國立雲林科技大學

一百一十二學年度第二學期

數位影像處理

第五次平時作業

班 級：四電子二A

學 號：B11013067、B11013031

學 生：林雅夫、楊孟哲

授課教師：藍呂興 老師

繳交日期：113年05月05日

[一. 中值濾波器 5](#_Toc165906906)

[1、 題目說明： 5](#_Toc165906907)

[2、 本程式之演算法： 6](#_Toc165906908)

[3、 製作方法： 7](#_Toc165906909)

[4、 討論與分析： 8](#_Toc165906910)

[5、 程式碼解說: 10](#_Toc165906911)

[6、 心得: 10](#_Toc165906912)

[A. B11013067: 10](#_Toc165906913)

[B. B11013031: 11](#_Toc165906914)

[二. 直方圖均質化 12](#_Toc165906915)

[1、 題目說明： 12](#_Toc165906916)

[2、 本程式之演算法： 12](#_Toc165906917)

[3、 製作方法： 13](#_Toc165906918)

[4、 討論與分析: 13](#_Toc165906919)

[5、 程式碼解說: 14](#_Toc165906920)

[6、 心得: 15](#_Toc165906921)

[A. B11013067: 15](#_Toc165906922)

[B. B11013031: 15](#_Toc165906923)

[三. 錯誤擴散 16](#_Toc165906924)

[1、 題目說明： 16](#_Toc165906925)

[2、 本程式之演算法: 17](#_Toc165906926)

[3、 製作方法： 18](#_Toc165906927)

[4、 討論與分析： 18](#_Toc165906928)

[5、 程式碼解說: 20](#_Toc165906929)

[1、 心得: 21](#_Toc165906930)

[五. 旋轉 24](#_Toc165906931)

[1、 題目說明： 24](#_Toc165906932)

[2、 本程式之演算法: 25](#_Toc165906933)

[3、 製作方法： 25](#_Toc165906934)

[A. 獲得新的大小 25](#_Toc165906935)

[B. 位置對應： 26](#_Toc165906936)

[C. 雙線性內插補點法： 26](#_Toc165906937)

[4、 討論與分析: 27](#_Toc165906938)

[5、 程式碼解說: 29](#_Toc165906939)

[A. 程式總覽 29](#_Toc165906940)

[B. 取得生成圖像大小、中心 29](#_Toc165906941)

[C. 雙線性插值 30](#_Toc165906942)

[D. 預先填入 30](#_Toc165906943)

[6、 心得: 31](#_Toc165906944)

[A. B11013067： 31](#_Toc165906945)

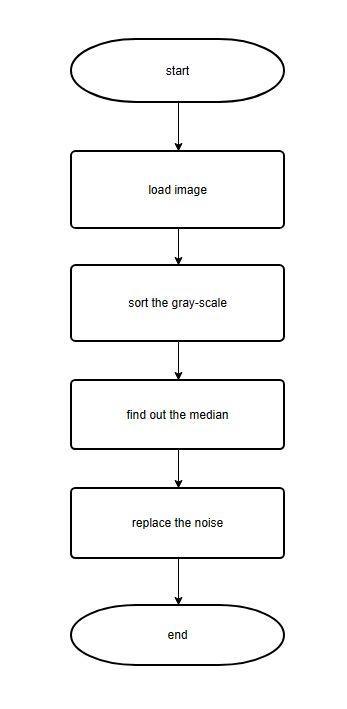
[B. B11013031： 31](#_Toc165906946)

# 中值濾波器

## 題目說明：

利用課堂學所的在一個可更改大小的濾波window内,收集所有像素點的灰階值。對這些灰階值進行排序,找出中位數作為新的中心像素點的灰階值。然判斷當前窗口內的中值是否為噪點。如果中值在最大值和最小值之間,則認為不是噪點，如果中心像素在最大值和最小值之間,則保留該像素;否則用中值代替該像素。

## 本程式之演算法：



## 製作方法：

對於每個像素位置，取其周圍指定大小的區域內的像素值，宣告濾波器的大小filter\_size，並計算濾波窗口的大小，即濾波器大小的平方。將其排序並取中間值作為該像素的新值。同時需要考慮圖像邊界的情況。並確定濾波窗口中心周圍的像素範圍，並檢查鄰近像素是否在圖像邊界內。然後再使用快速排序算法對資料進行排序。再來才計算中值的索引。再將中值丟給濾波後數據的對應位置。

## 討論與分析：

如果沒有使用qsort去重新排列的話圖像可能會沒有濾除噪點，並且反白，下圖是反白但沒有消除噪點的圖



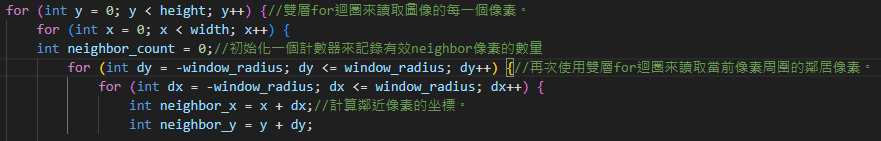
而宣告一個filter\_size的integer可以調整圖像的濾波器大小，以下為濾波器大小為3x3(左)以及13x13(右)的結果



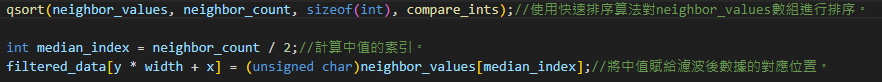
## 程式碼解說:



這裡先宣告濾波器的大小，方便控制濾波器大小已達成題目要求；以及為了計算周圍鄰近的像素，宣告了一個濾波窗口半徑，大小為濾波器的一半。



此為主要計算圖像的像素



這邊是利用qsort來做排序，避免圖像排序錯誤，導致圖像沒能達到要求；接著計算中值，並將他傳給濾波後的像素值。

## 心得:

## B11013067:

在利用老師上課所教授的算法後，沒能夠順利地產生我要的結果，因此在網路上參考了很多的資料，才得以解決問題。並且中值濾波器的計算複雜度較高,隨著濾波窗口大小的增加而增加。而且在某些需要實時處理的應用中,中值濾波器的效率可能沒有預期的那麼高。

## B11013031:

濾波器可以將圖像中的噪點過濾掉，藉此達到影像的平滑化。在改變不同遮罩大小後可以看到影像被造成了不同程度的平滑。而此算法的問題在於，慮波的過程中會造成影像的模糊，鑑於此，我不認為濾波器算是低失真的處理方法。

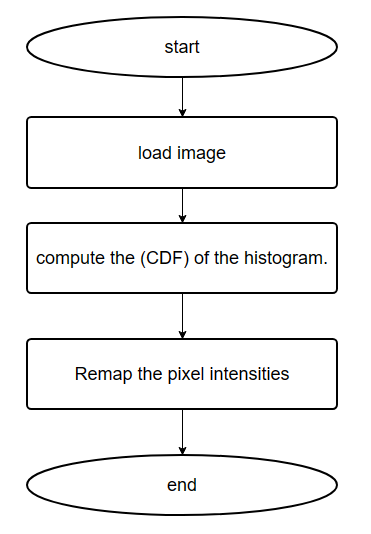
貢獻度:B11013067:70% ;B11013031:30%

# 直方圖均質化

## 題目說明：

改善影像的對比度和亮度分佈。是一種強度轉換方法，通過重新分配像素的亮度值，使得影像的亮度值分佈更均勻，從而增強影像的對比度和細節。

## 本程式之演算法：



## 製作方法：

計算輸入圖像的直方圖, 然後計算直方圖的累積分佈函數(CDF)。再來使用 CDF 對像素強度進行重新映射,以拉伸出現頻率最高的強度值到整個動態範圍。這是通過將每個像素值映射到與其 CDF值成比例的新值來完成的。

## 討論與分析:

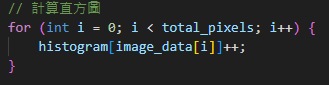
 

上圖左為原圖，右圖為直方圖均質化後的圖，經過直方圖均質化後明顯是不同張圖，各個細節都變得更加鮮明，明亮。直方圖均衡化拉伸像素值以涵蓋整個動態範圍,這減少最暗和最亮像素之間的差異。這讓影像看起來平淡、褪色或嘈雜,尤其是原始影像具有低對比度或解析度的情況。

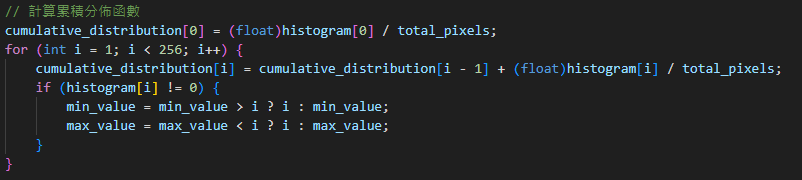
 

上圖是我利用別張圖所呈現出的亮暗之間明顯的差距，對比也更明顯。

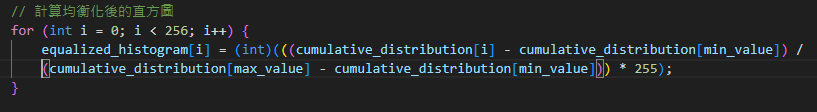
## 程式碼解說:



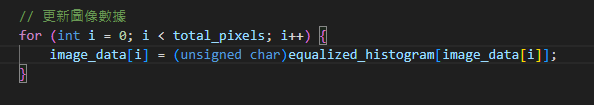
先計算原圖的直方圖



接著計算累積分布函數



再計算經過直方圖均值化後的直方圖



最後再把值回傳回去

## 心得:

## B11013067:

在編寫直方圖均值化的程式碼，一路上沒有遇到太大的問題，唯獨在理解以及計算累積分布函數的時候花了不少的時間，這當中涉及太多數學的原理，也慶幸是生在網路發達的時代，網路上有很多累積分布函數的解釋可以幫我理解這個數學原理。

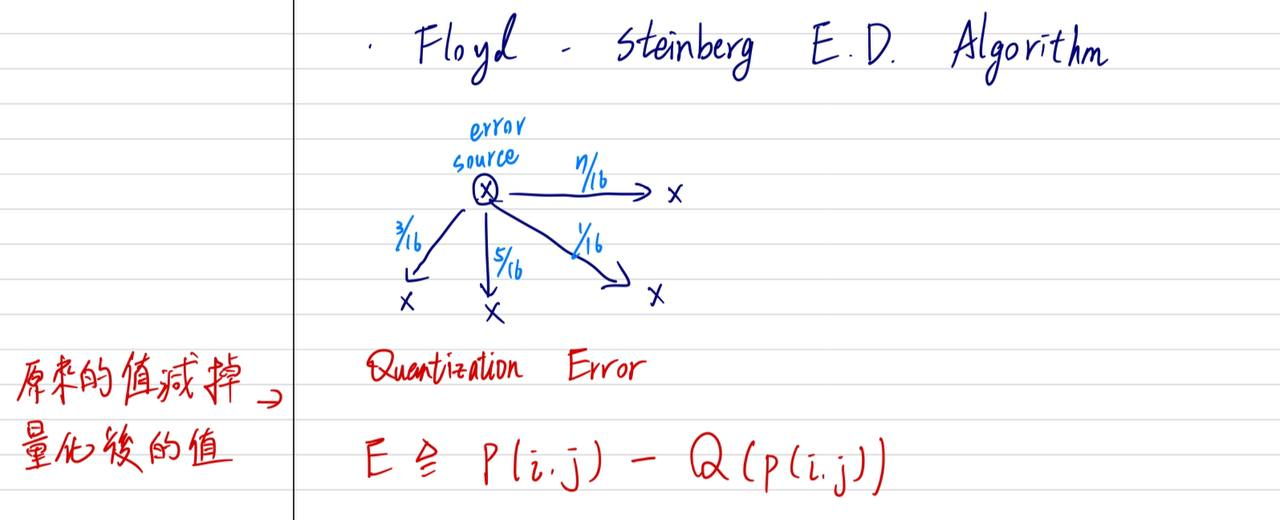
# B11013031:

使用了直方圖均值化後，能將圖像灰值的上下限拉大，也就是增強了對比度，將原本難以辨識的圖片以此法處裡後，能生成更易於識別的圖像。

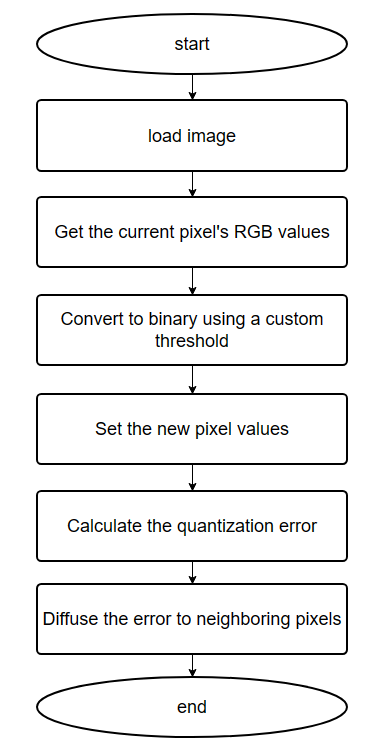
貢獻度:B11013067:70% ;B11013031:30%

# 錯誤擴散

## 題目說明：

比較像素的實際顏色與其最接近的顏色,並將差異(誤差)擴散到鄰近的像素。將部分誤差分配給鄰近的像素,確保擴散的總誤差不超過1(Floyd-Steinberg)。計算量化後的值，並利用原像素點減掉量化後的值，得到量化誤差(quantization error)。接著利用Floyd-Steinberg的演算法去計算像素值。下圖為Floyd-Steinberg E.D Algorithm和量化誤差計算公式。其中各項比例分別為3/16, 5/16, 1/16, 7/16加起來剛好為1。

## 本程式之演算法:



## 製作方法：

先獲取原圖像素的RGB，再利用一個臨界值使他二值化(0跟255)，再設置新的像素值(量化後的值)，接著計算量化誤差(原圖像素值減掉新的像素值)，最後再利用計算Floyd-Steinberg E.D Algorithm去計算新圖的值。

## 討論與分析：

對RGB的圖進行error diffusion時，要做三次，如果沒有這麼做，圖像會隨意排列，下圖為沒有使用gray-scale的算法對full-color進行error diffusion

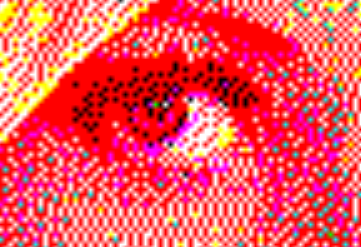
 

當顏色排列錯誤時會出現奇怪的顏色

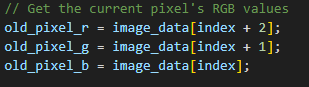
以下為準確的經過error diffusion所產生的圖:



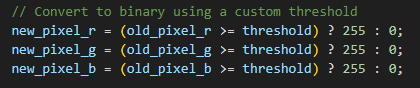


當圖片放大時可以明顯看到一些特殊的點

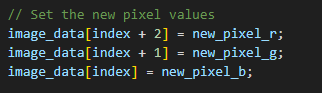
## 程式碼解說:



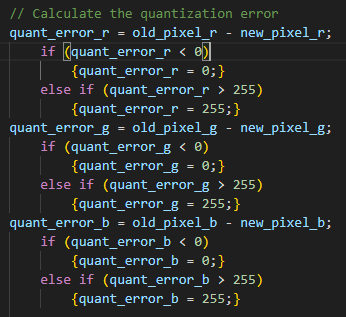
先獲取原圖像素的RGB



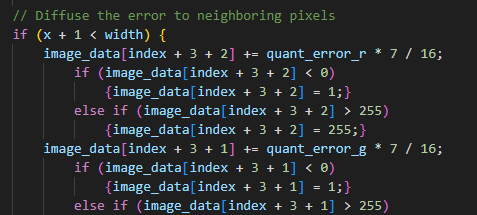
把它二值化



設置新的像素值(量化後的值)



計算量化誤差(原圖像素值減掉新的像素值)，並確保不會溢位



最後再利用計算Floyd-Steinberg E.D Algorithm去計算新圖的值，同樣確保他不會溢位。

## 心得:

對RGB的圖進行error diffusion時，要做三次，如果沒有這麼做，圖像會隨意排列，下圖為使用gray-scale的算法對full-color進行error diffusion

|  |  |
| --- | --- |
| Figure 19 | Figure 20 |
| 使用gray-scale的算法對full-color進行error diffusion | |

|  |  |
| --- | --- |
| Figure 21 | Figure 22 |
| 當顏色排列錯誤時會出現奇怪的顏色 | |

|  |  |
| --- | --- |
| Figure 23 | Figure 24 |
| Figure 23、Figure 24為準確的經過error diffusion所產生的圖 | |

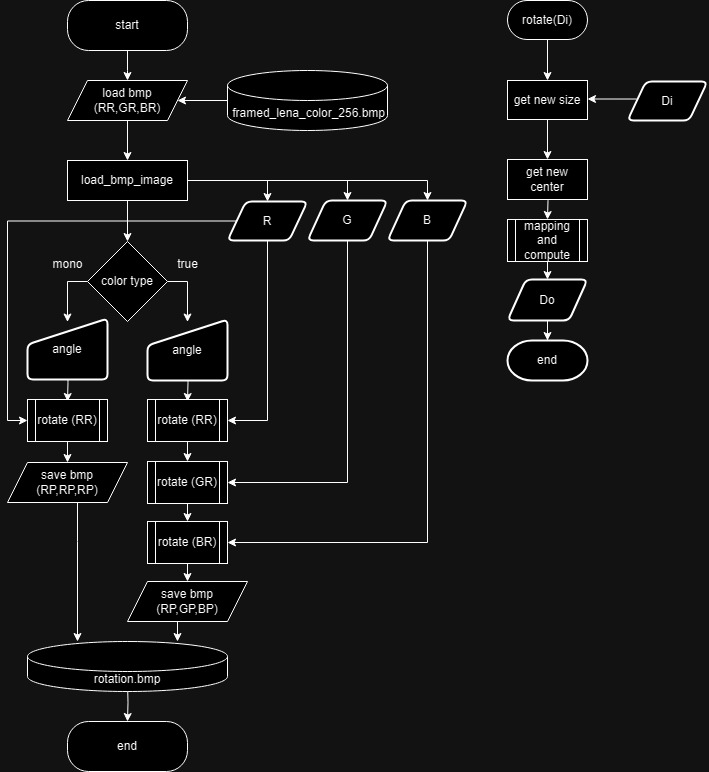
貢獻度:B11013067:30% ;B11013031:70%

# 旋轉

## 題目說明：

設計程式以將指定圖片之中心部分旋轉至任意角度，並利用老師上課所說的內插補點法修正色彩，

## 本程式之演算法:



## 製作方法：

## 獲得新的大小

先透過輸入的角度值，算出sin、cos以避免之後過多浮點數運算。接著計算原圖中心點、生成圖中心點以及旋轉後的圖像大小。

## 位置對應：

為要生成的圖片之陣列(src[new\_x,new\_y])取得其位置最接近的經旋轉原圖像素位置（src\_x,src\_y）

## 雙線性內插補點法：

取得其左(src\_x0)、其右(src\_x1)及其上(src\_y0)、其下(src\_y1)的整數值，dx,dy則為該點距其左、其上之距離。得到四點(c00,c01,c10,c11)以及兩值(dx,dy)後，分別計算四點的顏色影響量再將其相加為一整數值，及為該點經雙線性內插補點法後得出的顏色。((1-dx)\*(1-dy)\*c00)+((1-dx)\*dy\*c01)+(dx\*(1-dy)\*c10)+(dx\*dy\*c11)。

## 討論與分析:

|  |
| --- |
|  |
| 原圖 |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

## 程式碼解說:

## 程式總覽

|  |
| --- |
|  |
| 鎖定圖像的中心點以及對應的位置 |

## 取得生成圖像大小、中心

|  |
| --- |
|  |
| 計算角度所對應的三角函數值，以減少浮點數的計算量 |

## 雙線性插值

|  |
| --- |
|  |
| 實現雙線性內插補點法 |

## 預先填入

|  |
| --- |
|  |
| 確保新x,y的值不會被略過，每一個新值都有對應的數值 |

## 心得:

## B11013067：

在編寫圖片旋轉這份程式時，我根據老師上課所講的內插補點法得知可以應用不同的插值方法來處理旋轉後的像素值，以及旋轉矩陣可以根據旋轉角度計算。雖然線性代數以及其他理論的部分花了一些時間去理解，但經過這次的實作之後，對於線性代數這個科目的用意以及對於旋轉圖片的背後含意有了更深層的理解。

## B11013031：

在編寫旋轉的程式時，有許多不大清楚的算式，所幸這是許多人解過的問題，因此在網路上有不少的解題說明，最後除了解決缺點的問題外，也看懂了這個算法的意義。

貢獻度:B11013067:30% ;B11013031:70%