HW3

姓名：蔡育嘉 學號：D0641798

利用OpenCV寫出上課教的邊緣偵測: Canny Edge Detection

與 線段偵測: Hough Transform

# 一、程式碼：

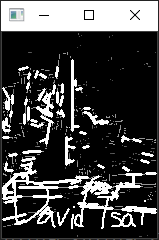
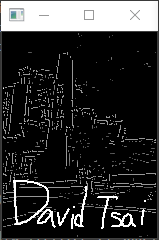
|  |
| --- |
| import cv2 import numpy as np import math  img=cv2.imread("./school.jpg",0) cv2.imshow("Original",img) img = cv2.GaussianBlur(img, (3,3),0)   img\_height,img\_width=img.shape  M=np.zeros([img\_height,img\_width],dtype="int16")  gradient\_x=np.zeros([img\_height,img\_width],dtype="int16")  gradient\_y=np.zeros([img\_height,img\_width],dtype="int16")  sobel\_y=np.array([(1,2,1),(0,0,0),(-1,-2,-1)], dtype='int') sobel\_x=np.array([(-1,0,1),(-2,0,2),(-1,0,1)], dtype='int')  #計算M for h in range(1,img\_height-1):  for w in range(1,img\_width-1):  temp\_x=0  temp\_y=0  for filter\_X in range(len(sobel\_x)):  for filter\_Y in range(len(sobel\_x)):   temp\_x=temp\_x+img[h-1+filter\_X][w-1+filter\_Y]\*sobel\_x[filter\_X][filter\_Y]  temp\_y=temp\_y+img[h-1+filter\_X][w-1+filter\_Y]\*sobel\_y[filter\_X][filter\_Y]   gradient\_x[h][w]=temp\_x  gradient\_y[h][w]=temp\_y    M[h][w]=((temp\_x\*temp\_x)+(temp\_y\*temp\_y))\*\*(0.5)  #Non-Maximum Supperssion for h in range(1,img\_height-1):  for w in range(1,img\_width-1):   gradient\_direction=math.atan(gradient\_y[h][w]/gradient\_x[h][w])  gradient\_direction=gradient\_direction/math.pi\*180  gradient\_direction=gradient\_direction+360  gradient\_direction=gradient\_direction%360  #print(gradient\_direction)  if gradient\_x[h][w]==0: #垂直方向    if M[h][w]>M[h-1][w]:  M[h-1][w]=0  if M[h][w]>M[h+1][w]:  M[h+1][w]=0 #M[h][w] 最大  else:  M[h][w]=0 #M[h+1][w] 最大  else:  M[h][w]=0  if M[h-1][w]>M[h+1][w]:  M[h+1][w]=0 #M[h-1][w] 最大  else:  M[h-1][w]=0 #M[h+1][w] 最大  elif gradient\_y[h][w]==0: #水平方向  if M[h][w]>M[h][w-1]:  M[h][w-1]=0  if M[h][w]>M[h][w+1]:  M[h][w+1]=0 #M[h][w] 最大  else:  M[h][w]=0 #M[h][w+1] 最大  else:  M[h][w]=0  if M[h][w-1]>M[h][w+1]:  M[h][w+1]=0 #M[h][w-1] 最大  else:  M[h][w-1]=0 #M[h][w+1] 最大    elif (gradient\_direction>0 and gradient\_direction<90) or (gradient\_direction>180 and gradient\_direction<270): #方向 右上 左下  if M[h][w]>M[h-1][w-1]:  M[h-1][w-1]=0  if M[h][w]>M[h+1][w+1]:  M[h+1][w+1]=0 #M[h][w] 最大  else:  M[h][w]=0 #M[h+1][w+1] 最大  else:  M[h][w]=0  if M[h-1][w-1]>M[h+1][w+1]:  M[h+1][w+1]=0 #M[h-1][w-1] 最大  else:  M[h-1][w-1]=0 #M[h+1][w+1] 最大       elif (gradient\_direction>90 and gradient\_direction<180) or (gradient\_direction>270 and gradient\_direction<360): #方向 左下到右上   if M[h][w]>M[h-1][w+1]:  M[h-1][w+1]=0  if M[h][w]>M[h+1][w-1]:  M[h+1][w-1]=0 #M[h][w] 最大  else:  M[h][w]=0 #M[h+1][w-1] 最大  else:  M[h][w]=0  if M[h-1][w+1]>M[h+1][w-1]:  M[h+1][w-1]=0 #M[h-1][w+1] 最大  else:  M[h-1][w+1]=0 #M[h+1][w-1] 最大   #雙門檻和連通成份連接斷掉的邊界  high\_threshold=150 low\_threshold=75 for h in range(1,img\_height-1):  for w in range(1,img\_width-1):   if M[h][w]<low\_threshold:  M[h][w]=0  #elif M[h][w]>120 and M[h][w]<325:  for h in range(1,img\_height-1):  for w in range(1,img\_width-1):   if M[h][w]>=high\_threshold:  if M[h+1][w]>low\_threshold: #上  M[h+1][w]=high\_threshold  if M[h-1][w]>low\_threshold: #下  M[h-1][w]=high\_threshold  if M[h][w-1]>low\_threshold: #左  M[h][w-1]=high\_threshold  if M[h][w+1]>low\_threshold: #右  M[h][w+1]=high\_threshold  if M[h+1][w-1]>low\_threshold: #左上  M[h+1][w-1]=high\_threshold  if M[h-1][w-1]>low\_threshold: #左下  M[h-1][w-1]=high\_threshold  if M[h+1][w+1]>low\_threshold: #右上  M[h+1][w+1]=high\_threshold  if M[h-1][w+1]>low\_threshold: #右下  M[h-1][w+1]=high\_threshold  M=np.uint8(M)  signature=cv2.imread("./me\_signature.png",0) signature\_y,signature\_x=signature.shape   cannyedge=np.copy(M) for i in range(signature\_y):  for j in range(signature\_x):  if signature[i][j]==255:#如果簽名檔為黑 則self\_blur同位置也是黑的  cannyedge[i][j]=255 cv2.imshow("Canny Edge Detection",cannyedge) #M canny edge的圖 #1 rho #np.pi/180 theta #1 最小投票數 #np.array([]) 替代字符? #5 線的最小單位 #10 線跟線之間的距離   #霍夫轉換 lines = cv2.HoughLinesP(M, 1, np.pi/180,5,np.array([]), 5,3) print(lines.shape) print(lines[0].shape) for i in lines:  x1=i[0][0]  y1=i[0][1]  x2=i[0][2]  y2=i[0][3]  print(str(x1)+" "+str(y1)+" "+str(x2)+" "+str(y2))  cv2.line(M, (x1, y1), (x2, y2), (255, 255, 255), 2) #在原圖上畫線  for i in range(signature\_y):  for j in range(signature\_x):  if signature[i][j]==255:#如果簽名檔為黑 則self\_blur同位置也是黑的  M[i][j]=255 cv2.imshow("Hough Transform",M) cv2.waitKey(0) |

# 二、執行結果：

original cannyedge Hough Transform



original cannyedge Hough Transform



original cannyedge Hough Transform



# 三、心得討論：

一開始在做sobel算梯度的時候，結果一直跟opencv算出來的不太一樣，後來發現opencv好像還有額外做幾個處理，不過後面就進行的蠻順利的，再找hough transform的api的時候，發現他有很多的參數可以調，這個是當初沒有想到的，那畫線的部分，則是直接透過houghtransformP的api中找到的直線兩點，透過opencv直接在圖上畫線，最後完成這項作業，最後根據結果圖發現，如果場景太複雜，在做雙門檻值跟連通成分分析，會把線連起來，導致圖片看起來一坨邊緣效果就被影響到。