影像處理 HW1

P76104231 蔣有為

Problem A : RGB Extraction & Transformation

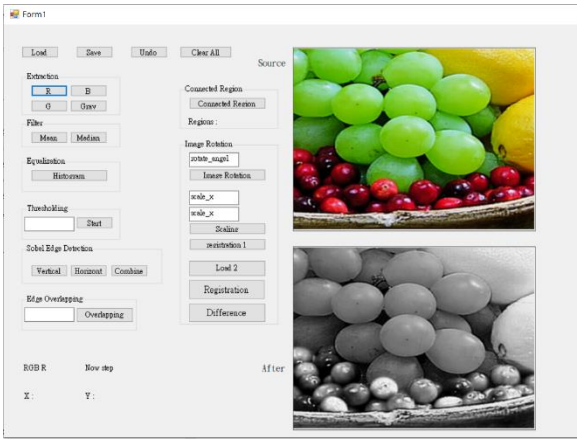
Method :

成功開啟照片後，讀取彩色原圖的每個 Pixel 的 RGB 三通到的色彩資訊，分別將 Pixel 分別儲存 RGB.R、RGB.G、RGB.B，建構出只具有 R、G、B channel 的圖片，而 gray scale 則是考量到人眼的亮度感受，採用具有權重的方式：Gray = 0.299 * RGB.R + 0.587 * RGB.G + 0.114 * RGB.B。

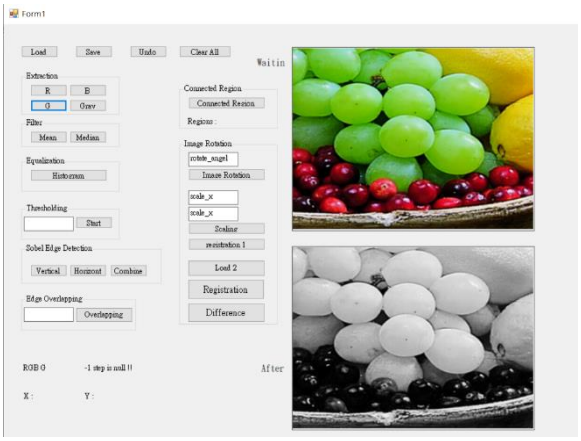
Result :

Color extraction

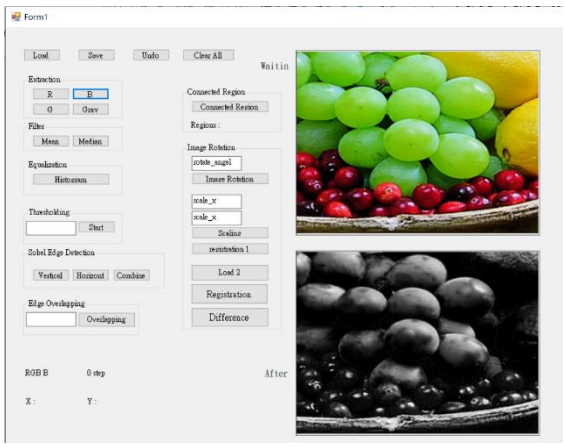
RGB.R :



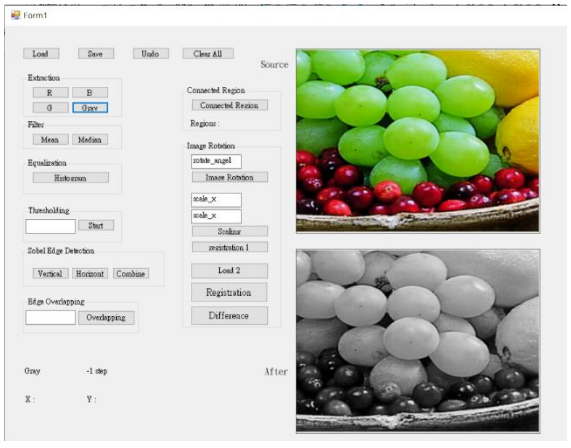
RGB.G :



RGB.B :



Color transformation , Gray Level :



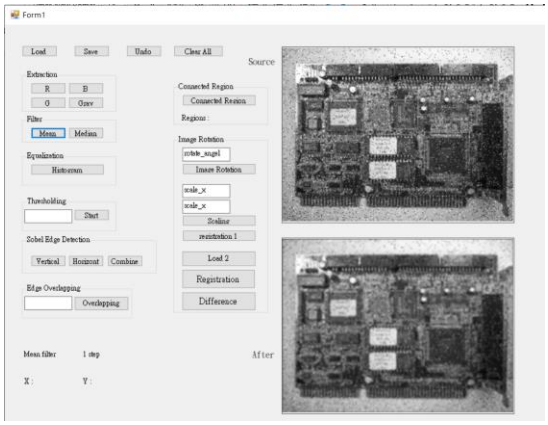
Discussion : 這題因為考量到後續 Undo 的設計，正常的使用流程會一張圖片只進行一次的結果，所以將 RGB 與 Gray level 處理後的照片分別用不同的按鈕顯示在同一個 Picture Box 上。

Problem B : Smooth filter (mean and median)

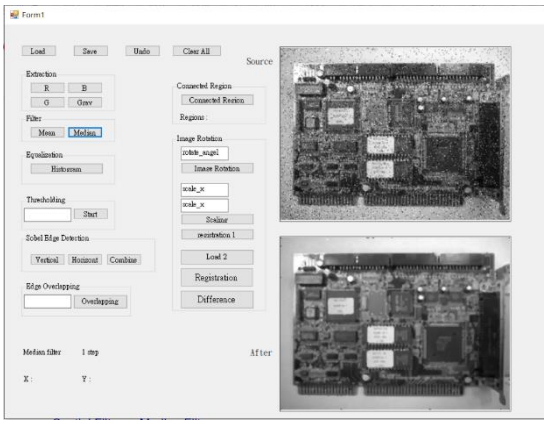
Method : 成功讀取照片後，歷遍圖片中每個 Pixel 為九宮格的中心，對 Kernel 中共 9 個點的數值算數平均與找中值，而 Kernel 若超出原圖則補 0(padding)。算數平均為加總 Kernel 內的數值再除以數值個數的加總，若數值為超出原圖之 0 則不計入術值個數加總；取中值方法為將 Kernel 的數值排序後再取裡面第五大之數值。

Result :

Mean filter



Median filter



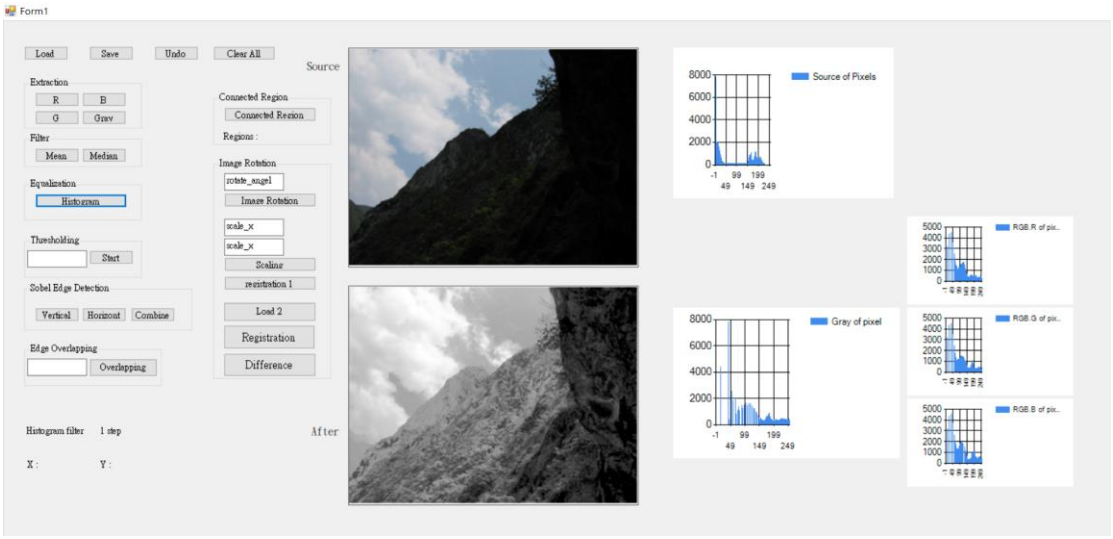
Problem C : Histogram Equalization

Method : 因為 Histogram Equalization 主要目的是將原始圖像像素的色彩強度均勻地映射到整個色彩範圍內，得到一個色彩強度分佈均勻的圖像，所以成功讀取照片後，對每個像素值分別統計各灰階值的出現次數，得到統計直方圖，了解該圖片的影像分佈，並且算出各個灰階值的 PDF，再將 PDF 做累加求出 CDF，將 CDF 的結果經過 4 捨 5 入後做出對照表，建立的對照轉換表，決定轉換後完各個灰階值的機率，再將所得到機率放置到表格上。將各 Pixel 帶入轉換表後即可求出新的灰階值。而直方圖則是將各 Pixel 值當作 x 軸，各 Pixel 值出現次數當作 y 軸作圖而成。

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{cdf_{max} - cdf_{min}} \times (L-1)\right)$$

Result :

Picture 1



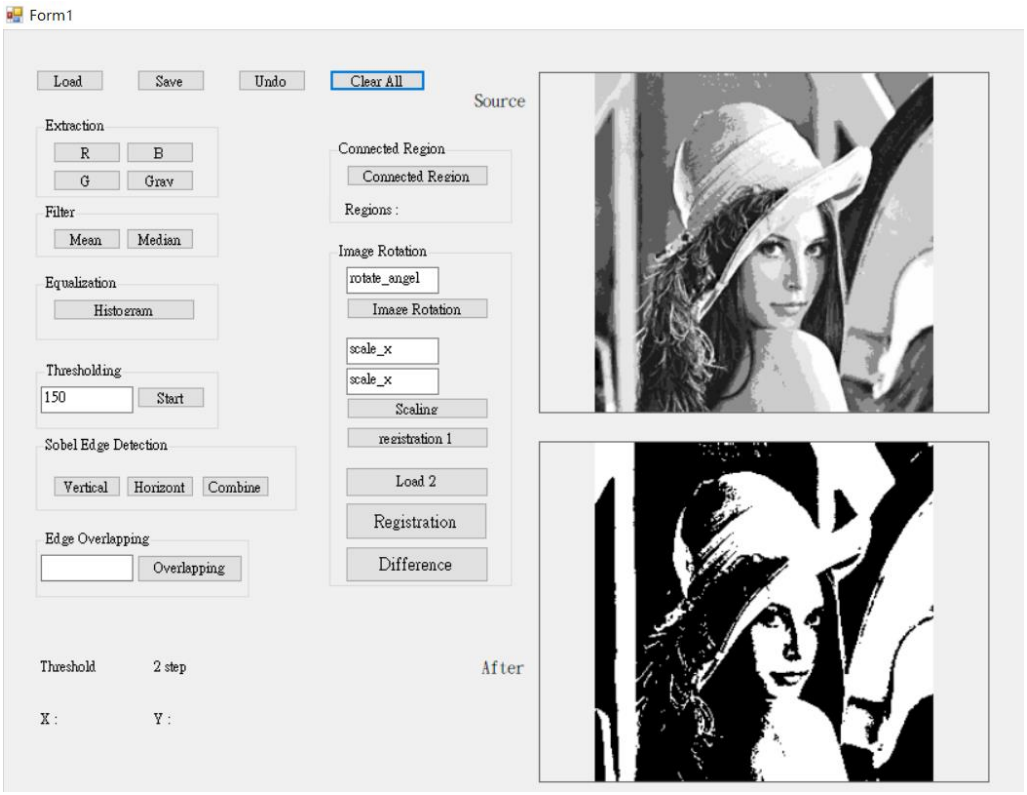
Picture 2

Discussion : 這題因為考量到提供的圖片為彩色圖片，因此額外時做出 RGB 三個通道分別的 Histogram，更能夠了解一張圖片中 RGB 的分布情況，並將結果呈現在 GUI 右方。

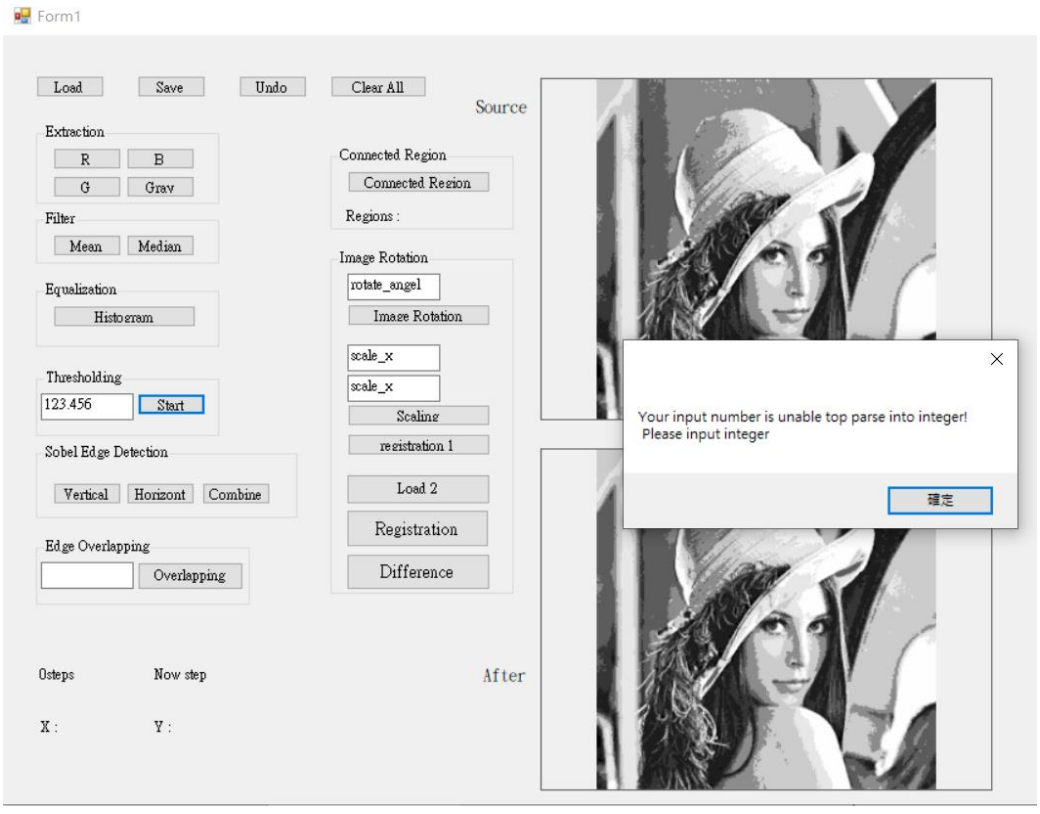
Problem D : A user-defined thresholding

Method : 根據使用者在 Threshold 輸入框所輸入的值，限制使用者輸入為整數，等待讀取圖片後，先將圖片中所有的 Pixel 轉為灰階值，同時若其 Pixel 值大於或等於 threshold 時，則將其值設為 white(255)，否則將其設為 black(0)。

Result :



Discussion : 這題因為有使用者輸入數字，因為圖片的 Pixel 基本上以 0~255 的正整數作為依據，需要檢查、限制使用者的輸入，我使用 Try and Catch 流程作為 thresholding 參數的檢查條件。

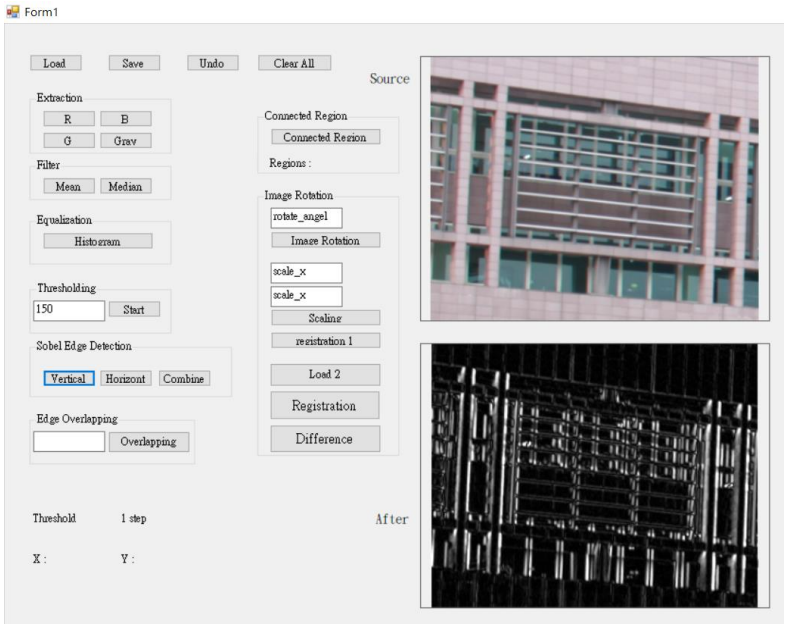


Problem E : Sobel edge detection (vertical, horizontal, and combined)

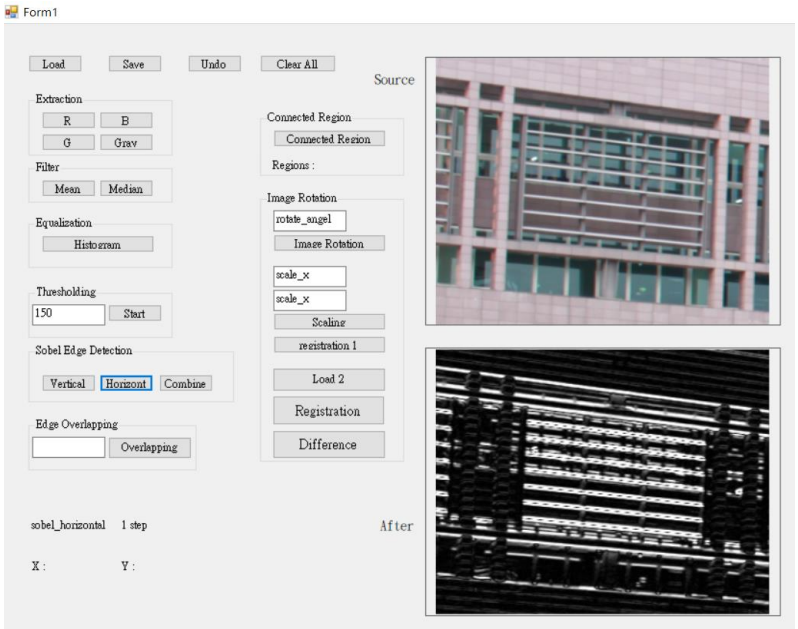
Method : 開啟圖片後，先將 Pixel 值轉為灰階值，對於每個 Pixel 做 Convolution，以該點的中心的 Kernel (image)上下左右取 8 個點，並利用 3x3 的遮罩矩陣 Sobel operator，x = [1, 0, -1, 2, 0, -2, 1, 0, -1]、y = [1, 2, 1, 0, 0, 0, -1, -2, -1]作運算得 Gx 和 Gy，若 Kernel 的範圍超出原來圖片的大小則以 0 代替，分別計算出 Gx 和 Gy，同時會檢查是否有超過 0~255 範圍的值，從而產生 Vertical、Horizontal image；而 Combine image 的 Pixel 值 G 則是由 Gx 與 Gy 的平方合開根號之結果。

Result :

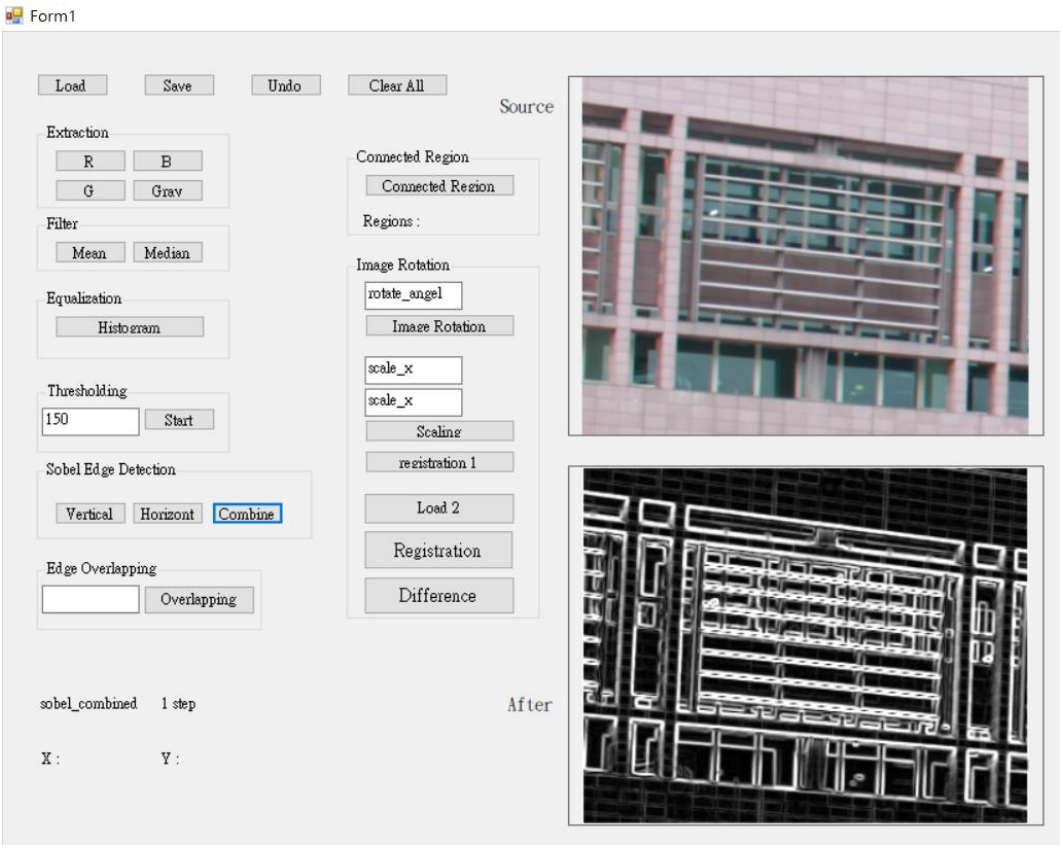
Vertical

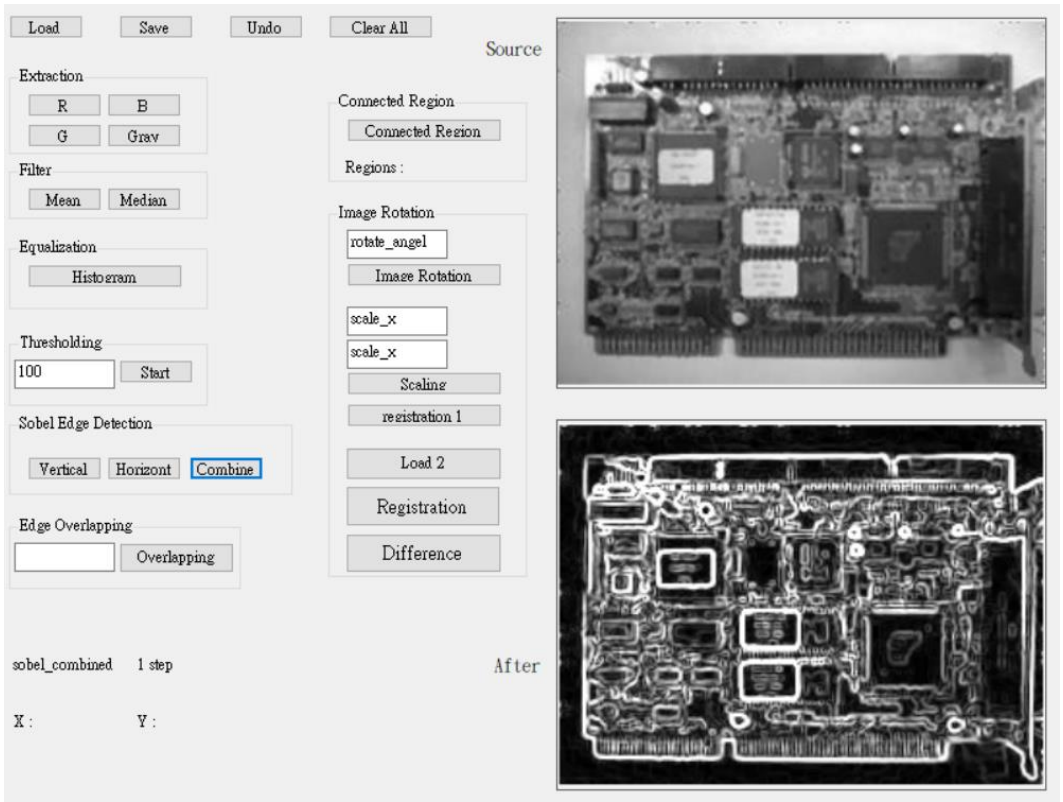


Horizontal



Combine



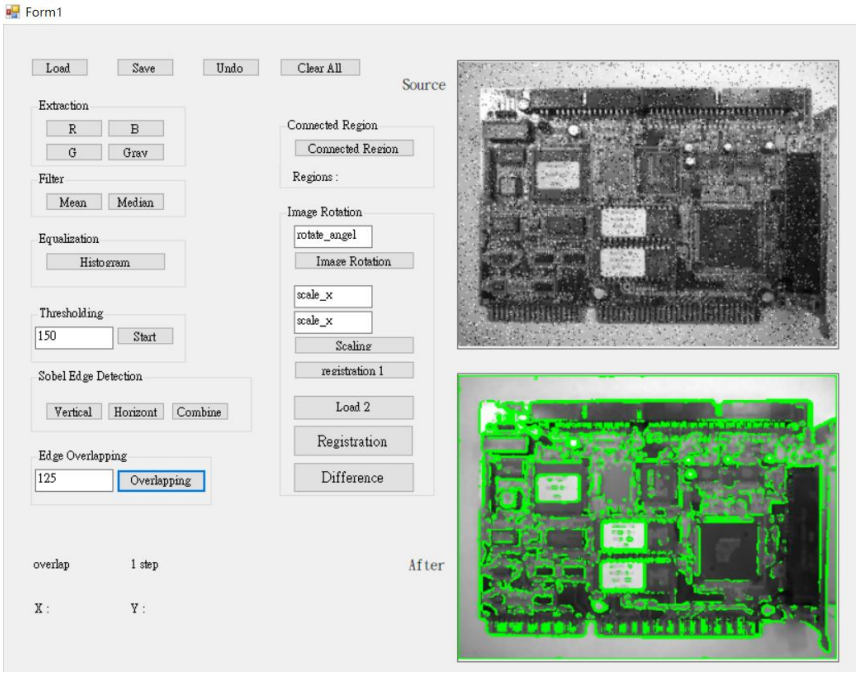


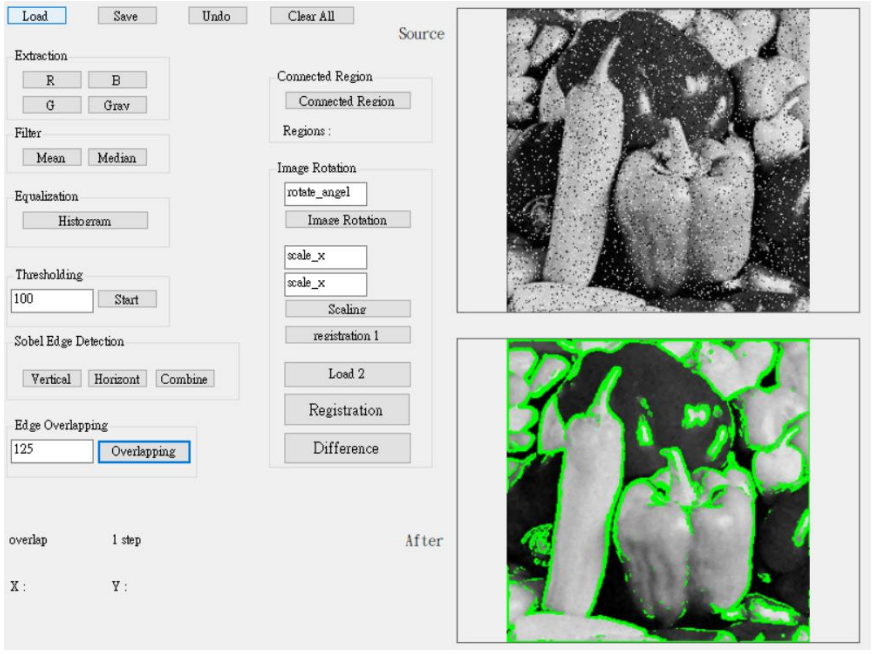
Discussion： 需要特別留意，經過 Sobel operation 之後，將大於 255 或小於 0 的值做調整，將變化明顯的部分留下來，進一步得到我們要的邊緣。

Problem F : Overlapping
(Threshold the result of (E) to binary image and overlap on the original image)

Method :由使用者輸入 Threshold 的參數後，開啟圖片後，然候先將圖片轉為灰階值，由於圖片具有雜訊，需要先經過 Median filter 和 Sobel edge detection，可以得到該圖片去雜訊後的邊緣，接著依 Convolution，每個 Pixel 會判斷若其值大於等於 threshold，則該 Pixel 之 RGB 值改設為綠色(0, 255, 0)，否則保持原 Pixel 之 RGB 值。

Result :





Problem G : Connected Component

(Count the number of connected regions in a binary image and paint it with different colors)

Algorithm :

8-Connected-Components，這個演算法是針對二值化後的影像去分區塊，將每一個區塊給一個標籤，把所有相鄰區塊標示為同一個標籤，最後計算出整張影像每個像素是屬於哪個標籤。換句話說，在二值化影像中，像素和像素之間灰階值的相似度就代表它的連通性，若 q 屬於集合 N8(p)，且 q 和 p 的灰階值為相等的。

Method :

開啟圖片之後，因為 8-Connected-Components 的要求，所以先將圖片灰階二值化，再將圖中的 Pixel 以 0 和 1 分類，針對每個 Pixel(以下以 p 做為表示)，尋找其鄰近的 4 個點，座標分別為：q : (x-1, y-1)、r : (x,y-1)、s : (x, y-1)、t : (x-1, y)，這樣一組稱為該 Pixel p :(x,y) 的 4-近鄰(4-neighbors)，用 N4(P) 表示。

Step 1：掃描所有的 Pixel (Scanning)

若 p=0，則掃描下一點。

當 p=1，

若 q=r=s=t=0，給 p 一個新標籤。

若 (q, r, s, t) 中只有一個為 1，把 p 點的標籤設得跟值為 1 那點的標籤相同。

若 (q, r, s, t) 中不只一個為 1，則把 p 點的標籤設得跟值為 1 的其中一點之標籤相同，而其他值為 1 的點，亦設為跟此點同樣的標籤。

Step 2 :

合併同類別 (merging classes)，選擇使用 Sequential algorithm (循序法)。

1.「由左而右，由上而下」掃瞄影像。

2. 若遇到像素值為 1 則：

a. 若目前處理中的像素的上面或左邊鄰近像素只有一個有標號，則將該標號複製給目處理中的像素。

b. 若上述兩個鄰近像素的標號相同，則將該標號複製給目處理中的像素。

c. 若上述兩個鄰近像素的標號不同，則將值較大的標號複製給目處理中的像素，並將該標號記錄在等值標號表內。

d. 若上述兩個鄰近像素都沒有標號，則指定一個新的標號給目前處理中的像素。

3. 若第一回合的掃瞄未完成，則回到 2.。

4. 搜尋等值標號表內，各組等值標號的最小值。

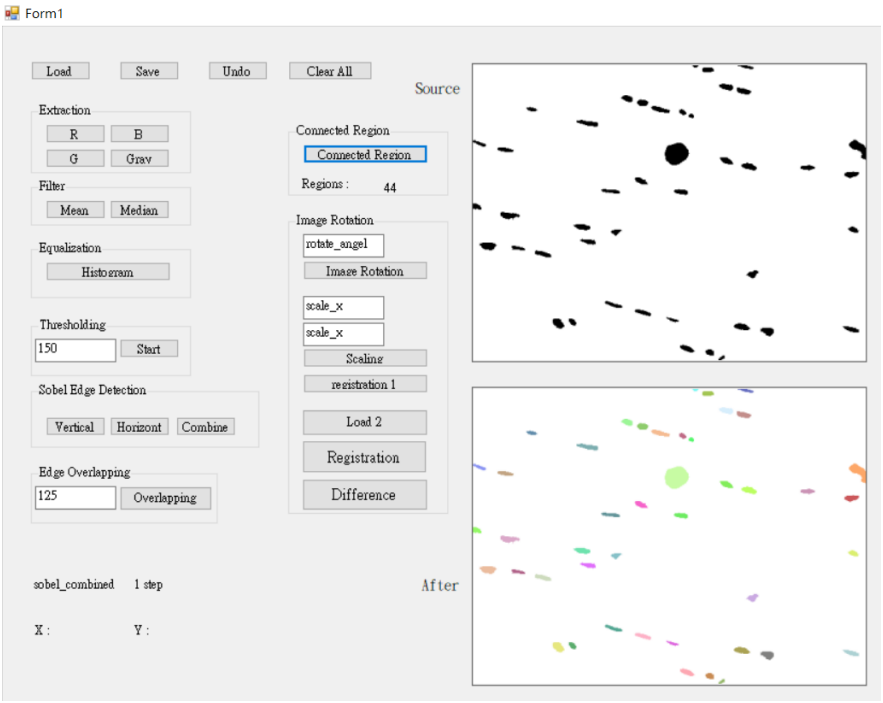
5. 進行第二回合的掃瞄，將各組等值標號以其最小值取代。

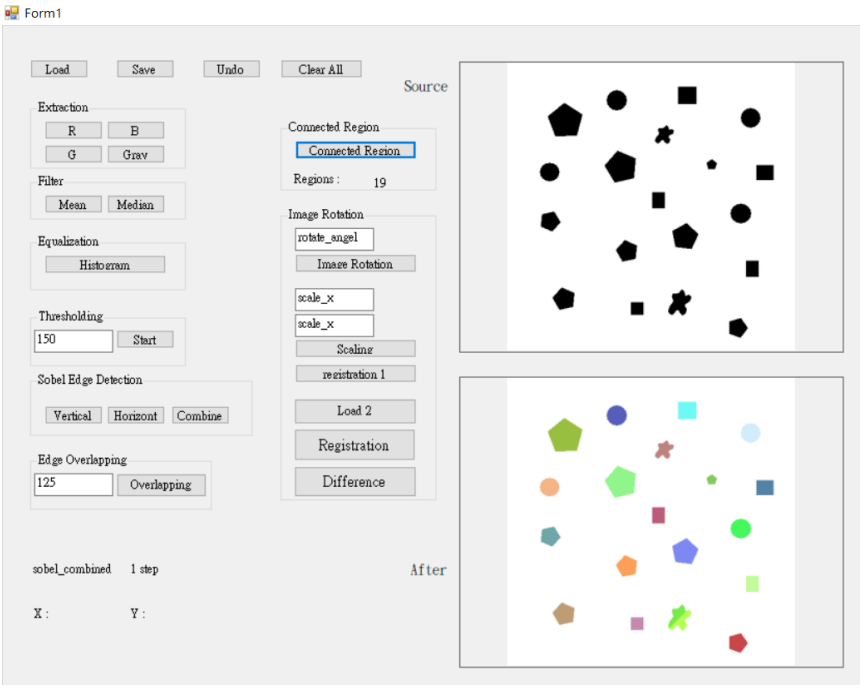
Step 3：

1. 從先前各組的等值標號表，統計目前的組別數量，透過隨機函數，隨機產生不同的顏色對應到每個組別，也就是組別顏色對照表。
2. 依據顏色對照表，逐步掃描圖片，都遇到非 0 個 Pixel 時，檢查其分類的組別，並且給予對應顏色的資訊。
3. 整張圖片的所有區塊依據組別顏色，具有不同的顏色。

參考：國立中央大學資訊工程學系碩士論文遺失物偵測且具回報能力之視訊監控系統

Result :





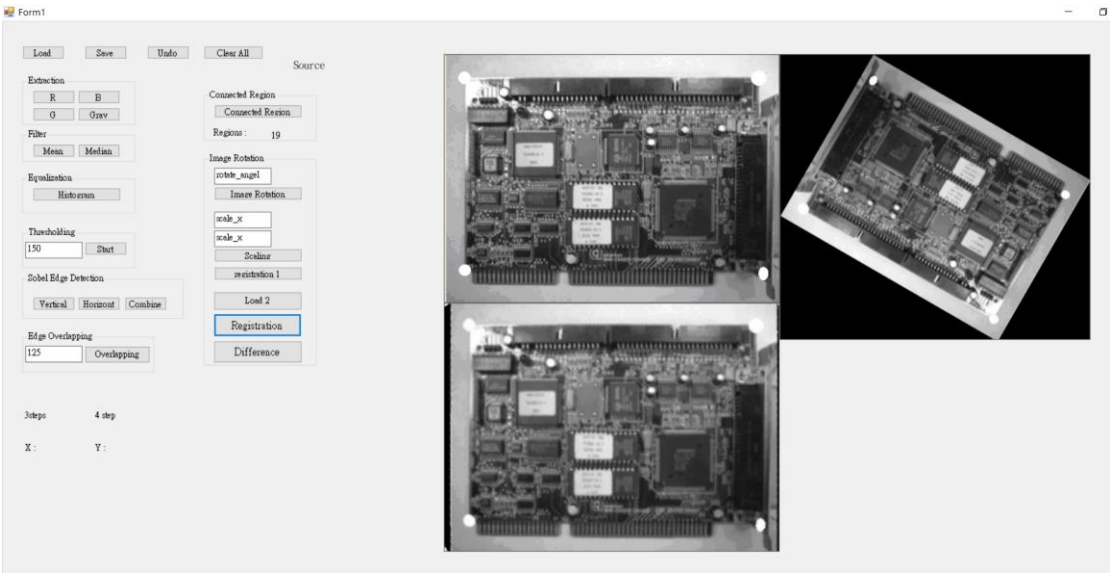
Problem G : Image registration

Method : 開啟 2 張圖片，分別為對應的原始圖片和經過旋轉和放大之後的圖片。首先，經使用者由滑鼠點擊，得到兩張圖片中的各 4 個點的座標 (x,y)，再根據左右兩個 PictureBox 的第 1 個點和第 2 個點，取直線斜率，得以計算分別對應到 XY 平面的角度，當旋轉轉正後，再根據先前計算的兩條直線距離作為對應的矩形短邊長度比例，此為放大比例，當放大完成後，在將圖片根據原始圖片的大小進行裁切，將黑色的區塊移除。最後，依 Convolution，每個 Pixel 會與旋轉放大後的圖片進行比較，計算兩張圖片的誤差值。

Result :

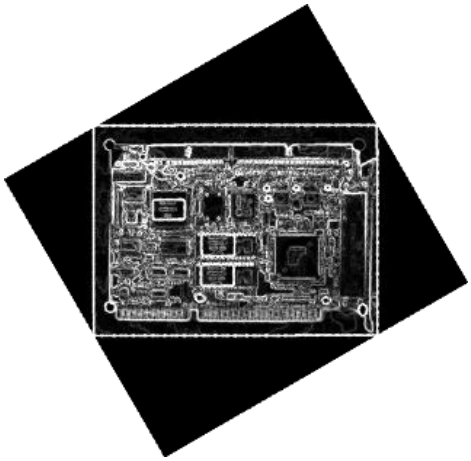
Picture 1，旋轉不超過 180 度

Picture 2，旋轉超過 180 度



Discussion :

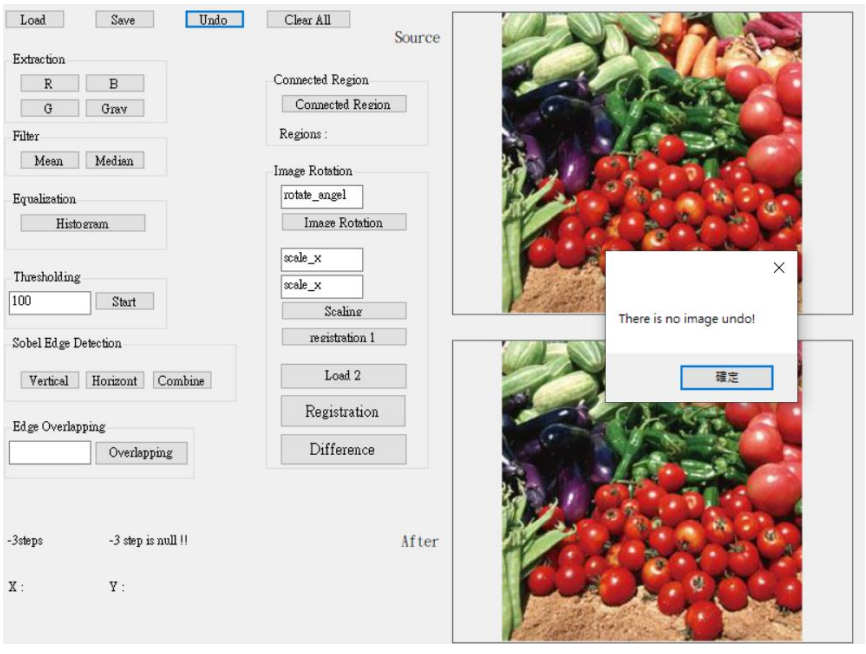
Scaling 部分需要將 Picture Box 的 Mode 調整為 Normal，以取得 4 個點的座標(x,y)時，對應原始圖片的位置資訊，認為使用 Sobel edge detection 去檢測邊緣，計算旋轉圖片的實際大小和座標(x,y)會比較準確，相較於使用滑鼠點擊取得四個點的座標。Rotating 部分，考量到圖片可能會旋轉超過 180 度，因此點擊的順序就顯得重要，由於我使用斜率去判斷，可以包含 360 度的所有範圍，且只需要使用者點擊 2 個點就能夠判斷出結果。



Problem H : Undo

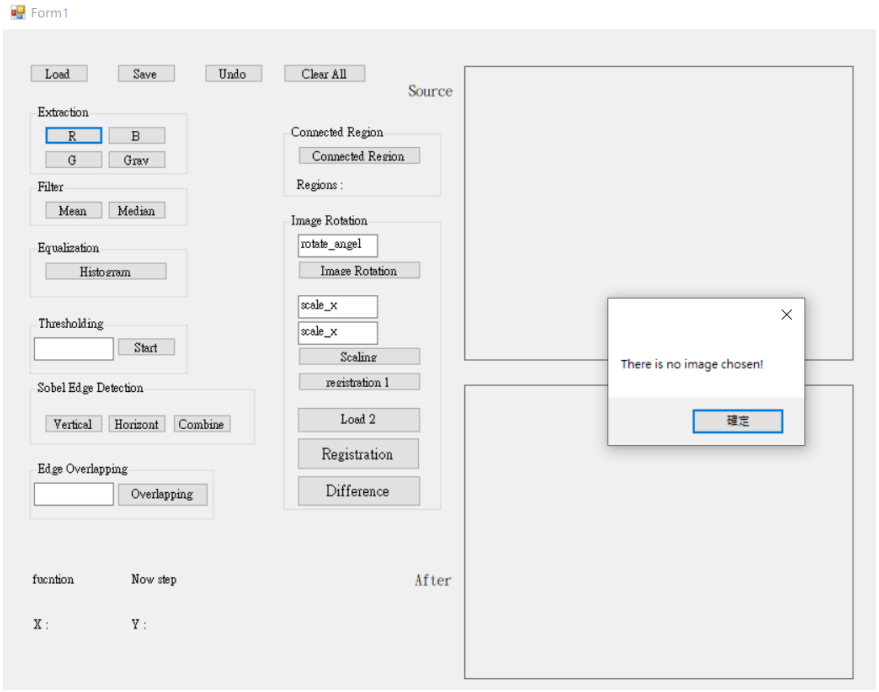
Method : 使用 Stack 先進後出的原則，將每次圖片的未處理和處理後的結果都先後存入(Push)，每當需要上一步時，就提出圖片(Pop)，達到效果。每當開啟新的圖片時，就將圖片放置於 2 個 PictureBox，每次處理只針對下方的 PictureBox 的圖片，然後依序放入 Stack 中。

Result :

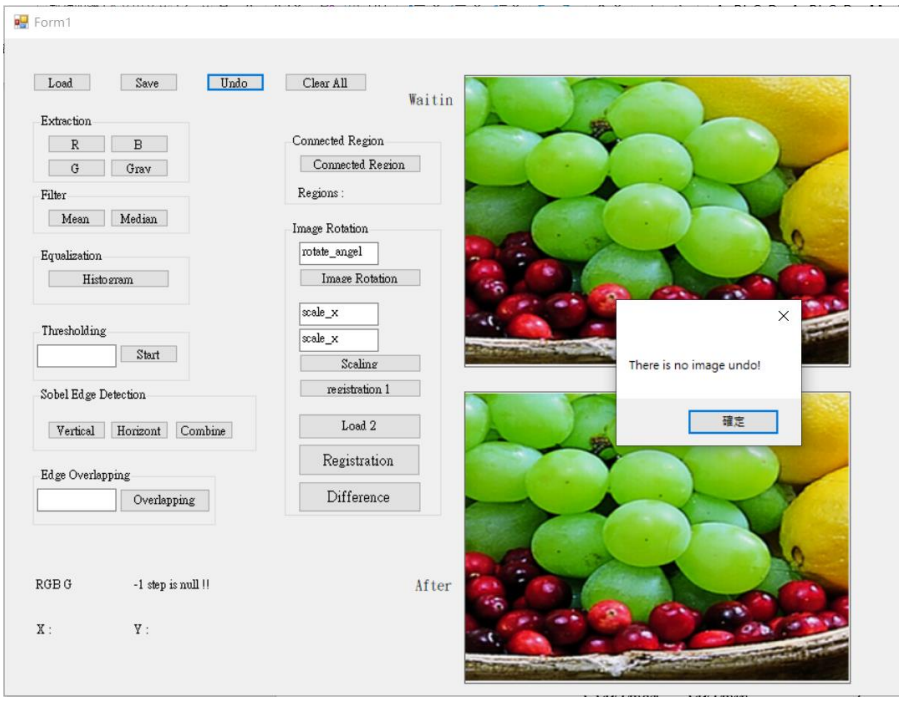


Additional function:

1.檢查是否有成功開啟照片



2.計算上一部的使用空間，查是否還有其他的照片在 Stack 中(目前的圖片數量)



3.設計 clear 按鈕

根據不同的功能所產生的畫面不同，可以使用此功能，回復到乾淨的使用視窗。

Conclusion :

這次的作業十分有趣，讓我不只學習新的語言 C#，也從中徹底學習影像處理的各個功能運作流程，實際上從每個 Pixel 去實作，才能夠清楚知道每個 Filter 實際上的用途和結果，也了解不同的功能互相結合後可以得到不同的功用。雖然一開始對於新的語言相當陌生，但是一邊做一邊學習，讓我滿有成就感的。實作連通元件的題目時，有找到相關的論文，就將其實做出來，對於我未來在論文的方向和內容有一定的幫助，也對於後面的題目，老師沒有太多的限制，讓我有很大的發揮空間，自己嘗試、應用學到的許多種方法去組合、去完成目標。簡言之，感謝老師出這次的作業，讓我完整的學習從影像處理中各個方面的內容和應用。