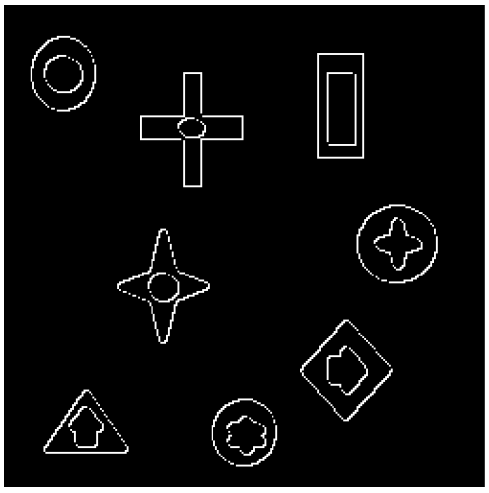
Digital Image Processing
HOMEWORK ASSIGNMENT #3

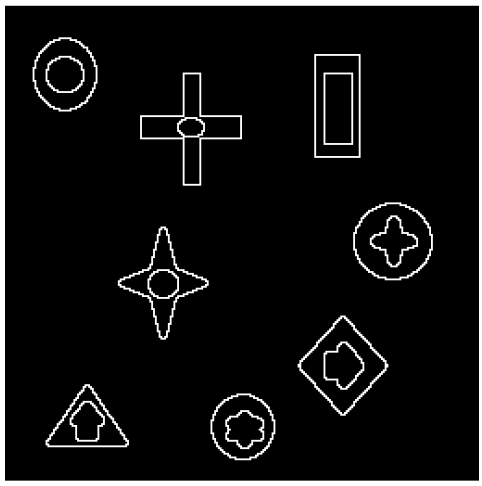
R06922097 資工碩一 鄭雅文

PROBLEM 1: Morphological Processing

(a) 先對原圖做 erosion： $G(j,k) = F(j,k) \ominus H(j,k)$，其中 $F(j,k)$為原圖I_1， $H(j,k)$為過濾 pixel 時所觀察的矩陣，若 $F(j,k)$點的相鄰 pixel 滿足 H 所有為 1 的點，則 $G(j,k) = 1$，否則 $G(j,k) = 0$。最後得到的 $G(j,k)$即為原圖縮減後的結果，下面兩張圖為原圖減掉H_4和H_8過濾後的結果，即是I_1的 boundary。

$$H_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad H_8 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

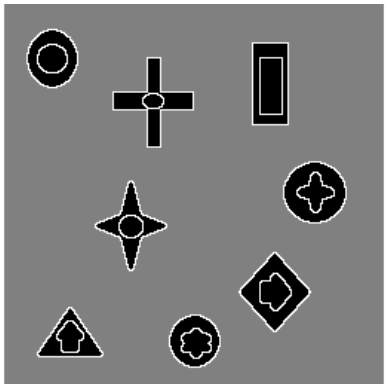




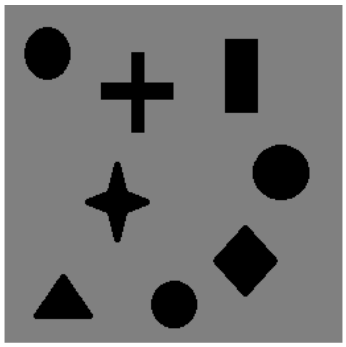
因為H_4考慮的 neighbors 較少，所以條件比較不嚴格，erosion 後的結果縮減較少，所以原圖和 erosion 後的圖相減會造成邊可能沒有完全密合，有鬚鬚的情況。H_8過濾後的結果邊緣較完整，較符合接下來題目的需求。

(b) 演算法：

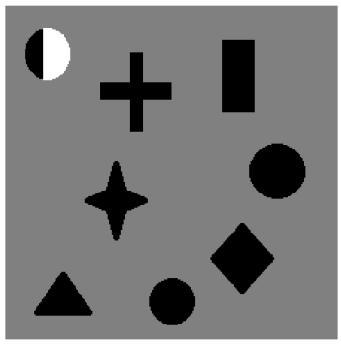
Step 1:先將背景都設為 0.5(灰色)。先將第(1,1)的 pixel 設為灰色，從左上開始如果左邊格或上面格為灰色且 F(i,j)==0，則塗成灰色。除了從左上角開始塗色之外還要從右下角開始塗色，才能顧到所有的方向。



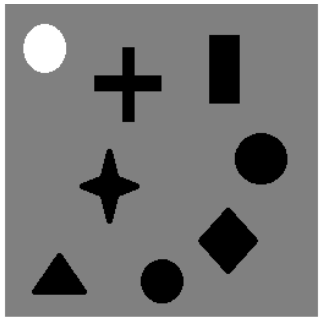
Step 2:將圖型中間的白色線條都塗黑。F(i,j)==1 即設為 0。



Step 3:遞迴搜尋黑色點，如果圖中有黑色點的話，即填滿數字 1~k，即為第幾個 object。若從頭搜尋，找到第一個黑色點時，設 count 為 1，接下來的 pixel 若有 neighbor 為 1 且該 pixel 為 0 的話，則設為 1。但如果從左上方開始搜尋的話，會發生下面這種情況：



圖形的左邊無法預知右邊的 pixel 是否為 1，所以必須從另一個方向(右下)再搜尋一次，找相鄰的 pixel 是否為 1，才能填滿整個圖型。



一直這樣搜尋直到整張圖都沒有黑色的 pixel。為了方便顯示將顏色做改變：(背景為黑，數到的 object 顏色由深到淺顯示)

(c)對原圖先找到要刪除的 candidate，將原圖全部搜尋完後刪除符合的 candidate。直到原圖不再縮減。

選擇要刪除的 candidate 的方式為：(及其相似 pattern)

P9	P8	P7
P2	P1	P6
P3	P4	P5

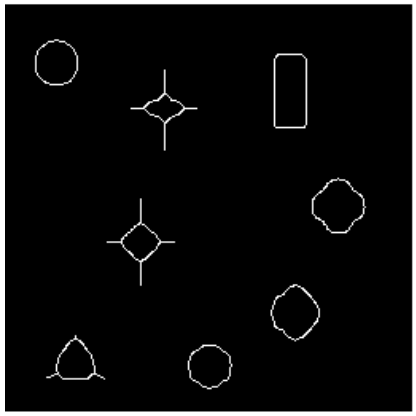
P9	P8	P7
P2	P1	P6
P3	P4	P5

P9	P8	P7
P2	P1	P6
P3	P4	P5

P9	P8	P7
P2	P1	P6
P3	P4	P5

P9	P8	P7
P2	P1	P6
P3	P4	P5

最後 skeletonize 後的結果如下：



Skeletonize 後的結果因為圖形中間都有缺口，所以圖形幾乎是繞著缺口及外圍的方式呈現。一般的正方形原本也有角到中心線的骨架，但這邊有缺口的正方形等於是比較方的甜甜圈形，只像甜甜圈形的骨架一樣在外圍和中間形狀繞一圈。

PROBLEM 2: Texture Analysis

(a)利用 4 種向量組成 9 種 filter：

L5 = [1 4 6 4 1];

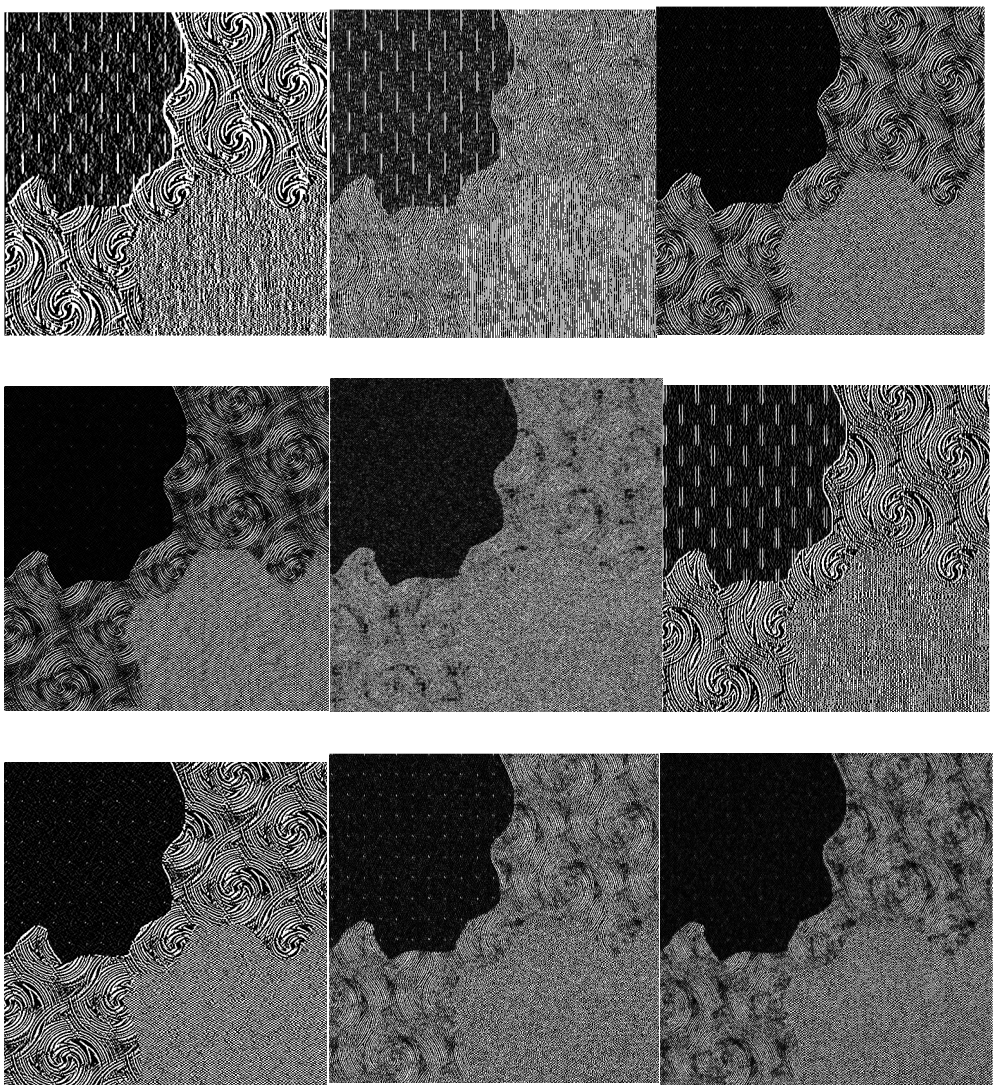
E5 = [-1 -2 0 2 1];

S5 = [-1 0 2 0 -1];

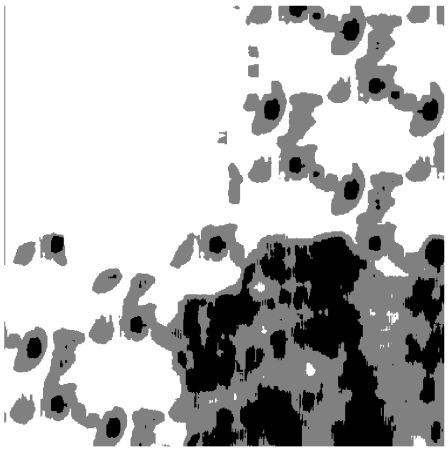
R5 = [1 -4 6 -4 1];

九種組合為：L5E5、L5R5、E5S5、S5S5、R5R5、L5S5、E5E5、E5R5、S5R5。

圖片經過九種 filter 後得到的結果如下：



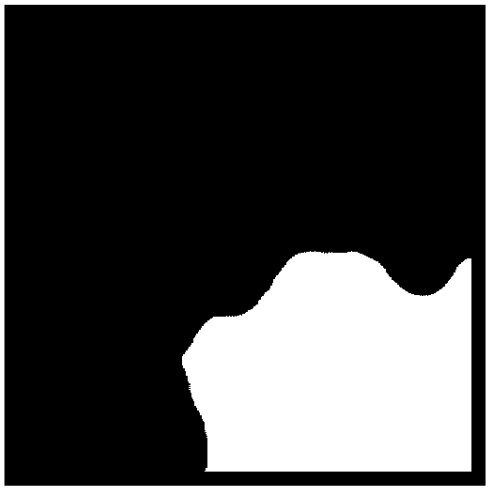
因為直接用 energy 的分類效果不好：



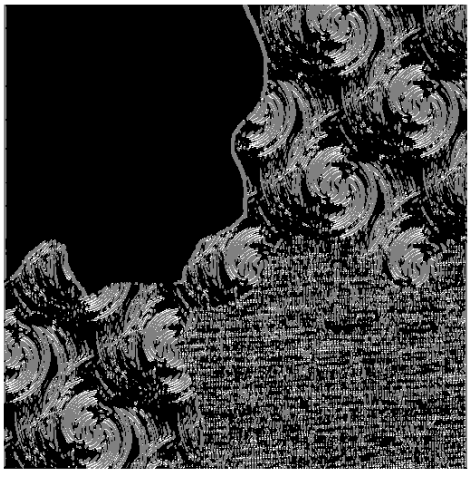
所以用兩種分法先分出兩類，剩下的則是第三類：

第一種分法是取 S5S5 filtered 過後的影像，算一個 31*31 的框格中的

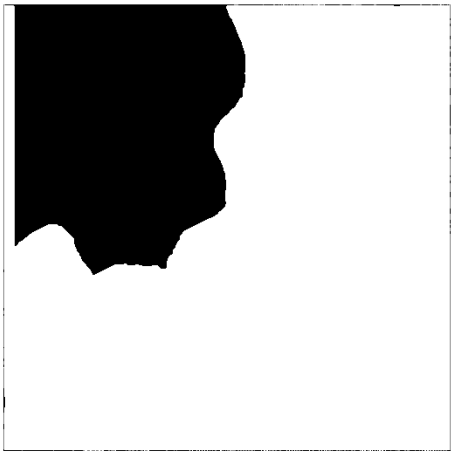
variance，再用 kmeans，分出右下角的 texture：



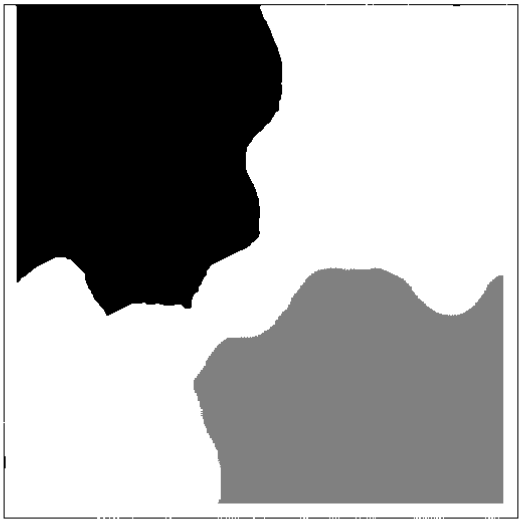
可能因為這個區域的 energy variance 較大，所以能分得出來。
第二種方法是用 9 個 filter 過後的每個 pixel 的 energy，當作該 pixel 的 features，再用算 root mean square 的方式跟取樣的 pixel 做比較，例如第一種 texture 取(100,101)這個點，第二個 texture 取(264,367)這個點，第三個 textrue 取(450,382)這個點，若距離最小則分為同一類。分出來的結果如下：



可以看到最左上角的 texture 可以被完整的分為同一類，再透過相鄰的 pixel 決定該格是不是位於其他區域。這邊是取周圍的 3 個 neighbors，若有大於 3 個 neighbors 為非零的數，則該格填 1。Iteration 10 次後結果如下：(因為只 iteration 一次還有些細小的 pixel 沒被分出來)



最後分類結果如下：



因為 filter 都太大的關係，所以邊緣分的不是很好。

(b)(沒時間做完…)