CV hw8 / 電機所 R06921082 陳與賢

Description:

利用 python 來處理 bmp 檔，首先要用高斯和 salt-and-pepper 來隨機地產生雜訊，再分別使用 box filter、median filter、opening and closing 的方式來消除雜訊。

Algorithm:

首先是製造出雜訊，這邊因為助教 ppt 上已經有給 code 了，而且 numpy 連 API 都做好，所以就直接拿來用，基本上跟助教給的沒什麼差別，code 如下：

```
for x in range(img_input.width):
    for y in range(img_input.height):
        pixels_gau10[x, y] = pixels_input[x, y] + int(10*np.random.normal(0.0, 1.0, None))
        pixels_gau30[x, y] = pixels_input[x, y] + int(30*np.random.normal(0.0, 1.0, None))
        if np.random.uniform(0.0, 1.0, None) < 0.05:
            pixels_snp005[x, y] = 0
        elif np.random.uniform(0.0, 1.0, None) > 1 - 0.05:
            pixels_snp005[x, y] = 255
        else:
            pixels_snp005[x, y] = pixels_input[x, y]
        if np.random.uniform(0.0, 1.0, None) < 0.1:
            pixels_snp01[x, y] = 0
        elif np.random.uniform(0.0, 1.0, None) > 1 - 0.1:
            pixels_snp01[x, y] = 255
        else:
            pixels_snp01[x, y] = pixels_input[x, y]
```

再來分別對四張雜訊圖來作處理，分別是 box filter、median filter、closing and then opening、opening and then closing

因為 box filter 跟 median filter 的差別只在於說一個是取平均、一個是取中位數，而且 numpy 也都有 API 可以用（偉哉 numpy）（P.S. 這應該有算是 hardcore programming 吧@@？因為取平均頂多也只是多寫一個 for，相加後除掉，取中位數也可以 sort 然後取中間就好，這兩個應該也都是 numpy 的實作方法），所以我直接把 box filter 及 median filter 寫在同一個 function，並且透過參數來決定 size，code 如下：

```python
################ filter
def imgfilter(option, size, img_input, pixels_input):
    filter_output = Image.new(img_input.mode, img_input.size)
    filter_pixels = filter_output.load()

    if size == 3:
        half = 1
    else:
        half = 2

    for x in range(img_input.width):
        for y in range(img_input.height):
            tmp = []
            for q in range(-1*half, half+1):
                for p in range(-1*half, half+1):
                    tmp_x = x + q
                    tmp_y = y + p
                    if tmp_x > -1 and tmp_x < 512 and tmp_y > -1 and tmp_y < 512:
                        tmp.append(int(pixels_input[tmp_x, tmp_y]))
                    else:
                        tmp.append(0)
            if option == "box":
                num = np.mean(np.asarray(tmp))
            else:
                num = np.median(np.asarray(tmp))

            filter_pixels[x, y] = int(num)
    return filter_output
```

根據尺寸，將原點設為中心點，然後算出 filter 之各點的相對距離，用 tmp_x、tmp_y 來紀錄，若在邊界之內則將 pixel 值放到 tmp 內，否則放 0 進去，最後再依據 option 看是 box 還是 median filter 來決定要取平均還是中位數，如此一來即可，至於 opening 跟 closing 之前作業寫過了演算法沒什麼需要改的所以就不多說明了。

最後我將需要處理的影像放到 list 裡面，跑一個 for 迴圈依序處理雜訊。
然後題目要求要算出 SNR，根據助教給的公式（如下）寫出即可

$$VS = \frac{\sum_{\forall n} (I(i,j) - \mu)^2}{\|n\|}$$

$$\mu = \frac{\sum_{\forall n} I(i,j)}{\|n\|}$$

$$VN = \frac{\sum (I_N(i,j) - I(i,j) - \mu_N)^2}{\|n\|}$$

$$\mu_N = \frac{\sum_{\forall n} (I_N(i,j) - I(i,j))}{\|n\|}$$

$$SNR = 20 \log_{10} \frac{\sqrt{VS}}{\sqrt{VN}}$$

Result:

SNR:

高斯10: 13.93052737286375
高斯10 opening then closing: 8.57448779833638
高斯10 closing then opening: 7.662142833586206
高斯10 box filter 3*3: 16.455027237114155
高斯10 box filter 5*5: 13.616968346614799
高斯10 median filter 3*3: 17.841516139301508
高斯10 median filter 5*5: 15.932207050178707

高斯30: 4.285158844118946
高斯30 opening then closing: 8.589414715001283
高斯30 closing then opening: 6.086593892991115
高斯30 box filter 3*3: 12.231931167531407
高斯30 box filter 5*5: 12.411389241825702
高斯30 median filter 3*3: 11.226120456264931
高斯30 median filter 5*5: 12.767571955752828

salt-and-pepper 0.1: -1.8816921300301679
salt-and-pepper 0.1 opening then closing: -2.417624252365165
salt-and-pepper 0.1 opening then closing: -2.3945012119250473
salt-and-pepper 0.1 box filter 3*3: 6.479975247389868
salt-and-pepper 0.1 box filter 5*5: 8.406992322246266
salt-and-pepper 0.1 median filter 3*3: 14.807162274269611
salt-and-pepper 0.1 median filter 5*5: 14.305297919242763

salt-and-pepper 0.05: 1.0442189088419838
salt-and-pepper 0.05 opening then closing: 4.408554812851394
salt-and-pepper 0.05 opening then closing: 4.295434508770716
salt-and-pepper 0.05 box filter 3*3: 9.354440634985034
salt-and-pepper 0.05 box filter 5*5: 10.650936525095045
salt-and-pepper 0.05 median filter 3*3: 18.57446862569331
salt-and-pepper 0.05 median filter 5*5: 15.730344219194219

gaussian10

gaussian30

salt-and-pepper 0.05

salt-and-pepper 0.1

Gaussian 10

Opening then closing

closing then opening

box filter 3*3

box filter 5*5

median filter 3*3

median filter 5*5

Gaussian 30

Opening then closing

closing then opening

box filter 3*3

box filter 5*5

median filter 3*3

median filter 5*5

salt-and-pepper 0.05

Opening then closing

closing then opening

box filter 3*3

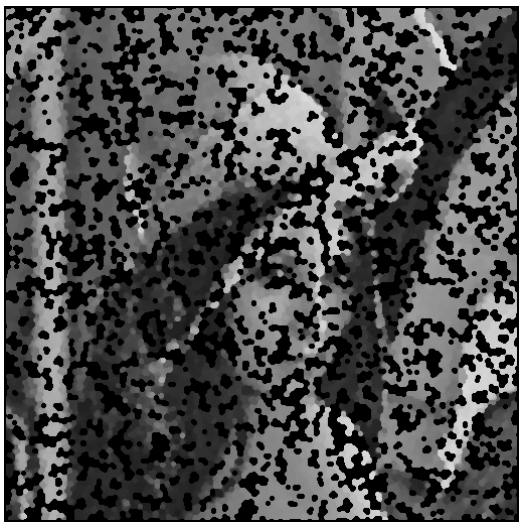
box filter 5*5

median filter 3*3

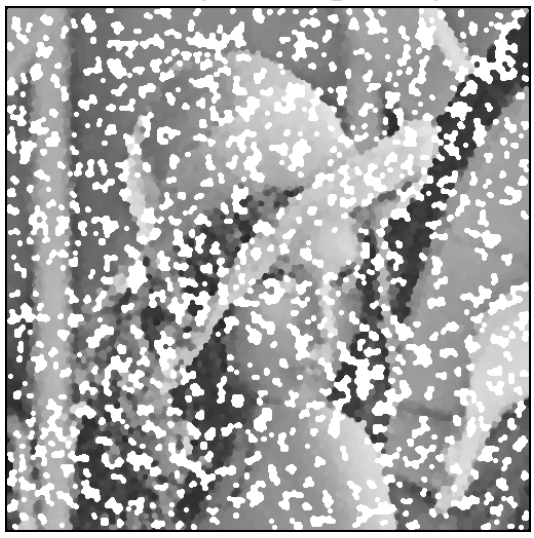
median filter 5*5

salt-and-pepper 0.1

Opening then closing



closing then opening



box filter 3*3

box filter 5*5

median filter 3*3

median filter 5*5