影像處理原理及應用 HW1

R11543063　葉乃綸

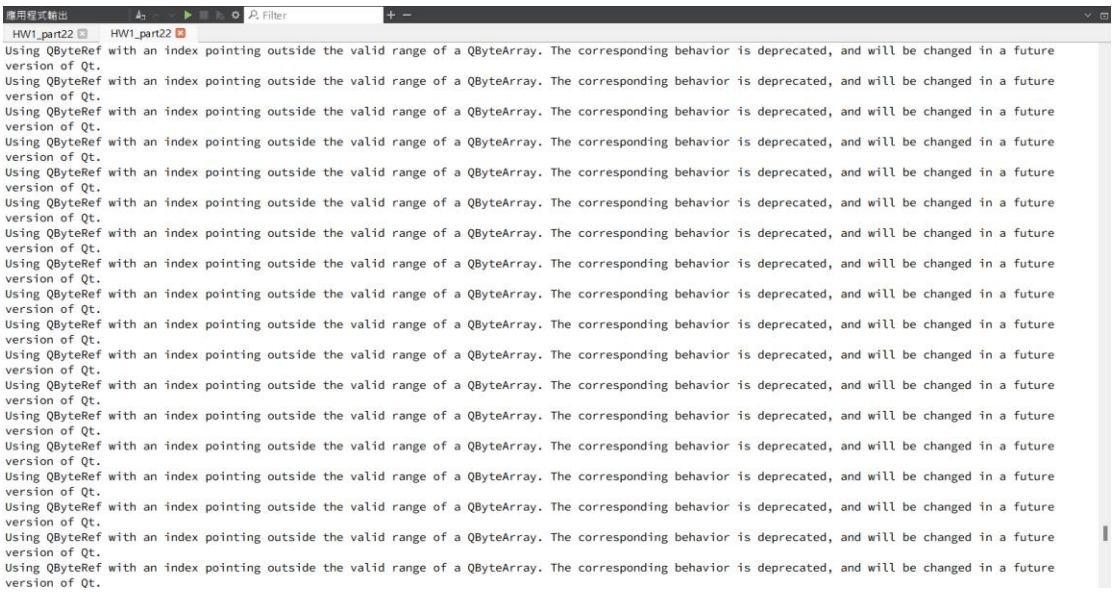
此次作業目標是要將題目所提供的.64 檔案透過程式轉成由灰階所組成的圖像，並另外計算每個灰階值的數量，作出 Histogram。

1. 程式介紹

雖然作業共有兩個小題，但程式碼皆大同小異，因此選擇將兩小題的要求合併於一個介面內完成。主要的演算法程序大致可分為開檔、讀檔後存入一維陣列、利用 for 迴圈將一維陣列轉為二維陣列（圖一）、將二維陣列轉換為灰階圖像、計算各灰階值的數量與作 Histogram 等步驟。此外，也加入了一些限制讓程式更為完善，例如執行上若無法順利讀檔時，跳出“failed reading”警語，並結束工作，避免程式持續反覆讀取（圖二）。

```
for(int i=0; i < 64; i++){
    for(int j=0; j<64; j++){
        location[i][j] = reader64[i*64+j];        //change 1-D Arrary to 2-D Array
        if(location[i][j] < 58)
            location[i][j] = location[i][j] - 48; //把ASCII碼轉成灰階值
        else {
            location[i][j] = location[i][j] - 55;
        }
    }
}
```

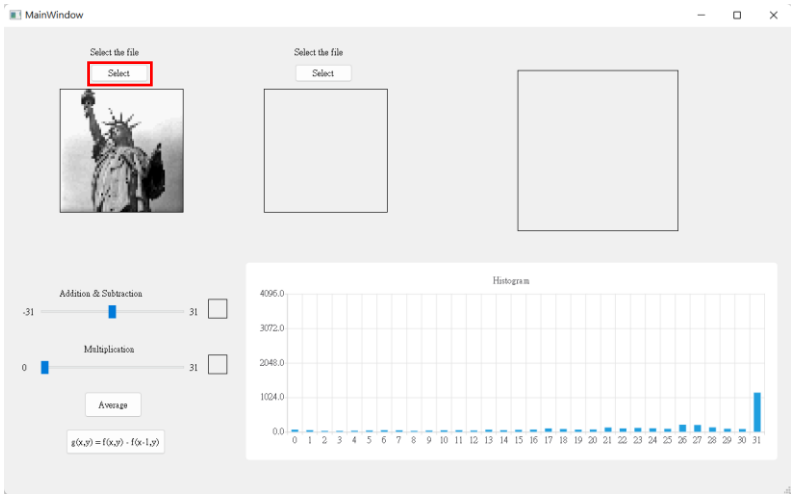
圖一、利用 for 迴圈將一維陣列轉為二維陣列



圖二、未加入限制前，若無法順利讀檔程式將持續反覆讀取

2. UI 介紹

對作業的第一小題而言，按下視窗最左上角的 Select 即可選擇欲讀取的.64 檔。開啟檔案後，該檔案經轉換後的灰階影像即呈現於下方方框內，而其 Histogram 則顯示於視窗的右下角（圖三）。



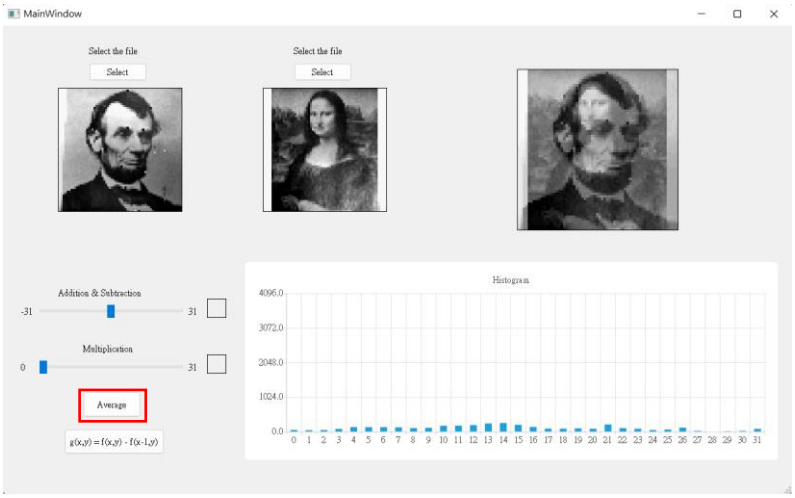
圖三

對作業的第二小題而言，先於視窗最左上角的 Select 按鈕選擇 input 檔案，接著拉動視窗左下角的滑動軸即可對 input 檔案的影像進行處理，並將 output 影像與 Histogram 分別呈現於視窗最右上角與右下角的方框內。其中，上放的滑動軸可將 input 影像每個像素的灰階值加一定值（-31~31）後呈現；而下放的滑動軸則可將 input 影像每個像素的灰階值乘以一定值（0~3.1）後呈現（圖四）。

圖四

此外，亦可透過視窗最左下角的兩個按鈕對影像進行處理，並同樣將 output 影像與 Histogram 分別呈現於視窗最右上角與右下角的方框內。其中，“Average”按鈕按下後可將視窗上方兩處 Select 按鈕所選之兩個影像的所有灰階值作平均處理後得到 output 影像（圖五）；而“$g(x,y) = f(x,y) -$

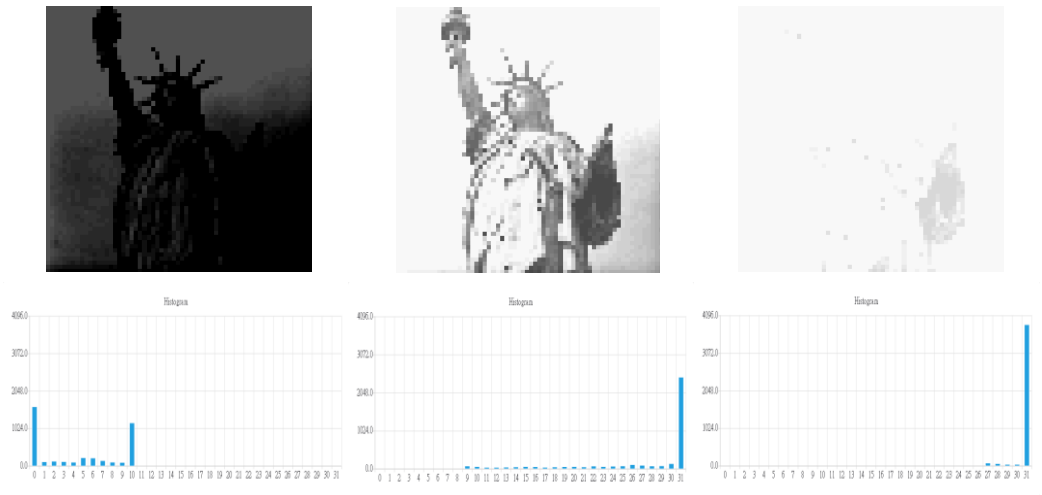
$f(x-1,y)$”按鈕按下後則可將視窗最左上方 Select 按鈕所選影像的每個像素灰階值與前一像素灰階值相減後得到 output 影像。



圖五、使用“Average”按鈕平均處理兩個 input 影像

3. 結果與討論

在作業的第二小題中可以發現，當對一 input 影像每個像素的灰階值加上一常數 n，若該常數為正數，則愈大影像愈明亮且 Histogram 分布愈靠右；反之，若該常數為負數，則愈小影像愈黑暗且 Histogram 分布愈靠左（圖六）。



n = -21　　n = 9　　n = 27

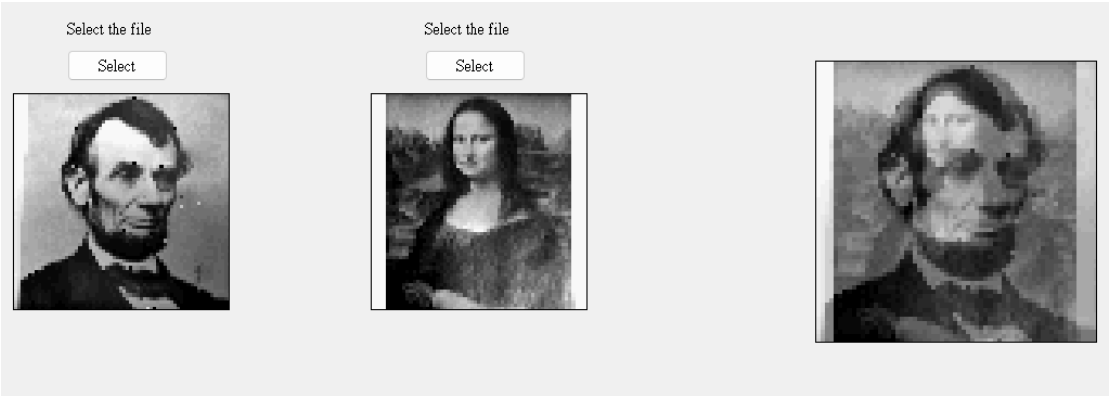
圖六

當對一 input 影像每個像素的灰階值乘以一非負常數 m 時，m 的值愈大，影像愈明亮且 Histogram 分布愈靠右。而當 m=0 時，由於全部像素的灰階值皆變為 0，故影像呈現全黑（圖七）。

m = 0　　m = 0.9　　m = 3.1

圖七

當對兩 input 影像每個像素之灰階值取平均後，所得到的 output 影像正如同將兩張底片重複曝光，有疊加的效果（圖八）。



圖八

當把 input 影像的每個像素灰階值與前一像素灰階值相減（即 $g(x,y) = f(x,y) - f(x-1,y)$ ）時，所得之 output 影像會在原 input 影像之「右亮左暗的明暗交界處」有較大的灰階值，其餘地方的灰階值皆極小，故 output 影像可粗略勾勒出 input 影像之輪廓線（圖九）。

input　　output

圖九