作業 3

課號: CO6041

課名: 數位影像處理

教師:唐之瑋

學號末三碼:009

姓名: 鄧翔冠

一、 實驗步驟說明

實驗步驟如圖 3.1，一開始讀取 24 bits 的影像(圖 4.1 和圖 4.14)，將矩陣的 B、G、R 三維的矩陣資料分別存起來，個別處理每一維度並進行高通濾波，如圖 4.2 和 4.3 為針對排球照片(圖 4.1)Blue 這一維度進行 Sobel X 和 Sobel Y 的處理，最後兩圖相加取得圖 4.4，此為高通濾波後的結果，而 G、R 兩維進行同樣的處理，結果見圖 4.4~4.10；依此類推於魔術方塊，使用相同的處理方式。

將邊緣(edge)取得後，將沒有必要的記憶體釋放出來，因為在記憶體配置上，會有大小的限制，如圖 1.1，在做 1000×1000×3 的排球影像時，如果沒有將記憶體釋放掉，會導致 malloc 分配記憶體時，heap 會溢出，導致程式執行一半當機。

將每一維度(BGR)的 Sobel X 和 Y 相加起來，此為最後的邊緣偵測，而後取一臨界值(threshold)，將此影像進行二值化、中值濾波(median filter)、低通濾波(Gaussian filter)，進行完前處理後，使用 Connected Component 的演算法，本次實驗使用八近鄰的方式，找出每一個區塊和物體，之後對其加上標籤。

有每一個物體的標籤後，其標籤在值上面有高低的分界，用此分界做為 Watershed Algorithm 的臨界值，對不同標籤做上色的部分(24 bits)，將影像儲存成彩色的影像，最後結果與討論，討論此流程、方法的結果是否好用、有效率等等。

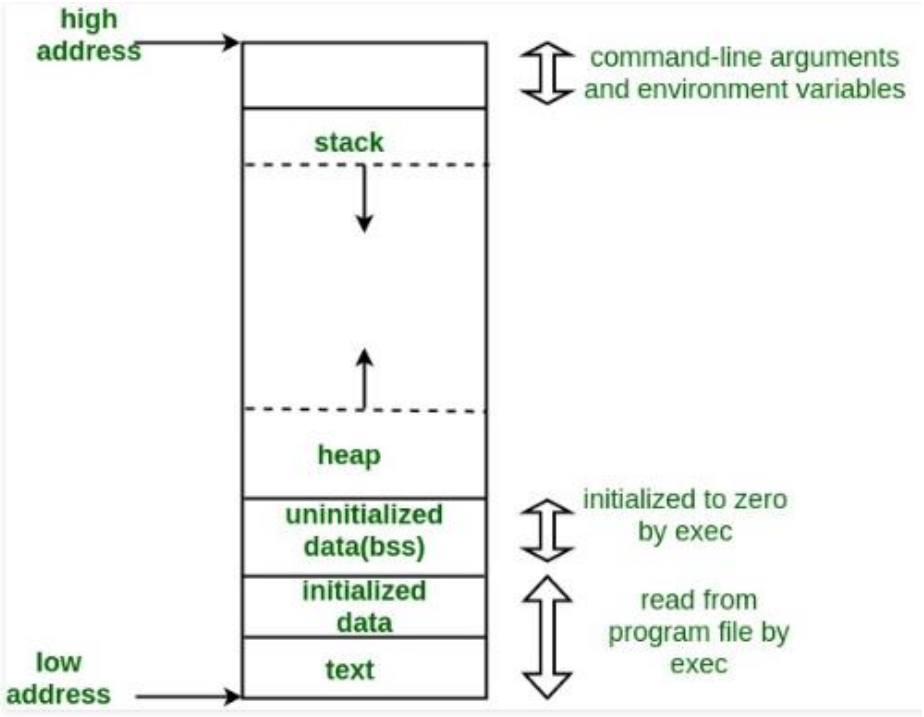

圖 1.1 記憶體配置參考圖

二、 學習目的

本次實驗的主要目的主要為實現 Watershed Algorithm 和 Image Segmentation，並具備對圖片進行前處理的能力，如低通濾波、高通濾波、中值濾波等等，如果前處理沒有做好，對於後續將圖片切割、上色會有很大的影響。

三、 實驗步驟流程圖

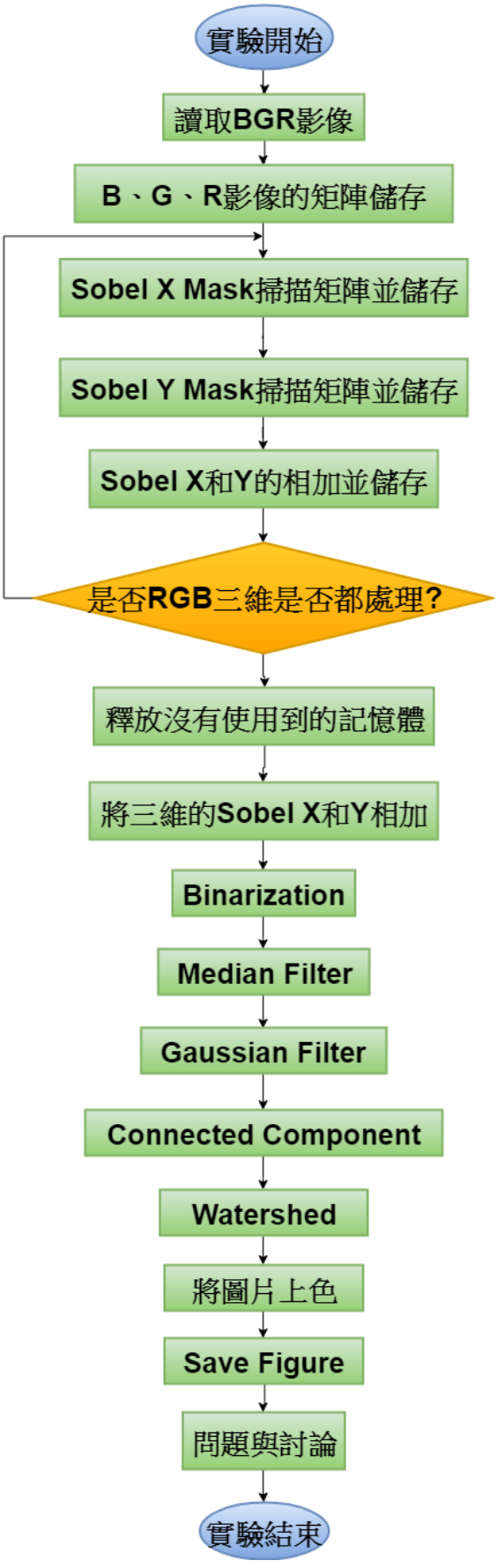

圖 3.1 整體實驗流程

四、 實驗結果

排球執行 command line 顯示結果:

```
Identifier is : 19778
FileSize is : 3000054
Reserve is : 0
BitmapDataOffset is : 54
BitmapHeaderSize is : 40
Width is : 1000
Height is : 1000
Planes is : 1
BitsPerPixel is : 24
Compression is : 0
BitmapDataSize is : 1000000
H_Rosolution is : 0
U_Rosolution is : 0
UsedColorsSize is : 0
ImportantColors is : 0

===== Main 1 =====
Image Max value is 255
Image Min value is 0
marker Max value is 11
marker Min value is 0

===== END =====
```

圖 4.1 排球原始圖

圖 4.2 排球 Blue 維度 Sobel X 的結果

圖 4.3 排球 Blue 維度 Sobel Y 的結果

圖 4.4 排球 Blue Sobel X 和 Y 相加

圖 4.5 排球 Green 維度 Sobel X 的結果

圖 4.6 排球 Green 維度 Sobel Y 的結果

圖 4.7 排球 Green Sobel X 和 Y 相加

圖 4.8 排球 Red 維度 Sobel X 的結果

圖 4.9 排球 Red 維度 Sobel Y 的結果

圖 4.10 排球 Red Sobel X 和 Y 相加

圖 4.11 銳化過後(using sobel mask)

圖 4.12 HW3-1-1 去雜訊、平滑、侵蝕、膨脹後的結果

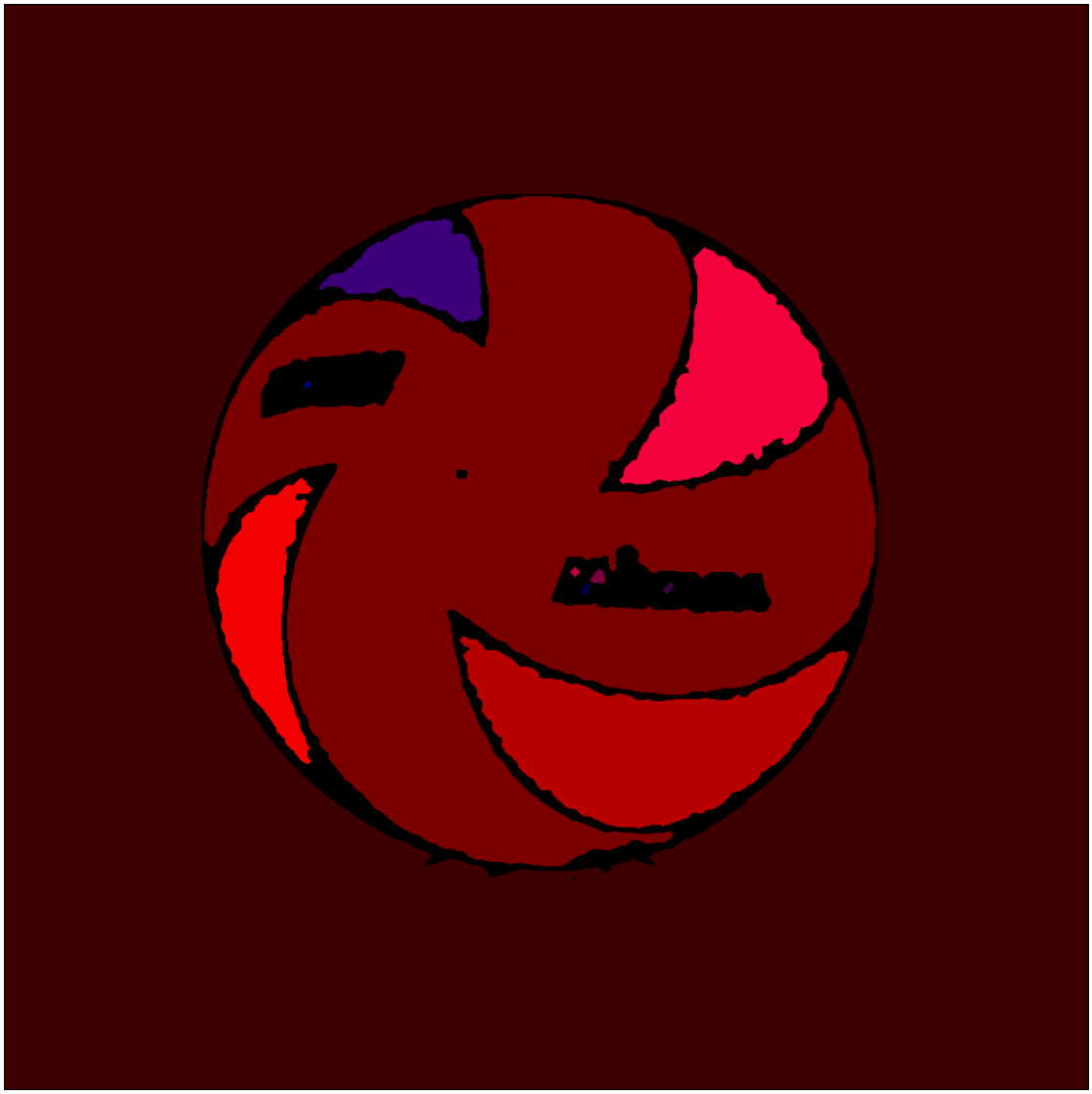

圖 4.13 HW3-1-2 將圖 4.3 上色後結果

魔術方塊執行 command line 顯示結果:

```
Identifier is : 19778
FileSize is : 270054
Reserve is : 0
BitmapDataOffset is : 54
BitmapHeaderSize is : 40
Width is : 300
Height is : 300
Planes is : 1
BitsPerPixel is : 24
Compression is : 0
BitmapDataSize is : 90000
H_Rosolution is : 0
U_Rosolution is : 0
UsedColorsSize is : 0
```

```
ImportantColors is : 0

===== Main 2 =====
Image Max value is 255
Image Min value is 0
marker Max value is 29
marker Min value is 0

===== END =====
```

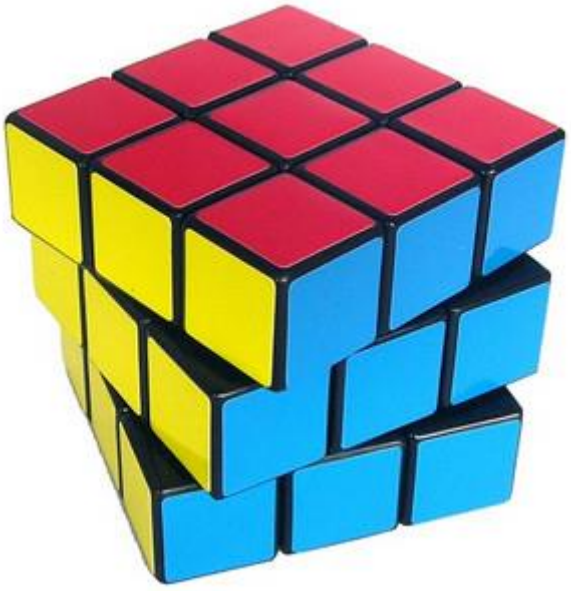

圖 4.14 魔術方塊原始圖片

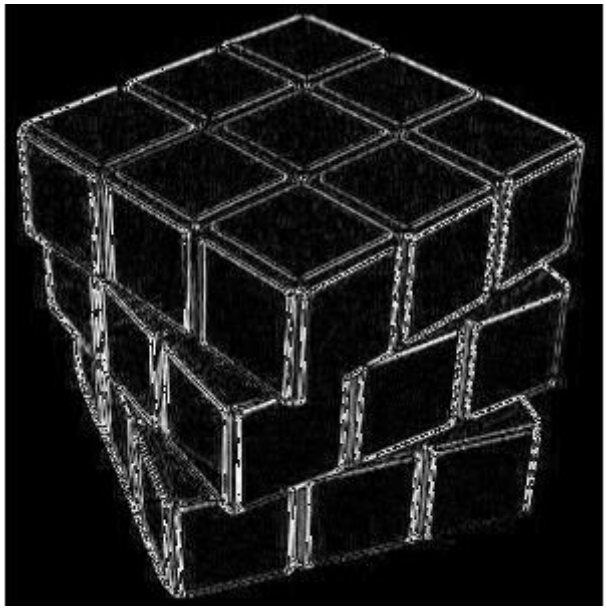

圖 4.15 魔術方塊 Blue 維度 Sobel X 的結果

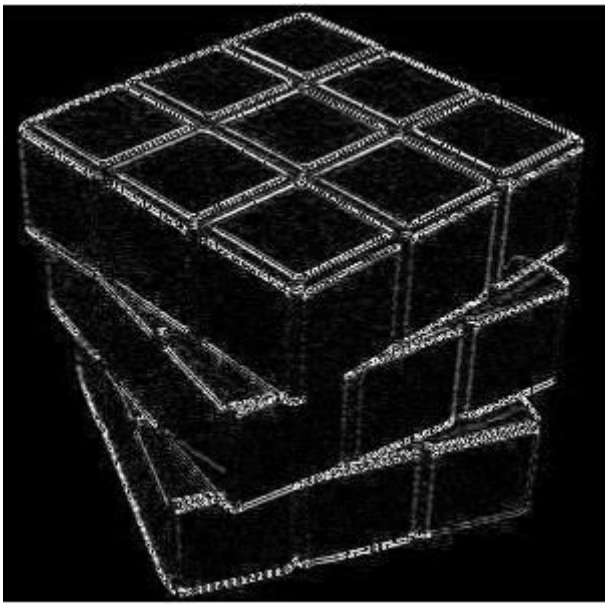

圖 4.16 魔術方塊 Blue 維度 Sobel Y 的結果

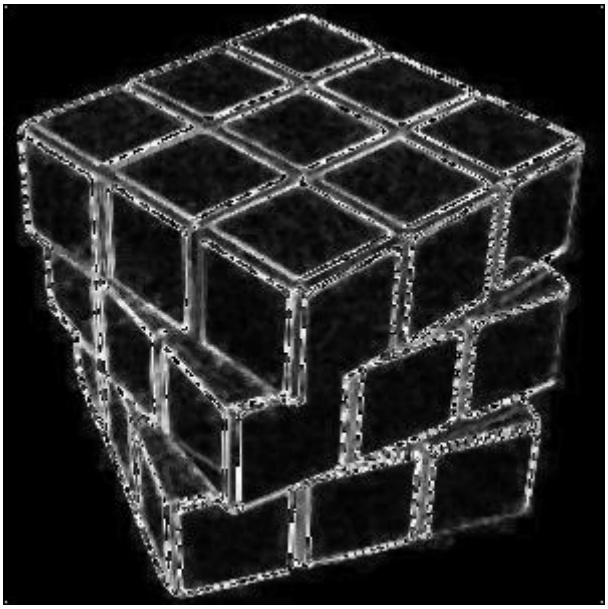

圖 4.17 魔術方塊 Blue 維度 Sobel X 和 Y 相加

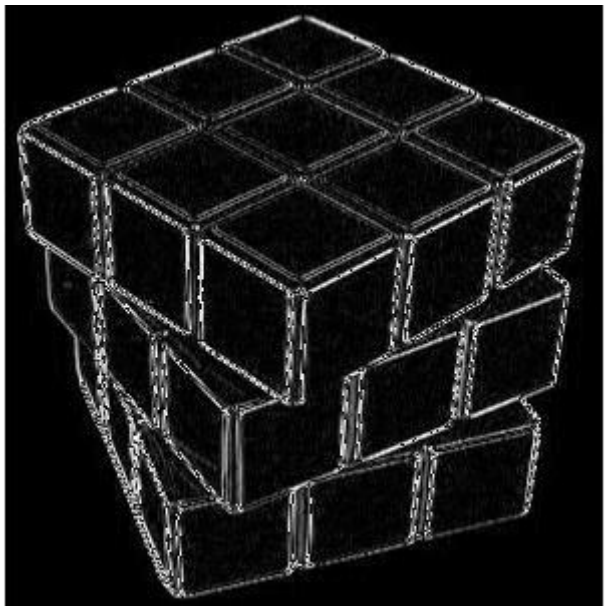

圖 4.18 魔術方塊 Green 維度 Sobel X 的結果

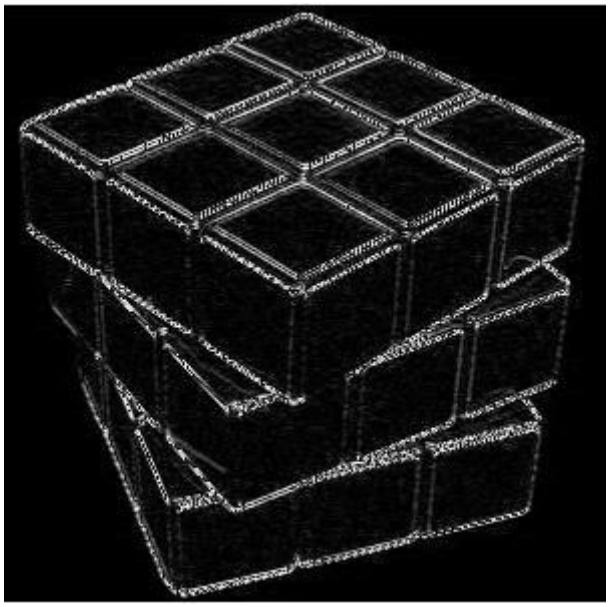

圖 4.19 魔術方塊 Green 維度 Sobel Y 的結果

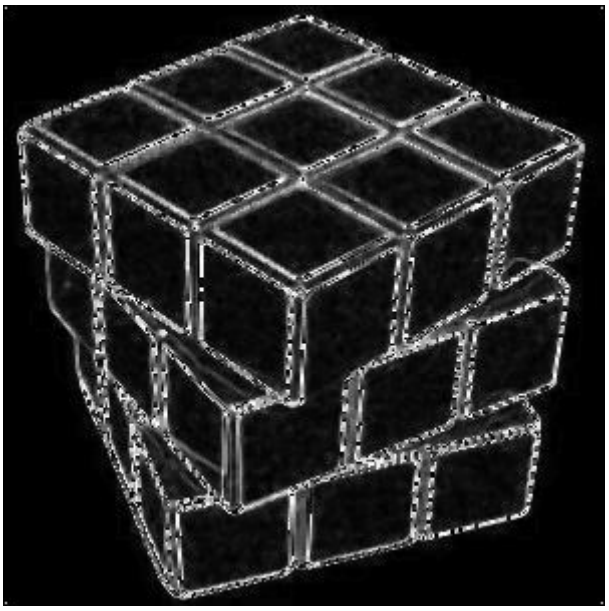

圖 4.20 魔術方塊 Green 維度 Sobel X 和 Y 相加

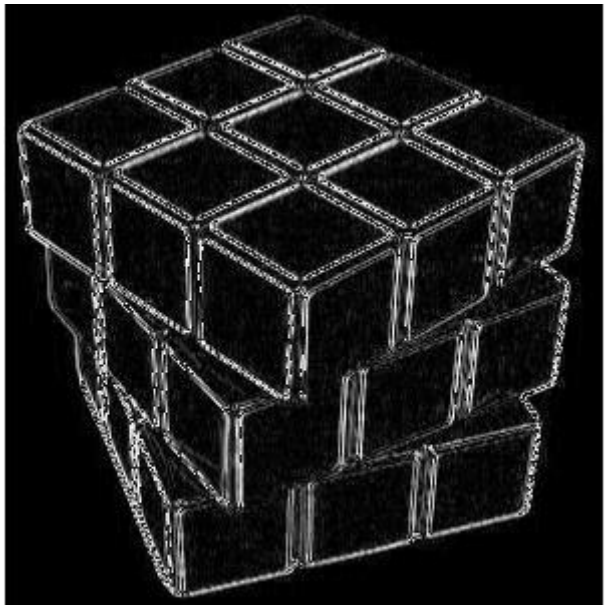

圖 4.21 魔術方塊 Red 維度 Sobel X 的結果

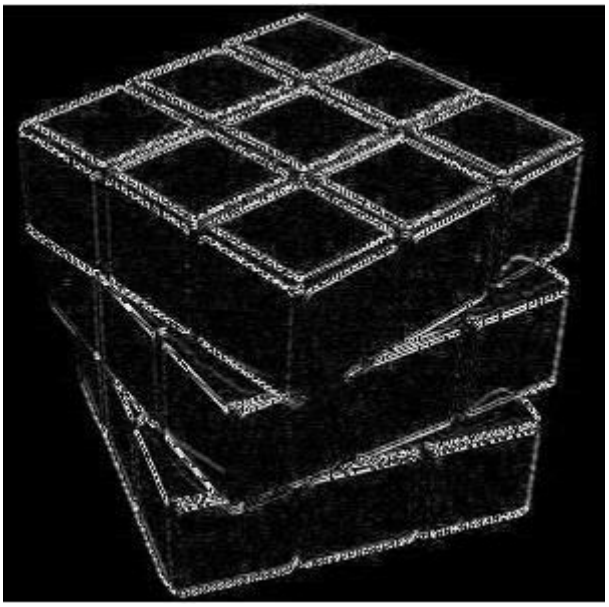

圖 4.22 魔術方塊 Red 維度 Sobel Y 的結果

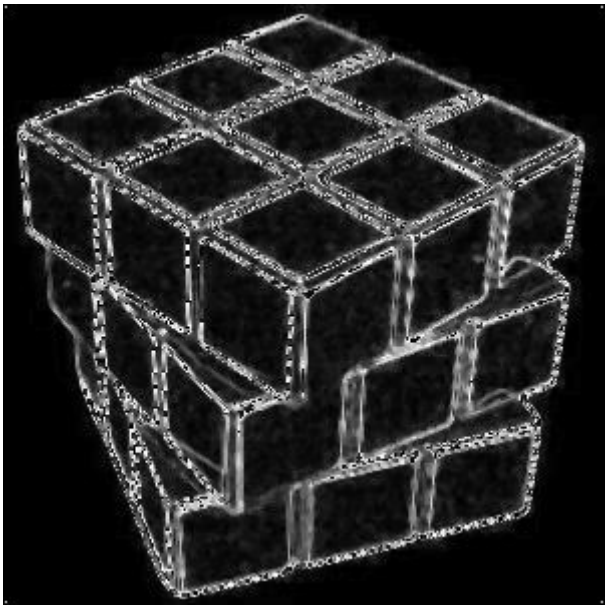

圖 4.23 魔術方塊 Red 維度 Sobel X 和 Y 相加

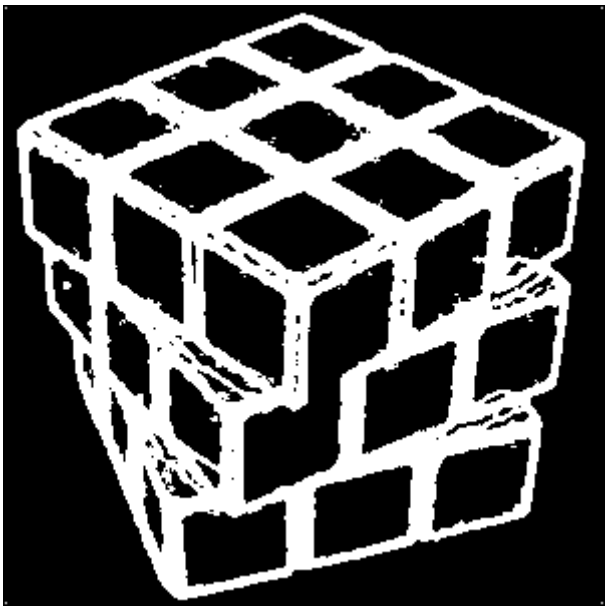

圖 4.24 銳化過後(using sobel mask)

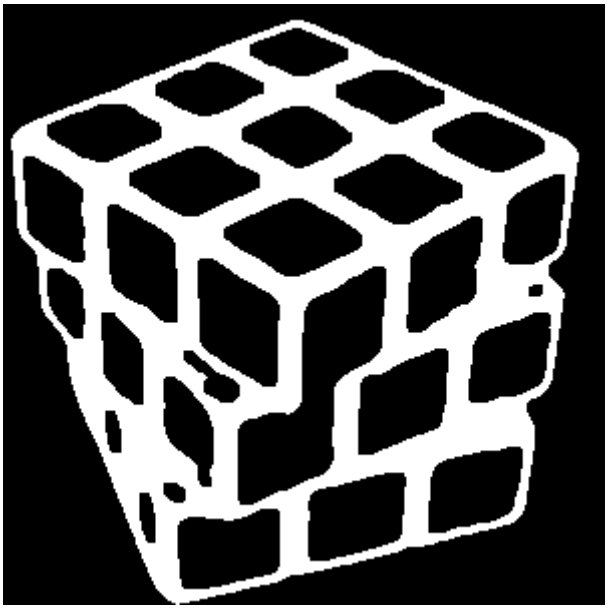

圖 4.25 HW3-2-1 去雜訊、平滑、侵蝕、膨脹後的結果

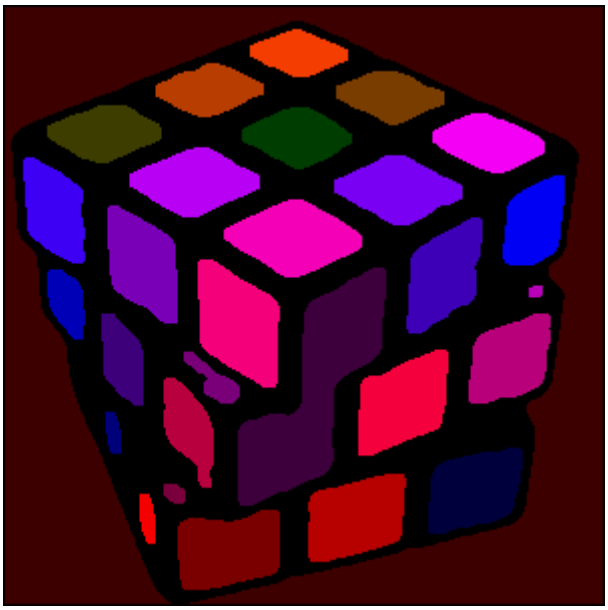

圖 4.26 HW3-2-2 將圖 4.7 上色後結果

1. 本次實驗 Filter Mask 的遮罩矩陣為:
   I. 高斯平滑的遮罩(Mask)

```
int8_t gaussianMask[3][3] = {
    // need to divide 16
    {1,2,1},
    {2,4,2},
    {1,2,1},
};
```

   II. Sobel Mask (X axis)

```
int8_t sobelMaskX[3][3] = {
    {-1,-2,-1},
    {0,0,0},
    {1,2,1},
  };
```

   III. Sobel Mask (Y axis)

```
int8_t sobelMaskY[3][3] = {
    {-1,0,1},
    {-2,0,2},
    {-1,0,1},
  };
```

   IV. 侵蝕膨脹用的 Mask - 1

```
  uint8_t mMask1[3][3] = {
    {0,1,0},
    {1,1,1},
    {0,1,0},
  };
```

V. 侵蝕膨脹用的 Mask - 2

```
uint8_t mMask2[3][3] = {
  {1,1,1},
  {1,1,1},
  {1,1,1},
};
```

2. 開發環境:
   i. 電腦廠牌 : Lenovo T470p
   ii. 作業系統 : Windows10
   iii. CPU : Intel®Core™i7-7700HQ
   iv. RAM : 16G
   v. Compiler : gcc (MinGW.org GCC-6.3.0-1) 6.3.0
   vi. IDE : 文字編輯器 Atom
   vii. 執行 : 命令提示字元 (CMD)執行 *.exe
   viii. 程式碼與圖片:https://github.com/ZXPAY/DIP_Project

五、分析

本實驗所使用的邊緣偵測是使用 Sobel 的 Filter Mask，做高通濾波，以抓取邊緣和圖片的細節部分，矩陣詳細的內容如 4.1.II 和 4.1.III，由矩陣的內容可以知道，在掃描過程中，會計算上面會計算圖片 X、Y 軸的差異性，也就是對矩陣做微分，抓取整張圖片變化的部分。本實驗針對 B、G、R 圖片的這三維分別處理，產生圖 4.2 到圖 4.10，可以計算出不同顏色在排球這張圖的差異性，再將每一 B、G、R 分別對 sobel X 和 Y 的相加起來，取一臨界值做二值化變成圖 4.11，此張圖的矩陣資料不是 0 就是 255。

在圖 4.11 裡面，可以發現有蠻多雜訊點的，外圍的輪廓、邊界清晰可見，為避免雜訊點會造成後續 watershed algorithm 造成干擾和誤差，所以使用 median filter(中值濾波器)，此為一種非線性濾波器，本實驗使用 7×7、5×5、3×3 的中值濾波器，在掃瞄整張圖片的時候，取每個 mask 的中位數作為新的影像 pixel 的值，用此方式可以雜訊有效的清除而較不會影響整張圖片主要的特徵。

雜訊濾除後，使用高斯濾波的低通濾波器，平滑整張圖片，之後做侵蝕、膨脹，讓線條較破損的地方可以連結在一起，一些較不重要、雜訊可以被侵蝕掉，讓整張圖片的邊緣可以很清晰(如圖 4.12)。

之後為了將圖片每個區塊上色，必須知道每個區塊的位置，本實驗使用 Connected Component Algorithm，是將非背景的 pixel 做 mark 的動作，同時會檢查每一個 pixel 是否要給予新的 mark，本實驗使用八近鄰判別是否為同一個 mark，如果當下的 pixel 不是背景，而且此 pixel 周圍尚未有 mark，就給予新的 mark，此實驗演算法參考 wiki 的 Pseudocode code (虛擬碼)實現之。

完成二值化的圖片做 mark 的動作後，因每一區塊的 label 的值不一樣，表示其高度不相同，用 watershed 的演算法，將不同高度的 label 找出來，並使用 BGR 三種顏色調出都不相同的顏色，最後結果如圖 13 和圖 26，同顏色的部分代表 mark 上相同的值，也就是同一區塊的部分。

完成這次實驗後，學習到如何切割影像和如何上色，在此實驗中也發現有許多的問題需要去解決，將影像處理至流程圖的三維 B、G、R sobel X 和 Y 的處理後(見圖 3.1)，再將其取臨界值(threshold)，得出圖 4-11 和圖 4-24，這兩張圖中可以發現個別 B、G、R 做 Edge Detection，再將其結合在一起，有許多的雜訊需要去做處理，在計算上面相對於先轉灰階後再做 Edge Detection 複雜很多，在寫程式的過程中發生過幾次記憶體溢出導致程式當機的情況發生，因此，這是一部分要去思考的問題。

再者，在做二值化的時候，需要去取一臨界值，如果圖片燈光不穩定，使用相同的臨界值，會無法有效的將我們想要的部分切出來，導致後續做影像處理得時候會有很大的誤差、誤判等情形發生，所以，如果能以一能自動判別二值化的演算法去代替固定值得二值化方法，想必能將二值化轉換更具彈性，讓不同亮暗的圖片都能夠去適應。

- Pseudocode code about Connected-component labeling in wiki

```
algorithm TwoPass(data)
   linked = []
   labels = structure with dimensions of data, initialized with the value of Background

   First pass

   for row in data:
      for column in row:
         if data[row][column] is not Background

            neighbors = connected elements with the current element's value

            if neighbors is empty
               linked[NextLabel] = set containing NextLabel
               labels[row][column] = NextLabel
               NextLabel += 1

            else

               Find the smallest label

               L = neighbors labels
               labels[row][column] = min(L)
               for label in L
                  linked[label] = union(linked[label], L)

   Second pass

   for row in data
      for column in row
         if data[row][column] is not Background
            labels[row][column] = find(labels[row][column])

   return labels
```

六、Reference

1. Gonzalez, Rafael C., and Richard E. Woods, “Digital image processing, “ Prentice Hall, 2007.
2. 排球圖片下載：
   https://img2.momoshop.com.tw/goodsimg/0004/292/504/4292504_B.jpg?t=1518336721
3. 魔術方塊下載：
   https://cdn.kingstone.com.tw/cvlife/images/product/30600/3060000006655/3060000006655b.jpg
4. Connected component (graph theory) Wiki：
   https://en.wikipedia.org/wiki/Connected_component_(graph_theory)
5. Image Segmentation with Watershed Algorithm in OpenCV：
   https://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html
6. IMAGE SEGMENTATION AND MATHEMATICAL MORPHOLOGY：
   http://www.cmm.mines-paristech.fr/~beucher/wtshed.html
7. Memory Layout of C Programs：
   https://www.geeksforgeeks.org/?p=14268/
8. Connected-component labeling：
   https://en.wikipedia.org/wiki/Connected-component_labeling