DIP HW3　R06944023 吳尚真

PROBLEM 1: MORPHOLOGICAL PROCESSING

a) 對I_1做 boundary extraction，輸出 B

1) Your motivation and approach:

根據講義，boundary extraction 的做法為:

$$\beta(\mathrm{F(j,k)}) = F(j,k) - (F(j,k) \ominus H(j,k))$$

所以，步驟如下:

- 求出原圖 F(j,k)對 structuring element H(j,k)的 erosion，將此結果存為 G(j,k)。根據講義，erosion 可以用 Sternberg definition 或是 Serra definition，我這裡用的是 Sternberg definition:

$$\mathrm{G(j,k)} = \cap\cap_{(r,c)\in H}\, T_{r,c}\{F(j,k)\}$$

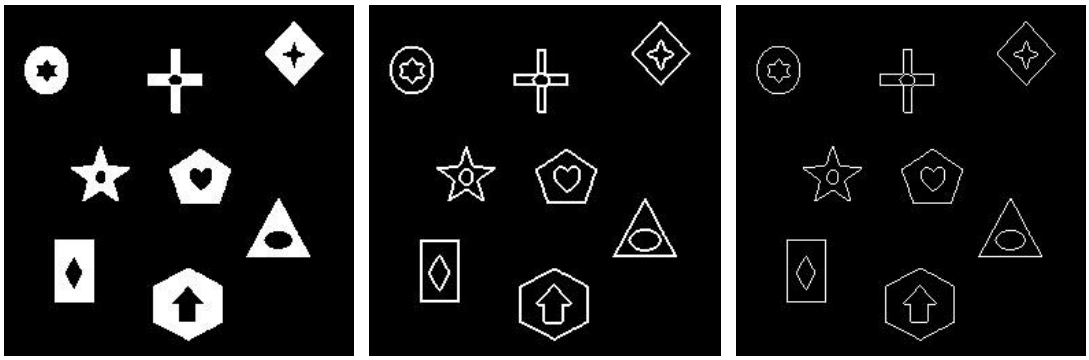
在 structuring element H(r,c)有值的地方，對原圖 F 移動 r,c，得到 translation T，因為 H 有值的地方可能很多，所以會產生許多不同的 T，再將這些 T 取交集。不過，因為在二元圖中，只要有一個是 0，取交集(and)後，(不管其他幾張 T 在同個位置上是否有出現過 1) 都會是 0，所以只需要考慮移動後是否有出現 0，如果有，就把那個位置設為 0，否則為 1。最後再將結果傳回 G(j,k)。

- 如果 G(j,k)=1，則輸出的新圖 B(j,k) = 0。因為 B=F-G，且 F、G 都是 grey-level image，pixel value 只能是 0 或是 255，所以可以直接取反。

- 對於可以自訂的 structuring element H，我分別使用兩種 H，如下所示:

$$H_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2) Original images　　3) Output images (使用 H1)　　使用 H2



4) Discussion of results:

可以發現使用比較大的 H，出來的 boundary 會比較粗，使用比較小的 H，出來的 boundary 會比較細。

b) 對I_1做 connected component labeling，輸出 C，每一個 object 用不同顏色標

註。

1) Your motivation and approach:

根據中文版“數位影像處理”[1]的內容，connected component 的概念如下:

假設 Y 表示包含於集合 A 的一個 component，並假設已知 Y 的一個點 p，則可以使用下面的疊代方式來產生 Y 的所有元素:

$$X_k = (X_{k-1} \oplus B) \cap A \qquad k = 1,2,3,\ldots$$

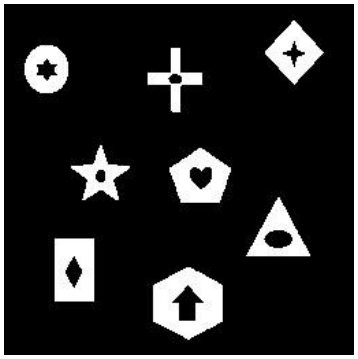
其中，$X_0 = p$，B 為 structuring element。

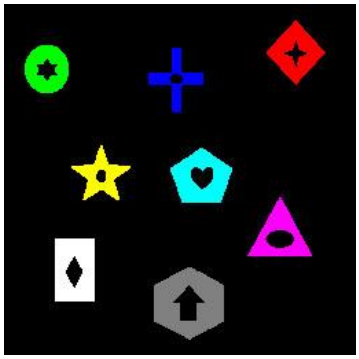
詳細步驟如下:

- 先找出一個在原圖上有值的點，對那個點進行擴張。擴張的範圍會根據我們自訂的 structuring element
- 將擴張後的圖與原圖取交集，只有兩個都為 1 時，才 assign 為 1。
- 接著比較擴張前與擴張後的圖，如果不管再怎麼擴張，結果都長一樣的話，表示已經 converge 了，則停止疊代，表示已經順利找到一個 component。
- 新創一個矩陣叫 label，用來記錄已經找到的 component，對於每一個不同的 component，在找到的位置上分別 assign 不同的值。
- 最後會找到 8 個不同的 component，也就是 label 矩陣會記錄 8 種不同的值。
- traverse 整個 label 矩陣，根據裡面的值，在不同的 component 輸出不同的顏色。 製造不同顏色的方法就是創立一個 256X256X3 的三維矩陣，分別對 [x][y][0], [x][y][1], [x][y][2] assign 不同的值。

其中，我的 structuring element 是選擇講義上提到的 $\mathrm{H} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

2) Original images 3) Output images





c) 對I_1做 thinning 跟 skeletonizing，並分別輸出D_1、D_2

1) Your motivation and approach:

Thinning:

根據老師上課講義，thinning 的作法是將原圖分別經過兩種不同的 3X3 filter 來決定是否要移除/保留該點。不過，如果要依循講義的作法，必須拿課本上列出的所有不同的 filter 來做，但是，講義只有列出一小部分，而我也沒有原文書，

所以改採用在網路上查到的 Zhung and Suen 作法[2]，步驟詳列如下:

- Traverse 原圖的每一點，用一個 3x3 的 mask 罩在這個點上，此點為P_1，如果此點滿足以下條件，就可以把這個點移除

$$\text{mask} = \begin{bmatrix} P_9 & P_2 & P_3 \\ P_8 & P_1 & P_4 \\ P_7 & P_6 & P_5 \end{bmatrix}$$

$$2 \le \mathrm{N}(P_1) \le 6, \qquad \mathrm{S}(P_1) = 1, \qquad P_2 \cdot P_4 \cdot P_6 = 0, \qquad P_4 \cdot P_6 \cdot P_8 = 0$$

其中，$\mathrm{N}(P_1)$代表去數$P_2 - P_9$有幾個 pixel value =1。當$\mathrm{N}(P_1) = 0$，表示P_1 為一孤立點，當$\mathrm{N}(P_1) = 1$，表P_1為端點，對孤立點或端點，不予刪除，否則物件會因細化而消失或導致線條退化。當$\mathrm{N}(P_1) > 6$時，則P_1為一內點，此情況也不予刪除，否則細化結果會產生 hole。

$\mathrm{S}(P_1)$代表去數$P_2 - P_9$這個順序中，0 變 1 的次數有幾次。若$\mathrm{S}(P_1) > 1$，則P_1必為物件中某些部份(components)之一“橋樑”(bridge)，若將之刪除將會造成斷點。

至於$P_2 \cdot P_4 \cdot P_6 = 0,\ P_4 \cdot P_6 \cdot P_8 = 0$是用來刪除右邊及下邊的邊點及左上的轉角點。可以更進一步推導如下:

$$(P_2 \cdot P_4 \cdot P_6 = 0) \cap (P_4 \cdot P_6 \cdot P_8 = 0)$$
$$\Rightarrow (P_2 \cup ((P_4 \cup (P_6)) \cap (P_4 \cup ((P_6 \cup (P_8))$$
$$= \ P_4 \cup P_6 \cup (P_2 \cap P_8)$$

- 類似於上一步，但條件稍微改變如下。

$$2 \le \mathrm{N}(P_1) \le 6, \qquad \mathrm{S}(P_1) = 1, \qquad P_2 \cdot P_4 \cdot P_8 = 0, \qquad P_2 \cdot P_6 \cdot P_8 = 0$$

$P_2 \cdot P_4 \cdot P_8 = 0, P_2 \cdot P_6 \cdot P_8 = 0$用來刪除左邊及上邊的邊點及右下的轉角點。如果滿足所有條件，就可以把這個點移除

- 重複上述步驟，直到無法再移除任何點。

Skeletonizing:

根據[1]，skeletonizing 的公式如下:

$$\mathrm{S(A)} = \bigcup_{k=0}^{k} S_k(A), \qquad S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$$

詳細步驟如下:

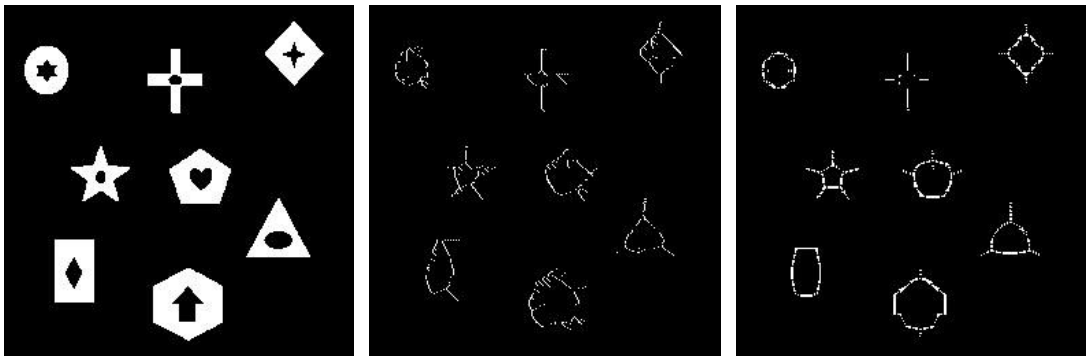
- 根據某個自訂的 structuring element B 對原圖 A 做 erosion。K 表示總共要執行幾次 erosion。換句話說，如果 k=2，$(A \ominus kB) = (A \ominus B) \ominus B$。k 的取法為:

使得 A 被完全侵蝕掉的前一步。實做上，我使用$\mathrm{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$，計算出來的 k=7。也就是說，總共會產生 7 個 erosion 後的結果，要分別存下來。

- 針對上一步得到的 7 個 erosion 後的結果做 opening。Opening 就是先做 erosion 再做 dilation。所以也可以想像成，對 k+1 的 erosion 結果再做 dilation。

- 將第一步的結果減掉第二步的結果，就可以得到$S_0, S_1, \dots, S_k$
- 最後再將$S_0, S_1, \dots, S_k$的結果取交集，即完成。

2) Original images 3) Output images(thinning) skeletonizing



4) Discussion of results:

由上面的結果圖可以觀察到，比起 thinning，skeletonizing 的結果會與其原圖的每一個 boundary 保持一樣的距離(例如右下角的三角形)；而 thinning 則是去找跟中間的洞最靠近的 outer boundary 的最小連接，所以形狀會因為中間的洞的形狀，而有所改變。在實做上，skeletonizing 會花比較多 memory，當 k 越大，就需要存越多圖。但是 skeletonizing 的結果是可以再導回去原圖，而 thinning 不行。

PROBLEM 2: TEXTURE ANALYSIS

a) 對I_2做 Law's method 來得到每一點的 feature vector

1) Your motivation and approach:

根據講義上 Law's method 可分為兩個步驟，詳述如下:

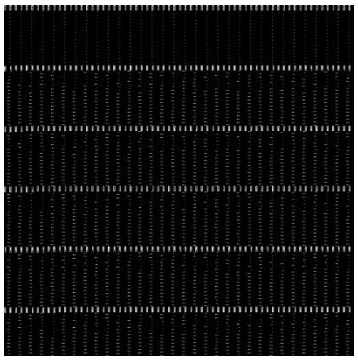
- 對原圖 F 做 convolution 得到 M。Convolution 的 mask 我依照講義，選用 3X3 的 filter，所以總共有 9 種不同的 H。
- 再對 M 做 energy computation，需自訂 window size w。得到 T，這個 T 即可視為 feature vector。

$$T_i(j,k) = \sum\sum_{(m,n)\in w} |M(j+m, k+n)|^2$$

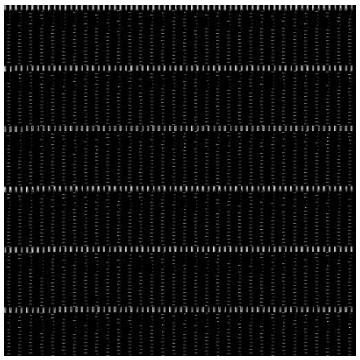
最後再將加總後的 T 除以 window size 的平方(取平均)。根據詢問助教的結果，我將 M[512][512][9]與 T[512][512][9]分別輸出 9 張[512][512]的灰階圖。

3) Output images

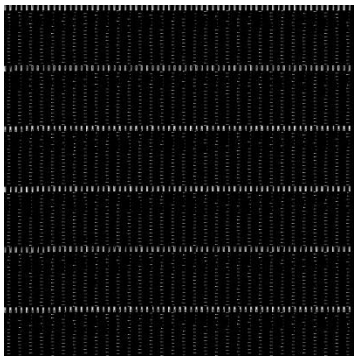
M1 M2 M3



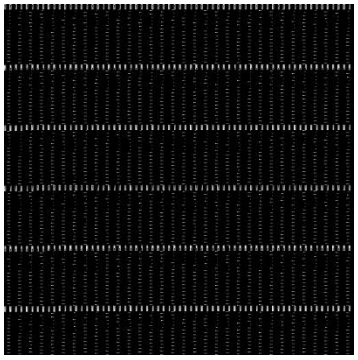
M4



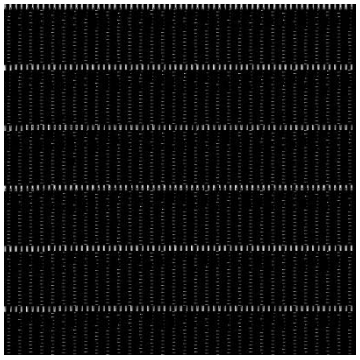
M5



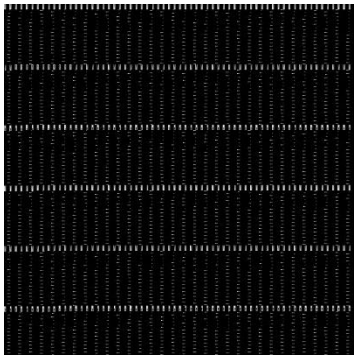
M6



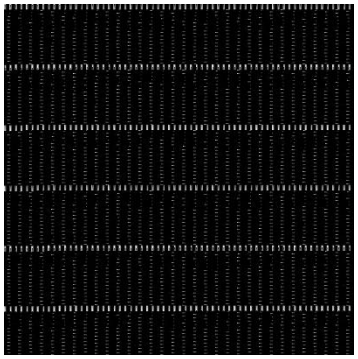
M7



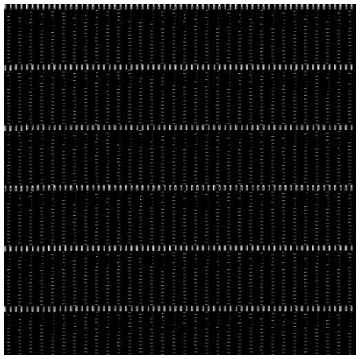
M8



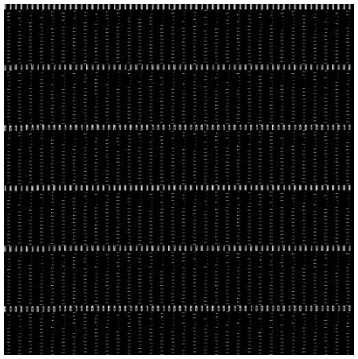
M9



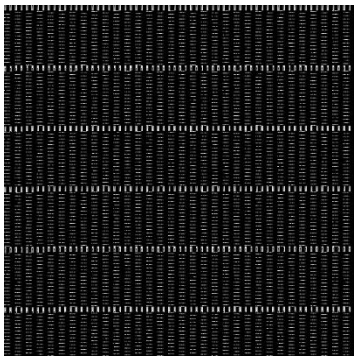
T1



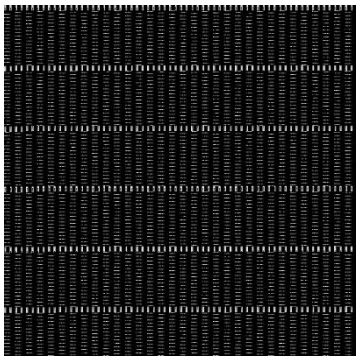
T2



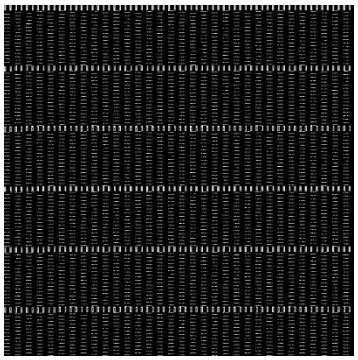
T3



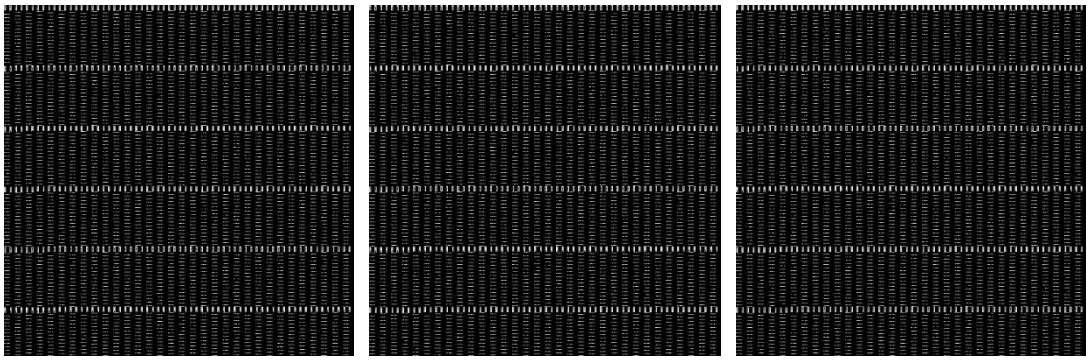
T4



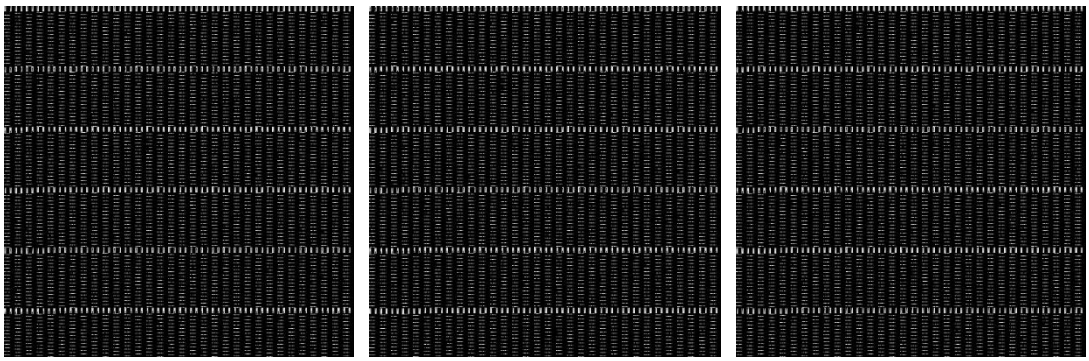
T5



T6



T7 T8 T9



4) Discussion of results:

發現用肉眼看 M 和 T 的灰階圖，每一張都很像，只有些微的差距。

b) 使用 k-means 來區分不同的 texture，並將不同 cluster 標成不同的灰階值

1) Your motivation and approach:

k-means 的步驟如下:

- 因為要分 4 個 clusters，且 T(feature)有 9 個，所以要建一個 4x9 的矩陣，來記錄每一個 cluster 的重心的位置
- 一開始將 T[x][y][feature_map]賦予 centroid[cluster_no][feature_map]。(x, y 可自訂)
- 接著計算 feature T 每一點與每一個 cluster 的 centroid 的歐基里德距離: 將 9 個 map 與 centroid 的差的平方加總起來，再開根號。
- 比較看看這個點與哪一個 cluster 的距離最小，就將此點 assign 給那個 cluster。 (開一個 512X512 的矩陣來記錄每一點分別屬於哪個 cluster)。
- 有了 cluster 後，重新計算 centroid => 將被標誌為同一個 cluster 的 T 加總起來，再取平均。
- 重複做 cluster 與計算 centroid 的步驟。 (iteration 數次)
- 檢查每個點被 assign 的 cluster 在每次更新後，是否有改變。如果更新後還是跟原先一樣，表示 converge，就可以停止 iteration。

2) Original images　　3) Output images(window=13)　　window=23

Window = 31　　Window =51　　Window =63

4) Discussion of results:

觀察到左上角的地方，當 window size 較大時，越能夠將同個區塊歸類到同個 cluster，但是不同區塊的 boundary 會變粗；而當 window size 較小時，左上角的區塊沒辦法完全分到同一個 cluster，但是區塊間的 boundary 看起來較細緻。

但是，當 window size 取較大時，computation effort 也會較大! 需要花較多時間執行!

c) 根據上一題的結果，交換每個區域的 features

1) Your motivation and approach:

步驟如下:

- 先定義 4 個用來存 texture 影像的二維矩陣。 (我是開 50X50)
- 用肉眼對應原圖，看看四個不同的 texture 分別在原圖的哪些位置，把原圖對應到的灰階值，按照位置，複製到 4 種 textures
- traverse 上一題用來記錄 cluster 的 512X512 的圖，若某點 cluster= 0，就把 texture 1 的 pattern 按照位置順序 assign 過去。以此類推: cluster=1 <- texture 2；cluster=2 <- texture 3；cluster=3 <- texture 0。
- 最後再將 Image output 出來即可

下面使用 window size = 51 做 law's method 的結果

(我將 b 小題的結果圖一起擺上，以方便對照)

左:原圖， 中: b 小題的結果圖 右:本小題結果

Original images　　cluster 的灰階圖　　交換 texture 的圖

4) Discussion of results:

比較 cluster 的灰階圖 與 交換 texture 後的結果圖，可以發現，在灰階圖上同一灰階值的地方，會被成功 assign 程某一種 texture。如果在前一步 (分 cluster 的地方) 做得很精準的話，交換 texture 後的結果圖就能越完整。

Reference:

[1] 中文版“數位影像處理” Gonzalez, Woods: Digital Image Processing, Prentice Hall, 2007

[2] T.Y. Zhang and C.Y. Suen, “A fast parallel algorithm for thinning digital pattern,” Communications of the ACM, Vol. 27, No. 3, pp. 236-239,1984