Prob 0 WARM-UP

(a)

先用 numpy.fromfile 把 raw 檔讀進來，接下來用 reshape 把原本為 1 維的陣列轉為 2 維陣列，再來用 cv2.imwrite 將 image 寫為 result.jpg 檔

此動作將原本不能呈現出圖片的 raw 檔轉成肉眼能見之 jpg 圖片檔。

(b)

先用 cv2.imread 將 sample.jpg 讀進來，接下來用.shape 能發現彩色的 sample.jpg 檔的確為 3 維陣列，因為要轉為 2 維的灰階圖片，先用 np.zeros 創造出一個空的 2 維陣列，接下來將彩色的 3 個 channel 合併為灰階的 1 個 channel

再來自己嘗試與查資料後，發現 R,G,B 3 個 channel 分別給予 0.299,0.587,0.114 的參數合成之後的灰階照片，表現為最佳。

一開始直覺給予 3 個 channel 平均的參數，但是結果慘烈，嘗試多次之後發現原來 3 個 channel 給予不同的比重能讓照片良好呈現

(original)

(output)

(c)

1.逆時針旋轉 90%

將照片 90%翻轉等同於將陣列 90%翻轉，所以同上先創造一個空的 2 維陣列，再將舊值一一填入翻轉後應該存在的陣列位置

New[511-j][i] = Original[i][j]

最後用 cv2.imwrite 寫成 result3.jpg

(original) (output)


2.對角線翻轉

此動作等同於將 image coordinate 的 x,y 對調

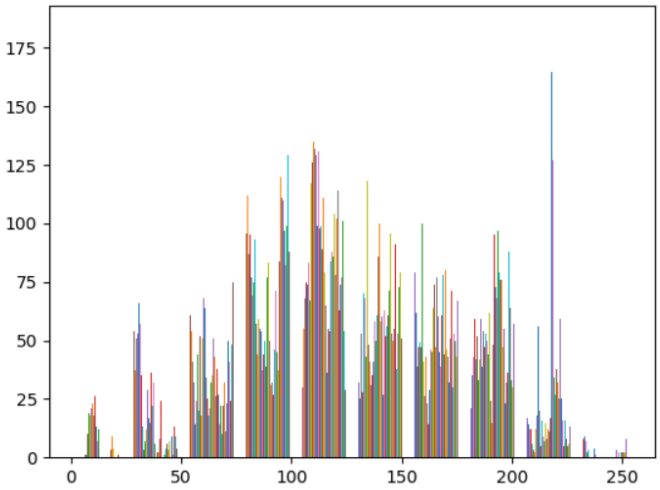
New[j][i] = Original[i][j]

(original) (output)


Prob 1 IMAGE ENHANCEMENT

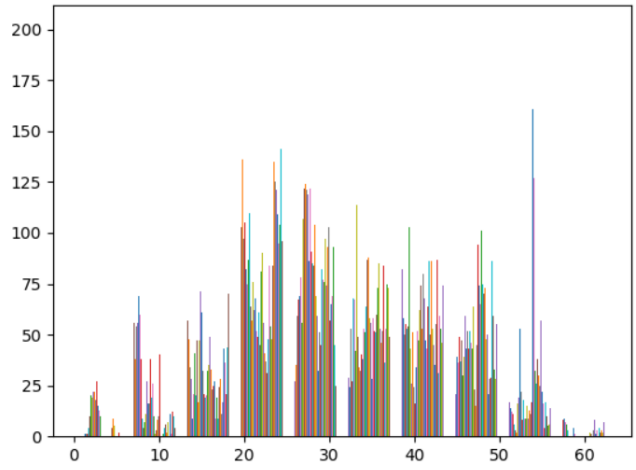
(a) Draw histogram

(result1.jpg)



(result1.hist)

(sample3.jpg)



(sample3.hist)

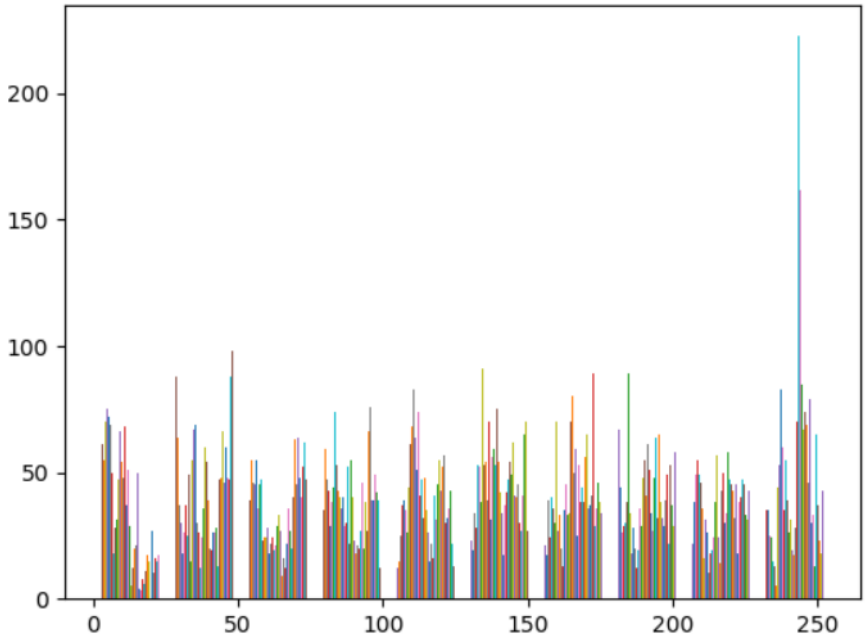
兩張照片很明顯的明暗不同，畫出的 histogram 圖型雖然很像，但是 X 軸表示的 dynamic range 範圍差很多，可以看到亮的 result1 x 軸最大值有到 250 左右，但是暗的 sample3 只有到 60 左右

做 contrast manipulation 能讓 sample3 趨近 result1 的結果

(b)

Original image

Output image



(result5.hist)

對 sample3 做 global histogram equalization 得出 result5，

為了讓 histogram 呈現 uniform 型式，先求出原圖的 cdf，

再帶入公式:round((cdf(v)-cdf.min)/(400*600-cdf.min)*255)

這裡的 v 是原本的 pixel value 值，而 cdf.min 是 112 因為 pixel value 為 0 的有 112 個，255 則是由最大 pixel 值 256 減 1 而得，最終得出的是新的 pixel value 值

會發現 histogram 變得較為 uniform，而圖片變得明亮，效果也好很多。

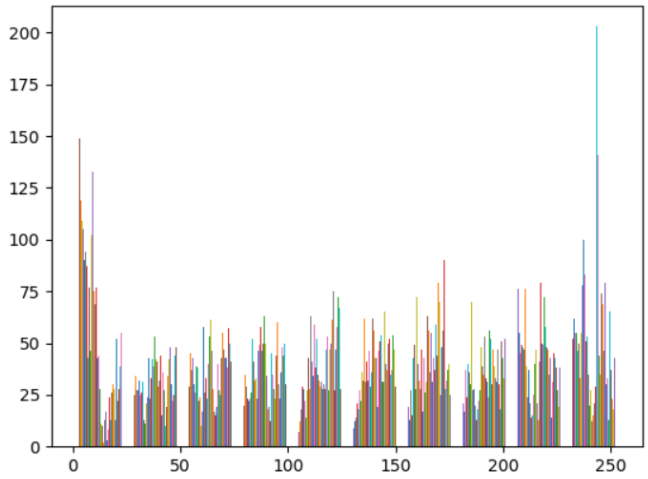
(c)

做 local histogram equalization 前，先將原始圖片切成數塊，分別進行 histogram equalization，在這裡我是將 400*600 的原圖切成 6 塊

200*200 的圖，接下來跟上一題一樣，先求出分塊圖片的 cdf 再帶入公式改善圖片效果，最後將 6 塊圖片在合成 400*600 的完整圖片，以下為結果:

Original image

output image



(result6.hist)

(d)

local 做出來的結果雖然跟 global 大同小異，但是放大仔細看，會發現用 local histogram equalization 做出來的圖片，因為是區塊跟區塊之間分開來做，比起全部一起做，較不會互相影響，所以黑白對比更為鮮明，相對起來有圖片更為清晰的視覺效果。

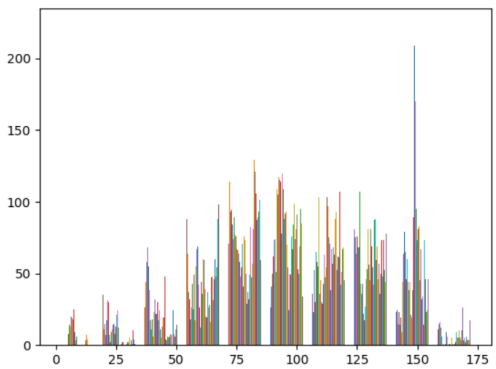
(e)

先使用 log transform 來 enhance sample3.jpg

(Original)

一開始使用 G(j,k)=c*log(F(j,k)+1)，這裡的 F(j,k)為原有 pixel value 壓縮到 0 到 1 之間的值，而 G(j,k)則為 output 出來新的 pixel value，
經由調整參數 c 之後，在 c=255 時，能得到下面的結果，可以看到圖片不僅比原本的明亮，圖片內容物也清晰許多。

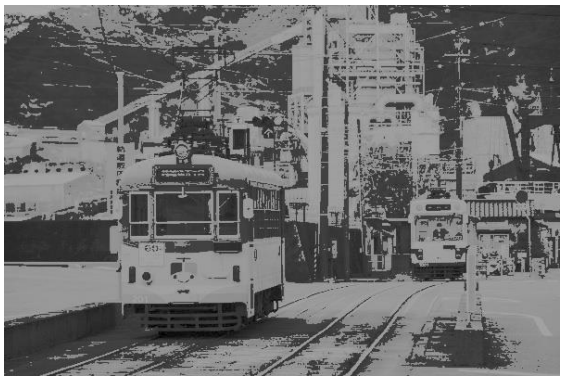
(result7.jpg)

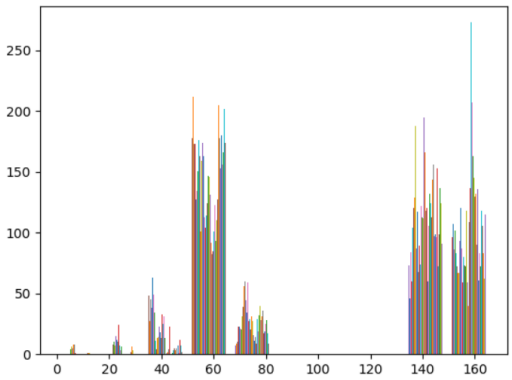


(result7.hist)

再來嘗試在原圖 pixel 值介於 0~30 之間時，使用函式

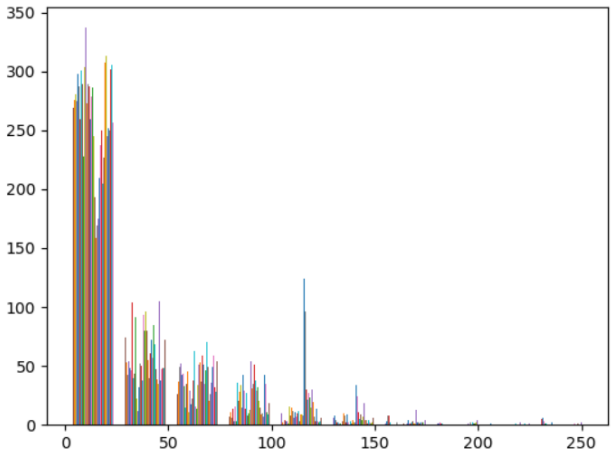
G(j,k)=c*log(F(j,k)+1)，這裡的 F 值則為未經壓縮的 pixel 值，且 c 值設為 20；處理剩下範圍的 pixel 值時則將 c 的參數值設為 40，能得到下列效果，像是用古老相機拍出來的相片，多了藝術感。





接下來使用 inverse log transform，inverse log 即為 exponential，經由試驗參數後，使用的函式為 1.095^(原 pixel value 值)，可以看到雖然圖片還是較為昏暗，但是物體的輪廓已比原圖清楚許多，且陰影這種對比較強的地方表現突出。

(result8.jpg)

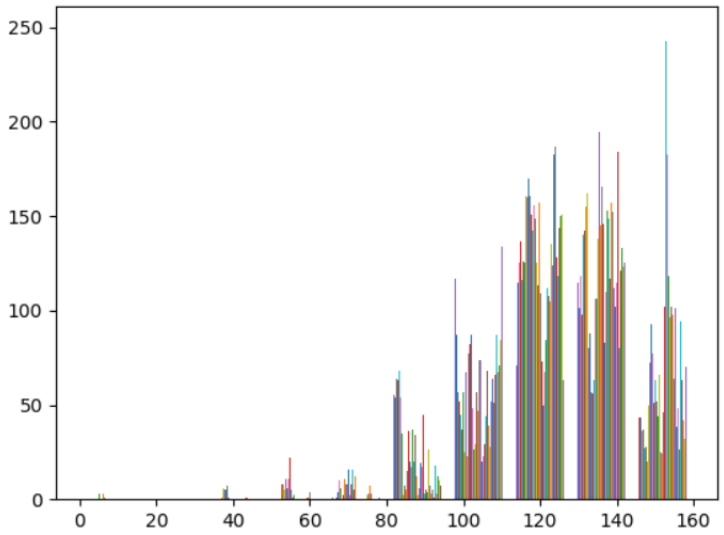


(result8.hist)

最後使用 power function transform 處理，這裡使用的函式為 G(j,k) = 256*(F(j,k)^c)，這裡的 F 值為壓縮至 0~1 的 pixel 值，c 則為次方數，當我們使用 1/2 次方時，結果如下，雖然比原圖亮一點，但是仍然像是陰天一般，昏昏暗暗的。

但是我們將次方數調整到 1/3 次方時，結果如下，明亮度能有所增加，且能呈現出類似下雪的效果。

(result9.jpg)



(result9.hist)

Prob2 NOISE REMOVAL
(a) Guassian noise

(original)

先用 np.random.normal 做出一個高斯常態分布，給予不同參數值之後，再將 noise 加入原有圖片，而得到下列兩個結果，第二個結果給的參數值較大，所以 noise 的情況會比第一張嚴重。

(resultG1.jpg)

(resultG2.jpg)

(b) Salt-and-pepper noise

一樣先將原始乾淨的圖片載入，先製造一定數量 pixel value 為 255 的 salt 點，再由隨機的方式加入原圖，接下來製造一定數量 pixel value 為 0 的 pepper 點，也一樣加入原圖。

第二張 noisy 圖因為給予的 amount 參數值較大，所以 noise 的影響比第一張更大。

(resultS1.jpg)

(resultS2.jpg)

(c) guassian noise removal

(oringinal)(ResultG1.jpg)

(resultR1.jpg)

處理這張 noisy image 我是用 3*3 且權重相等的 mean filter 來對每個 pixel 做一次，可以發現 guassian noise 的影響有下降，但是圖片清晰度也會稍微下降。

(Original)(resultG2.jpg)

(resultR2.jpg)

這張 noisy 圖則因為 noise 的情況較為嚴重，一開始我仍然使用 3*3 filter 去做，但效果不彰，在經過多次嘗試之後我選用 5*5 的 mean filter 且給予不相同權重，5*5 filter 中間給予較大權重，向外越來越小。

$$\begin{pmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 4 & 6 & 10 & 6 & 4 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{pmatrix}$$

可以看到結果有變好，但是照片變得相對更模糊了，所以我們可以得知用越大的 mean filter 越容易使照片變模糊。

(d) s-p noise removal

(Original)(resultS1.jpg)

(resultR3.jpg)

這裡我先做 boundary extension，然後我選用的基底是 3*3 的 median filter，一開始效果不錯，但是在取中位數時加入取 mean 的方法，能讓降躁效果更進一步提升。

(Original)(resultS2.jpg)

這張我選用和上一個相似的 median filter，也能得到不錯的降躁效果，清晰度也不受影響。

(resultR4.jpg)

(e)用 PSNR 的標準公式得出各個的 PSNR 值和比較

1. resultR1 的 PSNR 值為 27.74 dB

2. resultR2 的 PSNR 值為 22.81 dB

3. resultR3 的 PSNR 值為 25.02 dB

4. resultR4 的 PSNR 值為 22.15 dB

R1 處理的視覺效果比 R2 明顯的好，所以 PSNR 值較大，而因為 R3 將 salt and pepper 處理得比 R4 乾淨，所以 PSNR 值也較大。但是在嘗試各種大小 filter 和參數之後，PSNR 值仍然在 20~30 之間徘徊，還有進步空間。

#bonus

在處理這張下雨圖片時，一開始指使用跟前面一樣的 5*5 權重不等的 mean filter，雖然呈現一定效果，但是再加上 2D 的 Pseudomedian filter 處理各 pixel，可以有更好的效果。雨痕比較沒這麼明顯了，但是照片會變稍模糊。

(bonus.jpg)