影像處理作業 HW2

統計碩二 蔡耀德

Q1.

(1) 先使用 cv2.imread 工具讀取圖片 text-broken.tif，為方便處理因此轉為灰階影像。

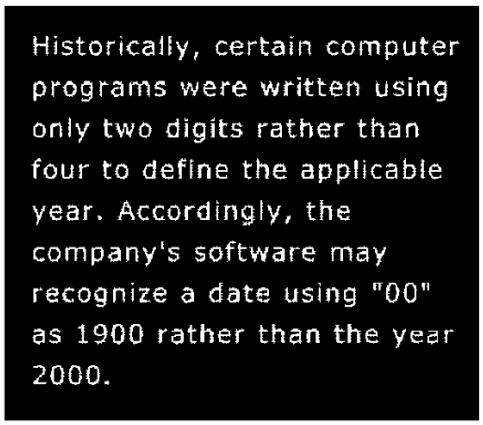
```
#Q1.1
# Read the image
imageQ1 = cv2.imread(path+'text-broken.tif', cv2.IMREAD_GRAYSCALE)
```

選取 3x3 的 Structuring element 作為 Kernel。

```
# Define kernel
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
```

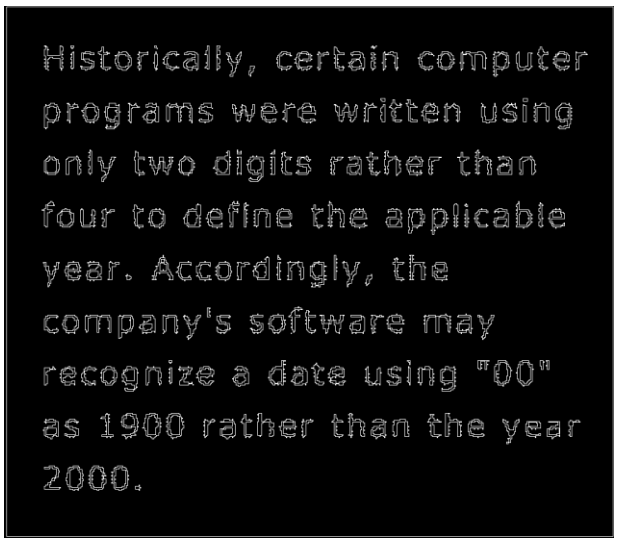
重複使用 Closing(先 Dilation 再 Erosion)，觀察修復狀況。

重複三次後影像如下，可見修復情形不錯。



(2) 先使用 Erosion 計算A ⊖ B 值，再以原影像A減去A ⊖ B，即可找出 boundary。

```
#Q1.2
# Perfom the edge of the image
img_erosion = cv2.erode(image, kernel, iterations=1)
boundary = image-img_erosion
```



Q2.

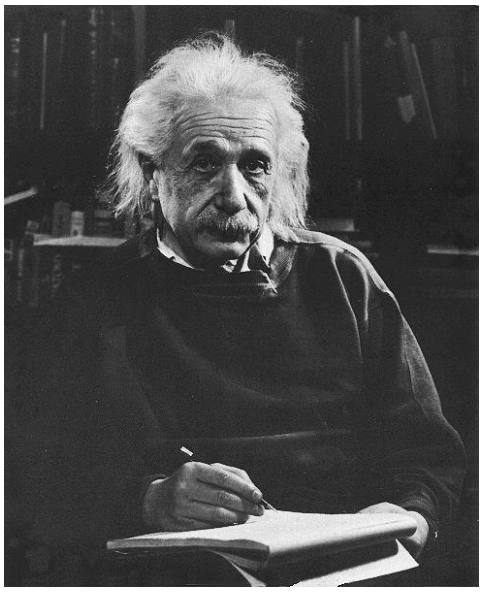
使用 cv2.imread 工具讀取圖片 einstein-low-contrast.tif

```
# Read the image
imageQ2 = cv2.imread(path+'einstein-low-contrast.tif', cv2.IMREAD_GRAYSCALE)
```

計算此影像之最小值及最大值，並依照拉伸公式將灰階值線性拉伸至 0~255 之間。

```
# Perform linear stretching
stretched_image = (imageQ2 - np.min(imageQ2)) * (255.0 / (np.max(imageQ2) - np.min(imageQ2)))
```

線性拉伸後影像如下



Q3

使用 cv2.imread 工具讀取圖片 einstein-low-contrast.tif。

```
# Read
imageQ3 = cv2.imread(path+'einstein-low-contrast.tif', cv2.IMREAD_GRAYSCALE)
```

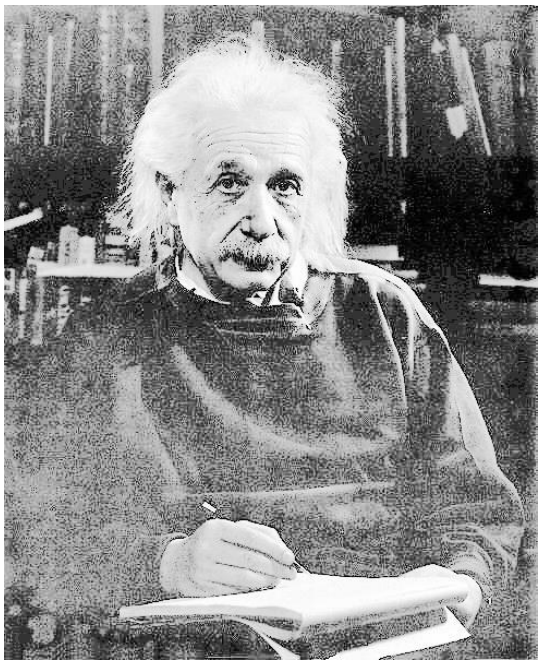
計算該影像灰階值之 histogram 及 CDF。

```
# Calculate the histogram
hist = np.zeros(256, dtype=int)

for i in range(imageQ3.shape[0]):
    for j in range(imageQ3.shape[1]):
        idx = imageQ3[i, j]
        hist[idx] += 1

# Calculate the CDF
cdf = [0]*256
for i in range(0,256):
    cdf[i] = sum(hist[0:(i+1)])/(imageQ3.shape[0] * imageQ3.shape[1])
```

將影像中每個 pixel 之灰階值帶入 CDF 後重新轉為 0~255 之灰階值。

```
# HE
HE_image = np.zeros_like(imageQ3)
for i in range(imageQ3.shape[0]):
    for j in range(imageQ3.shape[1]):
        idx_eq = imageQ3[i, j]
        HE_image[i, j] = int(cdf[idx_eq] * 255)
```

Histogram equalization 結果如下



Q4

使用 cv2.imread 工具讀取圖片 einstein-low-contrast.tif。

```
imageQ4 = cv2.imread(path+'aerialview-washedout.tif', cv2.IMREAD_GRAYSCALE)
```

先找出 median，透過灰階值之 histogram 中累計頻率超過總個數一半者藉以找出 median。

```
# Calculate the histogram
hist = [0]*256
for i in range(imageQ4.shape[0]):
    for j in range(imageQ4.shape[1]):
        idx = imageQ4[i, j]
        hist[idx] += 1

# Find the median of the image
total_pixels = imageQ4.shape[0] * imageQ4.shape[1]
cum_pixels, median = 0 , 0
for i in range(256):
    cum_pixels += hist[i]
    if cum_pixels >= int(total_pixels / 2):
        median = i
        break
```

分別計算灰階值從 0~208(median)及 209~255 之 CDF，並分別進行 histogram equalization。

```
# Calculate cdf of two sub-histogranms
cdf_Q4 = [0]*256
for i in range(0, 256):
    if i<=median:
        cdf_Q4[i] = sum(hist[0:(i+1)])/sum(hist[0:(median+1)])
    else:
        cdf_Q4[i] = sum(hist[(median+1):(i+1)])/sum(hist[(median+1):])

# Perform HE for sub-histograms
subHE_image = [[0 for i in range(imageQ4.shape[1])] for i in range(imageQ4.shape[0])]
for i in range(imageQ4.shape[0]):
    for j in range(imageQ4.shape[1]):
        idx_Q4 = imageQ4[i, j]
        if idx_Q4 <= median:
            subHE_image[i][j] = int(cdf_Q4[idx_Q4] * median)
        else:
            subHE_image[i][j] = int(cdf_Q4[idx_Q4] * (255 - median) + median)
```

分別進行 histogram equalization 後結果如下

Q5

使用 cv2.imread 工具讀取圖片 einstein-low-contrast.tif

```
imageQ5 = cv2.imread(path+'einstein-low-contrast.tif', cv2.IMREAD_GRAYSCALE)
```

參照論文 Two-dimensional histogram equalization and contrast enhancement (T. Celik 2012) 及課堂講義 CVCE version1，步驟如下：

先計算$h_x(l,k)$及$w_x(l,k)$以計算$H_x(l,k)$

```
# Calculate 2D (weighted) histograms h_x(l,k) & H_x(l,k)
x_1 = np.min(imageQ5) ; x_K = np.max(imageQ5)
weight_hist,hist = np.zeros((256,256)), np.zeros((256,256))
for l in range(256):
    idx = np.where(imageQ5==l)
    for k in range(256):
        weight = np.abs(l-k+1)/(x_K-x_1+1)
        for p, y in zip(*idx):
            if (p < 3 | (p + 3> imageQ5.shape[0]))|(y < 3 | (y + 3> imageQ5.shape[1])):
                continue
            else:
                hist[l,k] += (imageQ5[p-3:p+3+1,y-3:y+3+1]==k).sum()
                weight_hist[l,k] = weight * hist[l,k]
```

再將$H_x(l,k)$標準化後，計算$CPF_p(m)$

```
# Calculate H_x(l,k)
H_x = np.zeros_like(weight_hist)
sum_H_x=0
for l in range(256):
    for k in range(256):
        sum_H_x+= weight_hist[l,k]
H_x = weight_hist/sum_H_x

# Calculate CDF_x(m)
CDF_x = [0]*256
for m in range(256):
    for i in range(m):
        for j in range(m):
            CDF_x[m] +=H_x[i,j]
```

計算 Uniform distribution 之$H_u(l,k)$以及$CPF_u(m)$

```
# Calculate H_u(l,k) and CDF_u(m)
CDF_u = [0]*256
hist_u = np.full((256,256),1/256**2)
for m in range(256):
    for i in range(m+1):
        for j in range(m+1):
            CDF_u[m] += hist_u[i,j]
```

使用 argmin 函數求出$T(l) = argmin_{v\epsilon[0,255]}|CPF_x(l) - CPF_u(v)|$

```
# Calculate transformation function T(l)
T_l = [0]*256
for l in range(256):
    T_l[l] = np.abs(np.array(CDF_x[l])-CDF_u).argmin()
```

依照$T(l)$進行 2D histogram equalization

```
# Imple 2D-HE by using function T(l)
HE2D_image = np.zeros_like(imageQ5)
for l in range(256):
    idx = np.where(imageQ5==l)
    HE2D_image[idx[0],idx[1]] = T_l[l]
```

結果如下