Multimedia Systems and Applications

# Edge Detection

F74064012黃冠淳

1. **Basic issues**
   * 1. **Histogram equalization**

將影像直方圖均衡化，利用圖像直方圖對對比度進行調整

* + 1. **Sobel operator**

利用投影片slide09-Edge Detection-Supplementary第43頁的矩陣做邊緣偵測

gx: horizontal edge

[-1 -2 -1,

　　　0　0 0,

1　2 1 ]

gy: vertical edge

[-1 0 1,

　　　　　　　　-2 0 2,

-1　0 1 ]

* + 1. **Add your comments in source code**

見程式碼

1. **Bonus**
2. **His\_Clahe** **function**

對比有限的自適應直方圖均衡，用於優化His\_equ

1. **Gaussian\_filter function**

用於去雜訊 (但可能造成對比度下降)

1. **原始程式碼與大略的程式碼解說**

import cv2

import numpy as np

from matplotlib import pyplot as plt

"""

概念:

好的圖像將具有圖像所有區域的像素

如果存在較亮/較暗的圖像，則將所有像素限制在高/低值，使用此方法來修改圖像

將直方圖拉伸到兩端以改善圖像的對比度

"""

def His\_method(image):

img = cv2.imread(image, 0) # 讀取圖檔

equ = cv2.equalizeHist(img) # OpenCV函式，輸入為灰度影象，輸出為直方圖均衡影象。

# flatten() 將陣列變為一維

hist, bins = np.histogram(img.flatten(), 256, [0, 256]) # 256 表有 256+1 bin\_edges ， [0,256] 表範圍

cdf = hist.cumsum() # 將array變為累積分佈函數

cdf\_normalized = cdf \* hist.max() / cdf.max() # 標準化

# 創建一個轉換函數，將原始輸入像素均勻地分佈到整個區域

cdf\_m = np.ma.masked\_equal(cdf, 0)

cdf\_m = (cdf\_m - cdf\_m.min()) \* 255 / (cdf\_m.max() - cdf\_m.min())

cdf = np.ma.filled(cdf\_m, 0).astype('uint8')

img2 = cdf[img]

hist, bins = np.histogram(img2.flatten(), 256, [0, 256])

cdf = hist.cumsum()

cdf\_normalized2 = cdf \* hist.max() / cdf.max()

# 原始圖

plt.subplot(221)

plt.imshow(img, 'gray')

plt.subplot(222)

plt.plot(cdf\_normalized, color='b')

plt.hist(img.flatten(), 256, [0, 256], color='r')

plt.xlim([0, 256])

# 直方均衡化後

plt.subplot(223)

plt.imshow(equ, 'gray')

plt.subplot(224)

plt.plot(cdf\_normalized2, color='b')

plt.hist(equ.flatten(), 256, [0, 256], color='r')

plt.xlim([0, 256])

plt.show()

return

# 使用 OpenCV 的 Histograms Equalization

def His\_Equ(img):

img = cv2.imread(img, 0) # 讀取圖檔

Equ = cv2.equalizeHist(img) # OpenCV函式，輸入為灰度影象，輸出為直方圖均衡影象。

res = np.hstack((img,equ)) # 並排疊加圖片

cv2.imwrite('img\_HisEqu.jpg', Equ) # 寫入圖檔

return

"""

概念:

為改善His\_Equ有時會使對比度過大，使用自適應直方圖均衡來改善

將影象分為幾個小塊，稱“tiles”，

直方圖會限制在一個小區域(除非有噪聲)

為了避免噪音，如果任何直方圖bin超出指定的對比度限制，在直方圖均衡之前，這些畫素被裁剪並均勻地分佈到其他bin

均衡後，刪除邊界中的工件，採用雙線性插值。

"""

def His\_Clahe(img):

img = cv2.imread(img, 0)# 讀取圖檔

# create a CLAHE object (Arguments are optional).

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))# clipLimit表對比度的大小為title大小，(8, 8)為tilte大小

cl1 = clahe.apply(img)

cv2.imwrite('img\_advance.jpg', cl1)

return

"""

概念:

Gaussian\_Filter將給予各點不同的權值，愈靠近中央點的權值愈高，最後再以平均方式計算出中央點

用於消除噪音（平滑）

微分就是convolution

convolution與低通濾波器和intensity function結合

"""

def Gaussian\_filter(img):

image = cv2.imread(img) # 讀取圖檔

# 若增加權重，則輸入圖像會更加模糊

blurred=cv2.GaussianBlur(image,(3,3),0) # （3,3）是過濾器權重，權重必須為奇數

cv2.imwrite('img\_Gaussian.jpg',blurred)

return

"""

概念:

Median\_Filter找出除最中間那個點外的中間值

使用的點是個既存的像素而不是計算出來的像素，

可用在除噪功能上

"""

def Median\_Filter(img):

image = cv2.imread(img) # 讀取圖檔

blurred=cv2.medianBlur(image,3)

cv2.imwrite('img\_Median.jpg', blurred)

return

"""

概念:

主要概念

Sobel\_operator用於邊緣檢測

operator用兩個與原始圖像卷積的3×3內核來計算導數的近似值

如果該值大於 Container，則將其視為邊，反之亦然

"""

def Sobel\_operator(img):

Container = np.copy(img)

Size = Container.shape

for i in range(1, Size[0] - 1):

for j in range(1, Size[1] - 1):

Gx = (img[i - 1][j - 1] + 2 \* img[i][j - 1] + img[i + 1][j - 1]) - (img[i - 1][j + 1] + 2 \* img[i][j + 1] + img[i + 1][j + 1])

Gy = (img[i - 1][j - 1] + 2 \* img[i - 1][j] + img[i - 1][j + 1]) - (img[i + 1][j - 1] + 2 \* img[i + 1][j] + img[i + 1][j + 1])

Container[i][j] = min(255, np.sqrt(Gx \*\* 2 + Gy \*\* 2)) # 將梯度近似值與x和y方向的近似值組合

return Container

pass

return

# 蜘蛛圖

img='img/Spider.jpg'

Gaussian\_filter(img)

#Median\_Filter(img)

His\_method(img)

His\_Clahe('img\_Gaussian.jpg')

#His\_Equ('img\_Gaussian.jpg')

img = cv2.cvtColor(cv2.imread("img\_advance.jpg"), cv2.COLOR\_BGR2GRAY)

img = Sobel\_operator(img)

img = cv2.cvtColor(img, cv2.COLOR\_GRAY2RGB)

cv2.imwrite('Spider\_Gaussian.jpg',img)

#His\_Equ(img)

#His\_Clahe(img)

#Gaussian\_filter('img\_advance.jpg')

# 飛機圖

img='img/Airplane.jpg'

Gaussian\_filter(img)

#Median\_Filter(img)

His\_method(img)

His\_Clahe('img\_Gaussian.jpg')

#His\_Equ('img\_Gaussian.jpg')

img = cv2.cvtColor(cv2.imread("img\_advance.jpg"), cv2.COLOR\_BGR2GRAY)

img = Sobel\_operator(img)

img = cv2.cvtColor(img, cv2.COLOR\_GRAY2RGB)

cv2.imwrite('Airplane\_Gaussian.jpg',img)

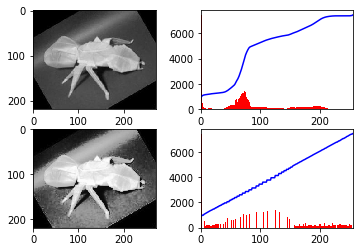
1. **Result**

**Spider**

**原圖：**

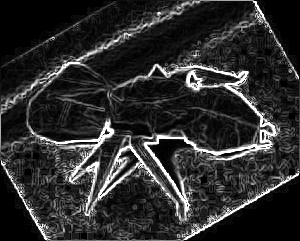
****

**Histogram equalization:**

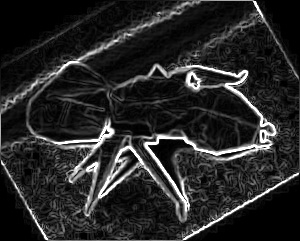
****

**Sobel operator :**

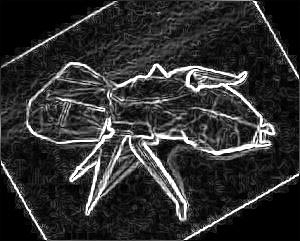
**His\_Equ + Sobel\_operator**

****

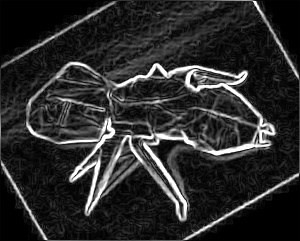
**His\_Equ + Sobel\_operator + Gaussian\_filter**

****

**His\_Clahe + Sobel\_operator**

****

**(最佳) His\_Clahe + Sobel\_operator + Gaussian\_filter**

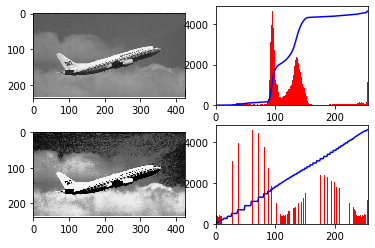
****

**Airplane**

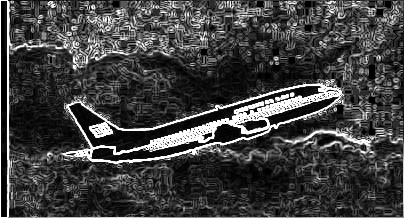
**原圖：**

****

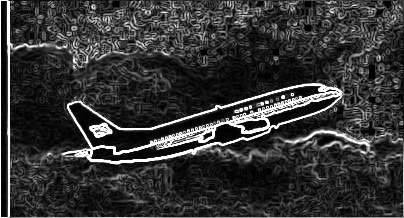
**Histogram equalization:**

****

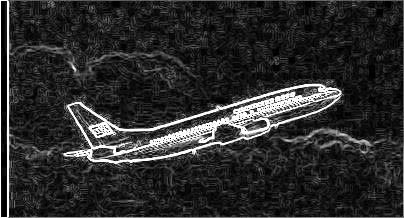
**His\_Equ + Sobel\_operator**

****

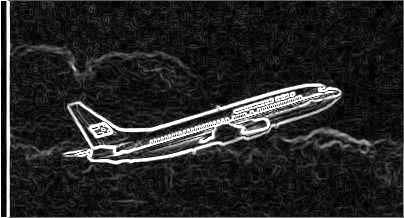
**His\_Equ + Sobel\_operator + Gaussian\_filter**

****

**His\_Clahe + Sobel\_operator：**

****

**(最佳) His\_Clahe + Sobel\_operator + Gaussian\_filter：**

****