影像處理作業報告

HW4
Spatial Image Enhancement

授課教授：柳金章
學　　生：楊憲閔
學　　號：613410047
Due date：2025/01/09
Date hand in：2025/01/02

目錄

Technical description

我們有時需要辨識影像內的物體或是分開彼此，故需要把物體的邊緣描繪出來(即邊緣偵測)，本次作業中即是要實作 Sobel operator 與 Canny operator 來達到邊緣偵測的目的。

1. Sobel operator

利用影像的梯度作為判斷依據，其中先一次微分所使用的 mask，如下所示：

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (1)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (2)$$

其中G_x是用來偵測水平邊緣，G_y是用來偵測垂直邊緣，我們可以使用 3x3 遮罩(filter mask)來實現G_x和G_y，如下所示：

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

圖(1) 左圖為G_x的遮罩，右圖為G_y的遮罩。

透過上面的遮罩計算完後，最後將兩者取絕對值相加，可得最後邊緣偵測的結果∇f，其公式如下。

$$\nabla f \approx |G_x| + |G_y| \tag{3}$$

效果如下：

a b
c d
FIGURE 10.10
(a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

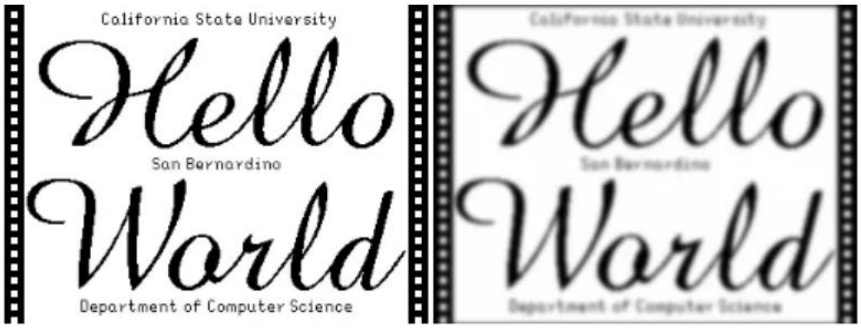
圖(2) 左上為原始影像，右上為G_x影像，左下為G_y影像，右下為$|G_x| + |G_y|$所得到的結果影像。

2. Canny operator

此為相當著名的邊緣檢測演算法，實作上分為了四大步驟：

(1) Gaussian Filter：

運用於濾除雜訊，原因就是降低了細節層次以達到去雜訊



經過Gaussian Filter後的圖像

圖(3) 視覺效果像是經過一個半透明屏幕在觀察圖像

雜訊+灰階　　濾除雜訊+灰階

圖(4) 可以觀察到明顯去除雜訊

(2) Gradient and Direction

此處為透過 Sobel 兩方向的 filter 計算 gradient(可參考上面所述)，再計算出此 gradient 之角度與方向，以濾出邊緣的強度與方向

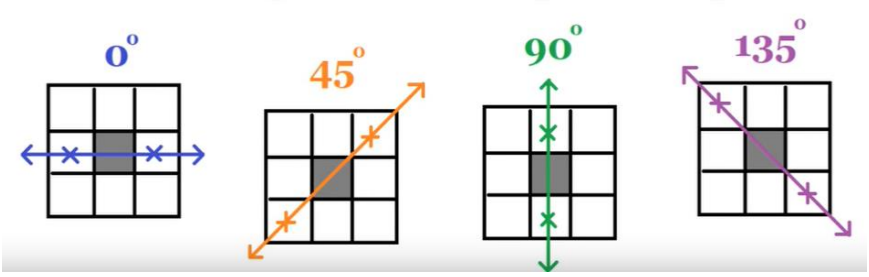
$$|\nabla S| = \sqrt{S_x^2 + S_y^2},$$

$$\theta = \tan^{-1}\frac{S_y}{S_x},$$

圖(5) 此步驟所使用的公式，Sx 與 Sy 為一次 gradient 之值

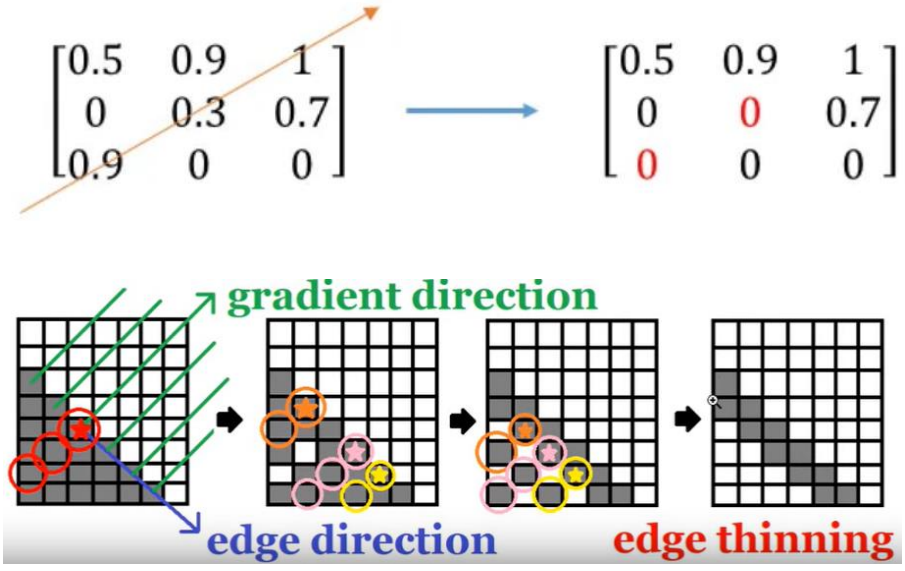
(3) Non-maximum suppression

此步驟為了簡化計算，因此所使用的梯度方向略分為 4 種：0 度、45 度、90 度、135 度



圖(6) 邊緣可能的方向

利用最大值抑制的演算法，去尋找個方向梯度變化最大的點，以下圖為例：已經確定此梯度方向為 45 度，就逐步尋找矩陣內，45 度方向強度最大的點，其餘 45 度方向的點皆歸 0。



圖(7、8) 最大值抑制示意圖

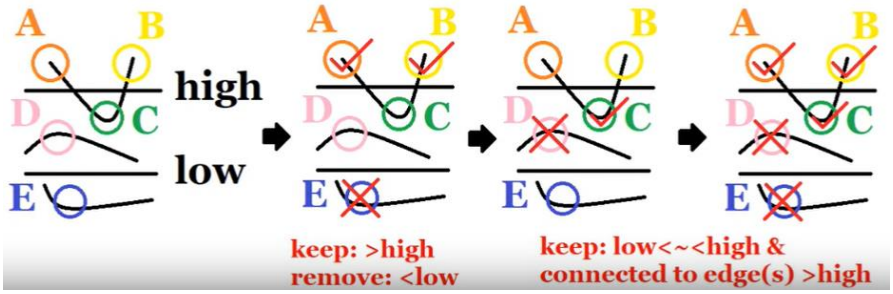
（4） Connect Weak Edge

透過Connect Weak Edge將線連起來，設定高界線與低界線，依循這樣的機制，找出邊緣。

a. 高於高界線:一定是邊緣

b. 低於低界線:一定不是邊緣

c. 介於高界線與低界線:若附近有兩點高於高界線的點，則此點也視為邊緣



圖(9) 連接邊緣機制示意圖

Experimental results

1. 程式執行流程:

(1) 確保已安裝相關 module，本次作業使用 module 如下所示:

```
import cv2
import os
import glob
import numpy as np
from matplotlib import pyplot as plt
import matplotlib
```

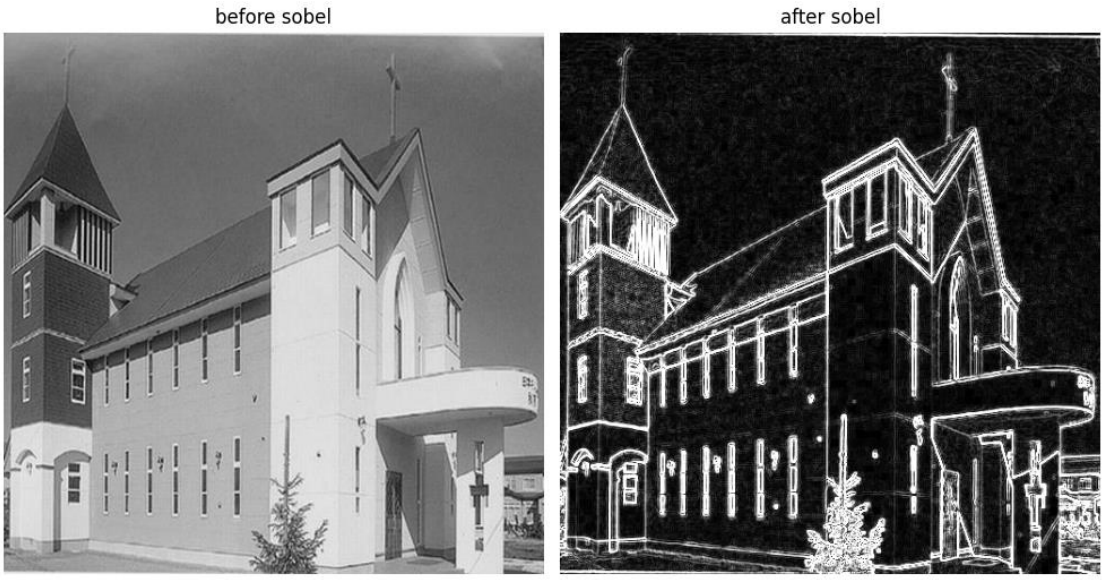
圖(10) 會使用到的 module

(2) 進到作業的目錄底下，會看到一個名為 HW4_test_image 的資料夾，一個 main.py，還有這份 pdf，點右鍵按在終端中開啟，輸入 python main.py，程式即開始執行。

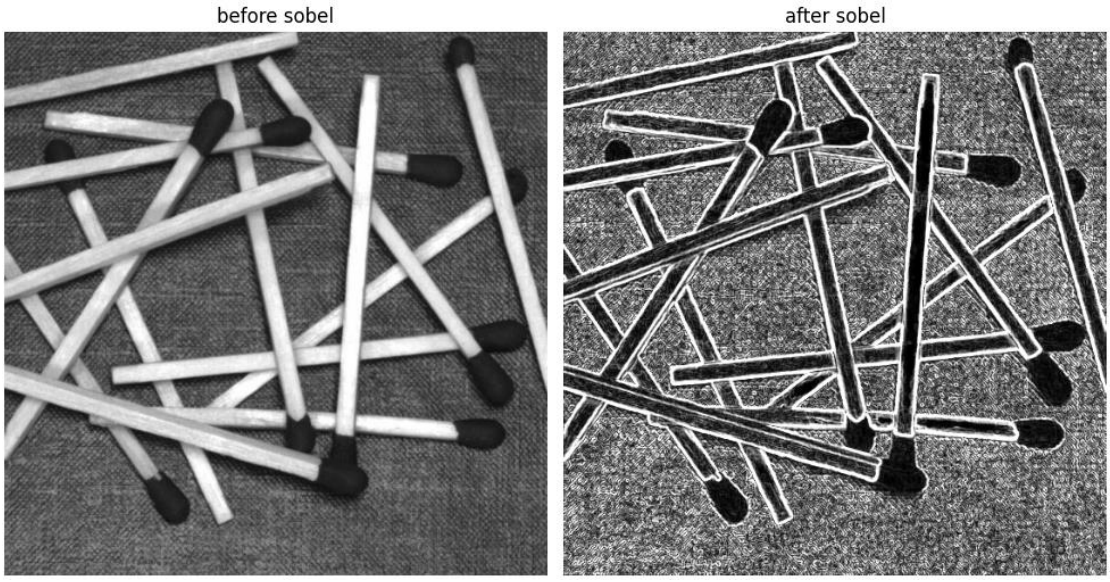
(3) 程式會讀取 HW4_test_image 資料夾底下的圖片，並輸出對每個圖片進行 Edge detection。順序會是讀一張圖片，輸出對該圖片運用 Sobel operator 進行處理後的結果，關掉視窗後會輸出對該圖片在 Canny operator 進行處理後的結果。到此一張圖片輸出結束，會繼續讀取下一張圖片，並做一樣的順序，直到所有圖片都被讀取完，即結束程式。

2. 程式執行結果:

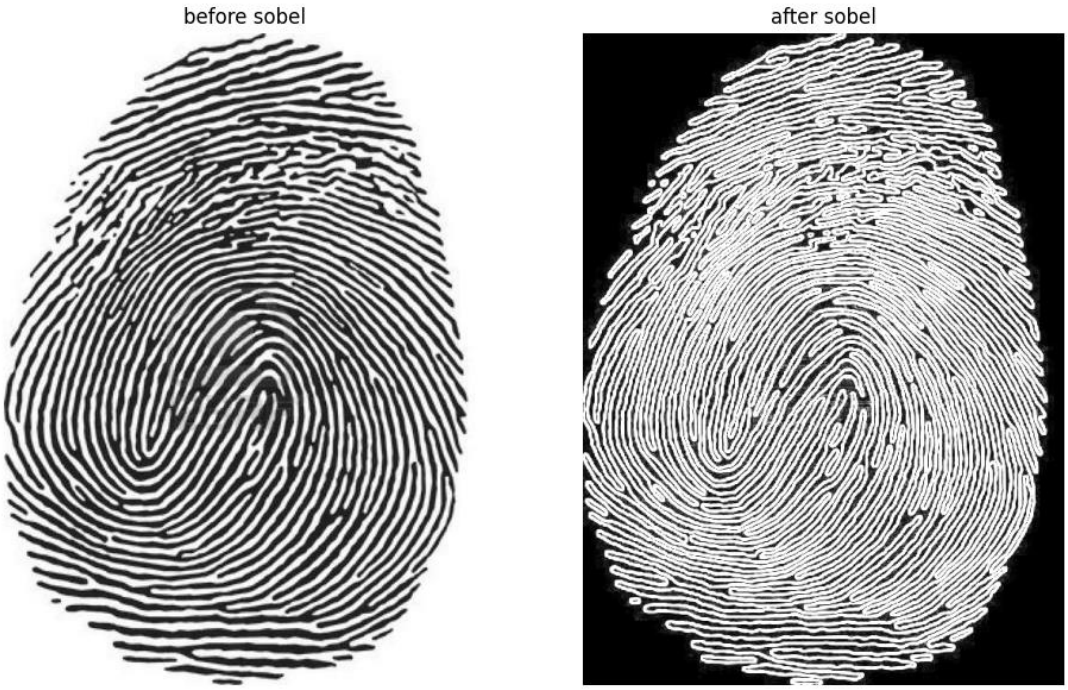
(1) Sobel:



圖(11) image1.jpg 原始以及使用 Sobel 邊緣偵測的結果。

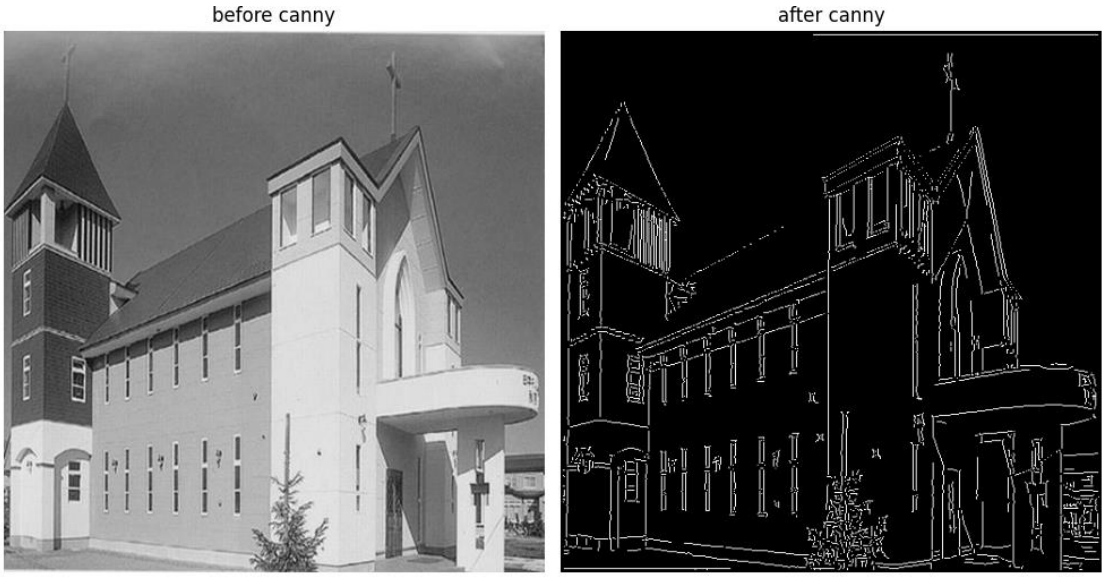


圖(12) image2.jpg 原始以及使用 Sobel 邊緣偵測的結果。

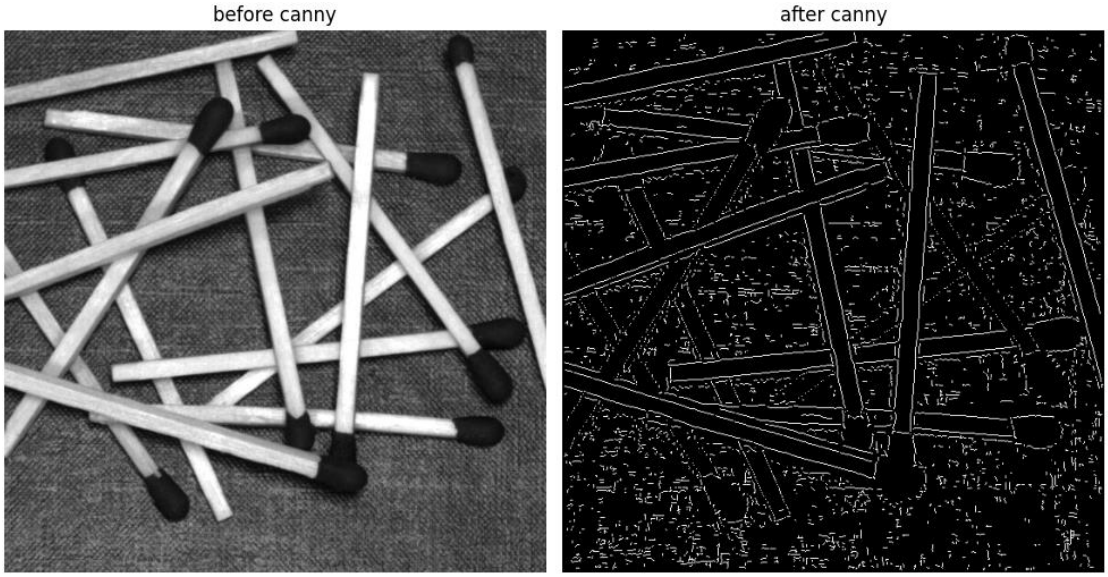


圖(13) image3.jpg 原始以及使用 Sobel 邊緣偵測的結果。

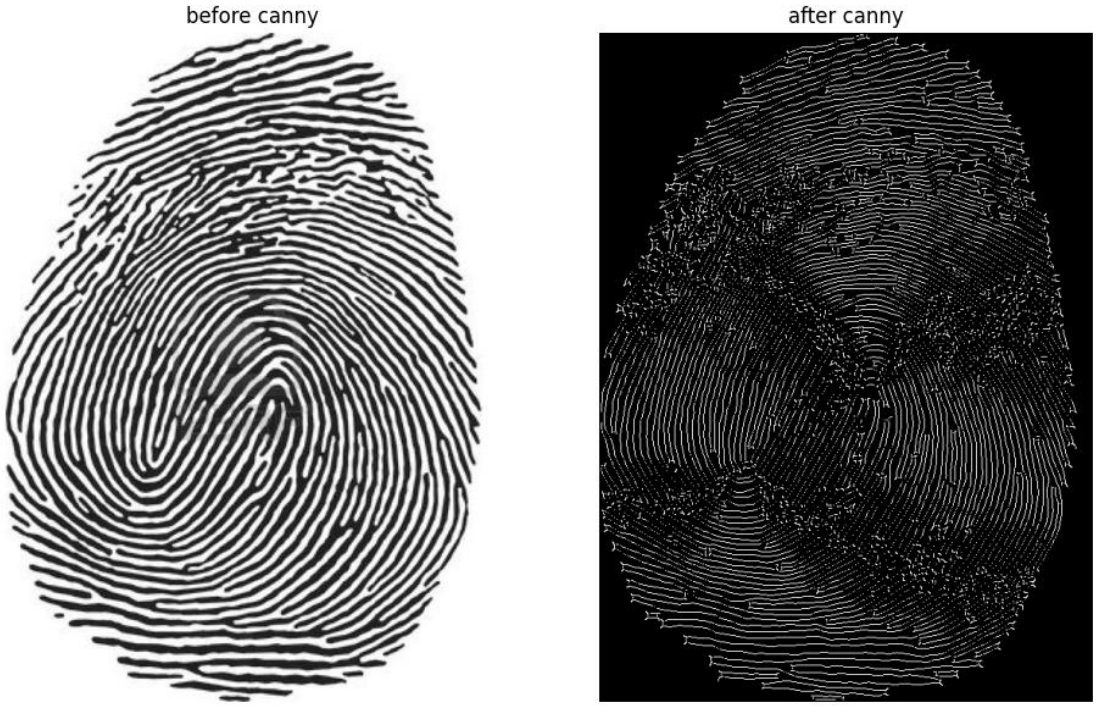
(2) Canny:



圖(14) image1.jpg 原始以及使用 Canny 邊緣偵測的結果。



圖(15) image2.jpg 原始以及使用 Canny 邊緣偵測的結果。



圖(16) image3.jpg 原始以及使用 Canny 邊緣偵測的結果。

Discussions

本次作業的 Sobel 沒什麼懸念，就是套 mask 給要弄的影像就會輸出物體的邊緣出來，但是 canny 因為結合了許多技術，因此在實作上較為困難，又因網路上大部分都為呼叫 opencv 套件的範例，因此較難 debug，bug 之處在於 double threshold 時會因值太小而導致 low threshold 將所有邊緣都殺掉，進而輸出為一個全黑的影像，經由輸出 threshold 前的影像就可發現，大部分邊緣之值沒有落在 low threshold 與 high threshold 間才會這樣。也因此在這次作業時學到了許多!

References and Appendix

canny

https://github.com/cynicphoenix/Canny-Edge-Detector/blob/master/Code/Canny%20Edge%20Detector.ipynb

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

Sobel

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