**Image Processing Homework 1 Report**

**P76111296 張中龍 使用語言: C#**

***1. RGB Extraction & Transformation***

**Problem**

把三通道的彩色的圖片轉換成只有單獨的R、G、B的Channel或Grayscale的圖片

**Method**

將RGB的圖片讀入Bitmap，分別取出其R、G、B 的Channel 的值，即可獲得純只有R、G或B Channel 的圖片；Grayscale則將每個pixle的RGB value 設置為( (R+G+B)/3 , (R+G+B)/3 , (R+G+B)/3)，即可呈現出Grayscale的效果

**Results**

Origin Image



R Channel G channel



B Channel Grayscale



***2. Smooth filter (mean and median)***

**Problem**

利用Smooth Filter消除圖片中noise

**Method**

在原圖的邊緣-1 pixl的範圍內進行下列filter

Mean Filter

=>對原圖在filter範圍內的pixel進行取平均 ( Mean )

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

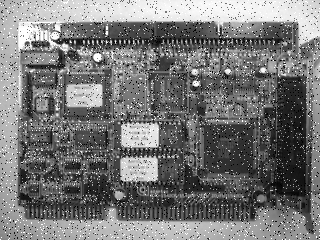
Median Filter

=>將原圖p1~p9 的pixel進行排序，挑選中位數 ( Median ) 後，把原圖filter大小內的pixel都設為中位數

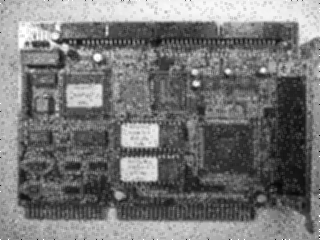
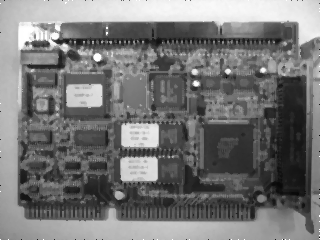
|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |
| P4 | P5 | P6 |
| P7 | P8 | P9 |

**Results**

Origin Image



Mean Filter Median Filter



**Discussion**

Mean Filter會使圖片變模糊，而Median Filter 圖片會比Mean Filter還要更清晰。

Median Filter 消除噪點效果會比較好

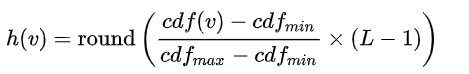
**3. Histogram Equalization**

**Problem**

Implement histogram equalization

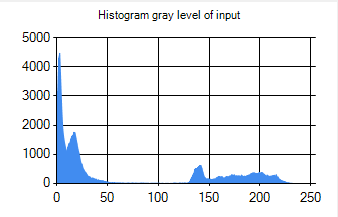
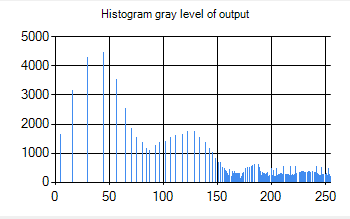
**Method**

計算input image中每個pixel的gray level值出現的頻率，再計算出每個pixel出現的pdf和cdf，最後使用下列公式進行equalization

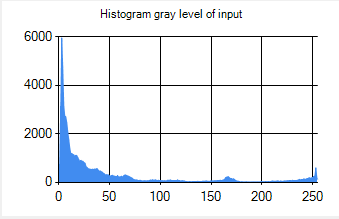
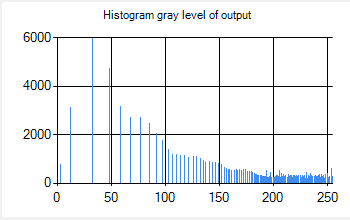
where L = 256

**Results**

原圖 處理後



原圖 處理後



**Discuss:**

從Histogram中可以看出，gray level密度較高的區域被拉較開，透過Histogram Equalization可以讓顏色糊成一塊的區域中物體的輪廓變較清晰，也就是增加對比度。

***4. A user-defined thresholding***

**Problem**

Implement Binary Thresholding with giving threshold value T.

**Method**

輸入threshold T，將pixel gray level value低於t的值設為0，大於等於t的值設為255。

**Results**

原圖 處理後(T=125)



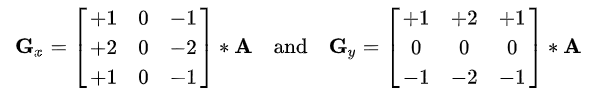
***5. Sobel edge detection***

**Problem**

Implement Sobel Edge in vertical, horizontal and combined

**Method**

透過vertical, horizontal and combined的filter在空間域上做convolution



horizontal

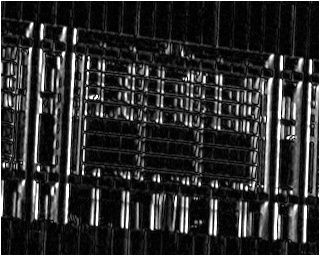
vertical

**Results:**

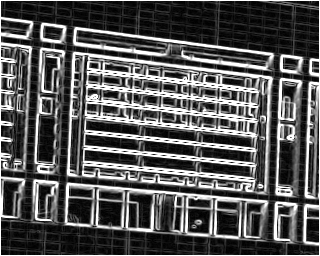
原圖



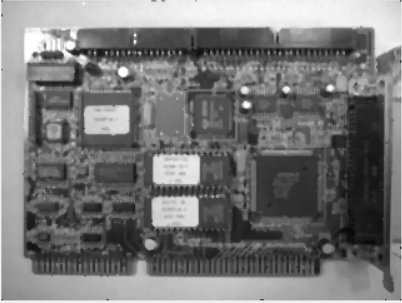
Vertical Horizontal



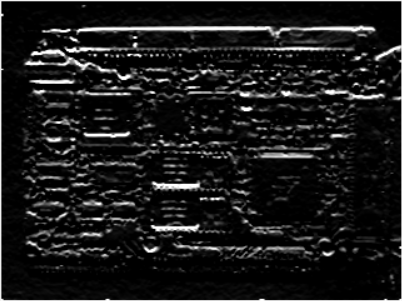
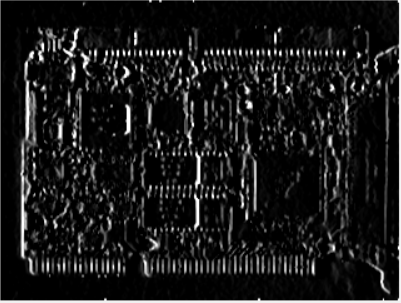
Combined



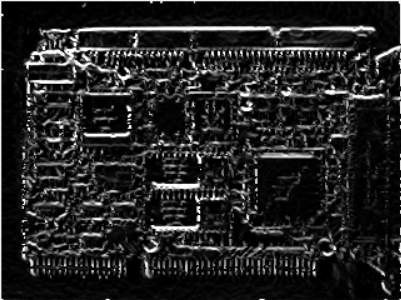
原圖



Vertical Horizontal



Combined



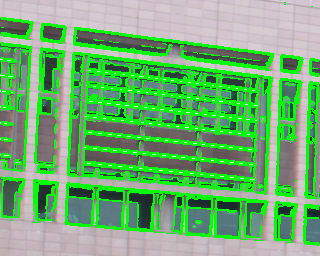
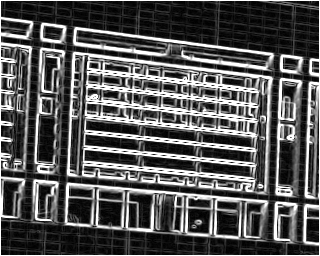
***6. Threshold the result of (5) to binary image and overlap on the original image***

**Problem:** Threshold the Sobel filter result and overlap on the original image

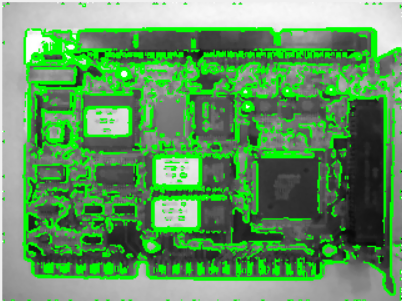
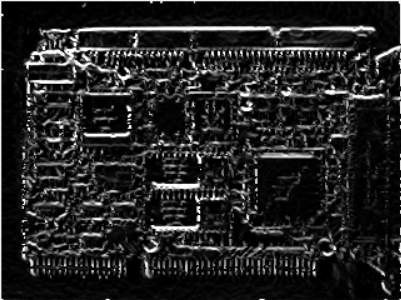
**Methods**:將5做完的圖片進行二值化處理後，留下來得即為物件的邊緣，再根據此邊緣以純綠色畫在原圖上。

**Results:**

原圖 處理後(T=125)



原圖 處理後(T=125)

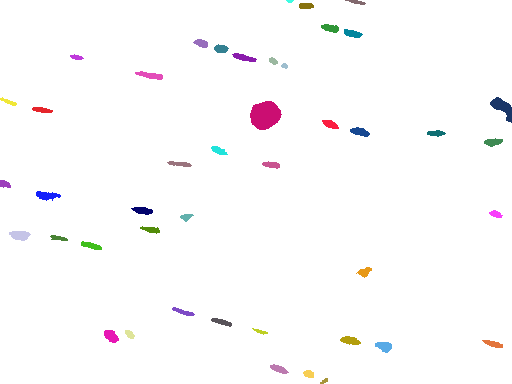
****

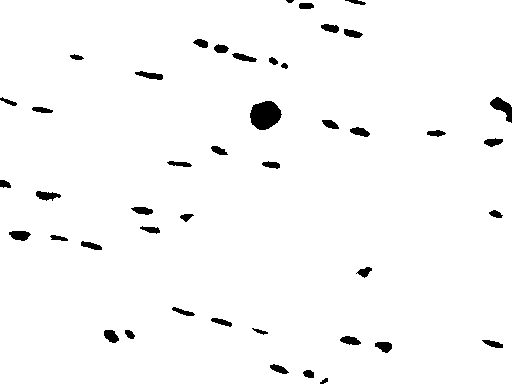
***7.*** ***Connected Component***

**Problem:** Count the number of connected regions in a binary image and paint it with different colors

**Methods**:從左而右，由上往下，掃遍所有pixle，如果pixle是黑色且周圍沒有其他不黑不白的顏色，將其標上新顏色(非黑白)，若周遭有其他顏色(非黑白)，則目前pixle改成此顏色。另外，若是掃到的pixle是黑色，但是要填的顏色與右上顏色(非黑白)不同，則將圖中所有與右上相同顏色的pixle改成當前pixle的顏色

**Results:**

****原圖 處理後(44 connected components)



**Discuss:**

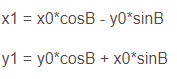
此方法的時間複雜度(設長寬皆n)會達O()，因此若圖像較大時，時間會很久，另外，由於有在填色前以while迴圈確認圖中填的隨機顏色不一樣，雖然視覺效果會好，但是若連通區域一多，耗時會更久。要改善的話，我認為可以以宣告二維的Label 陣列存儲並比對運算，可以減低access圖片pixle的次數。

***8. Image registration***

**Problem:**

依序點擊原圖的4個點 以及 旋轉縮放圖的4個點 ，讓旋轉縮放圖根據4個點的比例及位置資訊對回原圖

**Methods**:假設原圖及旋轉縮放圖的左上、右上、左下、右下點分別為(p1,p2,p3,p4)及(A,B,C,D)，求出及，先通過長度比較，可以得知scaling factor，再將兩項向量端點p1 ,A平移回原點，並將此兩同端點的向量帶入以下旋轉公式，算出angle。之後再將照片根據算出的角度和scaling factor旋轉並縮放，在根據angle及scaling factor求出左上角最後的座標，再以此座標為左上角標準點，裁減與原圖相同size的區域即可。

****

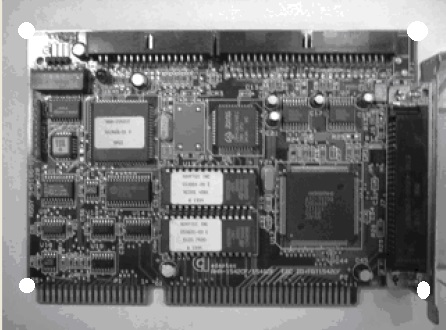
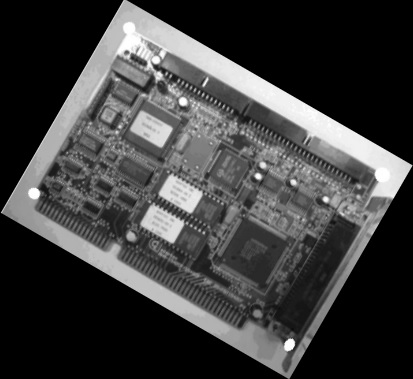
原座標: (x0,y0)

旋轉後座標: (x1,y1)

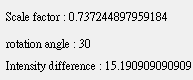
角度: B

**Results:**

原圖 旋轉縮放圖



Output Image



原圖

**Discussion**

若點的對應點不完美(非長方形)，那registration後的結果會不好，周圍會有一些黑邊存留或是裁減到原圖的區域。另外，由於過程中圖像經過旋轉及縮放，會造成許多pixle的資訊被破壞，因此結果會有一點模糊。

***Conclusion***

在這次的作業中可以看出許多同影像在不同處理下的結果，像是第二題對於雜訊的處理，Median Filter會比Mean Filter效果還要更好。以及有些時候會需要不同的方法互相搭配處理，像是假如要從含噪點的圖中獲得物件的邊緣，則可以先Median Filter再執行Sobel edge detection。另外，最後一題，也用到了一些向量的運算，活用了線性代數所教的東西，並且其他題目也可通過圖片直觀的觀察到一些數學公式及運算的意義。