Computer Vision HW1

612415013 電機碩一 蕭宥羽

一、 程式碼及方法

```
import numpy as np
import cv2
```

Import python libraries

```
def RGB2GRAY(image):
    return np.dot(image[...,:3], [0.21, 0.72, 0.07]).astype(np.uint8)
```

灰階值 = 0.21 * 紅 + 0.72 * 綠 + 0.07 * 藍

用 np.dot()計算 RGB 與三個參數的乘積總和，得到灰階值

用 astype(np.uint8) 將計算出的灰階值轉換為 8 位元的整數數據類型

```
def ReLU(x):
    if(x < 0): return 0
    else: return x
```

當 x 大於 0 時回傳 x，小於 0 時設為 0

即將負數設為 0，正數保持不變

```
def EDGE_DECT(image,kernel):
    height, width = image.shape[:2]
    k_hig, k_wid = np.shape(kernel)
    matrix=np.zeros((height-k_hig+1,width-k_wid+1))
    for i in range(height-k_hig+1):
        for j in range(width-k_wid+1):
            matrix[i,j]=ReLU((kernel*image[i:i+k_hig,j:j+k_wid]).sum())
    return matrix
```

kernal 跟 image 做卷積

kernel*image[i:i+k_hig, j:j+k_wid]).sum() 將區域內的每個像素與卷積核對應位置的值相乘，在將總和計算算出來

最後在經過 ReLU 函數

```python
def pooling(image):
    height, width = image.shape[:2]
    height_2=int(round(height/2))
    width_2=int(round(width/2))
    matrix=np.zeros((height_2,width_2))
    for i in range(height_2):
        for j in range(width_2):
            if(2*i+2<height):
                ri=2*i+2
            else:
                ri=2*i+1
            if(2*j+2<width):
                rj=2*j+2
            else:
                rj=2*j+1
            matrix[i, j] = np.max(image[2*i: ri, 2*j: rj])
    return matrix
```

做 max pooling，stride 為 2

np.max(image[2*i: ri, 2*j: rj])取範圍內的最大值，達到 max pooling 的作用

```python
def binarization(image, threshold):
    height, width = image.shape[:2]
    for i in range(height):
        for j in range(width):
            if(image[i ,j] < threshold):
                image[i ,j] = 0
            else:
                image[i, j] = 255
    return image
```

像素值小餘 threshold 設為 0，大於則設為 255

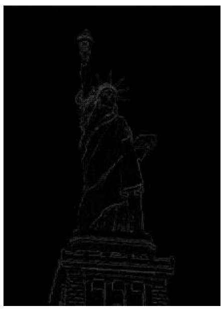
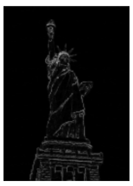
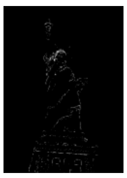
```python
img=cv2.imread("test_img/liberty.png")
gray_img = RGB2GRAY(img)
cv2.imshow('RGB Image To Grayscale', gray_img)
cv2.imwrite("result_img/liberty_Q1.png", gray_img)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()

kern=np.array([[0,1,0],[1,-4,1],[0,1,0]])
edge_img=EDGE_DECT(gray_img,kern)
cv2.imshow('EDGE_DECT', edge_img)
cv2.imwrite("result_img/liberty_Q2.png", edge_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

pool_img=pooling(edge_img)
cv2.imshow('pooling', pool_img)
cv2.imwrite("result_img/liberty_Q3.png", pool_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

bin_img = binarization(pool_img,128)
cv2.imshow('Binarizatio', bin_img)
cv2.imwrite("result_img/liberty_Q4.png", bin_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output Q1 -> Input Q2-> Input Q3 -> Input Q4
每做完一個動作便顯示圖片，關掉圖片後接續下一個動作

二、 結果

1. liberty.png

Origin	Grayscale	Convolution	Pooling	Binarization
				

2. temple.jpg

Origin	Grayscale	Convolution	Pooling	Binarization