影像視訊處理實驗作業結報

班　　級	電機	學　　號	M11107S08	姓　　名	王維澤
實驗題目	影像輪廓的擷取				

實驗內容　　*請勿抄襲，否則視為未交

一、　實驗簡介(至少 200 字)：

邊緣檢測 Edge detection 是影像處理，幫助電腦可以抓取圖片中的物體，這項技術以檢查影像中各像素點的顏色變化來區分邊界，有邊緣之後，可以再從這些交錯的線條得到物體輪廓。

邊緣檢測和輪廓檢測比較:

邊緣檢測：檢測圖像中像素的值有突然改變的地方。

輪廓檢測：檢測圖像中的物體邊界，也就像找出物體的邊界。

因此要知道有邊緣的地方不一定有邊界，有邊界的地方也不一定有邊緣。

二、　實驗動機及其解決方法(至少 500 字)：

邊緣檢測的本質是微分，當相鄰兩個像素點的灰度值差異越大時，也就是其斜率越陡，也就是微分值越大，進而通過這個來判斷邊緣，實際中常用差分，x 方向和 y 方向。

Python 的 opencv 提供三種邊緣檢測的方法：Laplacian、Sobel、Canny，這三個技術都是利用灰階圖片，基於每個像素灰值的不同，利用不同物體在其邊界處會有明顯的邊緣特徵來檢測。

Laplacian

可以針對「灰階圖片」，使用拉普拉斯運算子進行偵測邊緣的轉換。

Sobel

Sobel 邊緣檢測演算法使用影象梯度來預測和查詢影象中的邊緣。我們使用該演算法比較畫素密度以檢測邊緣。

我們計算函式的一階導數以找到峰值點。然後將它們與閾值進行比較。

在這種技術中，Sobel 運算元計算函式的梯度。它結合了高斯平滑微分。

通常，我們使用核心來平滑或模糊影象，但在這種情況下，我們將使用它們來計算梯度。沿 x 和 y 軸計算導數。

Canny

第一步：對原始圖像進行灰度化、對圖像進行高斯濾波

第二步：用一階偏導的有限差分來計算梯度的幅值方向

第三步：對梯度幅值進行非極大值抑制

第四步：用雙閾值算法檢測和連接邊緣

方法名稱	原理	技術	使用的圖片
Laplacian	Laplacian Method	透過計算零交越點上光度的二皆導函數	灰階圖片
Sobel	Gradient methods 梯度原理	透過計算像素光度的一皆導數差異	灰階圖片
Canny			

三、 程式碼(須附註解說明，截圖即可)：

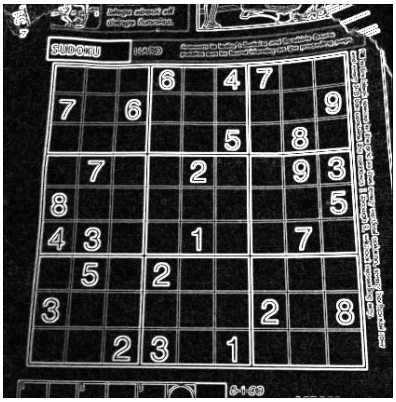
```python
def Sobel(image):
    grad_x = cv.Sobel(image,cv.CV_32F,1,0,ksize=3)  #計算xx方向的導數（末端的1,0表示xx方向）
    grad_y = cv.Sobel(image,cv.CV_32F,0,1,ksize=3)  #計算yy方向的導數（0,1，結尾表示yy方向）
    grad_xy = abs(grad_x)+ abs(grad_y)
    sobel = np.uint8(np.clip(grad_xy,0,255))  #將絕對值圖像轉換為8位元
    return sobel, grad_x, grad_y
```

```python
def Laplacian(image):
    laplacian = cv.Laplacian(image,cv.CV_8U,ksize=3)  #使用 cv2.CV_8U 型態做梯度運
    # cv2.Laplacian(img, ddepth, ksize, scale)
    # img 來源影像
    # ddepth 影像深度，設定 -1 表示使用圖片原本影像深度
    # ksize 運算區域大小，預設 1 ( 必須是正奇數 )
    # scale 縮放比例常數，預設 1 ( 必須是正奇數 )
    return laplacian
```

```python
def Edge_Detection_HoughLine(original_image,image):
    edges = cv.Canny(image,50,200)
    lines = cv.HoughLines(edges, 1, np.pi/180,200)
    if lines is not None:
        for i in range(len(lines)):
            for rho,theta in lines[i]:
                a = np.cos(theta)
                b = np.sin(theta)
                x0 = a*rho
                y0 = b*rho
                x1 = int(x0+1000*(-b))
                y1 = int(y0+1000*(a))
                x2 = int(x0-1000*(-b))
                y2 = int(y0-1000*(a))
                after_HoughLines = cv.line(original_image,(x1,y1),(x2,y2),(255,0,0),1)
    return edges, after_HoughLines

edges,img_line = Edge_Detection_HoughLine(img,img_gray)
cv_imshow(edges)
cv_imshow(img_line)
```

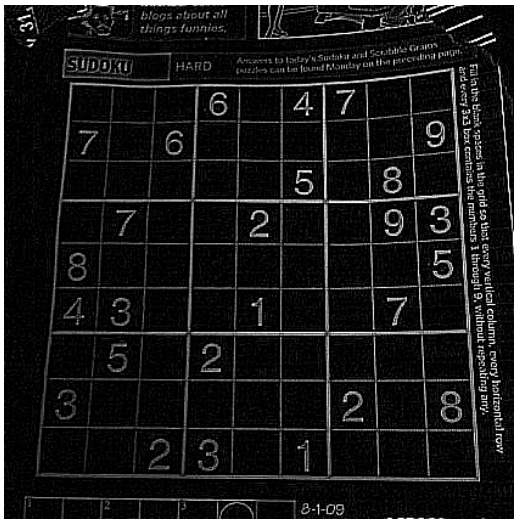
實驗結果(貼圖與簡述說明)：



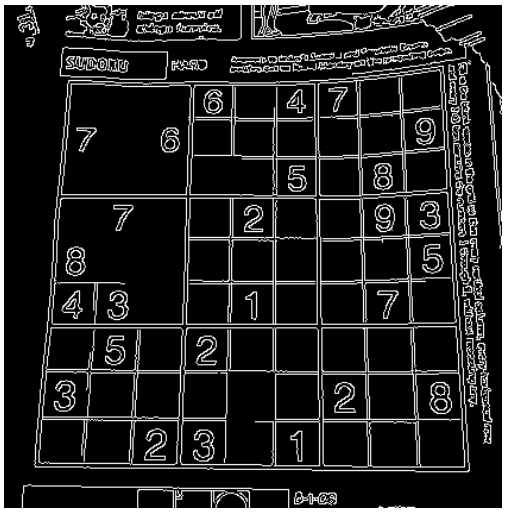
cv_imshow(img_sobel)

cv_imshow(grad_x)

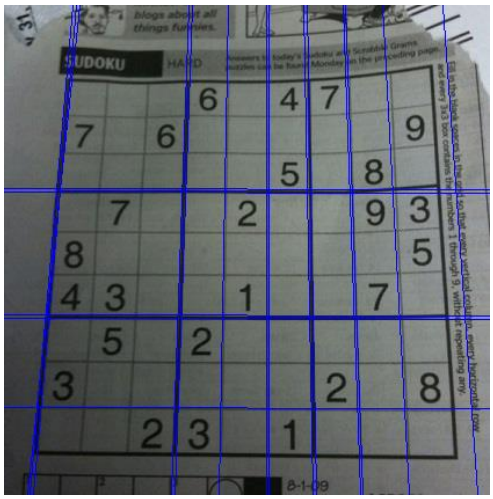
cv_imshow(grad_y)



cv_imshow(res)



cv_imshow(edges)



cv_imshow(img_line)

Hough_circle

(可自行增頁)

*內容字級為 12 字級、中文為標楷體、英文為 Times New Roman