**實驗報告**

**楊承尉**

1. **問題**

實作：

* 1. Color extraction & transformation
  2. Smooth filter ( mean & median )
  3. Histogram equalization
  4. Thresholding
  5. Sobel edge detection
  6. Edge overlapping
  7. Morphological operator ( erosion & dilation )

1. **方法**

視窗程式介面使用Microsoft Visual Studio 2013製作，並以Visual C#撰寫程式碼。影像的讀取與寫入使用到System.Drawing中Bitmap的方法：

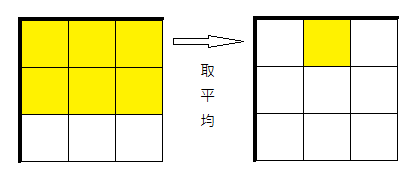
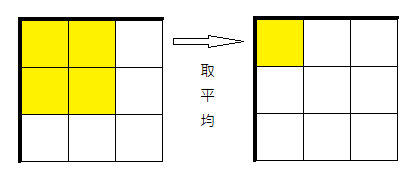
GetPixel( Int32, Int32 ) 和 SetPixel( Int32, Int32, Color )

前者輸入的參數為讀取的像素在原始影像中的座標，並回傳Color型態的像素值，再使用Color的屬性R、G和B，便可分別取得原始影像的紅色、綠色和藍色三個通道。而後者輸入的參數一樣是寫入像素的座標，而像素值可利用Color的方法FromArgb( Int32, Int32, Int32 )，來分別設定紅色、綠色和藍色三個通道值。

Color Extraction是取出像素的三個通道值之後，再組合成新的三張新的灰階圖片，例如：R Channel Extraction是把原始圖片的R通道值讀出來後，設為新圖的R、G和B三個通道值。

Color Transformation是取出像素的三個通道值之後，先計算出它們的平均值，再以此平均值組合新的灰階圖片。

Mean Filter是將每一個像素值都設定為該像素周圍3\*3個像素值的平均，但是因為四個邊的像素周圍的像素不到3\*3個，例如角落的像素周圍只有4個像素，因此就取這4個像素值的平均。



Median Filter的做法和Mean Filter類似，是將每一個像素值都設定為該像素周圍3\*3個像素值裡面的中位數，因此取得像素值之後，先使用插入排序法，再取出排在最中間位置的數作為中位數。而在影像中四個邊的像素就一樣找周圍4或6個像素值的中位數，然而因為觀察值有偶數個，所以中位數就取最中間的兩個數值的平均數。

Histogram Equalization會先計算出原始影像中的灰階數分布，再將灰階數的累積分布函數cdf算出來，之後將原始灰階分布中出現最小的灰階值min映射到0，其餘的灰階值按照累積分布函數的比例映射到新的灰階數分布中，映射函數H如下：

最後將原始影像的灰階數改為對應到的新的灰階數。

Thresholding利用scrollBar取得閥值，並逐一讀取像素值，大於等於閥值的設為白色，小於閥值的設為黑色。而修改閥值的時候也會呼叫Thresholding，輸出的圖片會跟著修改。

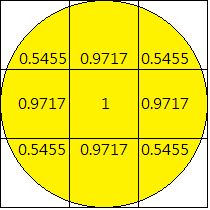
Sobel Edge Detection使用兩組3\*3的filter：

分別為vertical filter 以及 horizontal filter，再將之與圖像作平面卷積，取絕對值，得到水平與鉛直方向的邊界，再將水平與鉛直方向的結果利用 結合。而四個邊的像素因為無法套用filter，所以設為黑色。

Edge Overlapping讀取輸入的影像中白色的部分，將顏色改為使用者設定的顏色之後再覆蓋在原始影像上。

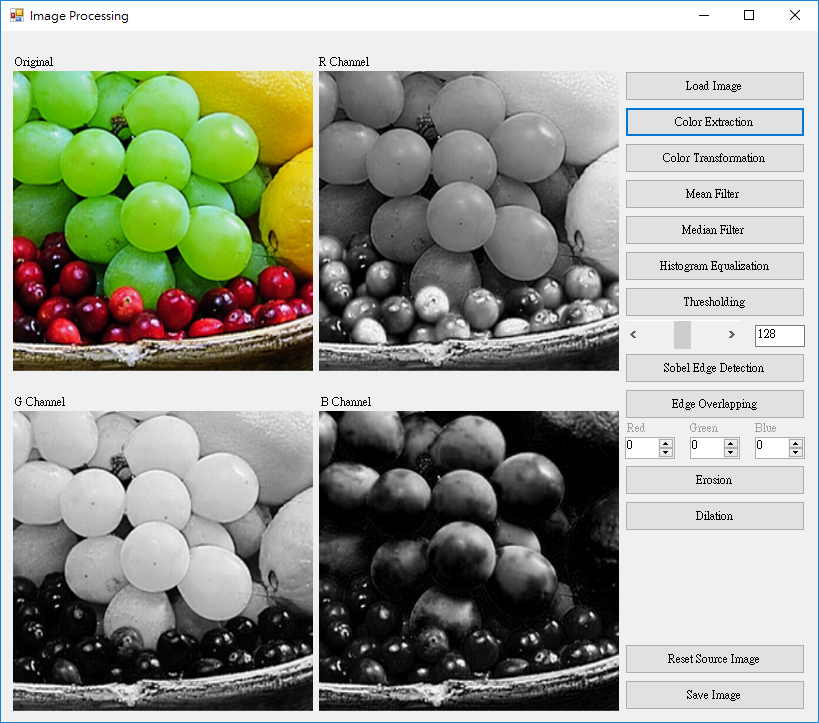
Erosion和Dilation使用3\*3的kernel：

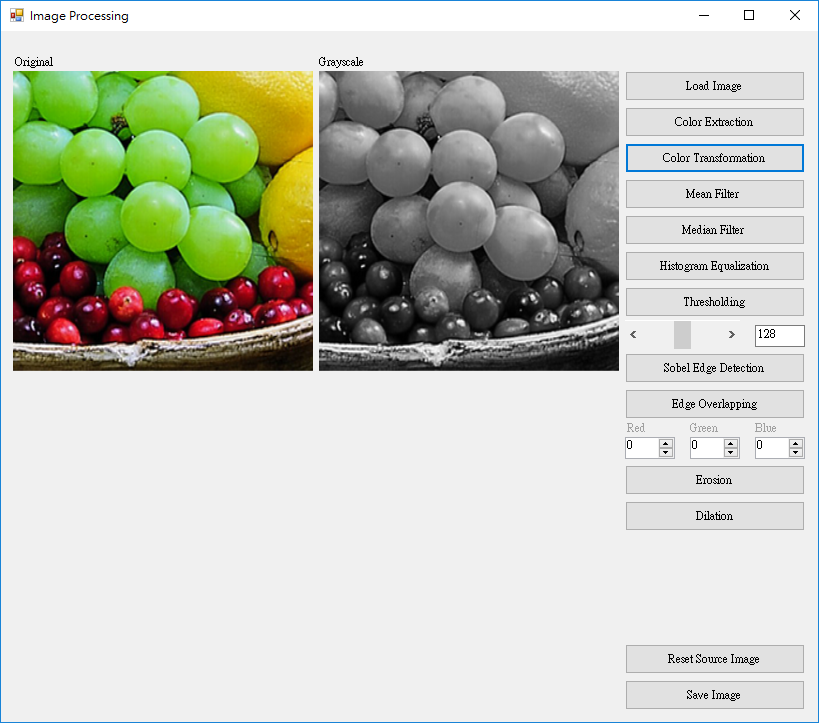
而其中的每個元素代表：在3\*3方格中的最大圓，佔每單位面積的比例。

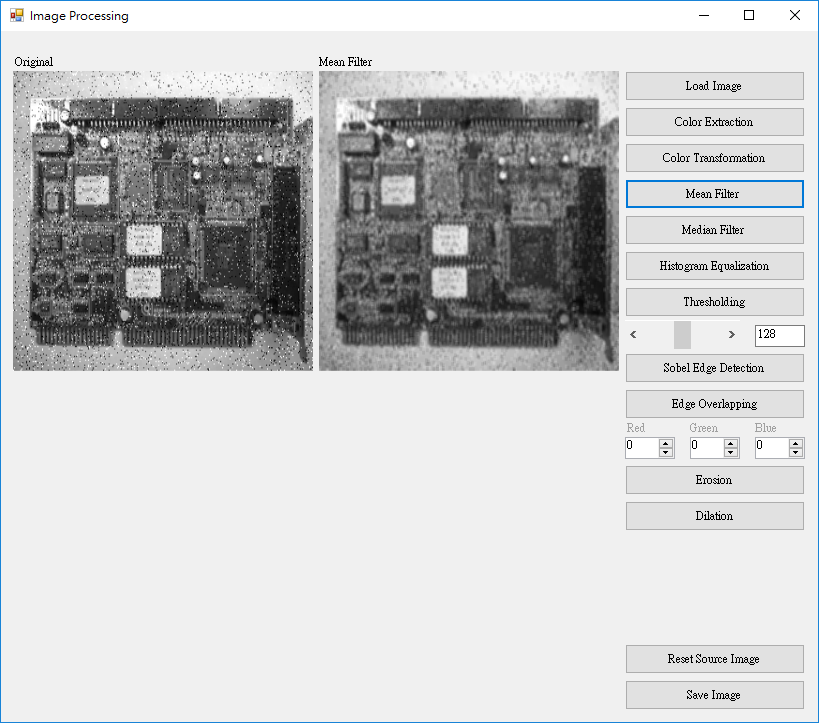


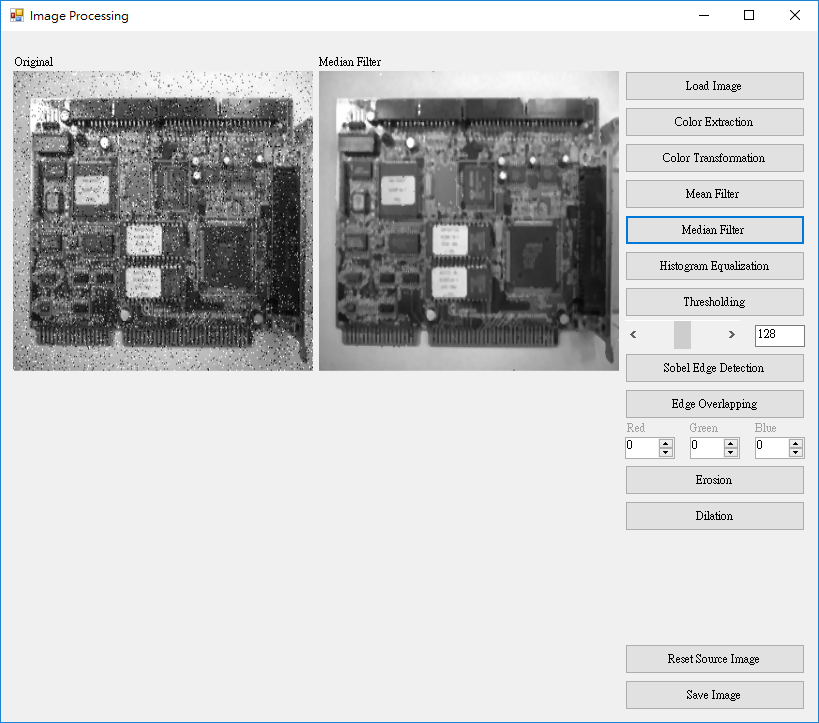
將其與圖像作平面卷積後，判斷每個像素的值，大於等於門檻的設為白色，小於門檻的設為黑色。Dilation的門檻設為 ；Erosion的門檻設為 ，其中7.0688是kernel裡的元素總和。

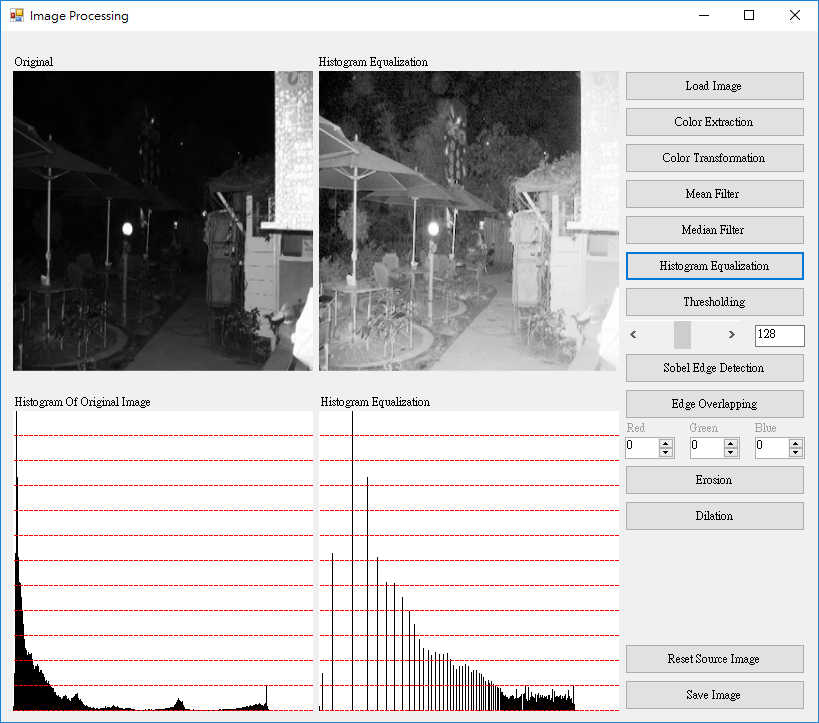
1. **結果**

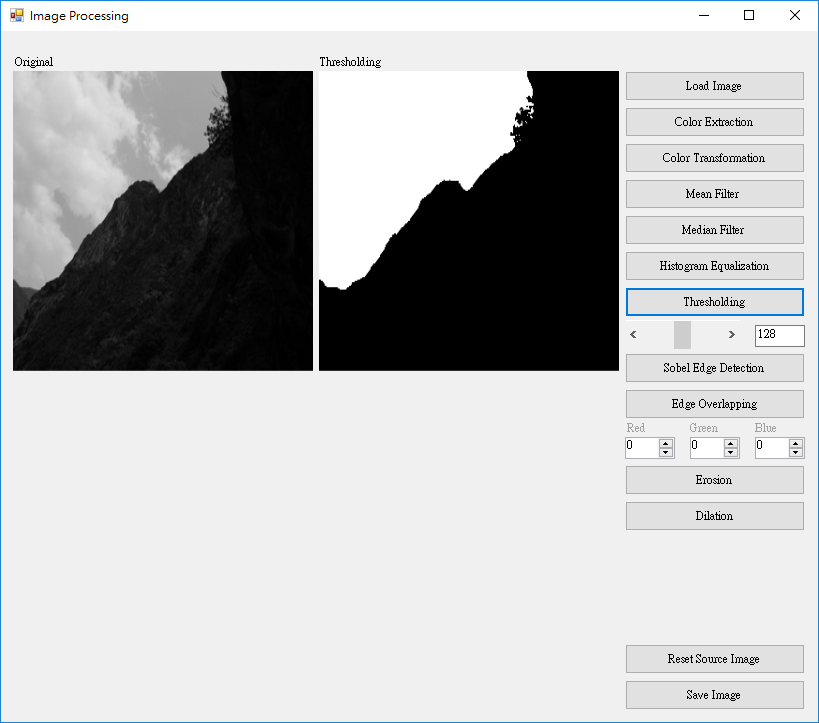


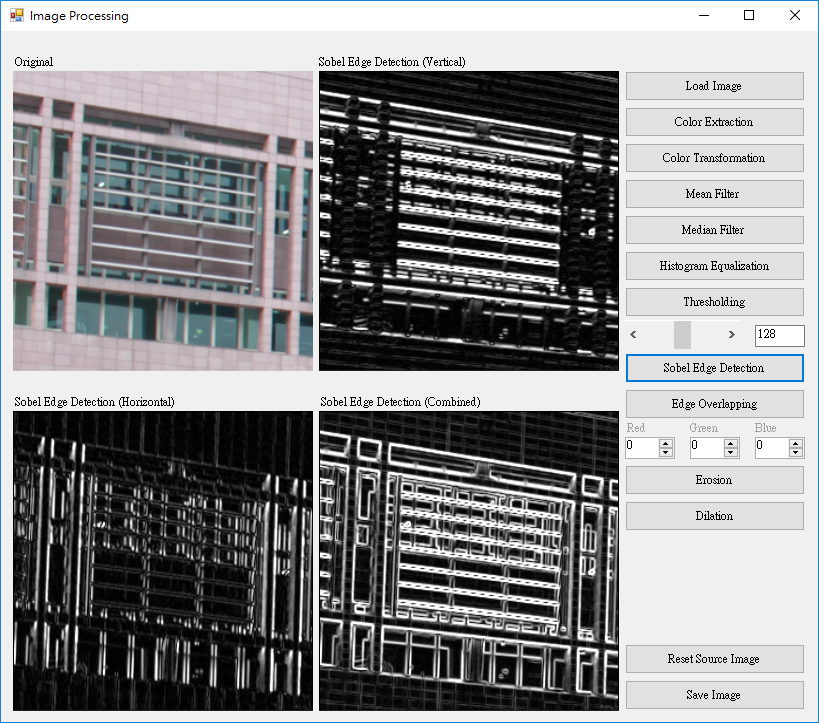


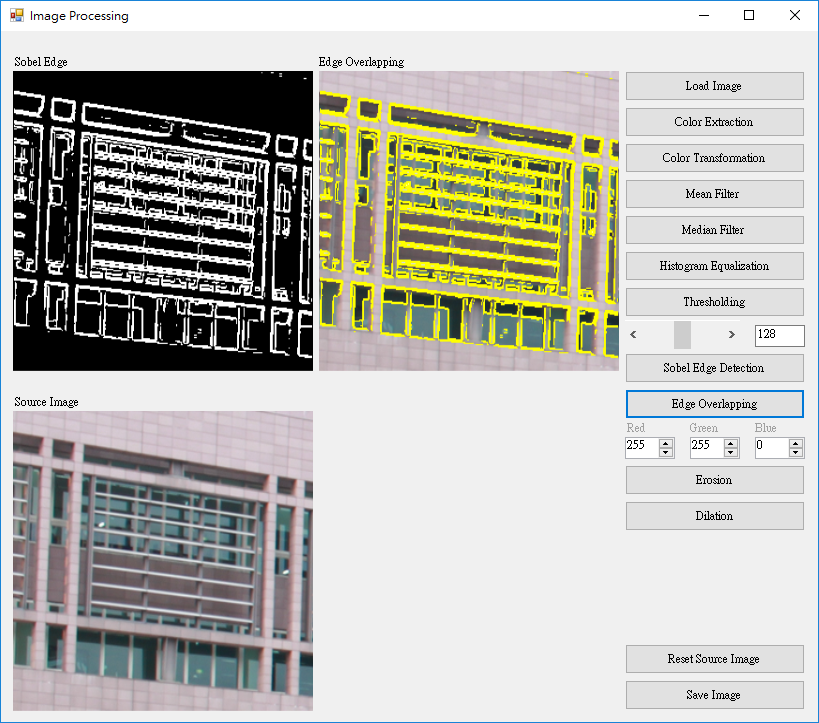


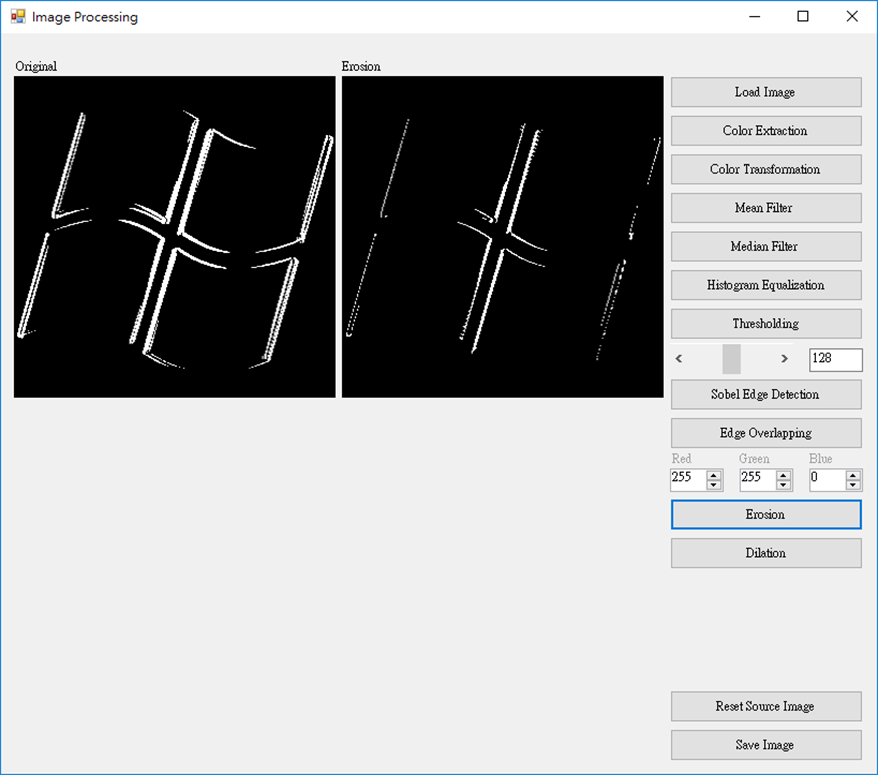


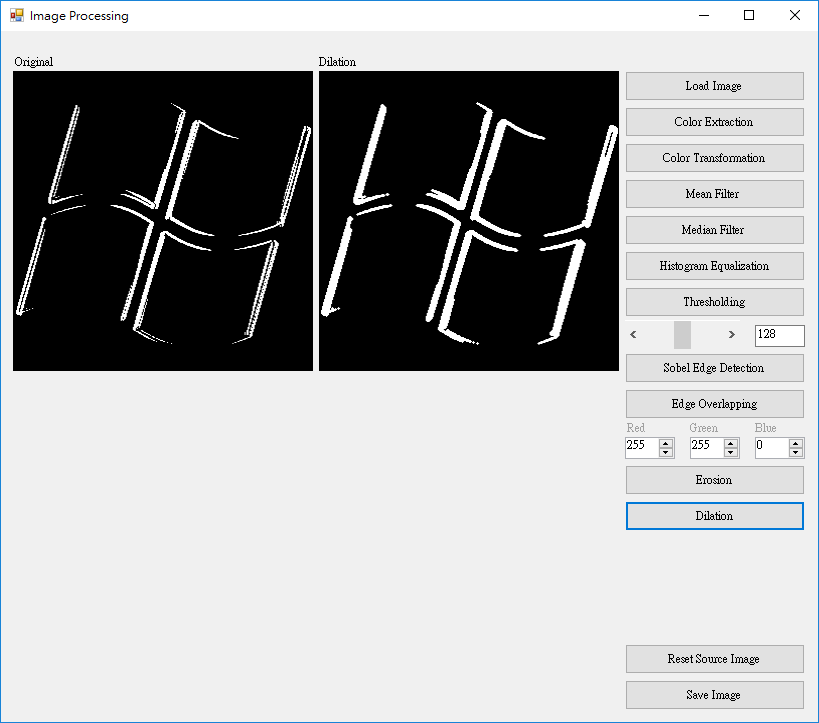






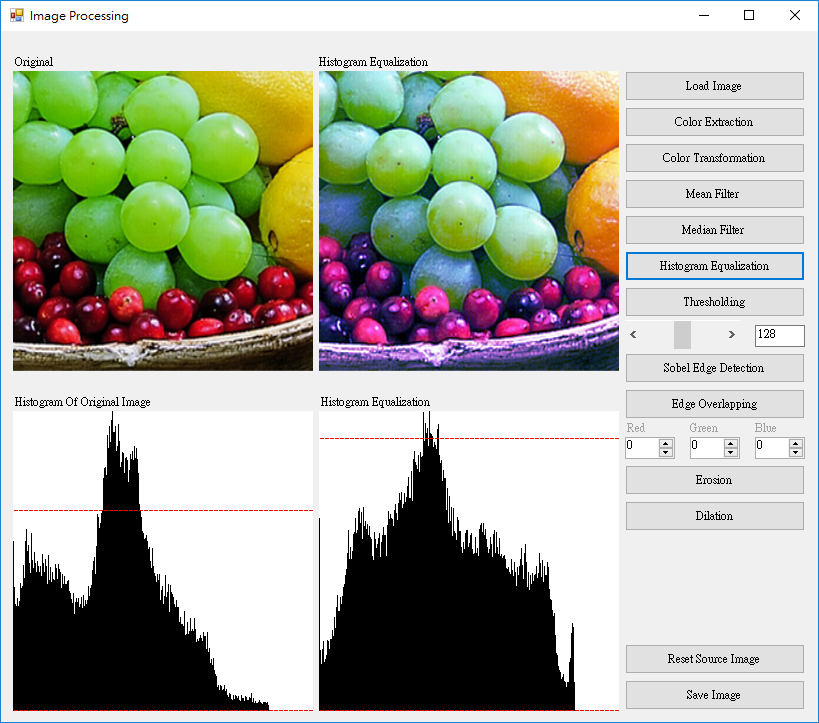






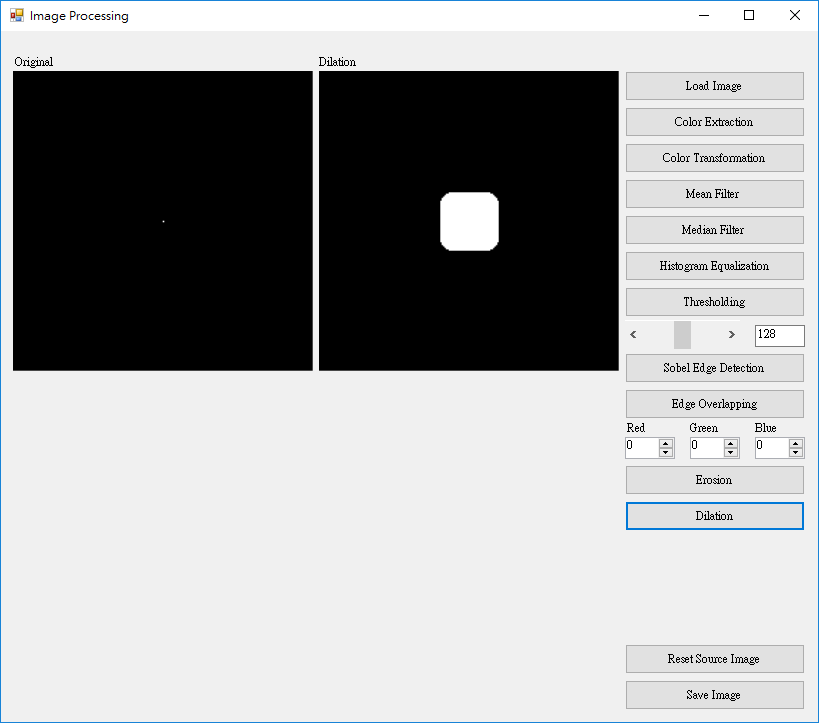
1. **討論**

雖然作業內容並沒有要求，但是在實作Histogram Equalization的時候，我是將原圖的R、G和B三個Channel取出來分別處理，再合成為一張圖，原本以為這樣就能夠處理彩色影像，但是實際上得出的結果並不理想，影像產生出了許多與原圖不相關的顏色：

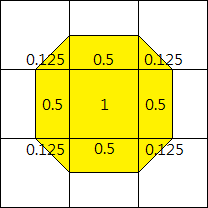


經過思考後，才發現如果使用這種作法，有可能會讓原本在同一個pixel裡的三個通道各自為政，因而變成完全不相關的顏色，所以我認為在做彩色Histogram Equalization的處理時，應該將三個通道的值按照一定的比例混合後，計算出對應的分布，再按照原圖的比例增加或減少，這樣或許就能保有原圖色彩的樣貌。

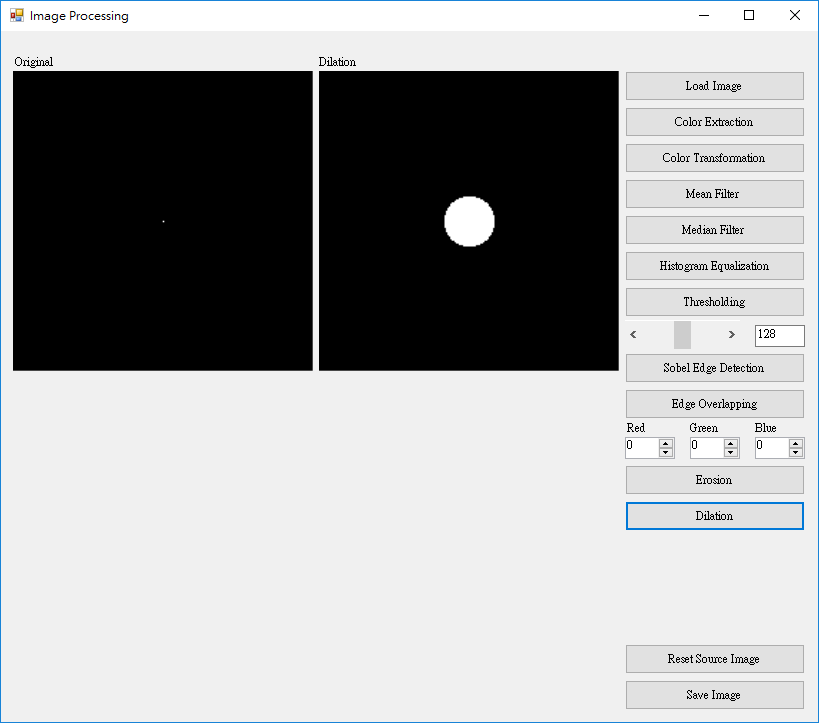
另外，再實作Erosion和Dilation時，我認為最佳的作法是讓每一個pixel可以向四面八方延伸或收縮，也就是讓一個白色pixel一直做Dilation後可以變成白色圓形，或是讓一個黑色pixel一直做Erosion後可以變成黑色圓形。但是因為kernel大小只有3\*3的關係，使得在重複做很多次同一個動作之後，kernel的形狀還是很明顯，Dilation20(onePixel)的結果如下：



重複換過許多kernel之後，發現有一個kernel效果很好：



Dilation20(onePixel)的結果如下：



1. **結論**

本次的實作的項目都有完成，也都達到了作業要求的目標。在製作這支程式的過程中，我學習到了許多影像處理的基本操作和概念，但是還有很多可以改進的地方，期待未來繼續進步。