Computer Vision HW1

111598066 資工碩一 許哲維

一、Explain Program and Method

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

Import three Python libraries.

```
def RGB2Gray(image):
    return np.dot(image[...,:3], [0.21, 0.72, 0.07]).astype(np.uint8)
```

Convert each bit of the RGB image to [0.21, 0.72, 0.07] to convert the grayscale image.(Base on luminosity method)

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

The ReLU operation is define when it smaller than 0 it will return 0, otherwise return by itself.

```
def Convolution(image, kernel):
    kernel = np.flipud(np.fliplr(kernel))
    result = np.zeros((int(image.shape[0]-2), int(image.shape[1]-2)))

    for y in range(0, result.shape[1]):
        for x in range(0, result.shape[0]):
            result[x, y] = ReLU((kernel * image[x: x + 3, y: y + 3]).sum())

    return result
```

In the convolution operation, the kernel needs to be rotated 180 degrees first, and then the operation is performed in sequence.

```
def MaxPool(image):
    result = np.zeros((int(round(image.shape[0]/2)), int(round(image.shape[1]/2))))

    x = y = 0
    while(y < result.shape[1]):
        while(x < result.shape[0]):
            if(x * 2 + 2 < image.shape[0]):
                rx = x * 2 + 2
            else:
                rx = x * 2 + 1
            if(y * 2 + 2 < image.shape[1]):
                ry = y * 2 + 2
            else:
                ry = y * 2 + 1

            result[x, y] = np.max(image[2*x: rx, 2*y: ry])
            x+=1
        y+=1
        x=0

    return result
```
In pooling operation, we select the max digit of the kernel size.

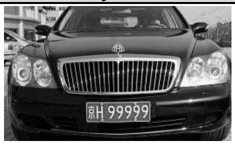
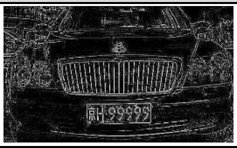
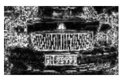
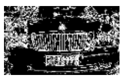
```
def Binarization(image, threshold):
    for y in range(image.shape[1]):
        for x in range(image.shape[0]):
            if(image[x ,y] < threshold):
                image[x ,y] = 0
            else:
                image[x, y] = 255
    return image
```
In binarization operation, when digit is smaller than threshold it will transfer to 0, otherwise it will be 255.

```
def main():
    path = input("Please Enter Path:> ")
    filename = input("Please Enter Filename:> ")
    if not(os.path.isfile(path+filename)):
        print("Error：Connot Find File！")
        return
    image = cv2.imread(path + filename)
    grayscale = RGB2Gray(image)
    cv2.imshow('RGB Image To Grayscale', grayscale) #Problem 1 Solution
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    print('Write File:> grayscale.png')
    cv2.imwrite(path + 'grayscale.png', grayscale)

    kernel = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
    convol = Convolution(grayscale, kernel)
    cv2.imshow('Convolution Operation With ReLU', convol) #Problem 2 Solution
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    print('Write File:> convolution.png')
    cv2.imwrite(path + 'convolution.png', convol)

    pool = MaxPool(convol)
```

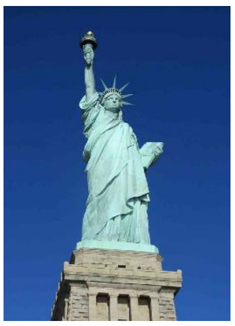
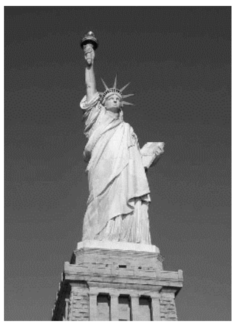
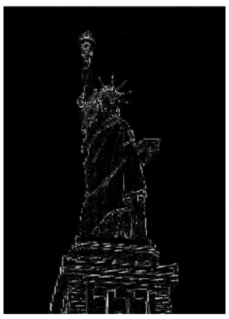
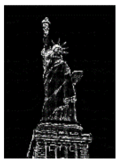
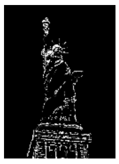
```
        cv2.imshow('Max Pool', pool) #Problem 3 Solution
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        print('Write File:> maxpool.png')
        cv2.imwrite(path + 'maxpool.png', pool)

        binar = Binarization(pool, 128)
        cv2.imshow('Binarization Operation ', binar) #Problem 4 Solution
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        print('Write File:> binarization.png')
        cv2.imwrite(path + 'binarization.png', binar)

main()
```
First, the program will ask you to enter the path and then the file name. If the file cannot be found, the program will jump out and end. Otherwise, it will perform grayscale conversion, convolution, pooling, binarization, and display one by one and write the files to the path.

二、Result Images

1. car.png

Origin	Grayscale	Convolution	Pooling	Binarization
				

2. liberty.png

Origin	Grayscale	Convolution	Pooling	Binarization
				

※ All the original outputs can be seen from the attached file.