Hw4_Edge Detection

學號:610415145 姓名:許晉偉

Date due:2022/01/06

Date handed in:2022/01/07

■Technical description

```
import numpy as np
import cv2
img = cv2.imread('image1.jpg',0)
Height, width = img.shape[:]
```

使用 opencv 函式庫進行基本圖片操作

使用 numpy 函式庫進行矩陣建立

```
Gaussian_img= np.zeros((Height-2, width-2, 1), np.uint16)
Gaussian_Filter=np.zeros((3, 3, 1),np.float16)
GX=np.zeros((Height-2, width-2, 1), np.int32)
GY=np.zeros((Height-2, width-2, 1), np.int32)
G=np.zeros((Height-2, width-2, 1), np.int32)
```

建立存放結果和 Sobel 濾波器所需矩陣

(1)　Sobel edge detection

首先運用 Gaussian Filter 濾除雜訊

```
sigma =0.5**0.5
y, x = np.mgrid[-1:2, -1:2]
total=0
```

設定 Gaussian Filter 所需參數

利用 np.mgrid 產生 3*3(x,y)值陣列

$$\begin{bmatrix} (-1,-1) & (0,-1) & (1,-1) \\ (-1,0) & (0,0) & (1,0) \\ (-1,1) & (0,1) & (1,1) \end{bmatrix}$$

```
for i in range(3):
    for j in range(3):
        Gaussian_Filter[i][j]=(np.exp(-(x[i][j]**2+y[i][j]**2)/(2*sigma**2)))/(2*np.pi*sigma**2)
        total=total+Gaussian_Filter[i][j]
for i in range(3):
    for j in range(3):
        Gaussian_Filter[i][j]=np.round((Gaussian_Filter[i][j]/total),3)
```

利用數學式計算出 Gaussian Filter 的 mask

$$G(x,y)=\frac{1}{2\pi}e^{-(x^2+y^2)}$$

$$\begin{bmatrix} 0.045 & 0.122 & 0.045 \\ 0.122 & 0.332 & 0.122 \\ 0.045 & 0.122 & 0.045 \end{bmatrix}$$

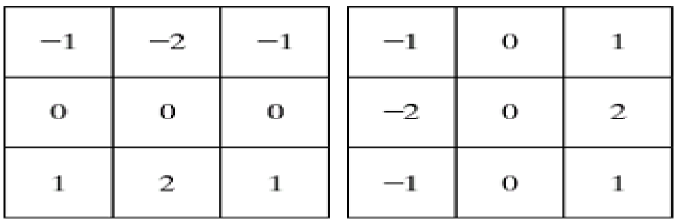
```python
def convolution(x,y,img):
    result=0
    for i in range(0,3):
        for j in range(0,3):
            result=result+img[x+i,y+j]*Gaussian_Filter[i][j]
    return float(result)

for i in range(0,Height-4):
    for j in range(0,width-4):
        temp=convolution(i,j,img)
        if(temp)<0 :
            temp=0
        if(temp)>255 :
            temp=255
        Gaussian_img[i,j]=np.round(temp,0)
cv2.imwrite('Gaussian.jpg', Gaussian_img)
```

並用 gaussian mask 對原圖 進行 convolution 即完成高斯模糊

```python
Gx = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
Gy = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
```

建立 Sobel 濾波器 Gx&Gy 用來計算梯度分量



−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

```python
def dot(x,y,img,direction):
    result=0
    for i in range(0,3):
        for j in range(0,3):
            if(direction=='x'):
                result=result+img[x+i,y+j,0]*Gx[i][j]
            elif((direction=='y')):
                result=result+img[x+i,y+j,0]*Gy[i][j]
    return result

for i in range(0,Height-4):
    for j in range(0,width-4):
        temp1=dot(i,j,Gaussian_img,'x')
        temp2=dot(i,j,Gaussian_img,'y')
        GX[i][j]=temp1
        GY[i][j]=temp2
```

以 Gx & Gy 分別進行 convolution 計算

得到 x 方向的邊緣與 y 方向的邊緣

```
for i in range(0,Height-2):
    for j in range(0,width-2):
        G[i][j]=np.sqrt(((GX[i][j]**2)+(GY[i][j]**2)))
```

最後再將兩者平方相加取根號即得出最後的 Sobel 結果

```
cv2.imshow('source', img)
cv2.imshow('sobel_img', G.astype(np.uint8))
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('image1_sobel.jpg',G.astype(np.uint8))
```

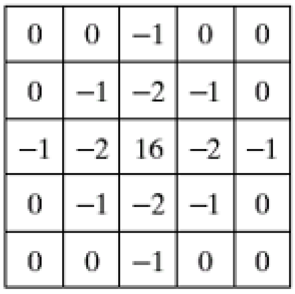
顯示出結果並儲存

(2) Laplacian of a Gaussian (LoG)

```
LOG_img= np.zeros((Height, width, 1), np.uint8)
LOG_mask=[[0,0,-1,0,0]
         ,[0,-1,-2,-1,0]
         ,[-1,-2,16,-2,-1]
         ,[0,-1,-2,-1,0]
         ,[0,0,-1,0,0]]
```

建立存放結果所需矩陣

建立課本提及之 LoG masks 用以達成 LoG 運算



0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

```
def convolution(x,y,img):
    result=0
    for i in range(0,5):
        for j in range(0,5):
            result=result+img[x+i,y+j]*LOG_mask[i][j]
    return result

for i in range(0,Height-4):
    for j in range(0,width-4):
        temp=convolution(i,j,img)
        if(temp)<0 :
            temp=0
        if(temp)>255 :
            temp=255
        LOG_img[i,j]=temp
```

並用 LoG mask 對原圖 進行 convolution 即完成 LoG

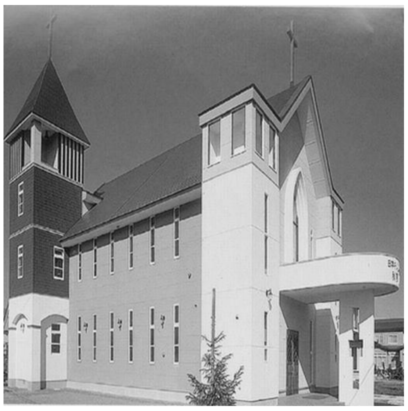
```
cv2.imshow('source', img)
cv2.imshow('LOG_img', LOG_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite('image1_LOG.jpg',LOG_img)
```

顯示出結果並儲存

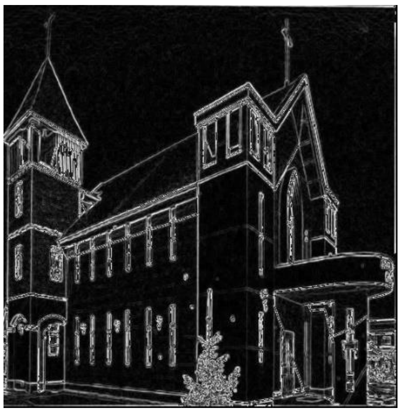
Experimental results

image1

原圖



Sobel



LOG

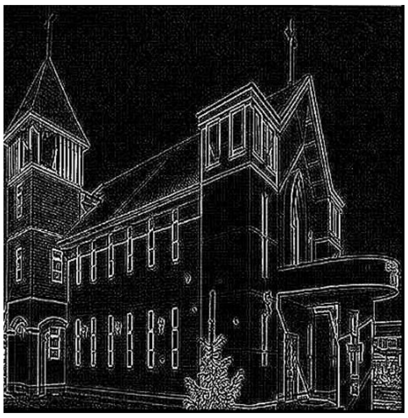
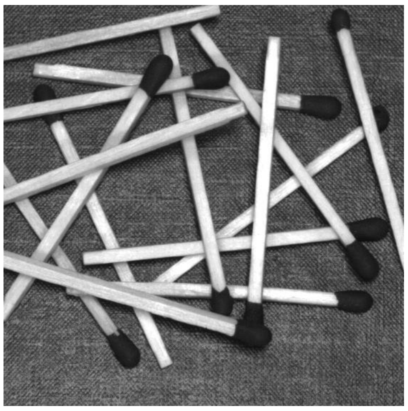
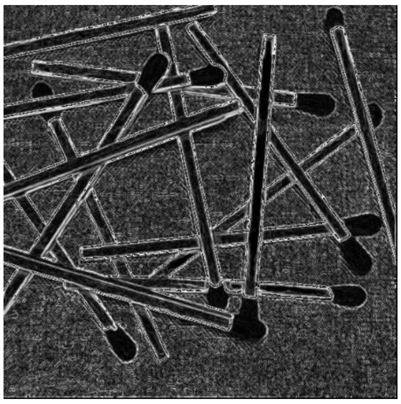


image2

原圖



Sobel



LOG

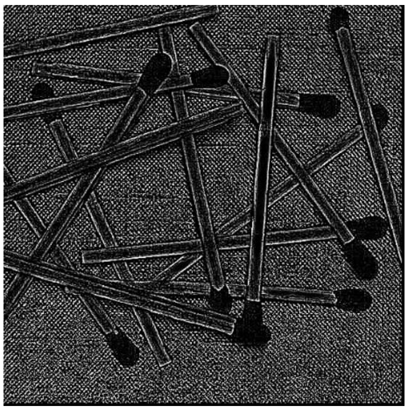
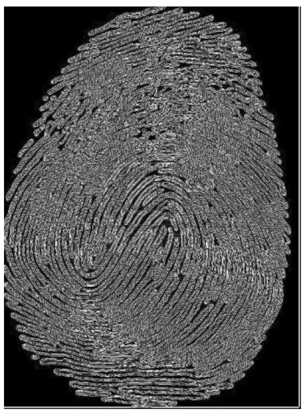


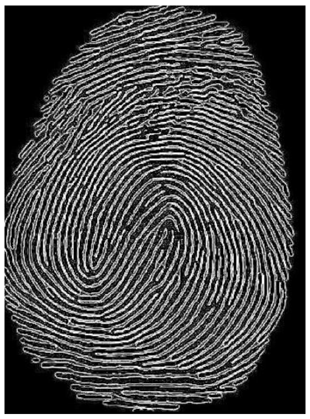
image3

原圖

Sobel



LOG



■Discussions

LOG 相比起 Sobel 好理解很多但是兩者卻沒有一定誰好誰壞在結果圖的比較中可以發現圖 1 和圖 3 使用 LOG 呈現出比較多細節但是在圖 2 是 Sobel 較好一些,所以我覺得這次讓我學到可以針對所要解決的問題選擇相對應演算法，得到最適合的結果。

■ References and Appendix

https://medium.com/@bob800530/python-gaussian-filter-%E6%A6%82%E5%BF%B5%E8%88%87%E5%AF%A6%E4%BD%9C-676aac52ea17

https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

https://medium.com/%E9%9B%BB%E8%85%A6%E8%A6%96%E8%A6%B

A/%E9%82%8A%E7%B7%A3%E5%81%B5%E6%B8%AC-

%E7%B4%A2%E4%BC%AF%E7%AE%97%E5%AD%90-sobel-operator-

95ca51c8d78a

https://medium.com/@bob800530/opencv-

%E5%AF%A6%E4%BD%9C%E9%82%8A%E7%B7%A3%E5%81%B5%E6%

B8%AC-canny%E6%BC%94%E7%AE%97%E6%B3%95-d6e0b92c0aa3

https://theailearner.com/2019/05/25/laplacian-of-gaussian-log/