影像處理HW3

Canny Edge Detection

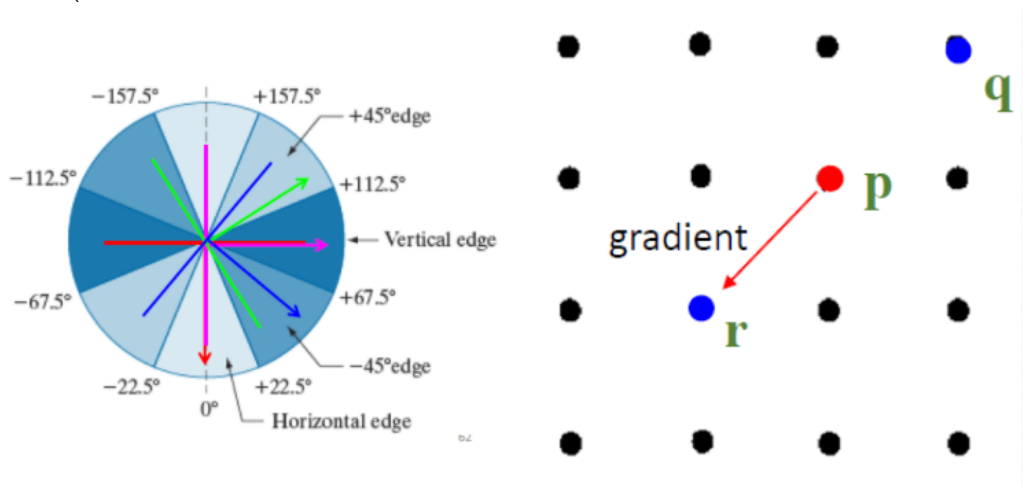
- 做 Gaussian Blur 將影像(模糊化濾掉高頻/雜訊/細節)
  - $Mx,y \approx gx+gy$

```
blur_img = cv2.GaussianBlur(img,(3,3),0)
```

- 計算梯度用Sobel(包含量值和方向性)

```
# 使用Sobel計算梯度(包含量值和方向性)
gx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize = 3)
gy = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize = 3)
# 梯度量值計算
magnitude = np.abs(gx) + np.abs(gy)
# 梯度方向計算
theta = np.degrees(np.arctan2(gy,gx))
```

- 非最大值抑制 (non maximal suppression, NMS)
  - 目的:來去除假的邊緣響應 (spurious response)
  - a.將邊緣依其法向量方向，即梯度方向分成四種：水平、垂直、 +45 、 −45，只保留同方向上連續點中的最大值
  - b.對於某一點 p 若它的梯度值沒有比它 gradient 方向 兩邊的點 q and r 都大，則將其梯度值設為 0 （ 抑制



```
for i in range(1,height - 1):
    for j in range(1,weight - 1):
        # 四個方向(上下、左右、正45度、負45度)，只保留同方向上連續點中的最大值

        if ( ( (theta[i,j] >= -22.5) and (theta[i,j]< 22.5) ) or
                ( (theta[i,j] <= -157.5) and (theta[i,j] >= -180) ) or
                ( (theta[i,j] >= 157.5) and (theta[i,j] < 180) ) ):
            magnitude_max = max(magnitude[i, j - 1], magnitude[i, j],
magnitude[i, j + 1])
            edge[i, j] = magnitude[i, j]
        elif ( ( (theta[i,j] >= 22.5) and (theta[i,j]< 67.5) ) or
                ( (theta[i,j] <= -112.5) and (theta[i,j] >= -157.5) ) ):
            magnitude_max = max(magnitude[i - 1, j - 1], magnitude[i, j],
magnitude[i + 1, j + 1])
            edge[i, j] = magnitude[i, j]

        elif ( ( (theta[i,j] >= 67.5) and (theta[i,j]< 112.5) ) or
                ( (theta[i,j] <= -67.5) and (theta[i,j] >= -112.5) ) ):
            magnitude_max = max(magnitude[i - 1, j], magnitude[i, j],
magnitude[i + 1, j])
            edge[i, j] = magnitude[i, j]

        elif ( ( (theta[i,j] >= 112.5) and (theta[i,j]< 157.5) ) or
                ( (theta[i,j] <= -22.5) and (theta[i,j] >= -67.5) ) ):
            magnitude_max = max(magnitude[i + 1, j - 1], magnitude[i, j],
magnitude[i - 1, j + 1])
            edge[i, j] = magnitude[i, j]
```

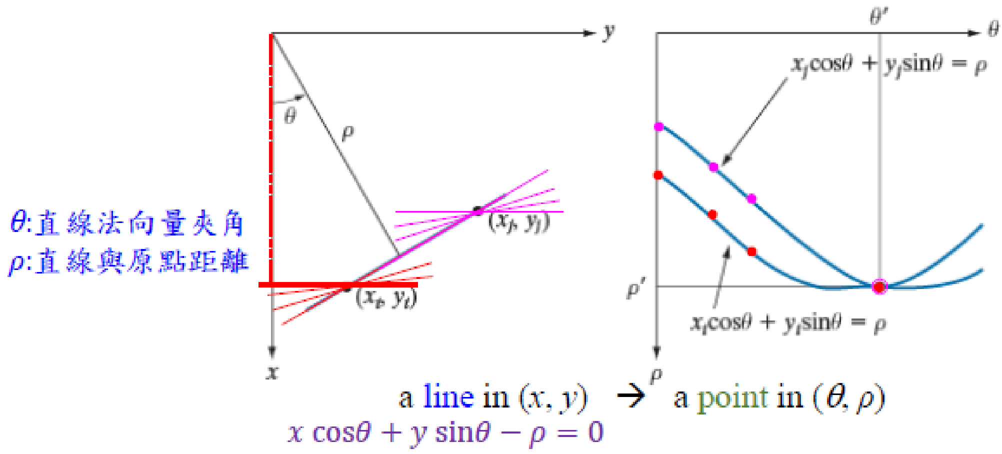
- 雙門檻和連通成份連接斷掉的邊界
    - 1.以高/低門檻 所偵測的邊緣點稱為強/弱像素(建議高低門檻為2:1)
    - 2.強像素皆保留為邊緣點。
    - 3.對於一邊緣點p，其所連接8連通) 的弱像素皆可標成邊緣點。

```
for i in range(height):
    for j in range(weight):
        if edge[i,j] >= 200:
            canvas[i,j] = 255
        elif edge[i,j] <= 100:
            canvas[i,j] = 0
        elif (( edge[i+1,j] < 200) or (edge[i-1,j] < 200 )or( edge[i,j+1] <
200 )or
            (edge[i,j-1] < 200) or (edge[i-1, j-1] < 200 )or ( edge[i-1,
j+1] < 200) or
                ( edge[i+1, j+1] < 200 ) or ( edge[i+1, j-1] < 200) ):
            canvas[i,j] = 255
```

Hough Transform：40%

- 通用之直線方程式：$ax+by+c=0$
- 以法線表示法 (normalrepresentation)：$x\cos\theta+y\sin\theta-\rho=0$
    - θ：直線法向量夾角
      - ρ：直線與原點距離

- 轉換至 (θ,ρ) 平面：line to point


θ:直線法向量夾角
ρ:直線與原點距離
y
x
θ
ρ
(x_i, y_i)
(x_j, y_j)
θ'
θ
$x_j\cos\theta + y_j\sin\theta = \rho$
ρ'
$x_i\cos\theta + y_i\sin\theta = \rho$
ρ
a line in (x, y) → a point in (θ, ρ)
$x\cos\theta + y\sin\theta - \rho = 0$

- 使用OpenCV函數，將 xy 座標影像轉換至Θρ 座標

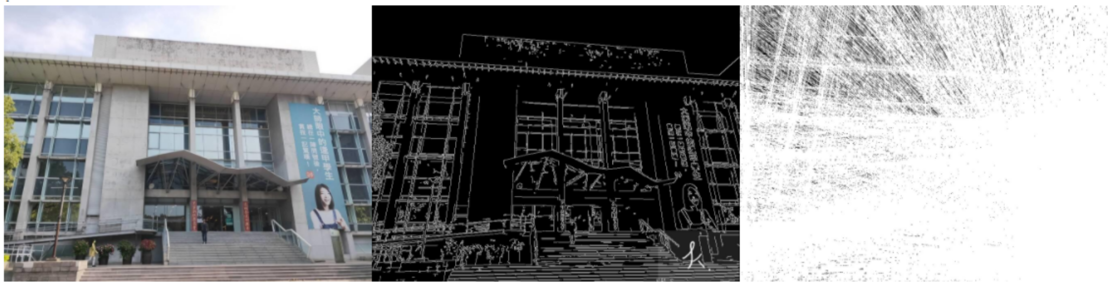
```
lines = cv2.HoughLines( img2, 1, math.pi/180.0, 135 )
if lines :
    a,b,c = lines.shape
    for i in range( a ):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = np.cos( theta )
        b = np.sin( theta )
        x0 = a*rho
        y0 = b*rho
        # line to point
        pt1 = ( int(x0 + 1000*(-b)), int(y0 + 1000*(a)) )
        pt2 = ( int(x0 - 1000*(-b)), int(y0 - 1000*(a)) )
        cv2.line( img2, pt1, pt2, (0, 0, 255), 1)
```

去背簽名檔：10%

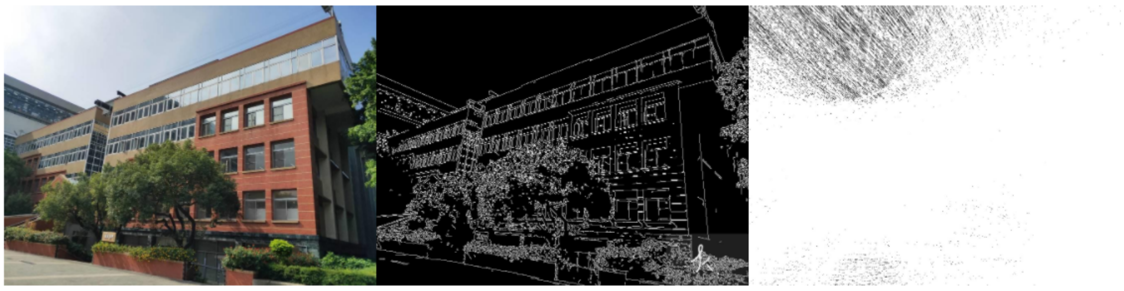
```
canvas[height - 125:,weight - 200:] = cv2.bitwise_or(canvas[height - 125:,weight - 200:],sign)**
```

結果圖：原圖、canny圖、Hough Transform圖

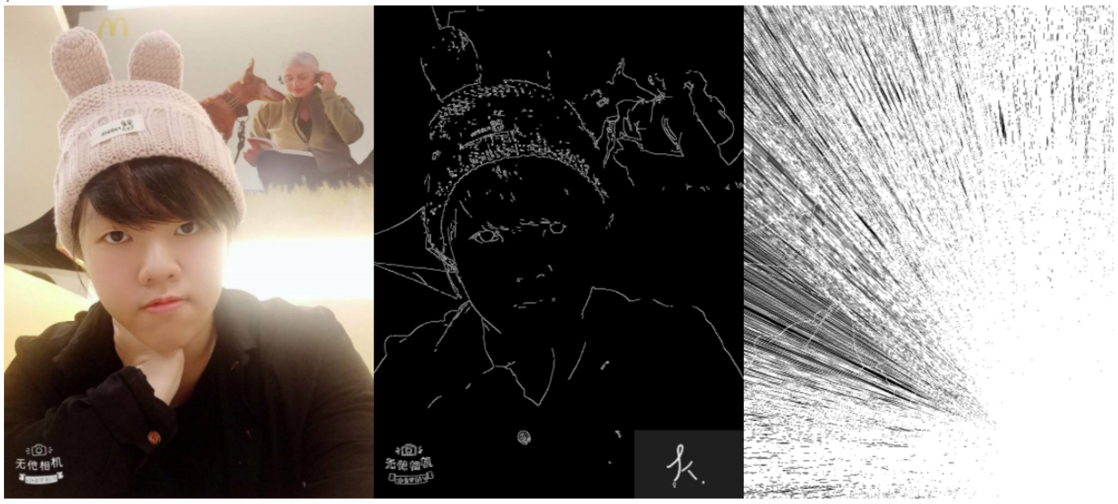
- picture1



- picture2



- picture3



心得

這個作業超級超級超級難的，我寫了六天....
先讀懂如何用sobel算梯度值與方向就花費我一天的時間，理解為

$$gx=\partial f\partial x=z7+2z8+z9-(z1+2z2+z3)$$
$$gy=\partial f\partial y=z3+2z6+z9-(z1+2z4+z7)$$

簡單矩陣乘積阿，不知為何都space error，最後問老師還好是可以直接用sobel，YES!!!!!!!!!!，終於可以繼續往下做，還有找梯度角度也想了超級久的，就是這一行小東西
`theta = np.degrees(np.arctan2(gy,gx))`
接下來做NMS、雙門檻與連通成份分析都蠻簡單、簽名檔都沒甚麼問題，唯一就是在放簽名檔的時候有出現編碼問題。再來的第二個題目霍夫轉換，直接用函式後，在座標圖上畫線，就是按照轉換公式取sin、cos，目的為line to point，只是白線真的超級多啊，有懷疑過是不是canny沒寫好的問題，直接用canny函式測試看看，結果也差不多白哈哈
最後希望不要再出作業了......真的好難哦QAQ

完整程式碼

```
from cv2 import cv2
import numpy as np
import math

# 分別讀取 picture1-3(兩張建築、一張自拍)
img = cv2.imread('picture1.jpg',0)
# 讀取簽名檔
sign = cv2.imread('sign.png',0)
# 模糊化濾掉高頻/雜訊/細節
blur_img = cv2.GaussianBlur(img,(3,3),0)
```

```
# 使用Sobel計算梯度(包含量值和方向性)
gx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize = 3)
gy = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize = 3)
# 梯度量值計算
magnitude = np.abs(gx) + np.abs(gy)
# 梯度方向計算
theta = np.degrees(np.arctan2(gy,gx))

# 非最大值抑制 (non maximal suppression, NMS)
height = magnitude.shape[0]
weight = magnitude.shape[1]
edge = np.zeros_like(magnitude)

for i in range(1,height - 1):
    for j in range(1,weight - 1):
        # 四個方向(上下、左右、正45度、負45度)，只保留同方向上連續點中的最大值

        if ( ( (theta[i,j] >= -22.5) and (theta[i,j]< 22.5) ) or
                ( (theta[i,j] <= -157.5) and (theta[i,j] >= -180) ) or
                ( (theta[i,j] >= 157.5) and (theta[i,j] < 180) ) ):
            magnitude_max = max(magnitude[i, j - 1], magnitude[i, j],
magnitude[i, j + 1])
            edge[i, j] = magnitude[i, j]

        elif ( ( (theta[i,j] >= 22.5) and (theta[i,j]< 67.5) ) or
                ( (theta[i,j] <= -112.5) and (theta[i,j] >= -157.5) ) ):
            magnitude_max = max(magnitude[i - 1, j - 1], magnitude[i, j],
magnitude[i + 1, j + 1])
            edge[i, j] = magnitude[i, j]

        elif ( ( (theta[i,j] >= 67.5) and (theta[i,j]< 112.5) ) or
                ( (theta[i,j] <= -67.5) and (theta[i,j] >= -112.5) ) ):
            magnitude_max = max(magnitude[i - 1, j], magnitude[i, j],
magnitude[i + 1, j])
            edge[i, j] = magnitude[i, j]

        elif ( ( (theta[i,j] >= 112.5) and (theta[i,j]< 157.5) ) or
                ( (theta[i,j] <= -22.5) and (theta[i,j] >= -67.5) ) ):
            magnitude_max = max(magnitude[i + 1, j - 1], magnitude[i, j],
magnitude[i - 1, j + 1])
            edge[i, j] = magnitude[i, j]

# 雙門檻與連通成份分析，高低門檻為200、100(2:1)
height = edge.shape[0]
weight = edge.shape[1]
canvas = np.zeros_like(edge)
for i in range(height):
    for j in range(weight):
        if edge[i,j] >= 200:
            canvas[i,j] = 255
        elif edge[i,j] <= 100:
            canvas[i,j] = 0
        elif (( edge[i+1,j] < 200) or (edge[i-1,j] < 200 )or( edge[i,j+1] < 200
)or
            (edge[i,j-1] < 200) or (edge[i-1, j-1] < 200 )or ( edge[i-1, j+1] <
200) or
                ( edge[i+1, j+1] < 200 ) or ( edge[i+1, j-1] < 200) ):
```

```
            canvas[i,j] = 255
canvas = np.uint8(edge)

# 去背簽名檔(size = 200*125)
height = canvas.shape[0]
weight = canvas.shape[1]
canvas[height - 125:,weight - 200:] = cv2.bitwise_or(canvas[height - 125:,weight
- 200:],sign)

# Hough Transform
img2 = canvas.copy( )
# 參數1 : 灰度圖、參數2與 : 分別是\rho和\theta的精確度、參數4:閾值T
lines = cv2.HoughLines( img2, 1, math.pi/180.0, 135 )
if lines :
    a,b,c = lines.shape
    for i in range( a ):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = np.cos( theta )
        b = np.sin( theta )
        x0 = a*rho
        y0 = b*rho
        # line to point
        pt1 = ( int(x0 + 1000*(-b)), int(y0 + 1000*(a)) )
        pt2 = ( int(x0 - 1000*(-b)), int(y0 - 1000*(a)) )
        cv2.line( img2, pt1, pt2, (0, 0, 255), 1)

# cv2.imwrite('canny3.jpg',canvas)
# cv2.imwrite('Hough3.jpg',img2)

cv2.imshow('canny',canvas)
cv2.imshow( "Hough", img2 )

cv2.waitKey(0)
cv2.destroyAllWindows()
```