影像處理

HW1

姓名: 張少鈞
學號: P76114545

1. RGB Extraction & transformation

1.1 Problem

將一張彩色影像的紅、綠、藍分別萃取出來，並將此影像轉為灰階影像。

1.2 Method

每張彩色影像均由紅、綠、藍，在 C#中將影像儲存成 Bitmap 的格式後，可以透過函式 1 取得影像各點的紅、綠、藍的數值，以此來達成顏色萃取的功能，並透過萃取出來的數值透過式 1 計算出相對應的灰階值，使原始的影像轉變為灰階影像。

$$\text{public System. Drawing. Color GetPixel (int x, int y);} \qquad \text{函式 1}$$

$$G = 0.07B + 0.72G + 0.21R \qquad \text{式 1}$$

1.3 Result

將彩色影像萃取出來的結果如圖 1。

(a) (b) (c) (d) (e)

圖 1 彩色影像萃取結果。(a)彩色原圖(b)紅色萃取(c)綠色萃取(d)藍色萃取(e)灰階影像

2. Smooth filter

2.1 Problem

使用均值濾波器(Mean filter)與中值濾波器(Median filter)將原影像進行濾波。

2.2 Method

均值濾波器採用鄰近平分法來進行，可以透過使用均值濾波遮罩(如圖 2)對原影像進行捲積來達成。

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

圖 2 均值濾波遮罩

中值濾波器則須在相鄰的像素素值中找到中位數來進行取代，如圖 3 的範例中相鄰的像素數值依序排列為$[2, 3, 5, 5, 5, 7, 8, 8, 11]$，數列的中位數為 5，因此正中心的數值將被更改為 5。

5	8	11
2	3	5
5	8	7

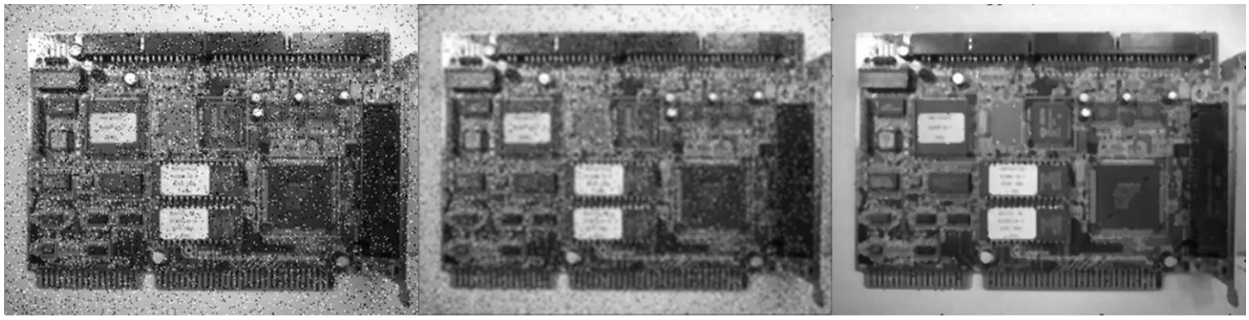
(a)

5	8	11
2	5	5
5	8	7

(b)

圖 3 中值濾波範例。(a)原始數值 (b)中值濾波後結果

2.3 Result



(a) (b) (c)

圖 4 平滑濾波。(a)原始影像 (b)均值濾波結果 (c)中值濾波結果

3. Histogram Equalization

3.1 Problem

實作直方圖等化(Histogram Equalization)增加影像的對比度。

3.2 Method

統計原始影像每個像素的灰階數值化出原始影像的直方圖，如圖 5。

考慮一個灰階影像$\{x\}$，讓n_i表示灰階i出現的次數，n為圖像所有的像素數量，則圖像中灰階i的出現機率可以表示為:

$$p_x(i) = p(x = i) = n_i/n, \qquad 0 \le i < 255 \qquad \text{式 2}$$

將對應於p_x的累積分布函數定義為:

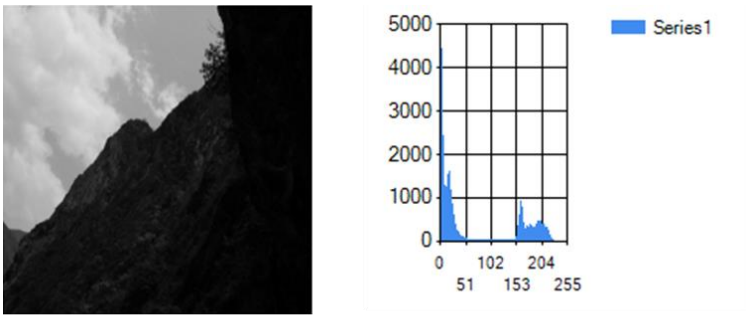
$$cdf_x(i) = \sum_{j=0}^{i} p_x(j), \qquad 0 \le i < 255 \qquad \text{式 3}$$

灰階值i透過累積分布函數重新定義為新的灰階值$G(i)$:

$$\mathrm{G}(i) = cdf_x(i) \times 255, \qquad 0 \le i < 255 \qquad \text{式 4}$$

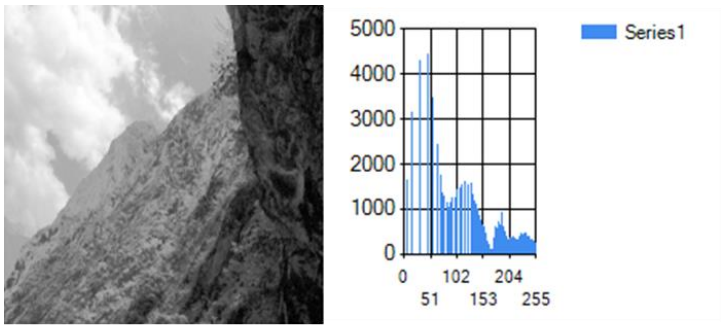
透過式 4 可以將原始的影像轉換為等化過後的影像，如圖 6。

3.3 Result



(a) (b)

圖 5 (a)原始影像(b)原始直方圖



(a) (b)

圖 6 (a)等化後影像(b)等化後直方圖

4. A user-defined thresholding

4.1 Problem

讓使用者透過滑桿決定一個 1~255 的閥值 t，將影像強度小於此閥值的像素設定為黑色，其餘設定為白色。

4.2 Method

使用 C#建立一個滑桿讓使用者設定閥值，當按下執行按鈕時透過函式 2 來讀取使用者的輸入數字後，使用函式 1 來確定每個像素的影像強度，小於閥值的像素設定為黑色，其餘設定為白色，可以使用函式 3 來進行像素的設定。

$$public\ int\ Value\ \{get; set;\}$$ 函式 2

$$public\ void\ SetPixel\ (int\ x, int\ y, System.Drawing.Color\ color);$$ 函式 3

4.3 Result

(a) (b)

圖 7 使用者自訂閥值。(a)原始影像(b)自訂閥值處理後影像

5. Sobel edge detection

5.1 Problem

將一張影像中的水平、垂直邊緣檢測出來。

5.2 Method

索伯算子(Sobel Operator)是利用梯度向量將邊緣萃取出來，分別設計水平與垂直的濾波遮罩如圖 8，將兩遮罩分別對原影像進行捲積萃取水平與垂直邊界。捲積過後的結果具有方向性，因此必須將結果取絕對值，在數值方面也要將超過 255 的數值設定為 255。

-1	0	+1
-2	0	+2
-1	0	+1

(a)

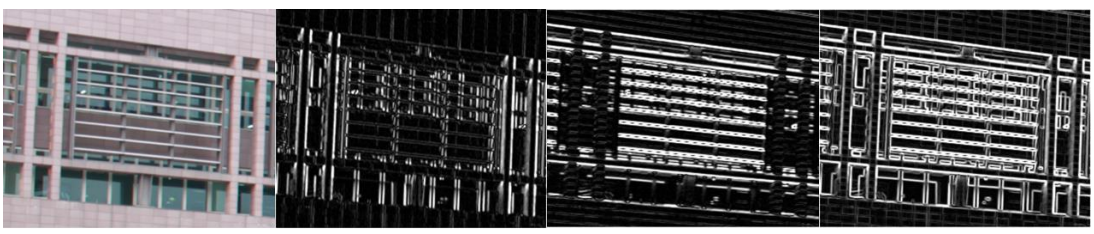
+1	+2	+1
0	0	0
-1	-2	-1

(b)

圖 8 索伯算子遮罩。(a)垂直遮罩(b)水平遮罩

分別將水平與垂直邊緣檢測過後的結果進行合併，即可在同一張圖上顯示出水平與垂直的檢測結果。

5.3 Result



(a) (b) (c) (d)

圖 9 索伯檢測。(a)原始影像(b)垂直邊緣檢測(c)水平邊緣檢測(d)邊緣檢測合併

6. Edge overlapping

6.1 Problem

原始圖片透過邊緣檢測得到結果後，讓使用者透過介面上的滑桿來設定一個閥值 t，利用這個閥值將邊緣檢測的結果二值化，最後重疊於原始影像中。

6.2 Method

透過 C#建立滑桿來讓使用者設定閥值 t，當按下開始鍵後檢查邊緣檢測後的影像，將影像強度小於閥值的像素設定為黑色，其餘設定為白色(如圖 10(c))，最後將顏色為白色的像素以綠色重疊於原始影像中並輸出(如圖 10(d))。

6.3 Result

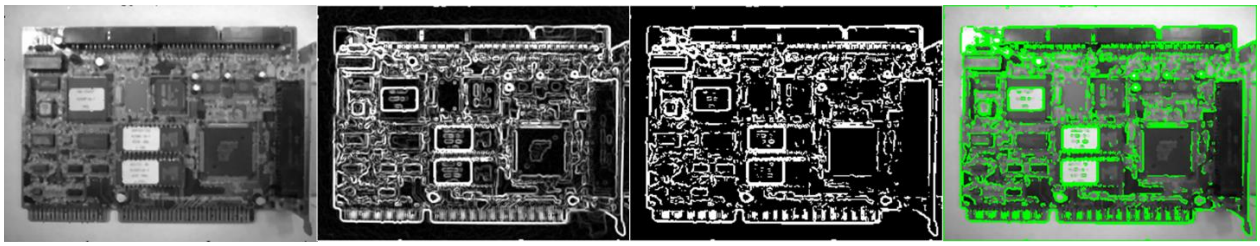


圖 10 邊緣重疊。(a)原始影像 (b)邊緣檢測後結果 (c)邊緣檢測結果透過閥值二值化 (d)將(c)的結果以綠色與原始影像重疊。

7. Connected Component

7.1 Problem

輸入一個只包含黑色與白色的二值化影像，將相鄰的黑色組成群組並標色，最終輸出分群結果的圖片與數量。

7.2 Method

從第一個像素開始進行搜尋，當碰到黑色時就會使用遞迴的方式來檢查周邊的鄰居進行分群並標色。

7.3 Result

圖 11 (a)原始影像 (b)分群結果

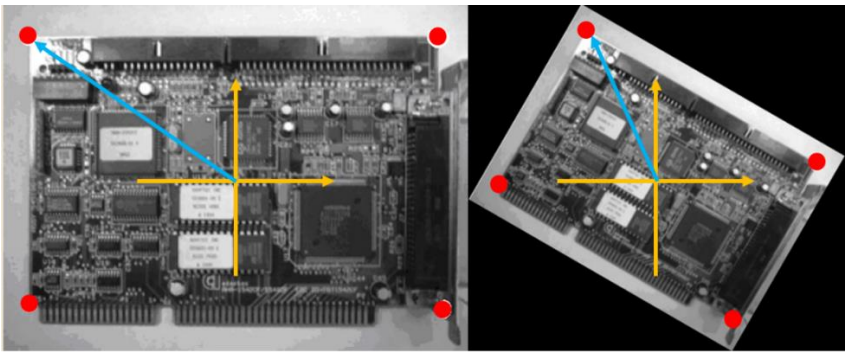
8. Image registration

8.1 Problem

給予兩張影像 A 與 B，影像 B 是由 A 經過等比縮放與旋轉後的結果，透過依序點選兩影像中的四個點，將 B 影像還原成 A 影像，並輸出縮放比例、旋轉角度與影像強度差。

8.2 Method

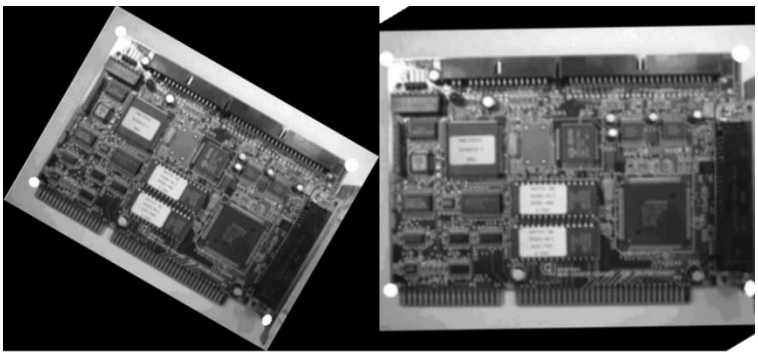
透過依序點擊的四個點以影像中心為原點產生向量(如圖 12 中的藍色箭頭)，透過兩個向量計算出旋轉的角度與縮放倍率，總共會計算出四組數值，將這四組數值取平均後即為影像的旋轉角度與縮放倍率。而影像的旋轉方向則由向量的外積方向來進行決定。依據計算出來的數值將影像 B 還原成影像 A，並透過計算各點的平均強度差來評斷還原的好壞。



(a) (b)

圖 12 影像定位。(a)影像 A (b)影像 B

8.3 Result



(a) (b)

圖 13 影像定位結果。(a)原始影像 B (b)B 影像還原結果

Discussion

本次作業透過自己實作各種影像處理的基本功能，比起以往使用函式來進行相同的影像處理，更能理解到其背後的原理與涵義，簡易的 UI 介面也能更方便觀察各個步驟的細微差異，對於影像處理有更深的了解。