Image Processing HW #1

**Spatial Image Enhancement**

611410122 許恒誠

Data due: Nov. 8, 2022

Data handed in: Oct. 26, 2022

Technical Description

**使用Python撰寫，利用3種方法來增強圖像**：

只有一支python，執行時會讀取 當前資料夾/HW1\_test\_image內的所有圖片。

並對這些圖片執行 (1) power-law (gamma) transformation, (2) histogram equalization, (3) image sharpening using the Laplacian operator.來增強圖像，並將圖像另存至 當前資料夾/output/<照種類對應資料夾>

**Package requirements:**

1. glob # 取得圖像路徑。
2. opencv # 讀取圖片。
3. numpy # 計算用途。
4. os #建立資料夾用途。

**Github Link:**

https://github.com/AUDOSt0ck1ng/ImageProcessing\_hw1.git

1. power-law (gamma) transformation

**說明:**

formula:

s = c(r+ε)^γ

----------------------------------

s: output

c: constant, default = 1

ε: offset, default = 0

γ: gamma, user-defined

c and γ are positive constants.

*def* power\_law\_transform(*img*, *γ*):

    if isinstance(*img*, np.ndarray) and isinstance(*γ*, *float*):

        return np.array(255\*(*img*/255)\*\**γ*, *dtype*='uint8')

    else:

        print("img: "+*str*(*type*(*img*)),*end*=" ")

        print("γ: "+*str*(*type*(*γ*)))

        print("type error.")

        exit()

調整參數大小後圖像的整體明暗程度也會隨之變化，其中的上限值可以在第8行做設定，預設是5。

1. histogram equalization

**說明:**

先把所有顏色的數量都統計成直方圖，並計算每個顏色的機率(像素個數/像素總數)。

再計算累計機率，乘上像素總數並4捨5入得到新的像素值。

*def* histogram\_equalization(*img*):

    hist, bins= np.histogram(*img*.ravel(), 256, [0, 255])    #直方圖 灰階:256bins，0 ~ 255

    pdf = hist/*img*.size   #計算每個顏色機率

    cdf = pdf.cumsum()

    equ\_value = np.round((cdf\*255)).astype('uint8')

    return equ\_value[*img*]

1. image sharpening using the Laplacian operator

**說明:**

padding先把矩陣補成(m+p\*2)x(n+p\*2)，convolution之後會變回mxn。

p = LM 的 kernel size/2

ex:

Laplacian matrix 3x3 (kernel size 3)

LM = [[0, 1, 0],[1, -4, 1], [0, 1, 0]]

p = 1

----------------------------------

formula:

new\_point = c·sum(LM convolution img) + point

計算後的值可能會超出範圍、以上下值取代。

*def* laplacian\_operator(*img*):

    M, N = *img*.shape

    k=3 # kernel

    c = -1

    p = k // 2

    new\_img = np.zeros((M + p \* 2, N + p \* 2), *dtype*=np.float64)

    new\_img[p: p + M, p: p + N] = *img*.copy().astype(np.float64)

    tmp = new\_img.copy()

    # laplacian matrix

    LM = [[0., 1., 0.],[1., -4., 1.], [0., 1., 0.]]

    for m in range(M):

        for n in range(N):

            point = tmp[p + m, p + n]

            con = tmp[m: m + k, n: n + k]

            new\_img[p + m, p + n] = c\*np.sum(LM\*con) + point

    new\_img = np.clip(new\_img, 0, 255)

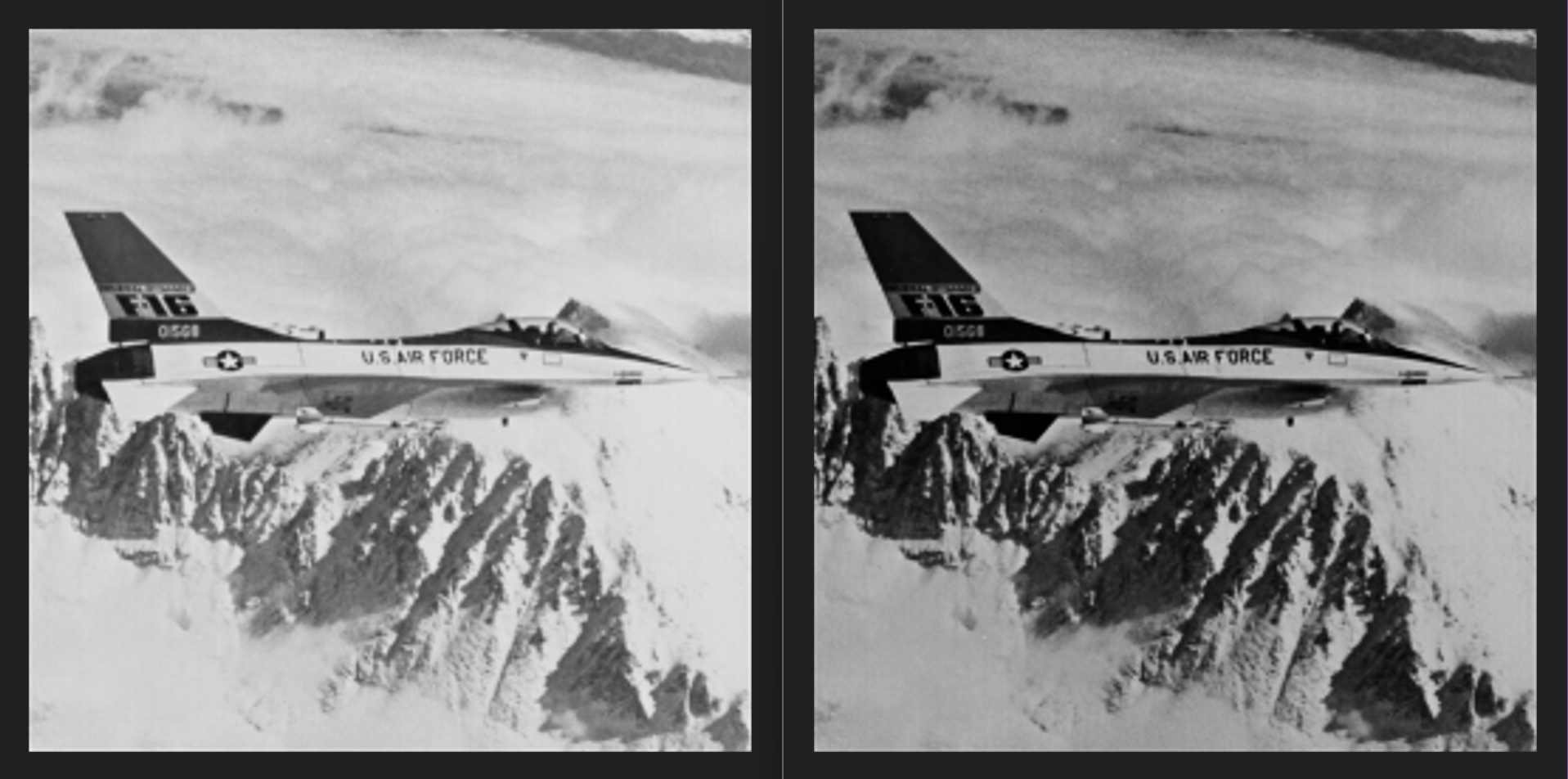
    new\_img = new\_img[p: p + M, p: p + N].astype('uint8')

    return new\_img

Experience Results

1. power-law (gamma) transformation

因為程式設計是平均值最接近128的圖像為最佳，所以原圖太亮，就會取得較暗的圖；原圖太暗，就會取得較亮的圖。





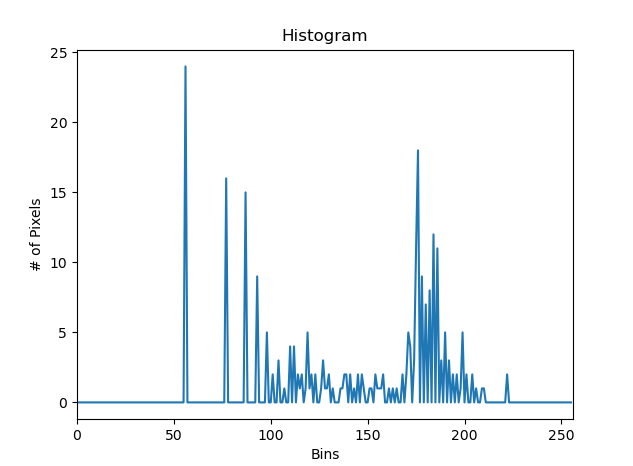
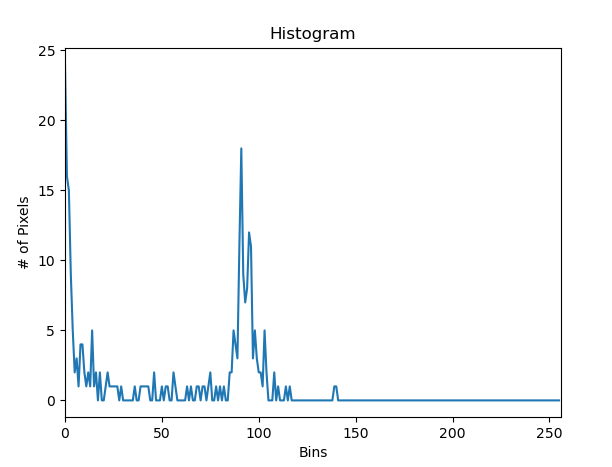


原圖V.S. power-law (gamma) transformation

1. histogram equalization

(我另外寫程式產生長條圖。)

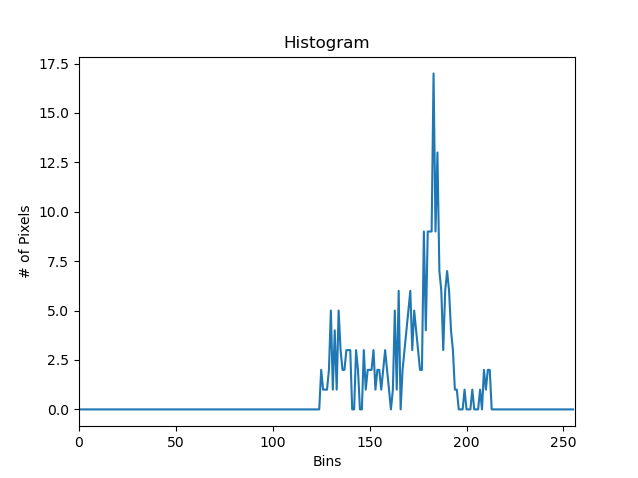
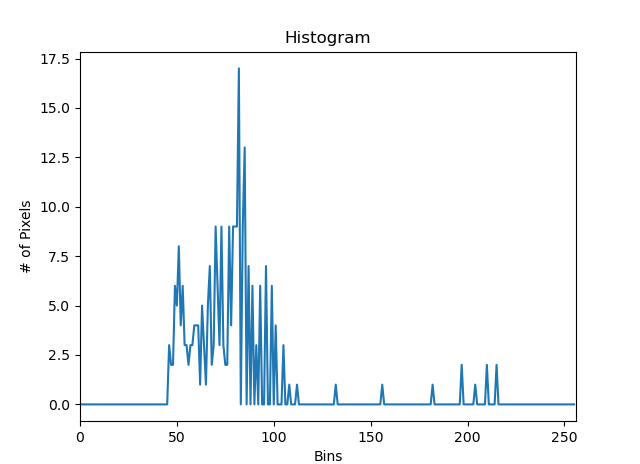
可看出做過histogram equalization的圖像分布較分散(除了Peppers)。



HE

Origin

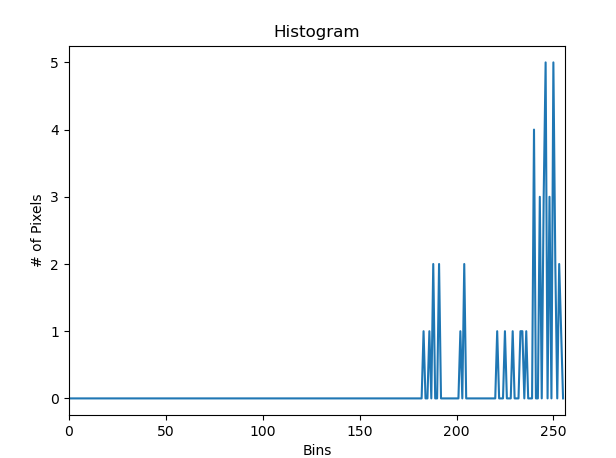
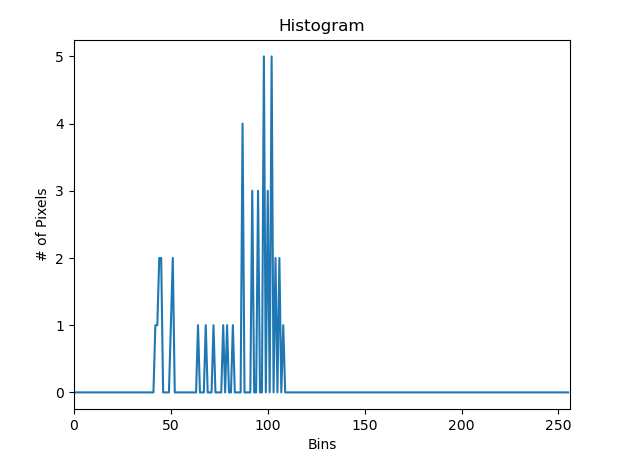
Lake



HE

Origin

Jetplane



Peppers

1. image sharpening using the Laplacian operator

與原本的比較，更著重在線條/輪廓表現。







原圖V.S. Laplacian operator

Discussions

power-law (gamma) transformation

**發現:**

一開始我使用中位數來代表整張圖，想找到中位數最接近128的圖片來當最佳選擇，但在圖片內像素對比太極端時，表現不太好，最後改以平均數接近128為最佳選擇。

References and Appendix

https://www.geeksforgeeks.org/python-intensity-transformation-operations-on-images/

https://codeinfo.space/imageprocessing/histogram-equalization/

https://pyimagesearch.com/2016/07/25/convolutions-with-opencv-and-python/