數位影像

Canny :

程式碼：

```python
def Canny_Edge_Detection(image, sign):  #
    # h = int(input("h(高度):"))
    # w = int(input("w(寬度):"))

    rows, cols, _c = image.shape  # (640,640,3)
    degree = np.zeros((rows,cols))
    edge = np.zeros((rows, cols))
    create = np.zeros((rows, cols, 1), np.uint8)  # 建立一個所須範圍的影像矩陣
    create2 = np.zeros((rows, cols, 1), np.uint8)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # print(cols,rows)
    image = cv2.GaussianBlur(image, (5, 5), 0)
    create_final = cv2.Canny(image,30,150)
    cv2.imwrite("img/canny_final.jpeg",create_final)
    for x in range(cols):
        for y in range (rows):
            if (x > 0 and y > 0 and x < (cols - 1) and y < (rows - 1)):
                # 取x方向
                pixel_x = abs((int(image[y+1][x-1]) + (int(image[y+1][x])*2) + int(image
                              (int(image[y-1][x-1]) + (int(image[y-1][x])*2) + int(image[
                # 取y方向
                pixel_y = abs((int(image[y-1][x+1]) + (int(image[y][x+1])*2) + int(image
                              (int(image[y-1][x-1]) + (int(image[y][x-1])*2) + int(image[
                create[y][x] = pixel_x + pixel_y # M(x,y)
```

```python
# 判斷角度取梯度方向
if(pixel_x==0 and pixel_y==0):
    degree[y][x] = -1.0
elif(pixel_x==0):
    degree[y][x]=0.0

elif (pixel_y==0):
    degree[y][x]= 90.0
else:
    degree[y][x] = math.degrees(math.atan(pixel_y/pixel_x))
    if((-22.5 <= degree[y][x] <= 22.5) or (-157.5 <= degree[y][x] <= 157.5)):
        degree[y][x] = 0.0
    elif((22.5 <= degree[y][x] <= 67.5) or (-157.5 <= degree[y][x] <= -112.5)):
        degree[y][x] = -45.0
    elif((67.5 <= degree[y][x] <= 112.5) or (-112.5 <= degree[y][x] <= -67.5)):
        degree[y][x] = 90.0
    elif((112.5 <= degree[y][x] <= 157.5) or (-67.5 <= degree[y][x] <= -22.5)):
        degree[y][x] = 45.0
    else:
        print("wrong")
```

```python
cv2.imwrite("img/M(x,y).jpg",create)
for x in range(cols):
    for y in range(rows):
        if (x > 0 and y > 0 and x < (cols - 1) and y < (rows - 1)):
            # 從角度去使用最大抑制
            if(degree[y][x] == 0.0):
                if ((create[y][x] > create[y-1][x]) or (create[y][x] > create[y-1][x]

                    create[y,x] = 0
            if(degree[y][x] == 45.0):
                if ((create[y][x] > create[y-1][x+1]) or (create[y][x] > create[y+1][

                    create[y, x] = 0
            if(degree[y][x] == 90.0):
                if ((create[y][x] > create[y][x+1]) or (create[y][x] > create[y][x-1]

                    create[y, x] = 0
            if(degree[y][x] == -45.0):
                if ((create[y][x] > create[y-1][x-1]) or (create[y][x] > create[y+1][

                    create[y, x] = 0

cv2.imwrite("img/Non-Maximum.jpg", create)
```

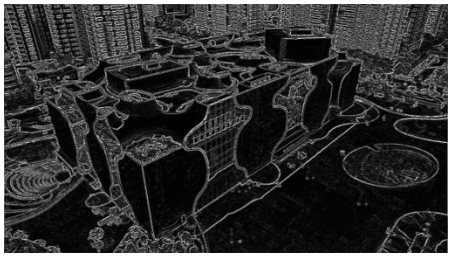
```python
# 雙門檻
for x in range(cols):
    for y in range(rows):
        if(create[y][x] >= 128) :
            edge[y][x] = 1
        elif(create[y][x] >= 64):
            edge[y][x] = 0
        else:
            edge[y][x] =-1

create2 = create
for x in range(cols):
    for y in range(rows):
        if(edge[y][x]< 1):
            create2[y][x] = 0
cv2.imwrite("img/threshold.jpeg", create2)
```

原理：

先利用高斯去除雜訊以及平滑化，再利用 Sobel 來取得梯度跟其向量，再利用非最大抑制去除邊緣效應，然後再設雙門檻和連通成分分析來偵測和連接邊緣。

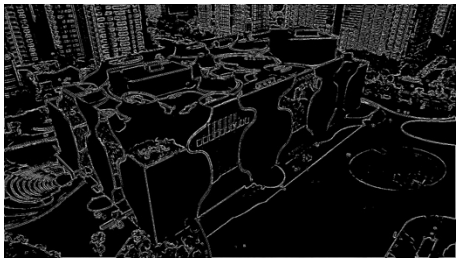
結果：

M(x,y)

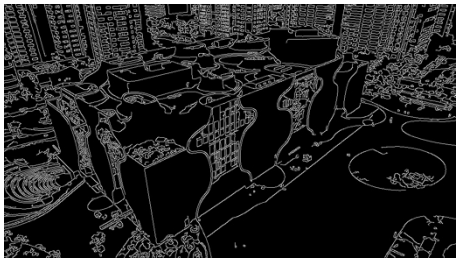


Non-Maximum

Threshold.



Final Canny(函式)



M(x,y)

Non-Maximum

Threshold.

Final Canny(函式)

M(x,y)

Non-Maximum

Threshold.

Final Canny(函式)

Hough transform

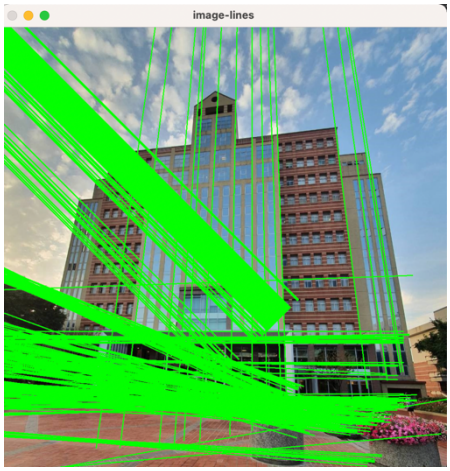
程式碼（函式）：

```python
def line_detection(image):
    print(1)
    gray = cv.cvtColor(image, cv.COLOR_RGB2GRAY)
    edges = cv.Canny(gray, 50, 150, apertureSize=3)  # apertureSize引數預設其實就是3
    cv2.imshow("edges", edges)
    cv2.waitKey(0)
    lines = cv.HoughLines(edges, 1, np.pi/180, 80)
    for line in lines:
        rho, theta = line[0]  # line[0]儲存的是點到直線的極徑和極角，其中極角是弧度表示的。
        a = np.cos(theta)  # theta是弧度
        b = np.sin(theta)
        x0 = a * rho  # 代表x = r * cos（theta）
        y0 = b * rho  # 代表y = r * sin（theta）
        x1 = int(x0 + 1000 * (-b))  # 計算直線起點橫座標
        y1 = int(y0 + 1000 * a)  # 計算起始起點縱座標
        x2 = int(x0 - 1000 * (-b))  # 計算直線終點橫座標
        # 計算直線終點縱座標 注：這裡的數值1000給出了畫出的線段長度範圍大小，數值越小，畫出的線段越短，數值越大，畫出的線段越長
        y2 = int(y0 - 1000 * a)
        cv2.line(image, (x1, y1), (x2, y2), (0, 255,0), 2)  # 點的座標必須是元組，不能是列表。
        cv2.imshow("image-lines", image)
        cv2.waitKey(0)
```

原理：

利用把原圖的每個點畫出他的多個直線，然後再讓其映射到（角度和距離）的另一個向量上，利用 Vote 計算其重複次數，再把投票出來的點返回到原圖的直線上。

結果：



心得：

發現在做的過程中，雖然知道原理，但有些在實作過程中卻時常發生問題，以至於最後只能使用函式，希望我之後有時間可以回來繼續把沒做完的補完。