影像處理 - 找出最長直線

5105056017 黃凱鴻

簡述

我們可以輕易的對影像處理，得到其邊緣的影像。但有時候可能會因為雜訊，或是解析度問題，這些線段會是不連續的。又或是想透過主要線段，對影像做角度校正至零度。這時候都需要找出影像中的特定線段。

在本次作業中，會先將影像中其亮度低於自定閥值的像素去掉，只留下足夠明顯的部份。再對每個對做 Hough Transform，轉到 pθ 空間，最後找出最多共線的點，將其標注出來。

實作

尋找邊緣

(左: 原始圖片 source.jpg; 右 : 尋找原圖的邊緣 edgeImg.jpg)

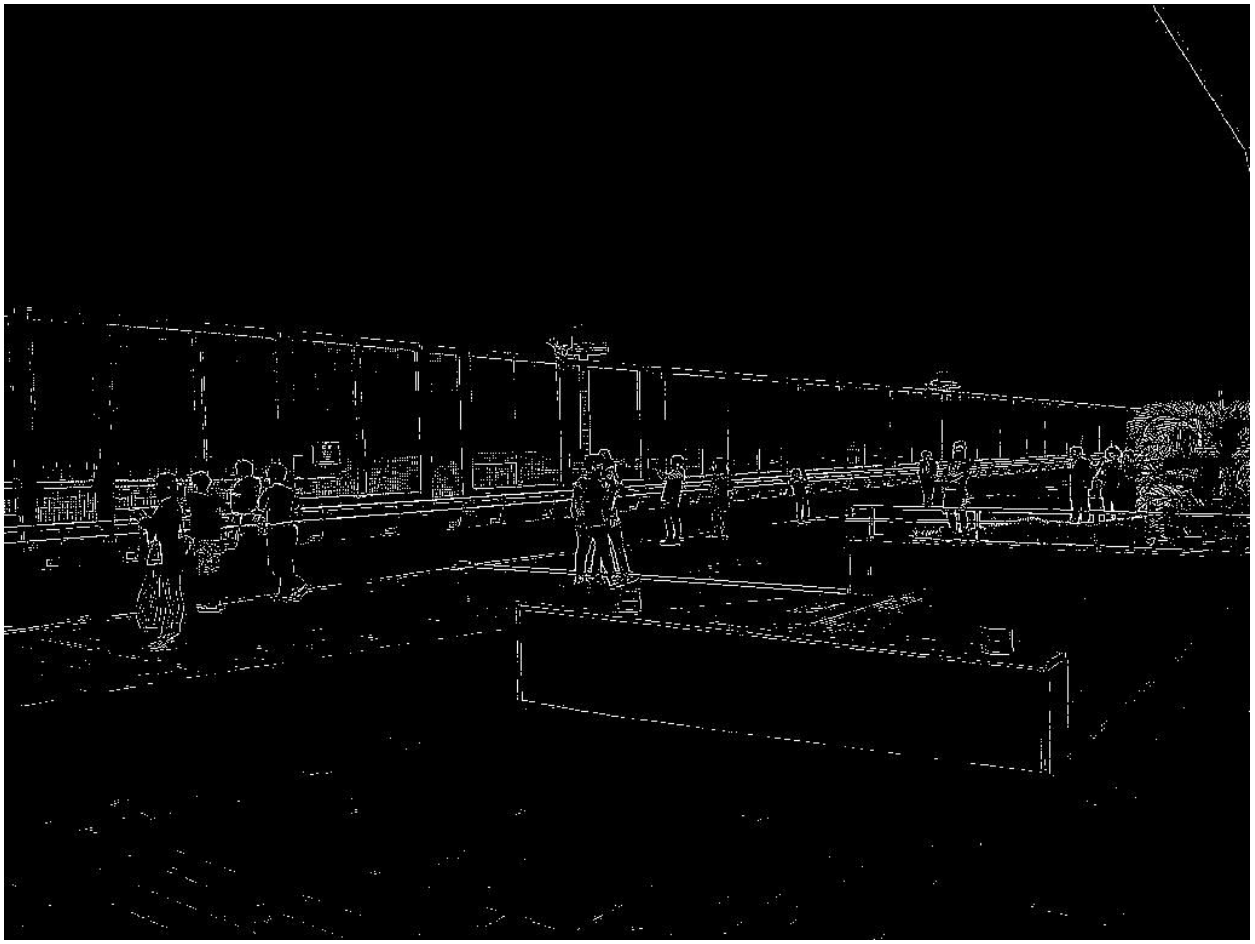
由於尋找邊緣已在上次作業實作過了，本次便直接使用 PIL 內建功能處理，不再重覆。

```
edgeImage = source.filter(ImageFilter.FIND_EDGES)
edgeImage.save('edgeImg.jpg')
```

去除微弱信號

對上一張圖片再掃過一次，只留下亮度 > 128的值，並增強至 255，其餘則降到 0 。以強化線條。

```
for j in range(img.height-2):
    for i in range(img.width-2):
        if pixels[i+1,j+1] < threshold:
            outputPixels[i,j] = 0
        else :
            outputPixels[i,j] = 255
output.save(fileName)
return output
```



(只留下足夠亮的線段 : binary.jpg)

計算 pθ 空間

對上一步的影像，每一個有值的像素做計算，去尋找其在 pθ 空間上的所有參數。然而理論上其實參數是無限多的，必須給定區間計算。

θ 會落在 [-π, π] 之間，本次將其切成 1000 等份，並與 (x, y) 代入 hough transform 得到 p。由於等所有點都計算之後才能得知 p 的區間，因此此時只先保存所有參數資料。

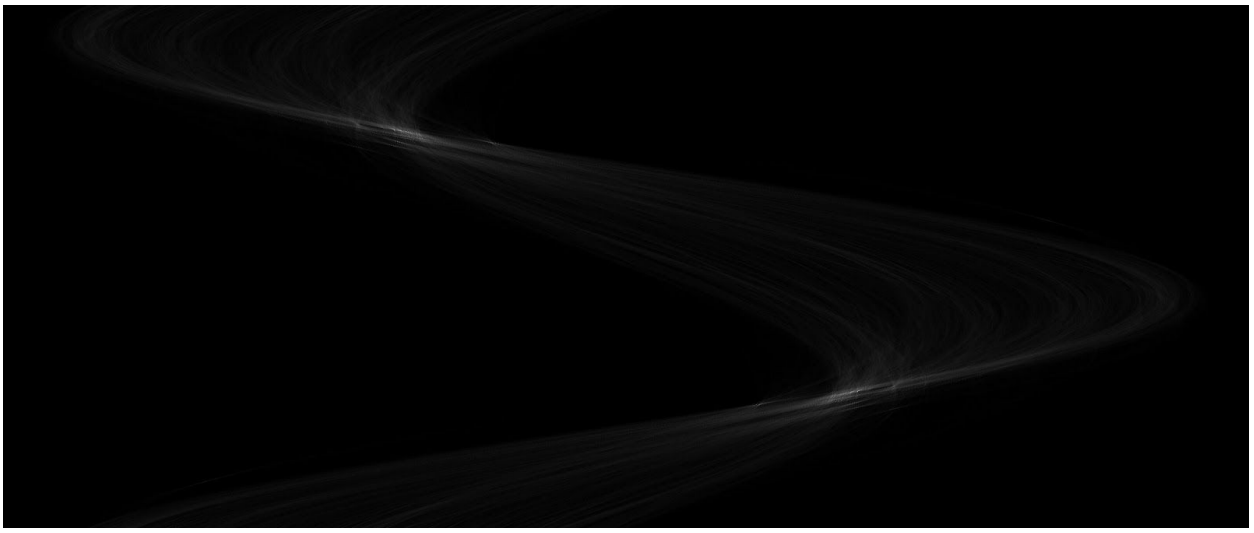
```
def computerImageHough(img):
    sliceNum = 1000
    #output = Image.new("L",(img.width,img.height))
    #outputPixels = output.load()
    pixels = img.load()
    outputPixels = []

    minP = 0
    maxP = 0

    for j in range(img.height):
        outputPixels.append([])
        for i in range(img.width):
            if pixels[i,j] != 0:
                lineP = []
                # 算 (i,j) pixel 在所有 angle 上的 p，並記錄下來
                for angleCount in range(sliceNum):
                    angle = (2 * angleCount / sliceNum - 1) * math.pi
                    p = houghTransform(i,j,angle)
                    #print(p)
                    lineP.append(p)
                    # 記錄上下限，以便後面產生圖片
                    if p < minP:
                        minP = p
                    if p > maxP:
                        maxP = p
                outputPixels[j].append(lineP)
            else :
                outputPixels[j].append([])


    return (outputPixels,minP,maxP)
```

在所有像素都計算出其在 pθ 空間上的參數後，便可得知 p 的最大值與最小值，因此便可得知其距離為 max - min = length。這值會在後面產生圖片時用到。

畫出 pθ 空間

建立一張新的圖片，其寬為前面自定的 1000，高為 max - min，也就是 p 的範圍。換句話說，每個 pixel 的 (x, y) 就對應著 pθ 空間的 (θ, p)。

接下來檢查所有 data，每一組線有經過的像素，其值都遞增 1。同時尋找同時經過最多條線的點與累計值。這是因為要轉成圖片時，其值域必須落在 [0, 255] 之間，因此最後要將累計值做正規化。



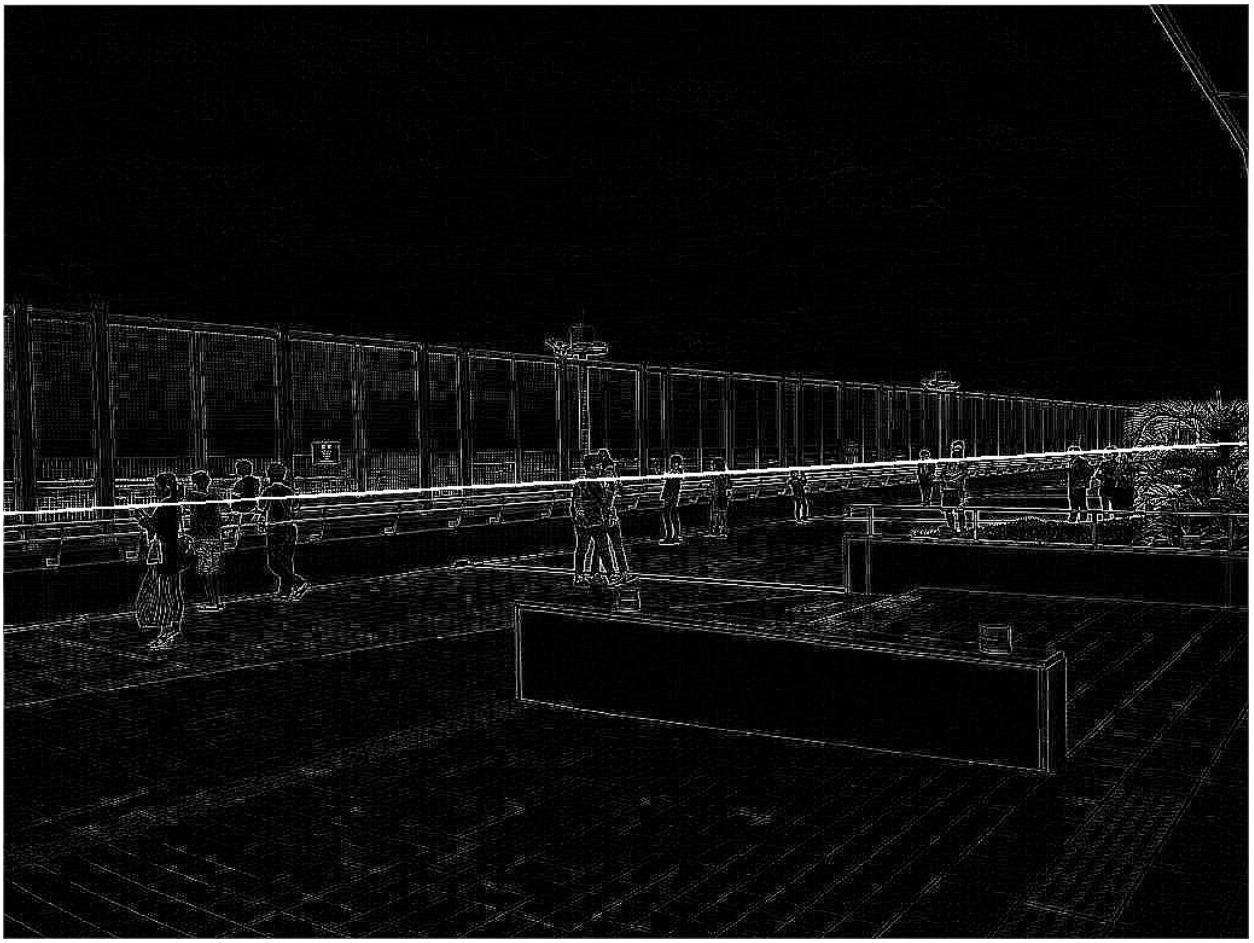
(從pθ空間轉換到2維圖片上的影像成果：Hough.jpg)

找出最長線段

在上一步中，我們已經知道在 pθ 空間中，哪一個點的值是最大的。於是可以再回頭去尋找，到底是哪幾組參數經過於它。再反推回去，便可得到一組原圖中的點座標。

將這些點座標串連起來，便可在原圖中畫出一條線段，即我們想得到的結果。

```
def findAllLine(data, passPoint, minP):
    angle = passPoint[1]
    p = passPoint[0]

    points = []

    for i in range(len(data)):
        row = data[i]
        for j in range(len(row)):
            line = row[j]
            if (len(line) > angle) and int(line[angle] - minP) == p:
                points.append((j,i))
    return points
```



(將所有共線的點畫出一條直線 : result.jpg)