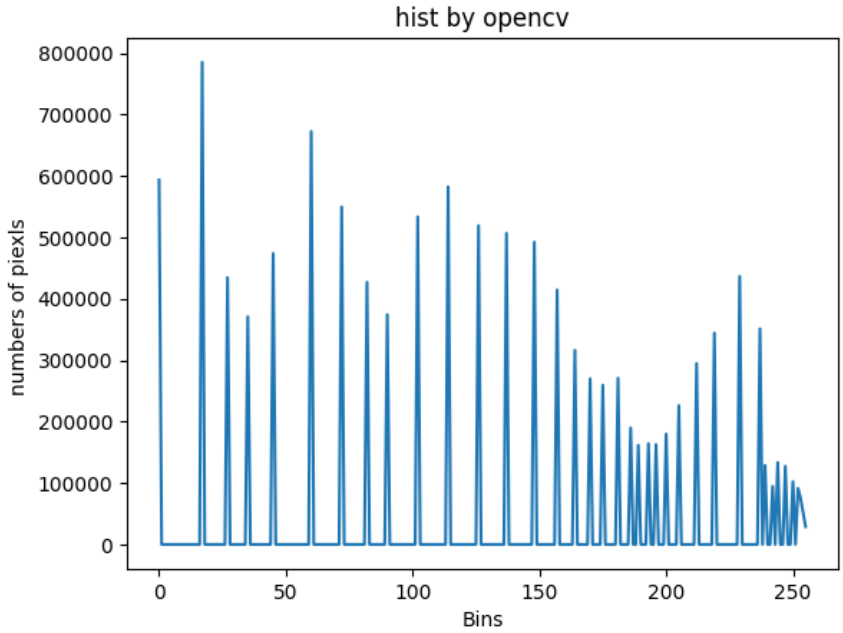
結果：

OpenCV 直方圖等化

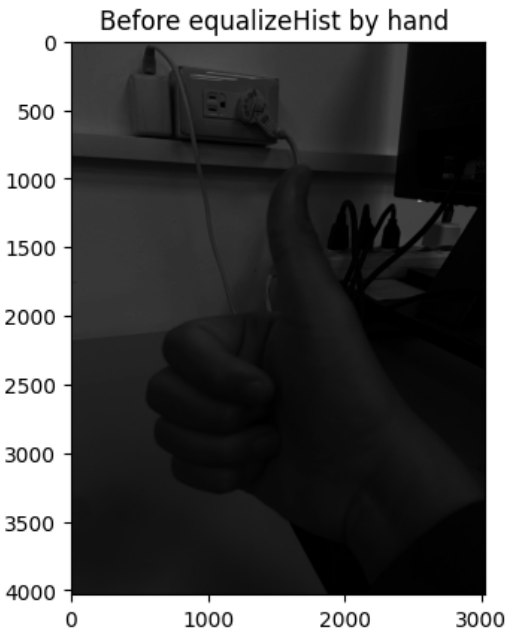


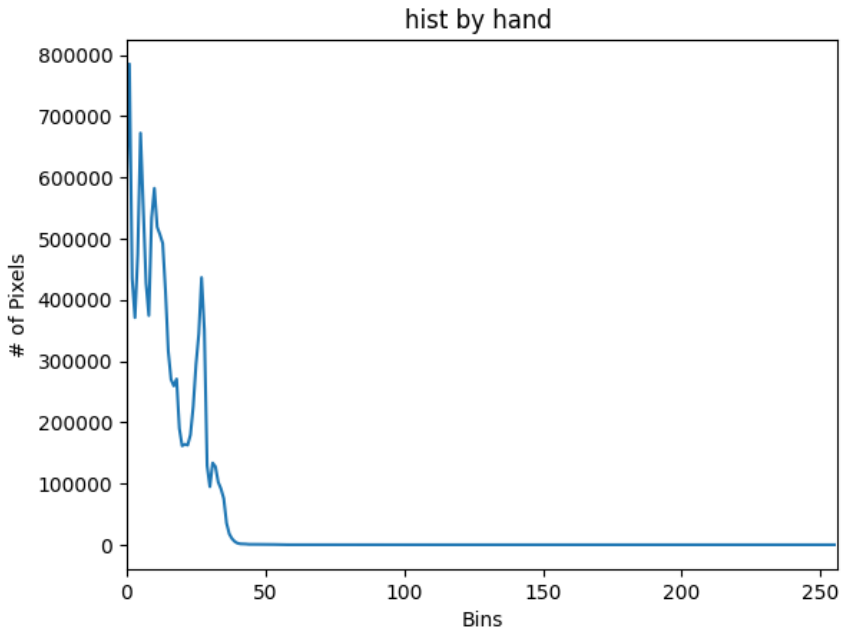


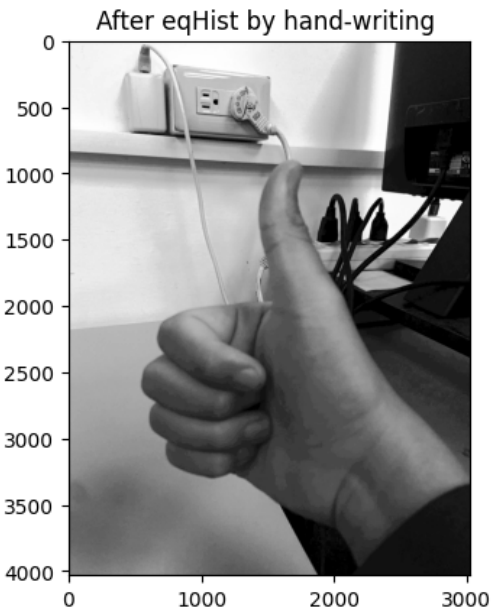


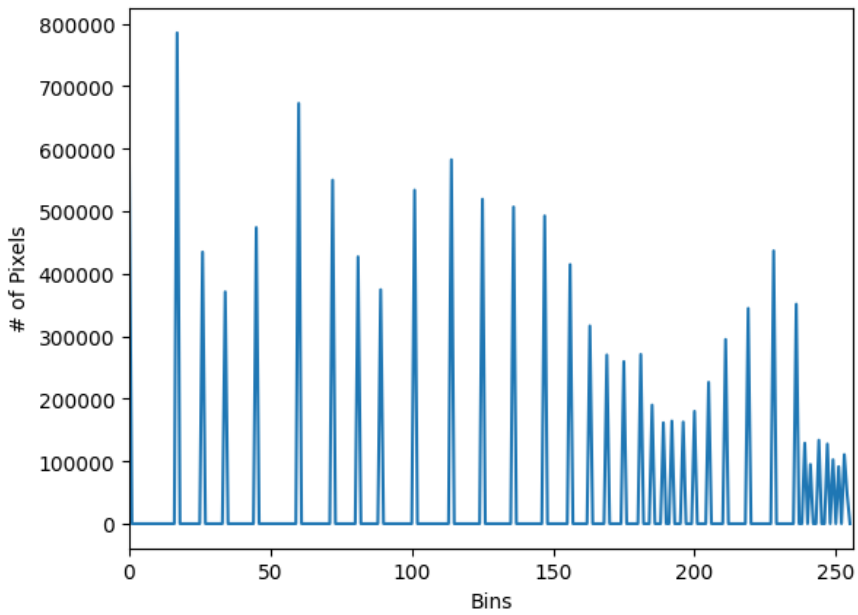


手刻法直方圖等化









結論:

直方圖等化適合對那些過度曝光的圖片進行前處理，從上面的圖片中可以很清楚地發覺圖片的變化很大，另外根據實據測試的結果，opencv的函數運算速度比較快，而我自己的手刻法在執行上速度慢許多。