

Computer Vision HW2

111598066 資工碩一 許哲維

I、Explain Program and Method

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
```

Import four Python libraries.

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

Convert every RGB image's bit to [0.21, 0.72, 0.07] to get the grayscale image. (Base on luminosity method)

```
def Mean_Filter(image):
    kernel = np.array([[1/9, 1/9, 1/9], [1/9, 1/9, 1/9], [1/9, 1/9, 1/9]])
    imagepad = np.zeros((int(image.shape[0]+2), int(image.shape[1]+2)))
    imagepad[1:image.shape[0]+1, 1:image.shape[1]+1] = image
    result = np.zeros((int(image.shape[0]), int(image.shape[1])))

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

    return result
```

The Mean Filter operation is using a 3*3 box to compute the mean of image sequentially.

```
def Median_Filter(image):
    imagepad = np.zeros((int(image.shape[0]+2), int(image.shape[1]+2)))
    imagepad[1:image.shape[0]+1, 1:image.shape[1]+1] = image
    result = np.zeros((int(image.shape[0]), int(image.shape[1])))

    for y in range(0, result.shape[1]):
        for x in range(0, result.shape[0]):
            result[x, y] = np.median(imagepad[x: x + 3, y: y + 3])

    return result
```

The Median Filter operation is using a 3*3 box to compute the median of image sequentially.

```
def Image_Histogram(image, path, filename):
    #print('Write File:> '+filename)
    plt.hist(image.tolist(), 256, histtype='barstacked')
    plt.savefig(path + filename)
    plt.show()
```

The function will turn the inputted image into histogram image.

```
def main():
    path = input("Please Enter Path:> ") #輸入圖片路徑，如 D:/cv/cv_hw2/
    filename = input("Please Enter Filename:> ") #圖片檔名，如 noise_image.png
    if not(os.path.isfile(path+filename)):
        print("Error : Connot Find File ! ")
        return
    image = cv2.imread(path + filename)
    grayscale = RGB2Gray(image)
    print('Write File:> grayscale.png')
    cv2.imwrite(path + 'grayscale.png', grayscale)
    Image_Histogram(grayscale, path, 'noise_image_his.png')




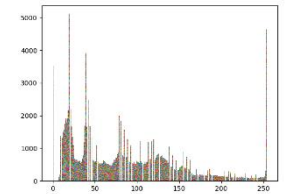
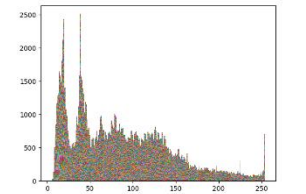
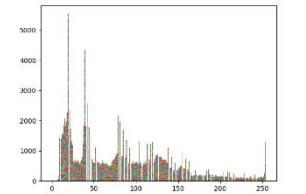
    output1 = Mean_Filter(grayscale)
    print('Write File:> output1.png')
    cv2.imwrite(path + 'output1.png', output1)
    Image_Histogram(output1, path, 'output1_his.png')

    output2 = Median_Filter(grayscale)
    print('Write File:> output2.png')
    cv2.imwrite(path + 'output2.png', output2)
    Image_Histogram(output2, path, 'output2_his.png')

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, mean filter, median filter, create histogram image, write the files to the path.

II 、 Result Images

	Noise image _(grayscale)	Mean filter	Median filter
Output Image			
Histogram			

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

III 、 Compare

From the output histogram, we can find that the Mean filter fills the Histogram gaps of the Noise image, and the Median filter is similar to the result of the Noise image histogram. It can also be found from the output image that in images with Salt & Pepper Noise, the effect of the Median filter is better than that of the Mean filter.