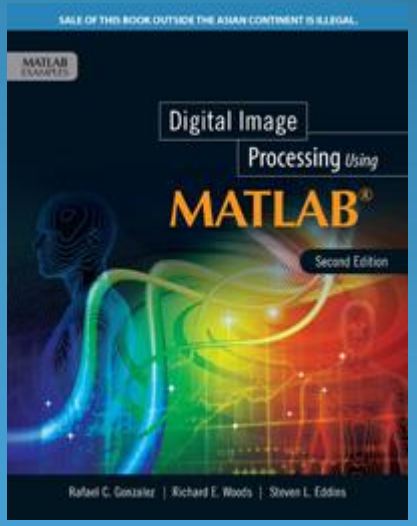
Medical Imaging

Medical Image Processing III

Yi-Li Tseng

FJU-EE

Fall 2017



References:

1. Digital Image Processing Using MATLAB, 2nd ed.: Rafael C. Gonzalez

第九章

形態學影像處理

Outline

- 預備知識
- 膨脹和侵蝕
- 結合膨脹和侵蝕
- 標記連通成分
- 形態學重建
- 灰階形態學

預備知識

預備知識

- 集合理論的基本概念，利用圖9.1說明

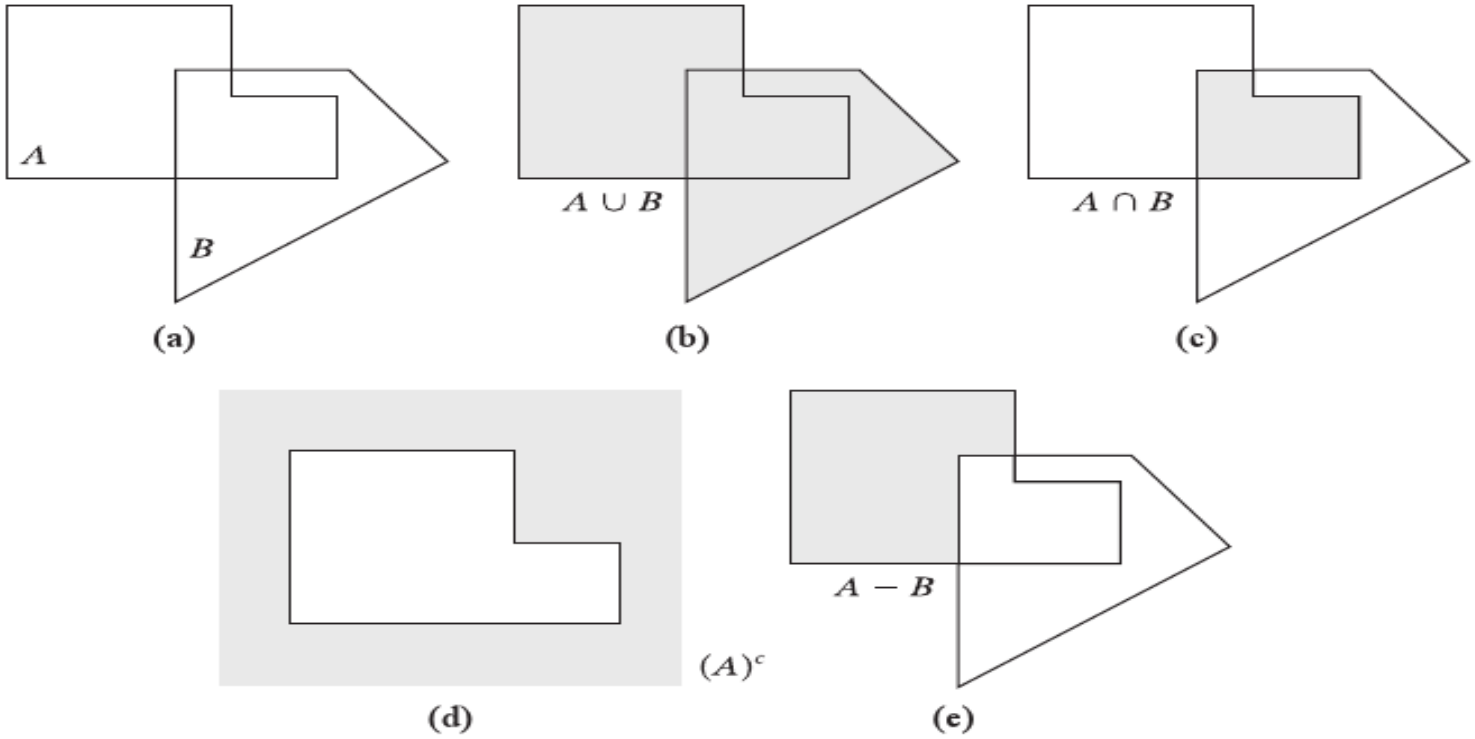


圖9.1 (a) 兩個集合A與B；(b) A 與B的聯集；(c) A 與B 的交集；(d) A的補集；(e) A 與B 的差集

預備知識

- 集合定義，利用圖9.2說明

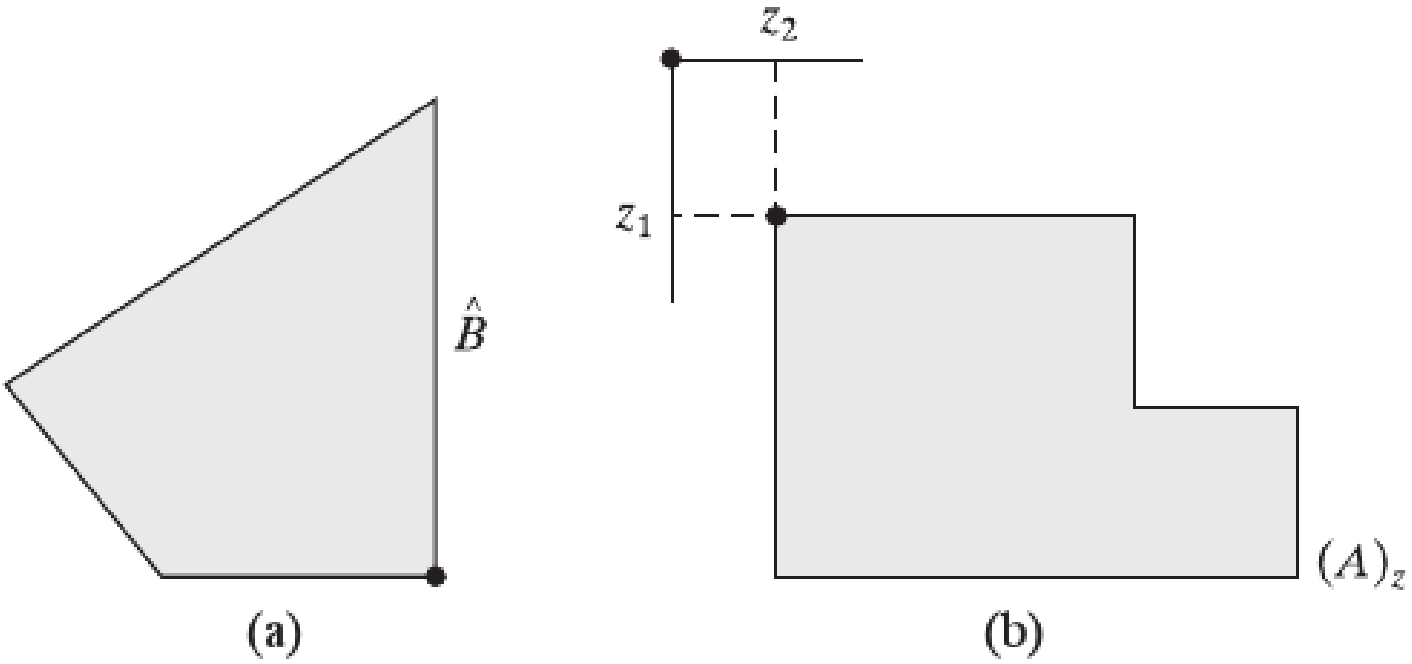


圖9.2 (a) B 的反射；(b)A 對z 的平移，A 和B 是來自圖9.1 的集合，黑色點表示其原點

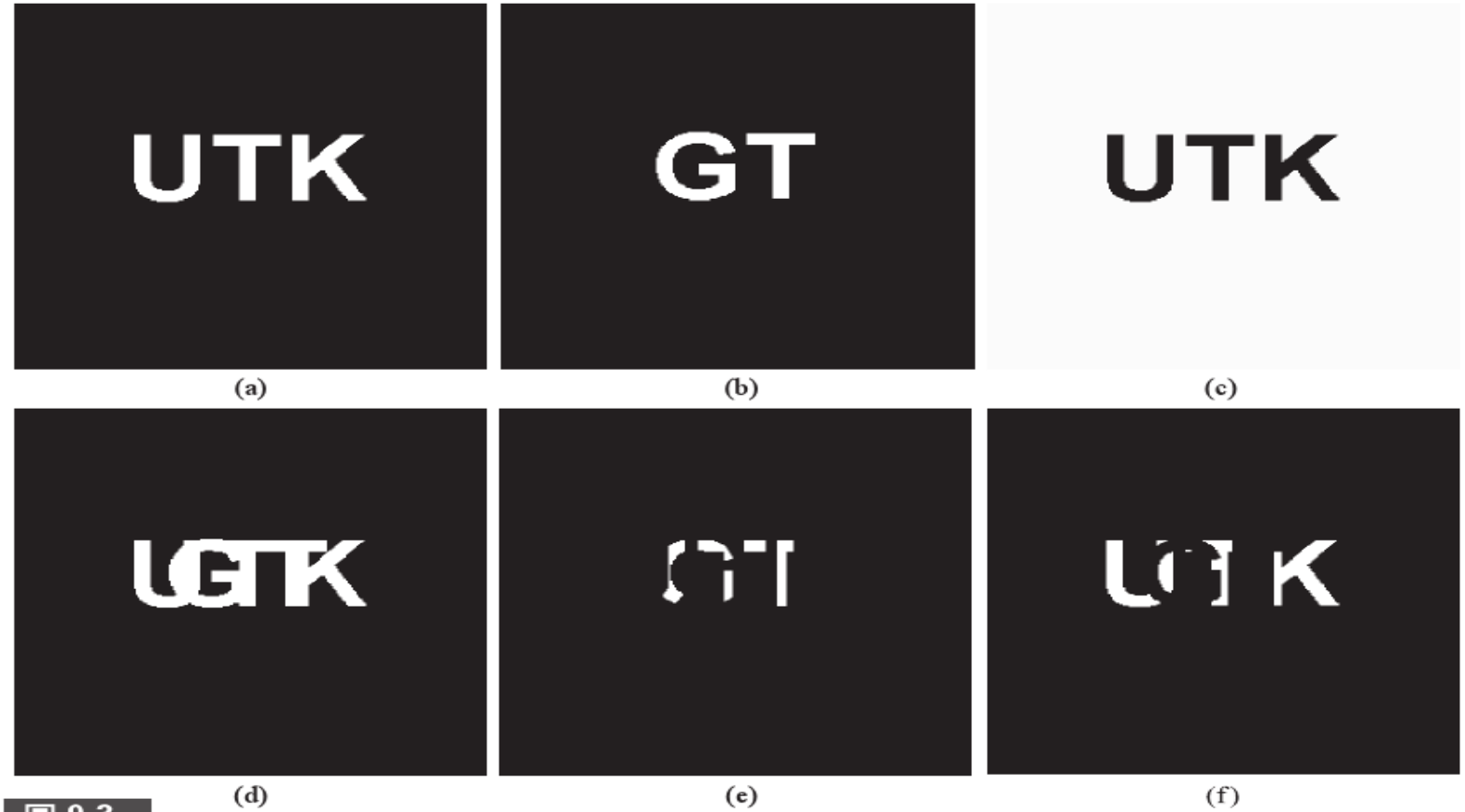
預備知識

- 使用MATLAB中的邏輯表示法來完成二元影像中的集合運算，如表9.1所示

集合運算	二元影像的 MATLAB表示法	名稱
$A \cap B$	A & B	AND
$A \cup B$	A | B	OR
A^c	~A	NOT
$A - B$	A & ~B	DIFFERENCE

預備知識

- 範例圖9.3顯示出運用幾個邏輯運算子到含有文字之二元影像上的結果



9.3

(a) 二元影像 A；(b) 二元影像 B；(c) A 的反相；(d) A 與 B 的反相之聯集；(e) A 與 B 的交集；(f) A 與 B 的反相之差集

膨脹和侵蝕

膨脹和侵蝕

- 圖9.4 說明膨脹如何運作

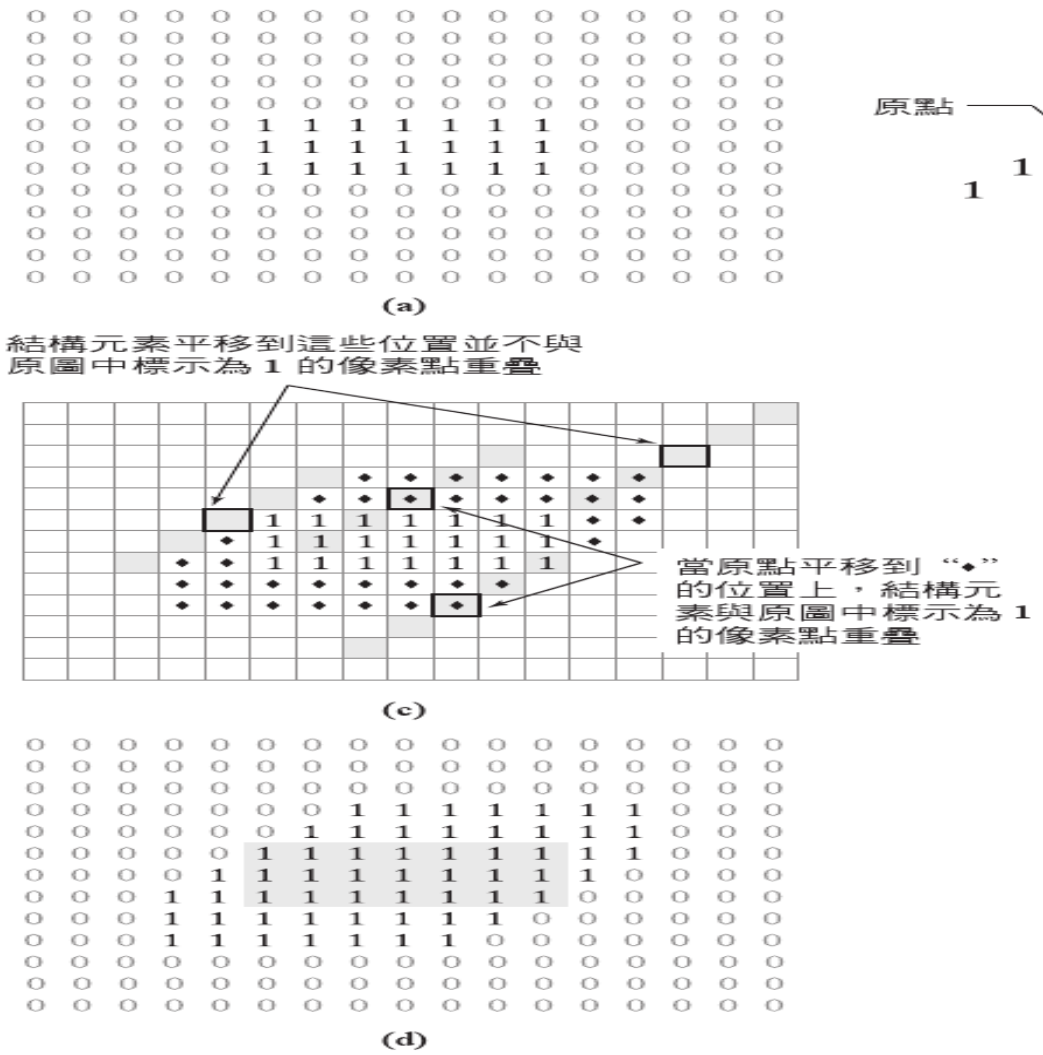


圖 9.4 膨脹的圖示說明 (a) 含有矩形物件的原圖；(b) 結構元素為五個像素長的對角線，其原點或中心點以黑框表示；(c) 結構元素平移到影像上的幾個位置；(d) 輸出影像上的陰影區域表示在原圖中標示為 1 的位置

膨脹和侵蝕

- 圖9.5顯示一個非對稱的結構元素及其反射

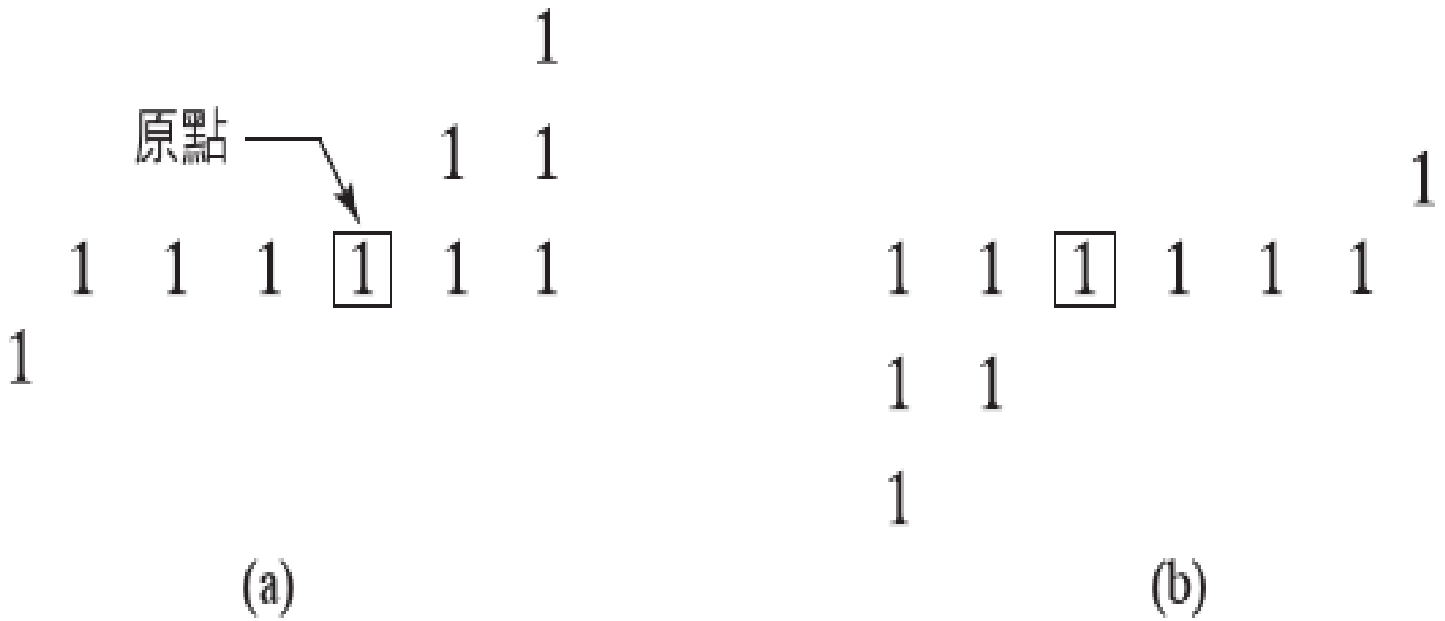


圖9.5 (a) 非對稱的結構元素；(b) 結構元素對其原點進行反射

膨脹和侵蝕

- 範例9.1膨脹的一個應用

圖 9.6(a) 顯示一個含有文字與破碎字元的二元影像。我們將使用函數 imdilate 根據下列結構元素將影像進行膨脹運算：

$$\begin{matrix} 0 & 1 & 0 \\ 1 & \boxed{1} & 1 \\ 0 & 1 & 0 \end{matrix}$$

以下指令將由檔案中讀取影像、組成結構元素矩陣、完成膨脹運算與顯示運算結果。

```
>> A = imread('broken_text.tif');
>> B = [0 1 0; 1 1 1; 0 1 0];
>> D = imdilate(A, B);
>> imshow(D)
```

圖 9.6(b) 顯示運算後所得的影像。

膨脹和侵蝕

- 範例9.1膨脹的一個應用

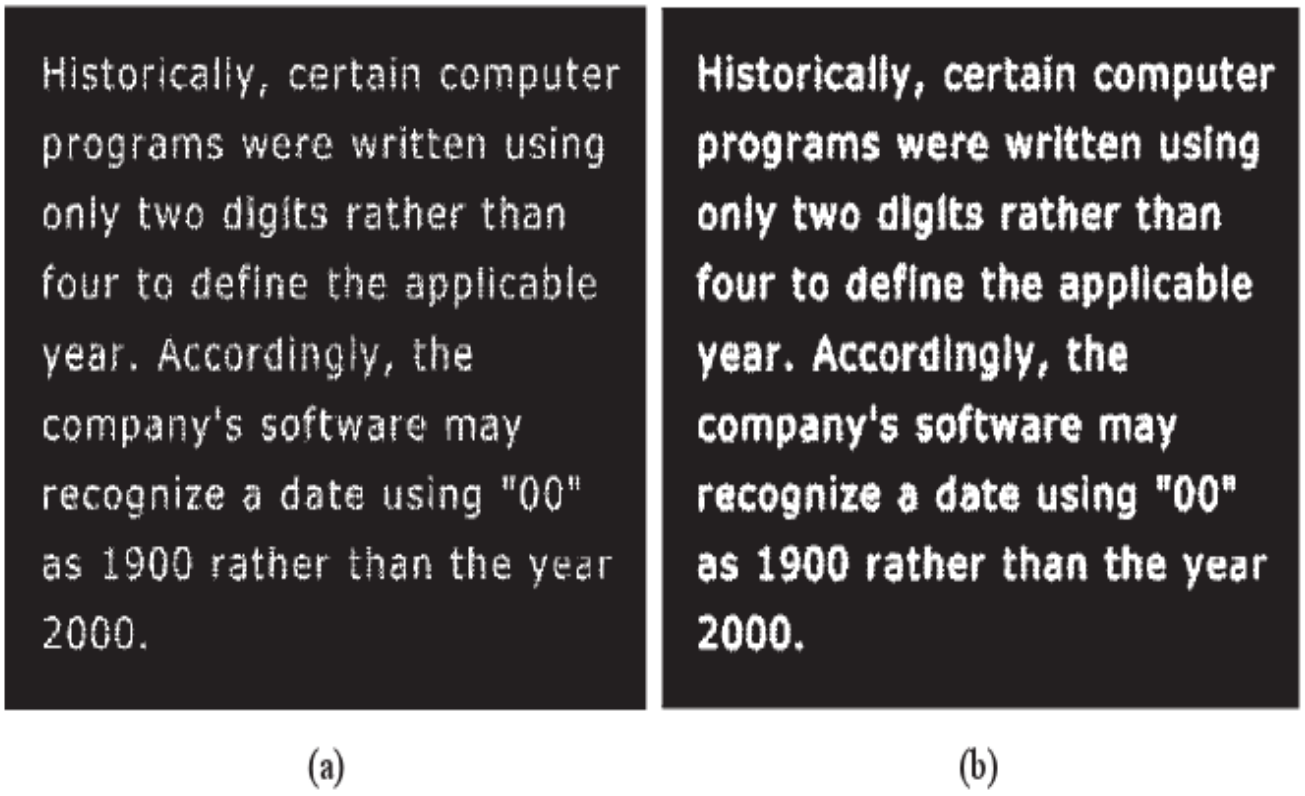


圖 9.6

膨脹的範例 (a) 包含破碎文字的影像；(b) 膨脹後的影像

膨脹和侵蝕

- 表9.2 strel函數的各種不同語法形式

語法形式	描述
`se=strel('diamond',R)`	產生一個平坦的菱形結構元素，參數 R 表示由結構元素的原點到菱形極端點的距離。
`se=strel('disk',R)`	產生一個半徑為 R 的圓盤型結構元素。（額外的參數可用來設定圓盤，詳細資訊請參看 strel 的參考頁面。）
`se=strel('line',LEN,DEG)`	產生一個平坦的線性結構元素，其中參數 LEN 表示長度；參數 DEG 表示與水平軸逆時針方向測量的角度（以度數表示）。
`se=strel('octagon',R)`	產生一個平坦的八角形結構元素，其中參數 R 表示由結構元素的原點到八角型邊緣的距離，此距離是沿著水平與垂直軸測量，且 R 必須是 3 的倍數。
`se=strel('pair',OFFSET)`	產生一個具有兩個構成分子的結構元素，一個構成分子表示原點的位置，另一個構成分子的位置由向量 OFFSET 指定，此向量必須是含有兩個元素的整數向量。
`se=strel('periodicline',P,V)`	產生一個平坦、含有 2*P+1 個構成分子的結構元素；V 是一個具有兩個元素的向量，其元素為整數值，用以表示列與行的偏移量。一個結構元素的構成分子用來標明原點的位置，其它的構成分子則標示 1*V, -1*V, 2*V, -2*V, ..., P*V 和 -P*V 的位置。
`se=strel('rectangle',MN)`	產生一個平坦的矩行結構元素，其中 MN 表示此矩形的大小，MN 必須是一個由非負整數所構成之兩個元素的向量，MN 的第一個元素表示結構元素的列數，第二個元素則表示結構元素的行數。
`se=strel('square',W)`	產生一個長寬為 W 個像素的正方形結構元素，W 必須是一個非負整數。
`se=strel('arbitrary',NHOOD)` `se=strel(NHOOD)`	產生一個任意形狀的結構元素，NHOOD 是一個由 0 與 1 所組成的矩陣，用來表示結構元素的形狀。第二個較簡單的語法形式顯示執行相同的運算。

表 9.2 strel 函數的各種不同語法形式。**平坦 (flat)** 這個字表示二維的結構元素（也就是元素的高度為 0）這只對灰階影像的膨脹與侵蝕有意義。相關內容將在 **9.6.1** 節討論

膨脹和侵蝕

- 圖9.7侵蝕的圖示說明

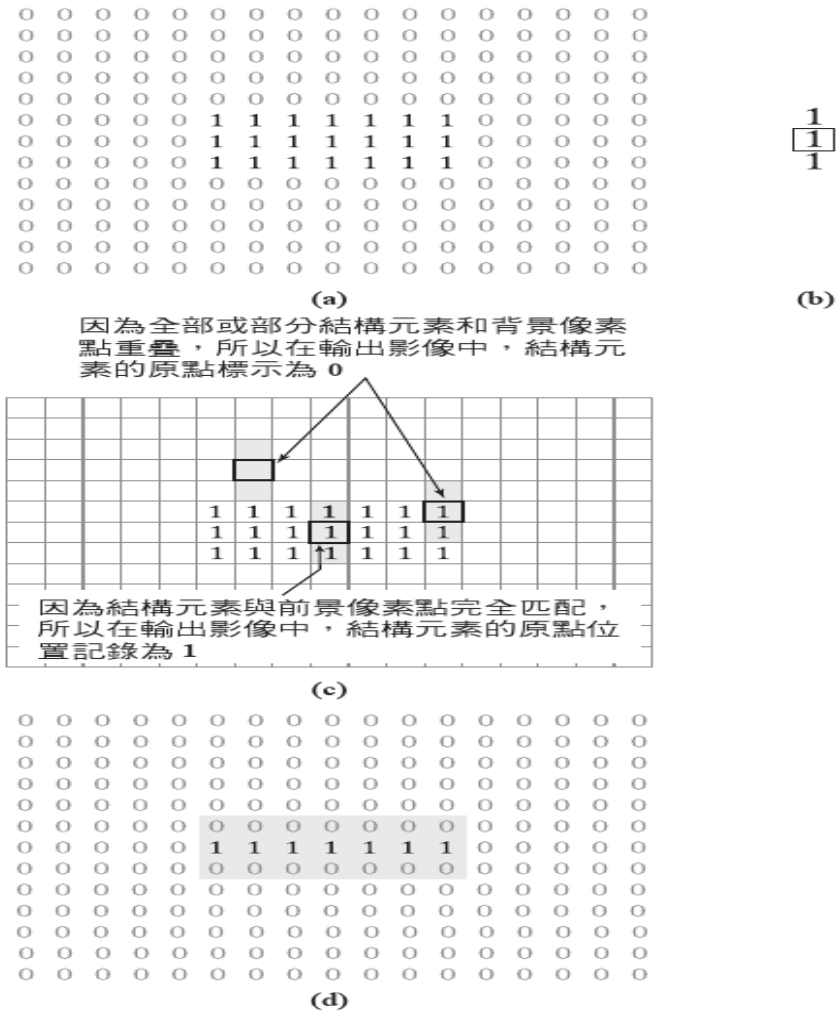


圖 9.7

侵蝕的圖示說明 (a) 含有矩形物件的原始圖像 (b) 由三個像素排列成直線的結構元素，其原點以黑框標示 (c) 結構元素平移到影像中的幾個位置 (d) 輸出影像。其中陰影區域表示在原圖中標示為 1 的像素點位置

膨脹和侵蝕

- 範例9.3一個侵蝕的圖例說明

侵蝕運算可以使用工具箱函數 imerode 完成，其語法跟在 9.2.1 節所討論的函數 imdilate 相同。假如我們希望在圖 9.8(a) 的二元影像中保持其它影像結構且將其中的細線移除，我們可以選擇滿足以下條件的結構元素：小到可以放入中心矩形、大於邊界導線但不會大到不能放入細線中。使用以下指令：

```
>> A = imread('wirebond_mask.tif');
>> se = strel('disk', 10);
>> E10 = imerode(A, se);
>> imshow(E10)
```

膨脹和侵蝕

- 範例9.3一個侵蝕的圖例說明

如圖 9.8(b) 所示，這些指令成功移除遮罩中的細線，圖 9.8(c) 顯示如果我們選擇的結構元素太小所造成的結果：

```
>> se = strel('disk', 5);
>> E5 = imerode(A, se);
>> imshow(E5)
```

在這個例子中，某些導線沒有被移除。圖 9.8(d) 顯示如果我們選擇的結構元素太大所造成的結果：

```
>> E20 = imerode(A, strel('disk', 20));
>> imshow(E20)
```

在此情況下，導線被移除了，但是邊界導線也被移除了。

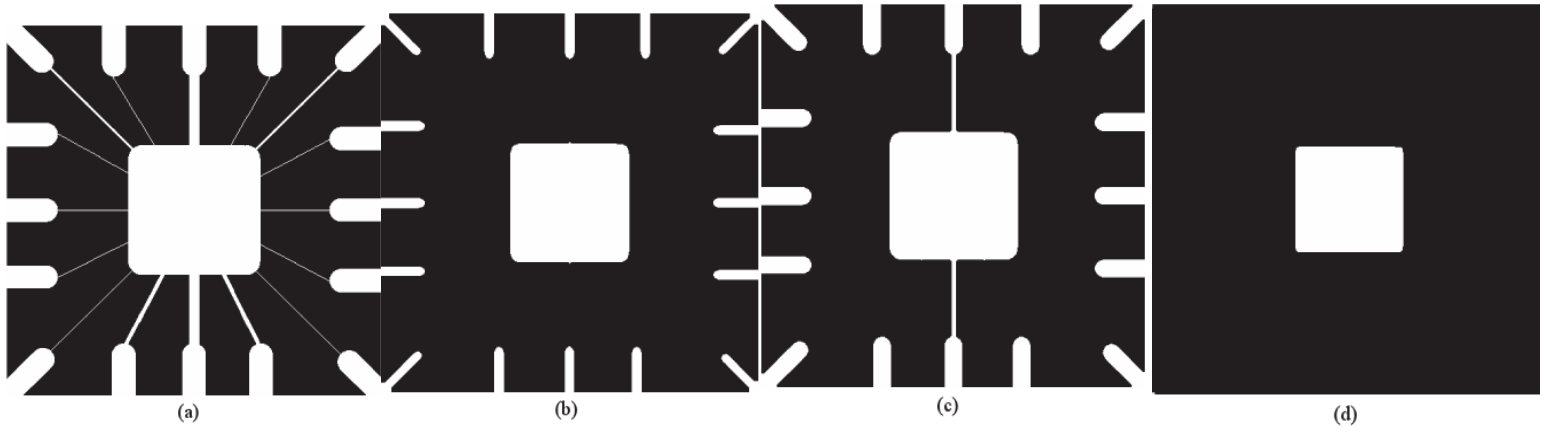


圖 9.8 一個侵蝕的圖例說明 (a) 大小為 486 × 486 的原圖 (b) 以半徑為 10 的圓盤進行侵蝕運算 (c) 以半徑為 5 的圓盤進行侵蝕運算 (d) 以半徑為 20 的圓盤進行侵蝕運算

結合膨脹和侵蝕

結合膨脹和侵蝕

- 斷開與閉合可以視為經結構元素平移後的聯集，利用圖9.9說明

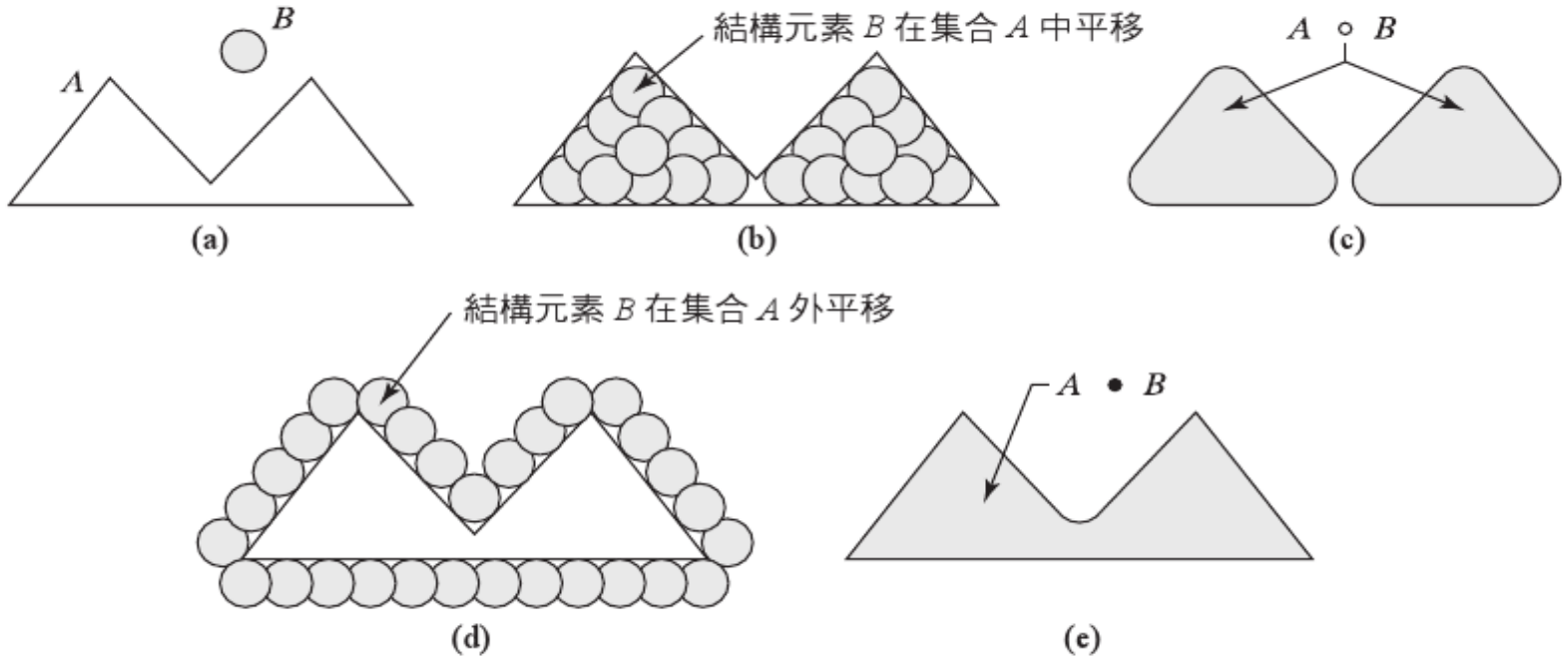


圖 9.9

斷開與閉合可以視為經結構元素平移後的聯集 **(a)** 集合 A 和結構元素 B；**(b)** 結構元素平移後可完全匹配於集合 A 內；**(c)** 完整的斷開結果（陰影部分）；**(d)** 結構元素 B 在集合 A 的邊界上平移；**(e)** 完整的閉合結果（陰影部分）

結合膨脹和侵蝕

- 範例9.4說明函數imopen與imclose的使用方法

這個範例說明函數 imopen 與 imclose 的使用方法，圖 9.10(a) 中影像的檔案名稱為 shapes.tif，影像中的物件有幾個特別設計來說明斷開與閉合運算效果的特點，如細小的突起、狹長的橋接部份、獨立的小孔、細小的獨立物件及不平整的邊緣。以下指令以大小為 20 × 20 的結構元素對影像中的物件進行斷開運算：

```
>> f = imread('shapes.tif');
>> se = strel('square', 20);
>> fo = imopen(f, se);
>> imshow(fo)
```

圖 9.10(b) 顯示運算結果，注意影像中細小的突起和物件邊界上朝外的不規則處都被移除掉了，同時狹長的橋接部份和細小的獨立物件也被移除了。使用以下指令

```
>> fc = imclose(f, se);
>> imshow(fc)
```

結合膨脹和侵蝕

- 範例9.4說明函數imopen與imclose的使用方法

產生圖 9.10(c) 顯示的運算結果。在此影像中，狹長的缺口、物件邊界上朝內的不規則處與小孔都被移除了，在斷開運算後進行閉合運算對影像中物件具有平滑化的效果：

```
>> foc = imclose(fo, se);
>> imshow(foc)
```

圖 9.10(d) 顯示運算後影像中平滑的物件。

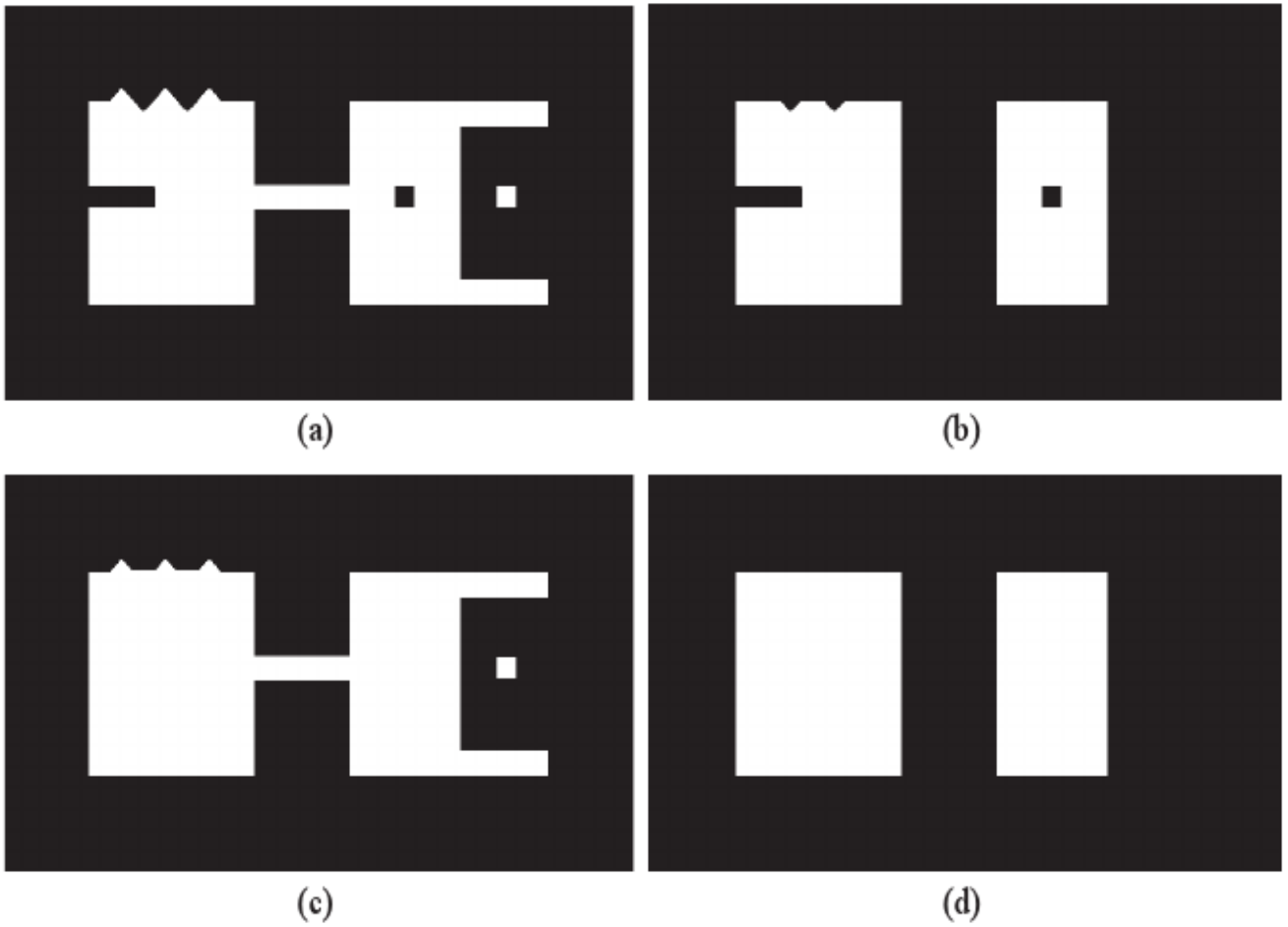
一連串斷開或閉合運算可以用於去除雜訊，以圖 9.11(a) 為例，對一個有雜訊的指紋影像使用以下指令

```
>> f = imread('Fig911(a)

.tif');
>> se = strel('square', 3);
>> fo = imopen(f, se);
>> imshow(fo)
```

結合膨脹和侵蝕

- 範例9.4說明函數imopen與imclose的使用方法



9.10

斷開和閉合運算的圖示說明

(a) 原始影像

(b) 斷開運算的結果

(c) 閉合運算的結果

(d) (b) 圖再經過閉合運算的結果

結合膨脹和侵蝕

- 範例9.4說明函數imopen與imclose的使用方法

產生圖 9.11(b) 中的影像。透過對影像進行斷開運算將影像上的雜訊點移除，但此運算在指紋脊上造成許多缺口。而這些缺口可以在斷開運算後進行閉合運算加以填補：

```
>> foc = imclose(fo,se);
>> imshow(foc)
```

圖 9.11(c) 顯示最後的處理結果，影像中絕大多數的雜訊都被移除掉了（付出在指紋脊上產生一些小缺口的代價）。

(a) (b) (c)

圖9.11(a) 有雜訊的指紋影像；(b) 經斷開運算後的結果；(c) 經斷開運算再進行閉合運算的結果

結合膨脹和侵蝕

- 交離轉換(hit-or-miss transformation)

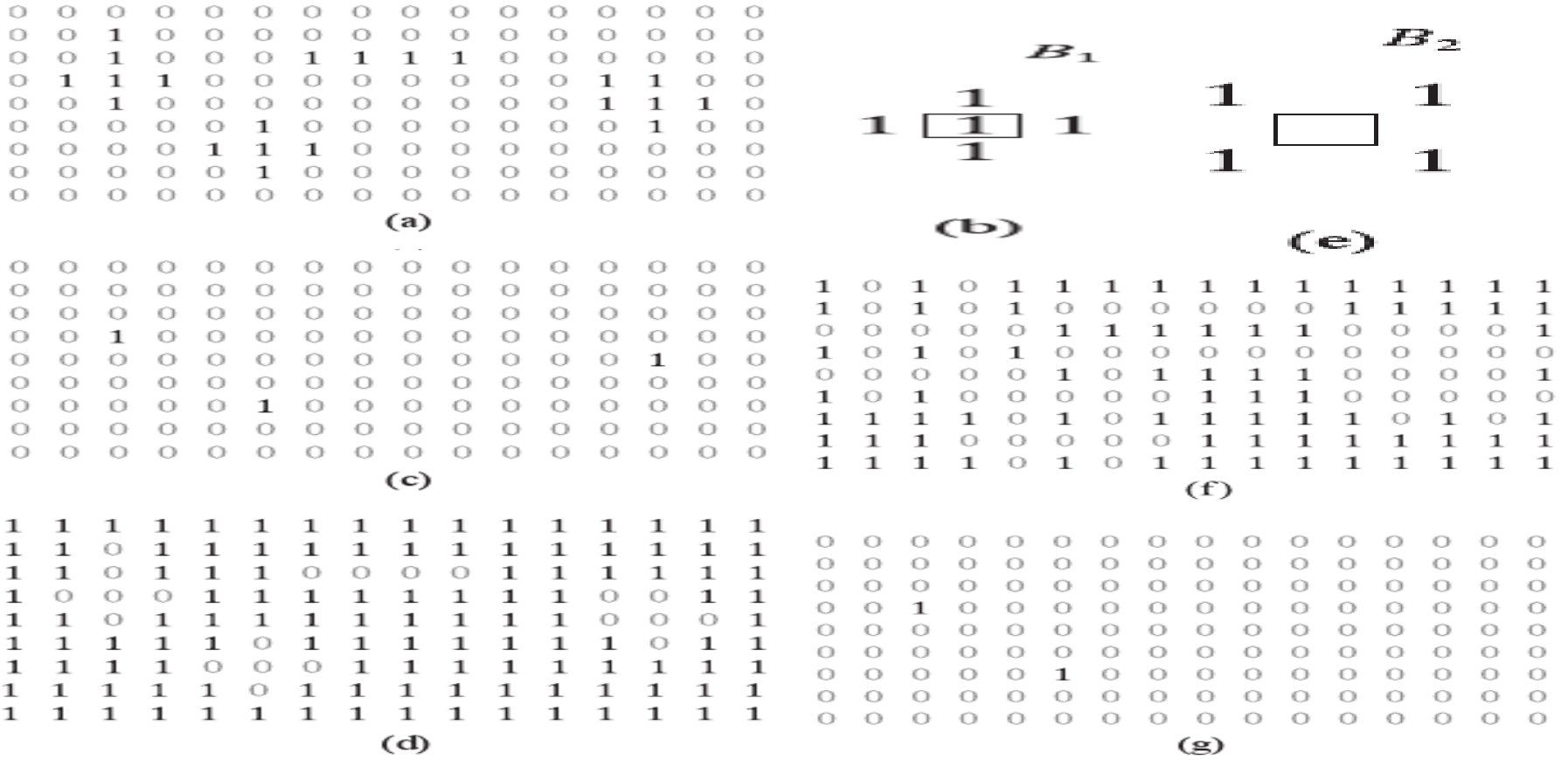


圖9.12　(a) 原始影像A　(b) 結構元素B_1 (c) 使用結構元素B_1對影像A進行侵蝕運算　(d) 原始影像A的補集A^C (e) 結構元素B_2　(f) 使用結構元素B_2對影像A^C進行侵蝕運算　(g) 輸出影像

結合膨脹和侵蝕

- 範例9.5使用函數bwhitmiss

想想如何使用交離轉換來找出位於影像中左上角正方形物件的像素位置。圖 9.13(a) 顯示一張含有各種不同大小正方形的影像，我們希望找到在東邊與南邊有相鄰點的前景像素點位置（這些稱為“命中”），同時該像素點在東北、北、西北、西或西南方向上沒有相鄰點（這些稱為“錯失”）。要完成此要求，需要以下兩個結構元素：

```
>> B1 = strel([0 0 0; 0 1 1; 0 1 0]);
>> B2 = strel([1 1 1; 1 0 0; 1 0 0]);
```

注意到這兩個結構元素在東南方向上都沒有相鄰點，這些相鄰點稱為**不理會 (don't care)** 像素，我們使用函數 bwhitmiss 來計算此轉換，其中 A 是圖 9.13(a) 中顯示的輸入影像：

```
>> C = bwhitmiss(A, B1, B2);
>> imshow(C)
```

在圖 9.13(b) 中的每一個像素點都表示在圖 9.13(a) 中每一個物件左上角像素點的位置，在圖 9.13(b) 中的像素點為了能清楚顯示所以被放大了。

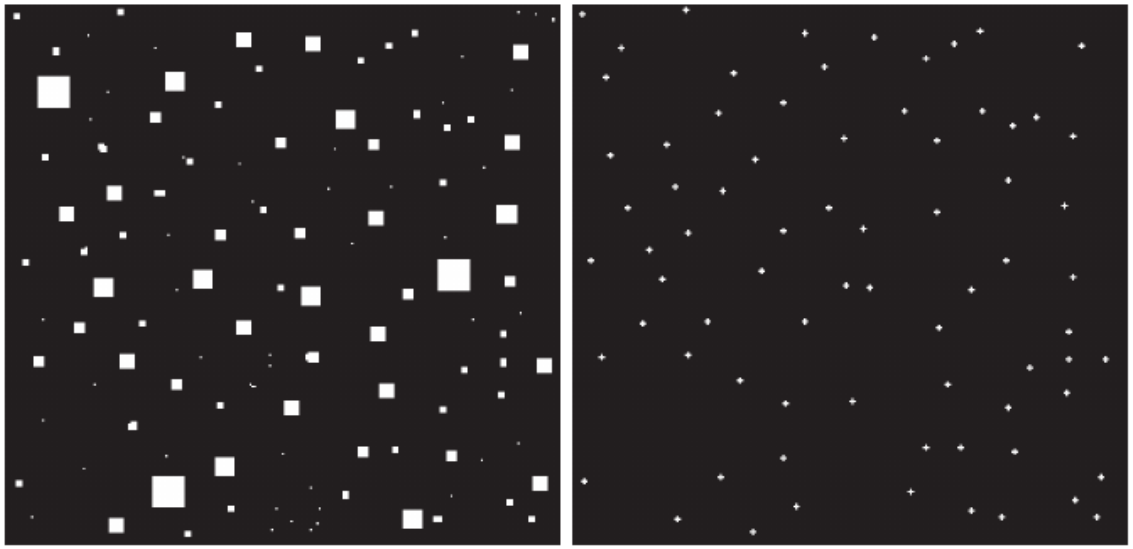
函數 bwhitmiss 的另一個語法為將結構元素 B1 和 B2 結合成一個**區間矩陣 (interval matrix)**，在此區間矩陣中的元素，如果在結構元素 B1 中標示為 1 則標示為 1；如果在結構元素 B2 中標示為 1 則標示為 −1；而**不理會**像素點則標示為 0，對應上述結構元素 B1 與 B2 的區間矩陣如下：

結合膨脹和侵蝕

- 範例9.5使用 函數bwhitmiss

```
>> interval = [-1 -1 -1; -1 1 1; -1 1 0]
interval =
     -1     -1     -1
     -1      1      1
     -1      1      0
```

使用此區間矩陣，輸出影像 C 可以使用語法：`bwhitmiss(A, interval)` 來計算。



(a) (b)

9.13

(a) 原始影像

(b) 進行交離轉換後的結果（為方便觀察將圖上的點放大顯示）

結合膨脹和侵蝕

- 圖9.14 展示函數endpoints的用法

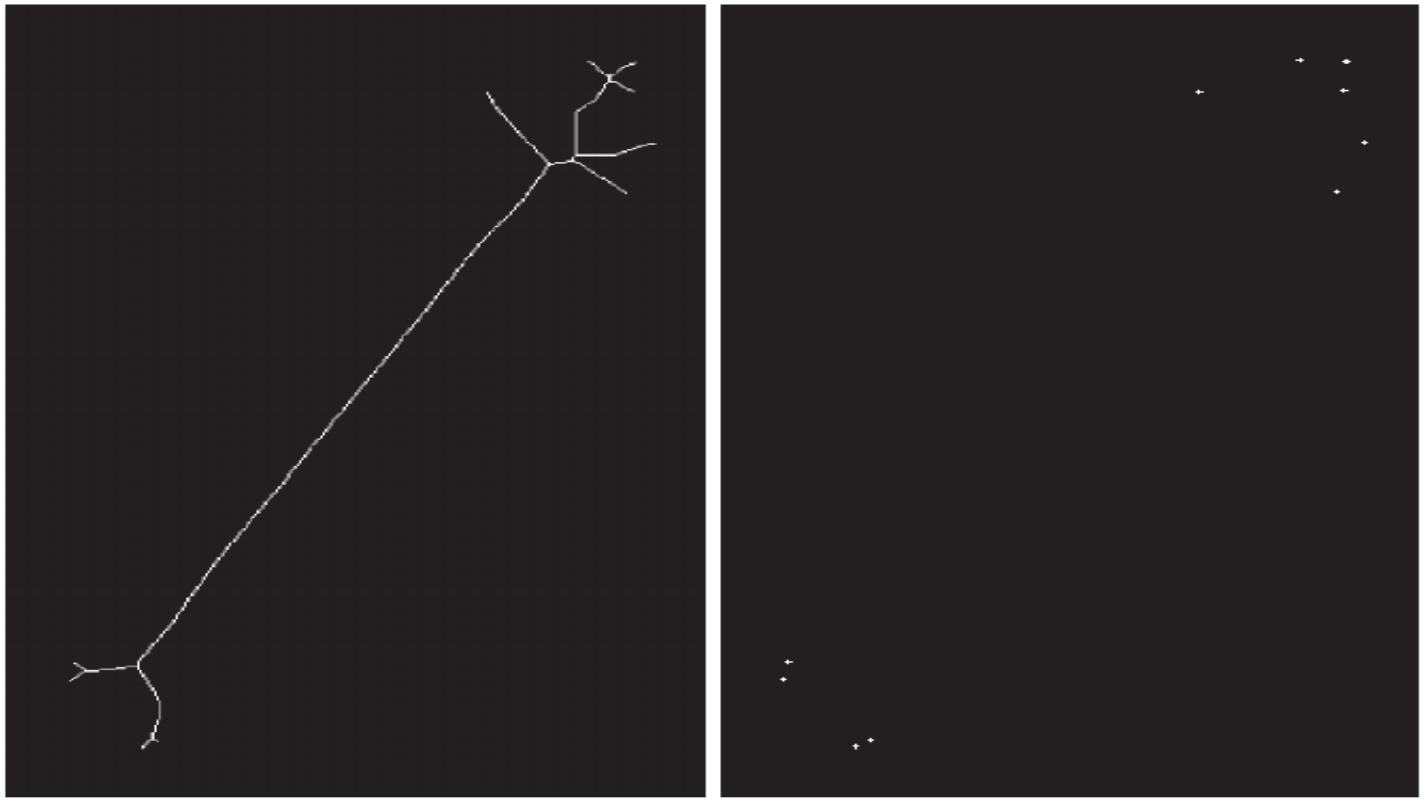


圖9.12 (a) 形態學骨架的影像；(b) 函數endpoints的輸出結果。在(b) 中的像素已放大以便清楚觀察

結合膨脹和侵蝕

- 表9.3函數bwmorph支援的運算

運算	描述
bothat	使用大小為 3 × 3 的結構元素“底帽”進行運算，如欲使用其它結構元素，則使用函數 imbothat（詳見 9.6.2 節）。
bridge	連結被一像素寬之缺口分開的像素。
clean	移除孤立的前景像素點。
close	使用大小為 3 × 3 的結構元素進行閉合運算，如欲使用其它結構元素，則使用函數 imclose。
diag	沿著對角連結的前景像素點周圍進行填補。
dilate	使用大小為 3 × 3 的結構元素進行膨脹運算，如欲使用其它結構元素，則使用函數 imdilate。
erode	使用大小為 3 × 3 的結構元素進行侵蝕運算，如欲使用其它結構元素，則使用函數 imerode。
fill	填補一個像素大小的破洞（前景像素點圍住背景像素點），較大的破洞則使用函數 imfill 填補（詳見 10.1.2 節）。
hbreak	移除 H- 連通的前景像素點。

結合膨脹和侵蝕

- 表9.3函數bwmorph支援的運算

運算	描述
majority	如果至少有 5 個像素點在 $N_8(p)$（詳見 9.4 節）中，則設定像素點 p 為前景像素點，反之則設定 p 為背景像素點。
open	使用大小為 3 × 3 的結構元素進行斷開運算，如欲使用其它結構元素，則使用函數 imopen。
remove	移除“內部”像素點（沒有背景相鄰的前景像素點）。
shrink	將沒有破洞的物件收縮成點狀；而有破洞的物件則收縮成環狀。
skel	對影像進行骨架化運算。
spur	移除影像中突刺的像素點。
thicken	將物件加厚，但不會將斷開的前景像素點連結起來。
thin	將沒有破洞的物件細化成最小的連結筆劃；而有破洞的物件則細化成環狀。
tophat	使用大小為 3 × 3 的結構元素“頂帽”進行運算，如欲使用其它結構元素，則使用函數 imtophat（詳見 9.6.2 節）。

結合膨脹和侵蝕

- 細線化運算(Thinning) ：指將一影像中的二元化物件或形狀簡化，使其剩下單一像素寬，如圖9.15

圖9.15：圖9.11(c) 中的指紋影像經過一次細線化；(b) 經過兩次細線化的影像；(c) 細線化直到穩定的影像

結合膨脹和侵蝕

- 骨架化運算(Skeletonization)：將二元影像物件簡化成一組細線條的另一種方法，且細線條中保留了關於原始物件形狀的重要資訊
  - 寄生成分(parasitic components)：細線化運算常會產生短的突刺(spurs)
  - 剪除運算(pruning)：清除(或移除)這些突刺的程序

結合膨脹和侵蝕

- 利用函數進行骨架化運算，如圖9.16

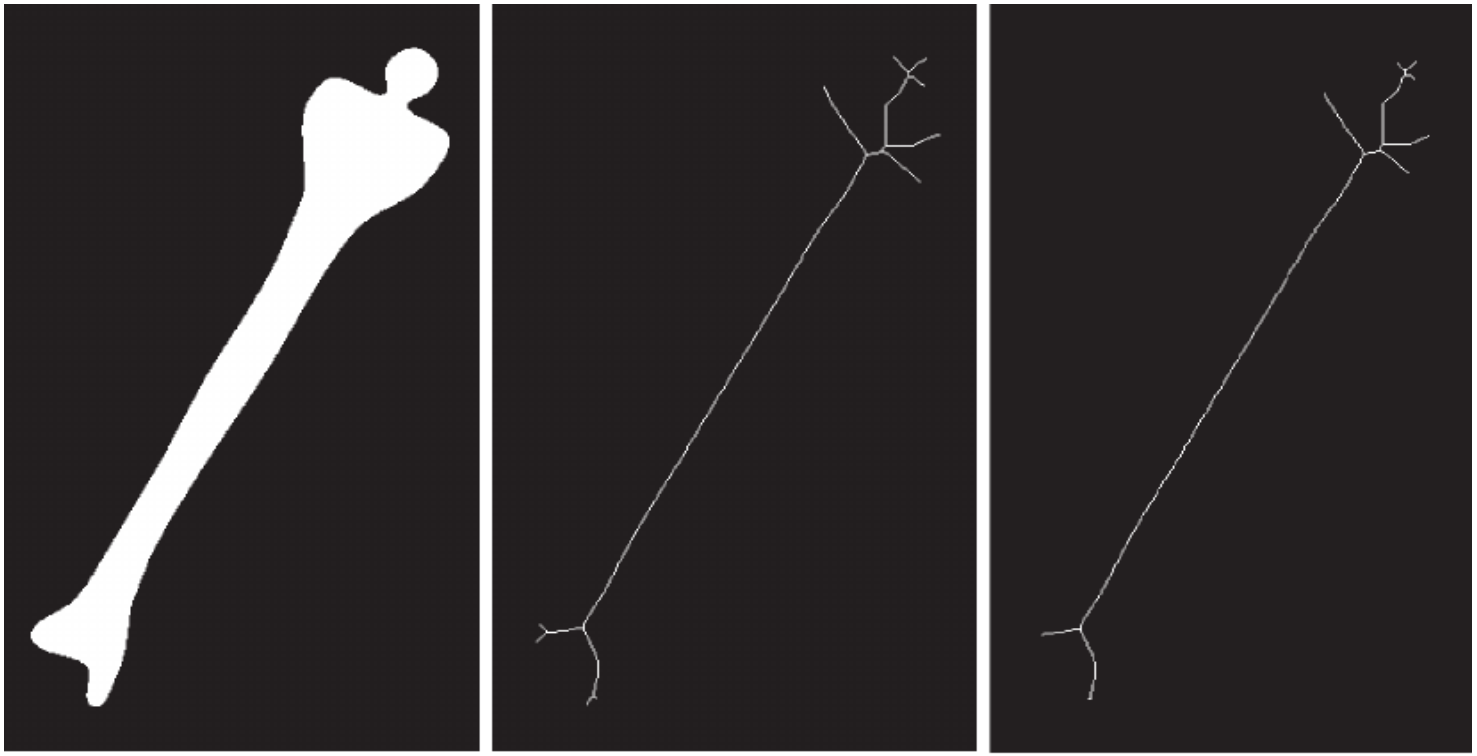
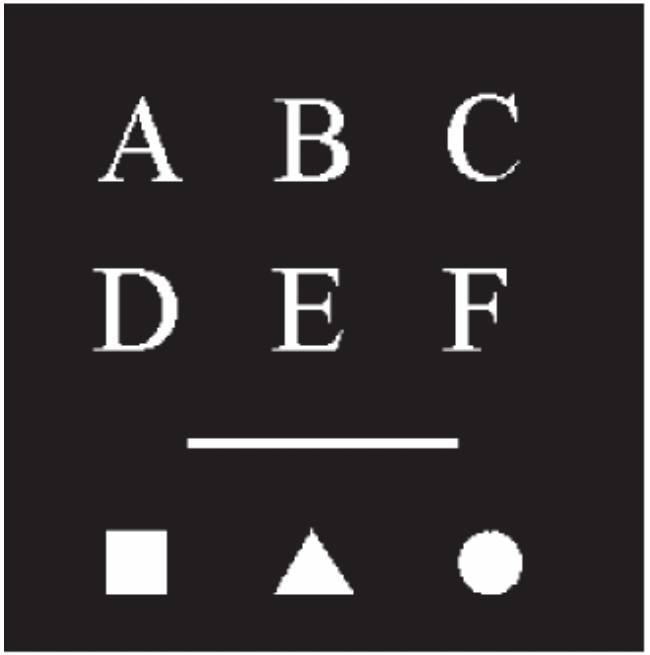


圖9.16：(a) 骨狀影像；(b) 使用函數bwmorph骨架化運算後得到的影像；(c) 使用函數endpoints修剪後的骨架

標記連通成份

標記連通成份

- 連通成分(connected components) ，也稱為物件(object)，如圖9.17、圖9.18和圖9.19



(a)

0	1	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0	0	0

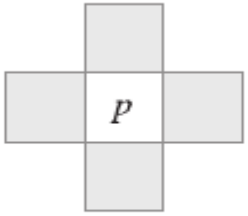
(b)

9.17

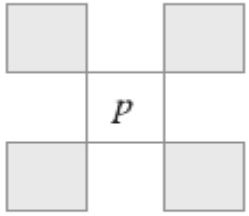
(a) 包含 10 個物件的影像；(b) 影像中像素點的一個子集合

標記連通成份

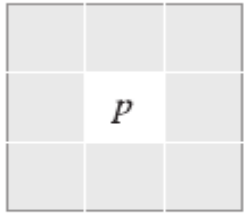
- 圖9.18



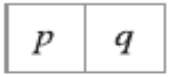
(a)



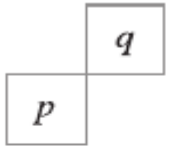
(b)



(c)



(d)



(e)

0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
1	1	1	0	0
0	0	0	0	0

(f)

0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
1	1	0	0	0
0	0	0	0	0

(g)

圖 9.18

(a) 像素點 p 及其 4 相鄰點；**(b)** 像素點 p 及其對角相鄰點；**(c)** 像素點 p 及其 8 相鄰點；**(d)** 像素點 p 和 q 是 4 鄰接及 8 鄰接的關係；**(e)** 像素點 p 和 q 是 8 鄰接的關係而不是 4 鄰接的關係；**(f)** 陰影像素點同時為 4 連通和 8 連通的關係；**(g)** 陰影像素點是 8 連通但不是 4 連通的關係

標記連通成份

- 圖9.19

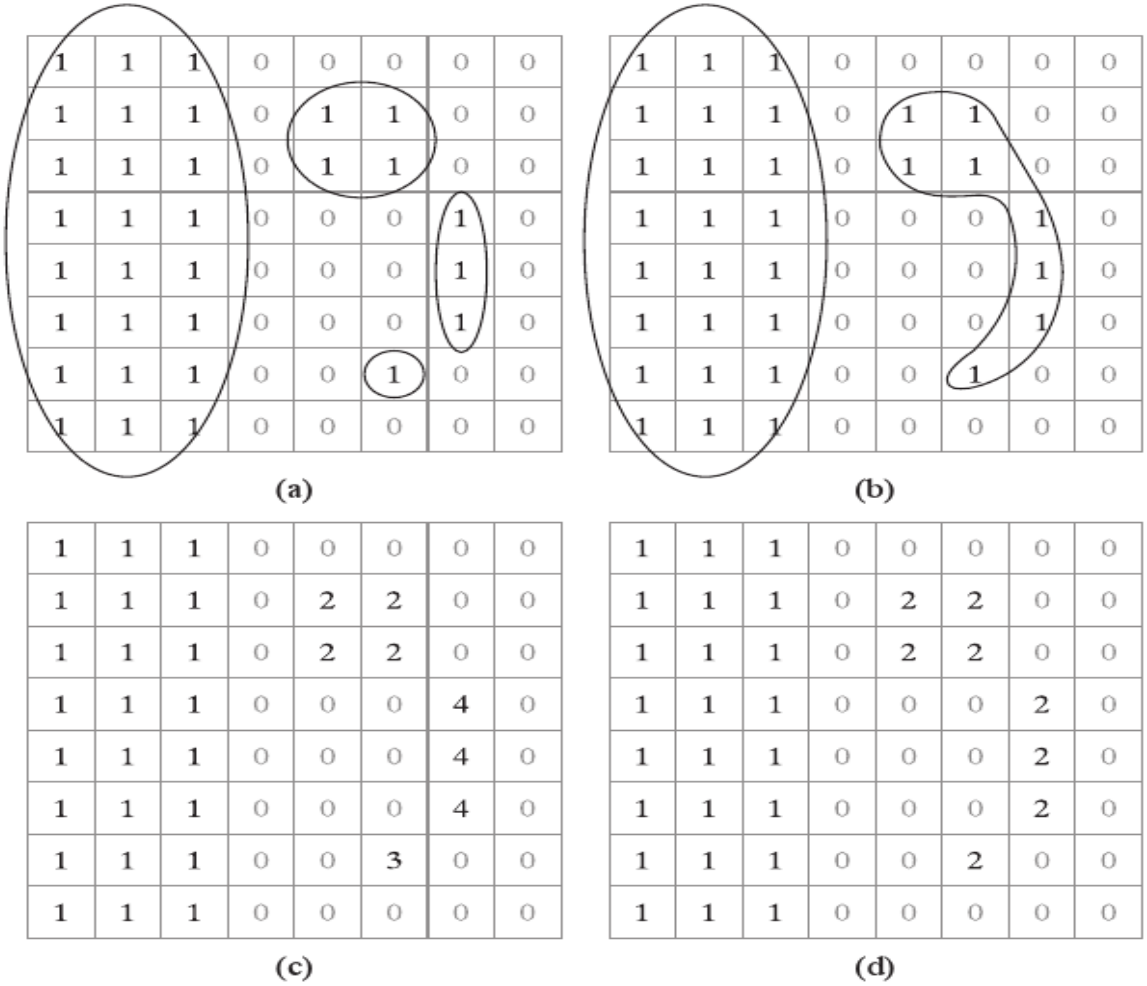


圖 9.19

連通成分

(a) 四個 4 連通成分

(b) 兩個 8 連通成分

(c) 使用 4 連通方式計算而來的標記矩陣

(d) 使用 8 連通方式計算而來的標記矩陣

標記連通成份

- 範例9.7計算且顯示連通成分的質量中心

這個範例說明如何計算與顯示圖 9.17(a) 中每個連通成份的質量中心。首先，我們使用函數 bwlabel 來計算 8 連通成分：

```
>> f = imread('objects.tif');
>> [L, n] = bwlabel(f);
```

函數 find（參見 4.2.2 節）對於標記矩陣的處理極為有用。例如以下呼叫函數 find 且傳回屬於第三個物件的所有像素點之行與列索引值：

```
>> [r, c] = find(L == 3);
```

函數 mean 以 r 和 c 作為輸入，接著計算出此物件的質量中心。

```
>> rbar = mean(r);
>> cbar = mean(c);
```

使用迴圈來計算與顯示影像中所有物件的質量中心。為了使質量中心與影像疊印時能清楚可見，我們使用填滿黑色的圓上放白色 "*" 作為標記來顯示質量中心，如下列指令：

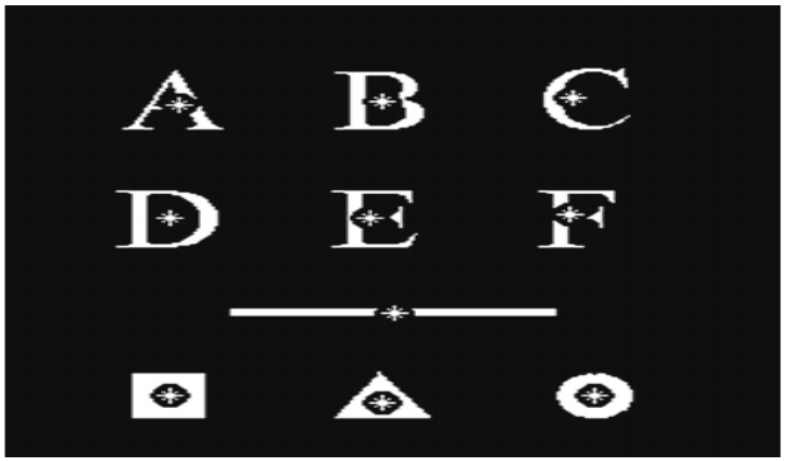
參見 11.4.1 節。其中討論到函數 regionprops 提供一個更快更方便的物件中心計算方法

標記連通成份

- 範例9.7計算且顯示連通成分的質量中心

```
>> imshow(f)
>> hold on % 如此一來，以後的指令會畫在影像最頂層的圖層上。
>> for k = 1:n
   [r, c] = find(L == k);
   rbar = mean(r);
   cbar = mean(c);
   plot(cbar, rbar, 'Marker', 'o', 'MarkerEdgeColor', 'k',...
        'MarkerFaceColor', 'k', 'MarkerSize', 10);
   plot(cbar, rbar, 'Marker', '*', 'MarkerEdgeColor', 'w');
end
```

圖 9.20 顯示運算結果。



9.20 質量中心（白色星號）以重疊方式顯示於所對應的連通成分上

形態學重建

形態學重建

- 標記(maker)：是轉換的起始點
- 遮罩(mask)：用以限制重建轉換
- 利用圖9.21來說明形態學重建

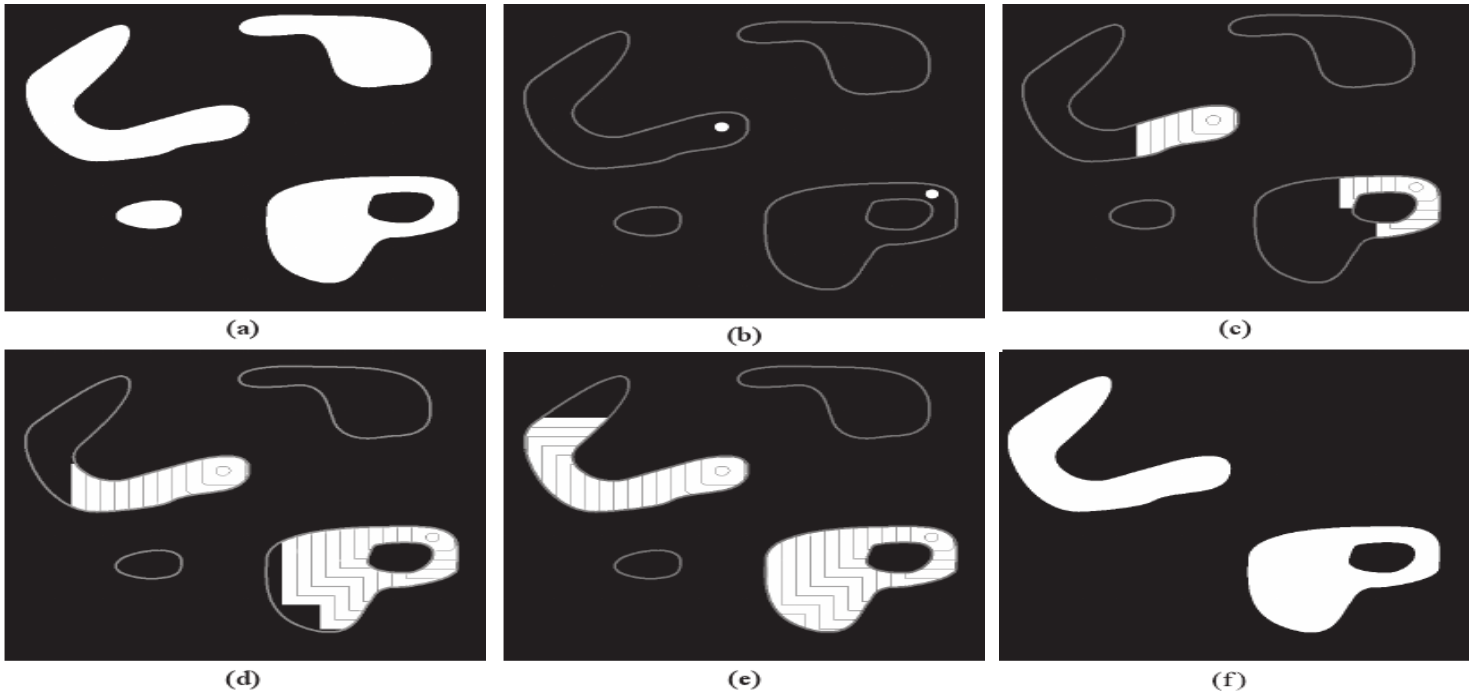


圖9.21：形態學重建(a) 原始影像(遮罩)；(b) 標記影像；(c) ~ (e) 個別為100,200和300 次迭代後的中間結果；(f) 最後結果(在遮罩影像中，物體的外觀重疊顯示於(b) ~ (e) 以供視覺上的參考)

形態學重建

- 範例9.8 重建執行斷開 (opening by reconstruction)運算

圖 9.22 顯示一張包含文字的影像，進行斷開運算及使用重建進行斷開運算的比較。在這個例子中，我們注意如何從圖 9.22(a) 中抽取含有垂直長筆劃字元。因為不論斷開運算或使用重建進行斷開運算都需要侵蝕運算，所以我們先使用長度與文字高度成正比的細長垂直結構元素完成此侵蝕步驟：

```
>> f = imread('book_text_bw.tif');
>> fe = imerode(f, ones(51, 1));
```

圖 9.22(b) 顯示運算結果，在圖 9.22(c) 中顯示以函數 imopen 完成的斷開運算：

```
>> fo = imopen(f,ones(51, 1));
```

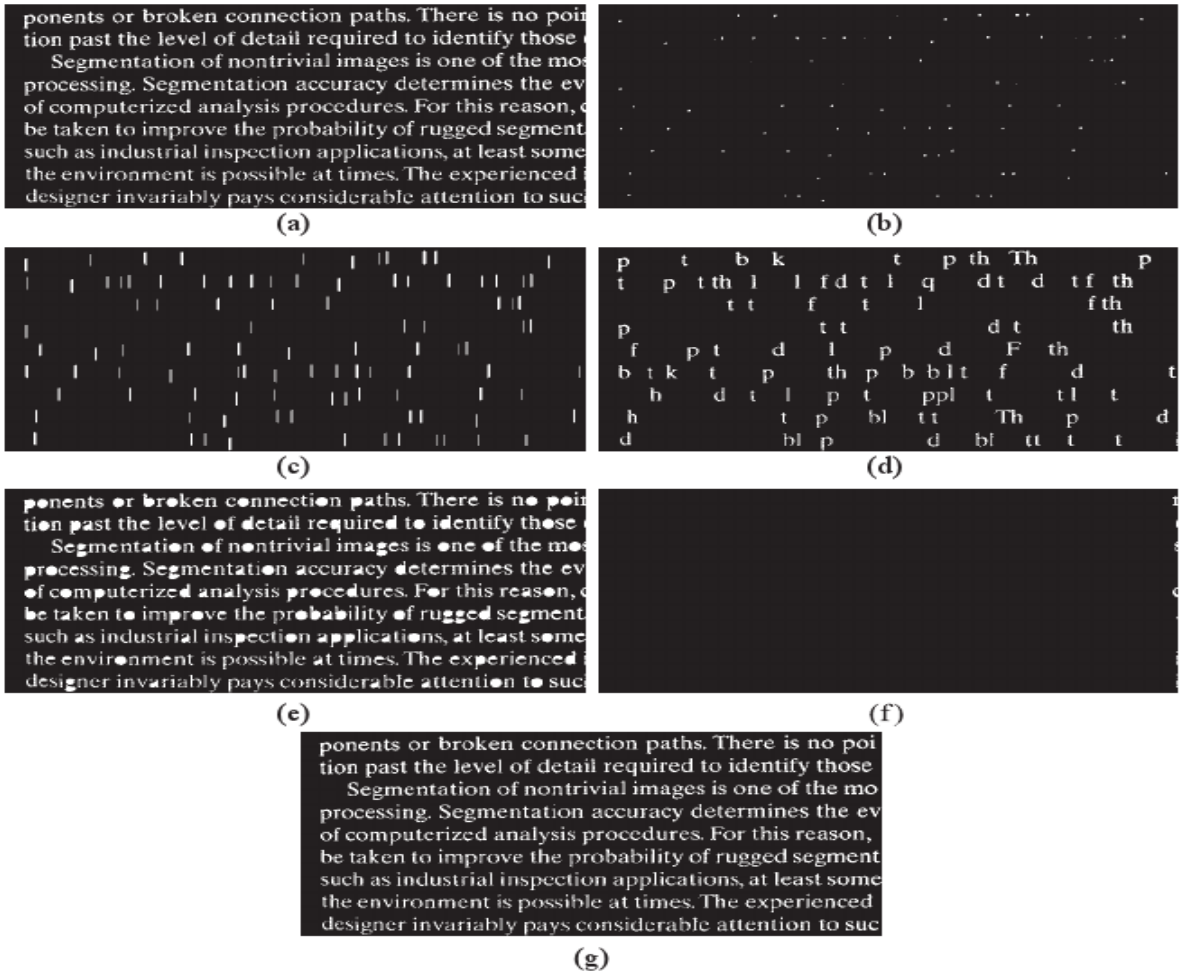
注意影像中垂直筆劃被修復了，但是其它包含這些筆劃的字元沒有被修復。最後，以下列指令得到重建的結果：

```
>> fobr = imreconstruct(fe, f);
```

圖 9.22(d) 中的結果顯示含有長垂直筆劃的字元被正確修復了，而其它所有字元都被移除了。圖 9.22 中剩下部份的運算將於接下來兩節中解釋。

形態學重建

- 範例9.8 重建執行斷開 (opening by reconstruction)運算



9.22

形態學重建

(a) 原始影像

(b) 以垂直線進行侵蝕後的影像

(c) 以垂直線進行斷開

(d) 以垂直線用重建進行斷開

(e) 填補破洞

(f) 連接邊界的字元（注意右側邊界）

(g) 移除邊界字元

灰階形態學

灰階形態學

- 膨脹和侵蝕運算
  - 灰階膨脹運算(gray-scale dilation)定義如下：

$$(f \oplus b)(x,y) = \max\{f(x-x', y-y') + b(x',y') \mid (x',y') \in D_b\}$$

  - 灰階膨脹運算的方程式簡化如下：

$$(f \oplus b)(x,y) = \max\{f(x-x', y-y') \mid (x',y') \in D_b\}$$

灰階形態學

- 灰階侵蝕運算(gray-scale erosion)定義如下：

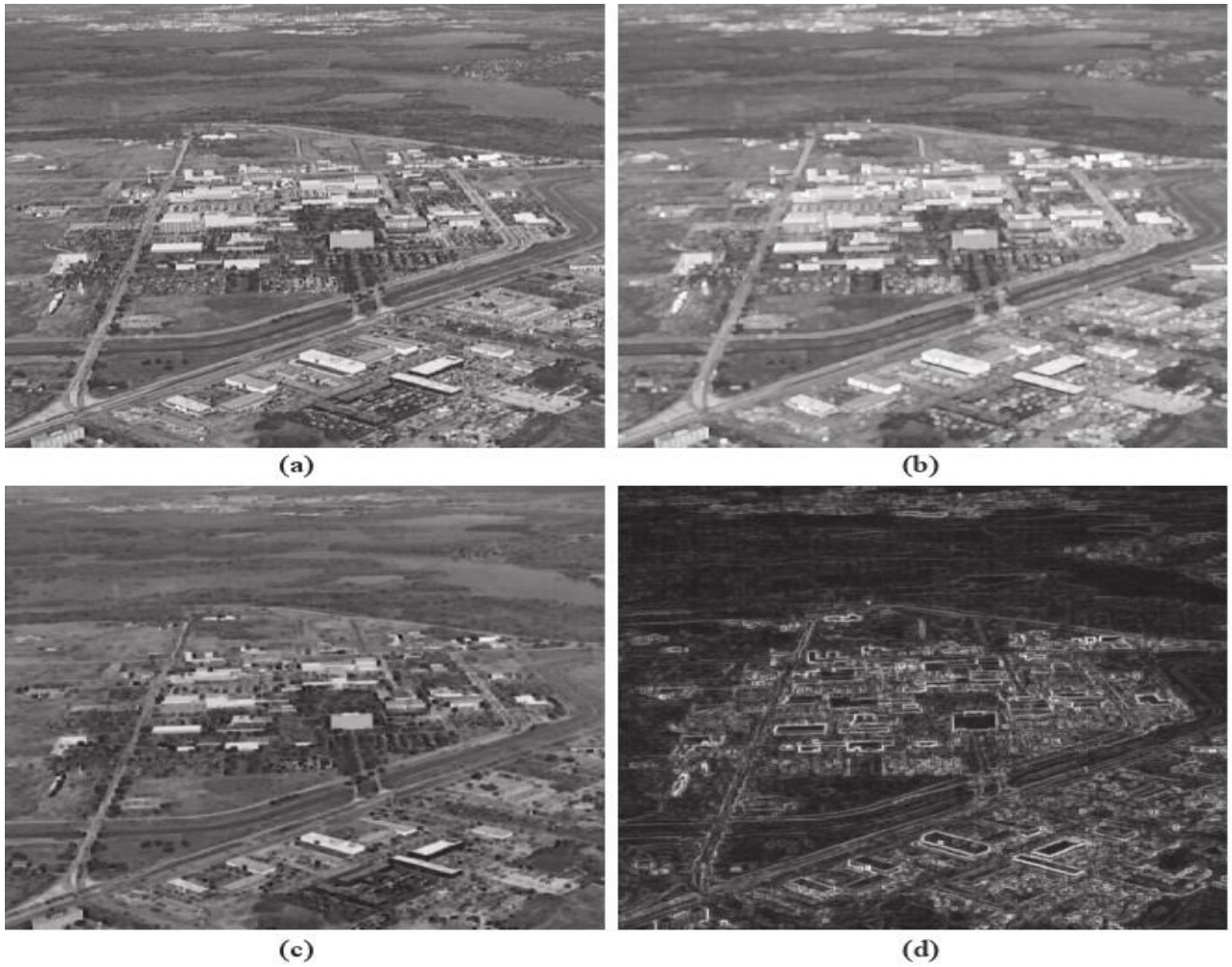
$$(f \ominus b)(x,y) = \min\left\{ f(x+x', y+y') - b(x',y') \,\middle|\, (x',y') \in D_b \right\}$$

- 灰階侵蝕運算的方程式簡化如下：

$$(f \ominus b)(x,y) = \min\left\{ f(x+x', y+y') \,\middle|\, (x',y') \in D_b \right\}$$

灰階形態學

- 圖9.23膨脹和侵蝕運算



9.23

膨脹與侵蝕運算 (a) 原始影像；(b) 膨脹運算後的影像；(c) 侵蝕運算後的影像；(d) 形態學上的梯度（原始影像由 NASA 提供）

灰階形態學

- 斷開和閉合運算
  - 斷開(opening) 運算，定義如下：

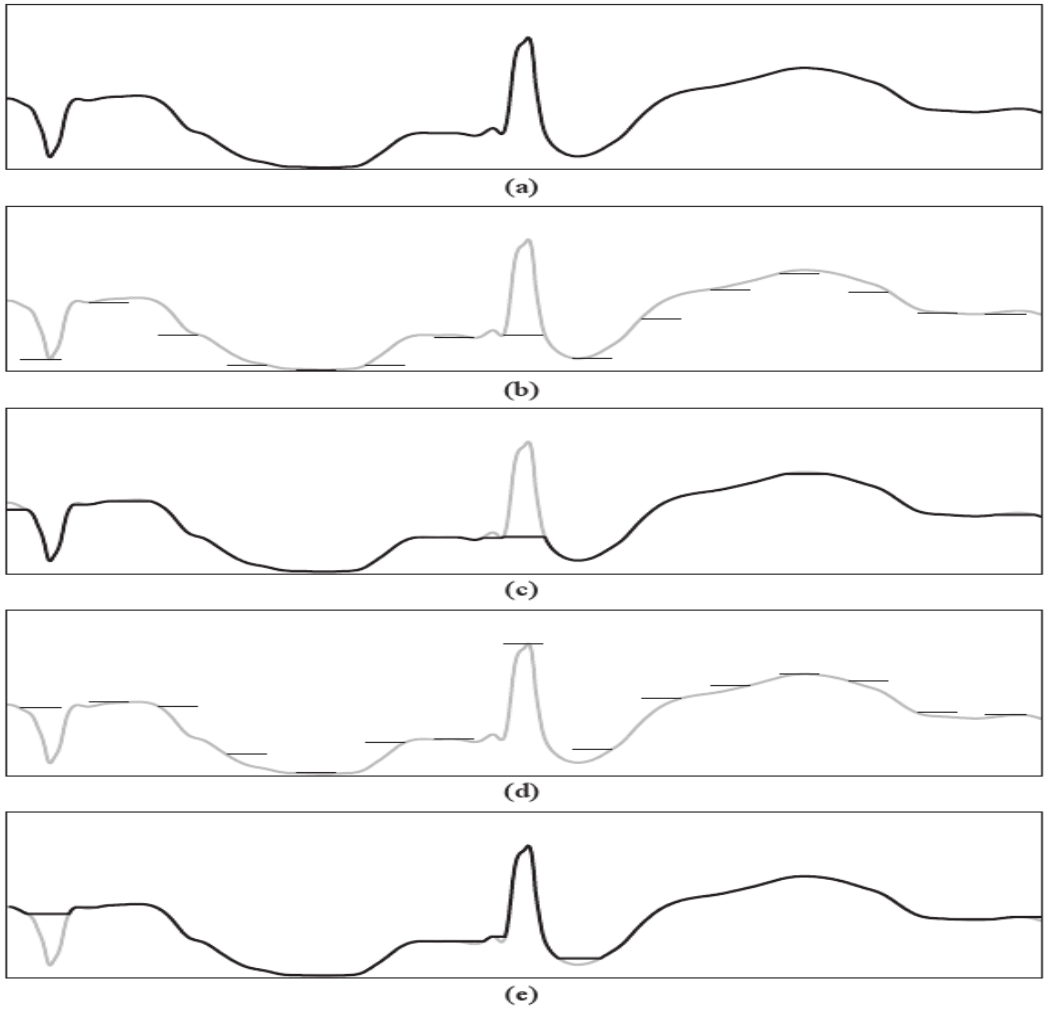
$$f \circ b = (f \ominus b) \oplus b$$

  - 閉合(closing) 運算，定義如下：

$$f \bullet b = (f \oplus b) \ominus b$$

灰階形態學

- 圖9.24 在一維空間中說明此概念



9.24

一組斷開與閉合 **(a)** 原始一維訊號；**(b)** 將訊號由下往上推的平坦結構元素；**(c)** 斷開運算；**(d)** 將訊號由上往下推的平坦結構元素；**(e)** 閉合運算

灰階形態學

- 範例9.9使用斷開和閉合運算完成形態學上的平滑化運算

因為斷開運算抑制小於結構元素的較亮細節，而閉合運算抑制小於結構元素的較暗細節，所以常常結合兩者運用於影像的平滑化與雜訊去除。在此例中，我們使用函數 imopen 與 imclose 對顯示於圖 9.25(a) 中的木塞影像進行平滑化運算。木頭紋理是這些木塞影像的主要特徵（以暗色條紋呈現），這些木紋疊印在一個適度均勻且較淡的背景上。記住圖 9.24 中說明的斷開與閉合運算之類推，將有助於理解以下指令的執行結果。

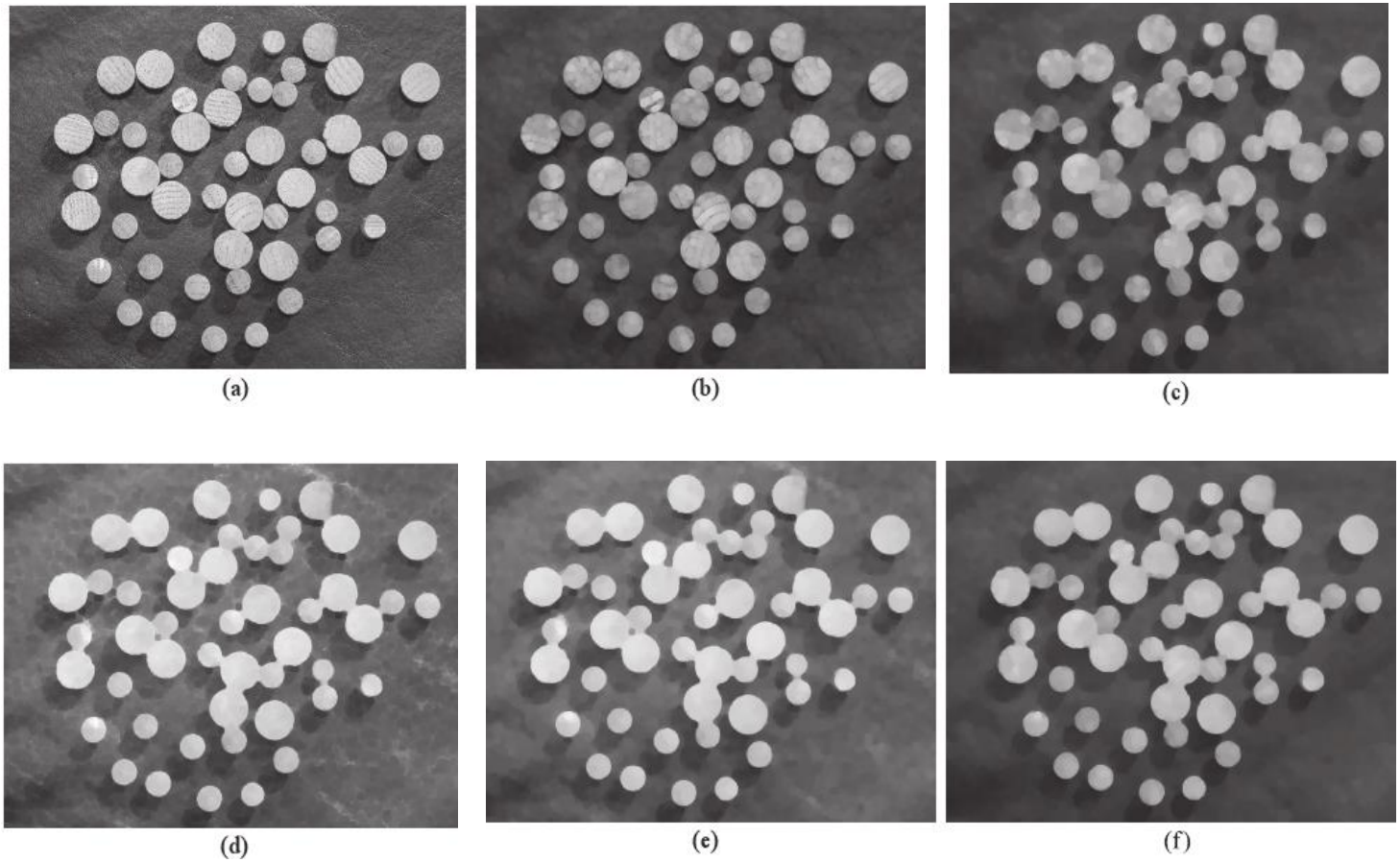
考量以下的執行步驟：

```
>> f = imread('plugs.jpg');
>> se = strel('disk', 5);
>> fo = imopen(f, se);
>> foc = imclose(fo, se);
```

圖 9.25(b) 顯示斷開運算後的影像 fo。在此。我們可以看到，顏色較亮的區域被淡化了（平滑處裡），在木塞上顏色較暗的條紋幾乎不受

灰階形態學

- 範例9.9使用斷開和閉合運算完成形態學上的平滑化運算



9.25

使用斷開和閉合運算進行平滑化運算 **(a)** 木塞的原始影像；**(b)** 使用半徑為 5 的圓盤作為結構元素進行斷開運算；**(c)** 完成斷開運算後進行閉合運算；**(d)** 對原始影像進行閉合運算；**(e)** 完成閉合運算後進行斷開運算；**(f)** 交替循環濾波後的結果

灰階形態學

- 範例9.9使用斷開和閉合運算完成形態學上的平滑化運算

另一個類似的處理程序，反轉了運算的順序，稱為**閉合－斷開濾波 (close-open filtering)**。圖 9.25(d) 顯示原始影像經過閉合運算後的結果，木塞上較暗的紋理已經被平滑掉，剩下較亮的細節部份（例如背景中較亮的條紋）。圖 9.25(d) 為圖 9.25(e) 經過斷開運算後的結果，顯示這些條紋被平滑化，且木塞的邊緣也被進一步的平滑化了。接下來是對原始影像進行全面性的平滑化運算。

另一種斷開與閉合運算組合的使用方式就是**交替循環濾波 (alternating sequential filtering)**。其中一個使用交替循環濾波的形式就是以一連串大小遞增的結構元素來進行斷開 - 閉合運算。下列的指令說明運作的過程，從一個小的結構元素開始，逐漸增加結構元素的尺寸大小，直到該結構元素與圖 9.25(b) 與 (c) 中使用的結構元素一樣大：

```
>> fasf = f;
>> for k = 2:5
       se = strel('disk' ,k);
       fasf = imclose(imopen(fasf, se), se);
   end
```

在付出額外處理的代價，得到比使用單獨一個斷開 - 閉合濾波器稍為平滑的結果，顯示於圖 9.25(f) 中。在這個特例中，比較使用這三種不同濾波器的效果，閉合 - 斷開濾波器產生最平滑的影像。

灰階形態學

- 範例9.10對非均勻背景進行補償處理

斷開運算可以用於補償背景不均勻的照度，圖 9.26(a) 顯示一張米粒花紋的影像 f，其底部背景比上面背景更暗，不均勻的照度造成影像臨界值（10.3 節）選取困難。例如，圖 9.26(b) 是一個選取臨界值且二元化的影像版本，在影像頂部的米粒與背景正確的分離開來，但是在影像底部的米粒卻不能適當的由背景分離出來。將影像進行斷開運算，只要結構元素夠大，不會被完整的匹配於米粒物件中，就可以合理產生普及於影像的背景之估測。例如使用下列的指令

```
>> se = strel('disk', 10);
>> fo = imopen(f, se);
```

產生圖 9.26(c) 中經過斷開運算的影像，將原始影像減去這個影像，可以產生有適當均勻背景的米粒影像：

```
>> f2 = f - fo;
```

灰階形態學

- 範例9.10對非均勻背景進行補償處理

圖 9.26(d) 顯示運算結果，而圖 9.26(e) 顯示新的影像選取臨界值（且二元化）後的結果，注意其效果改善明顯優於圖 9.26(b)。

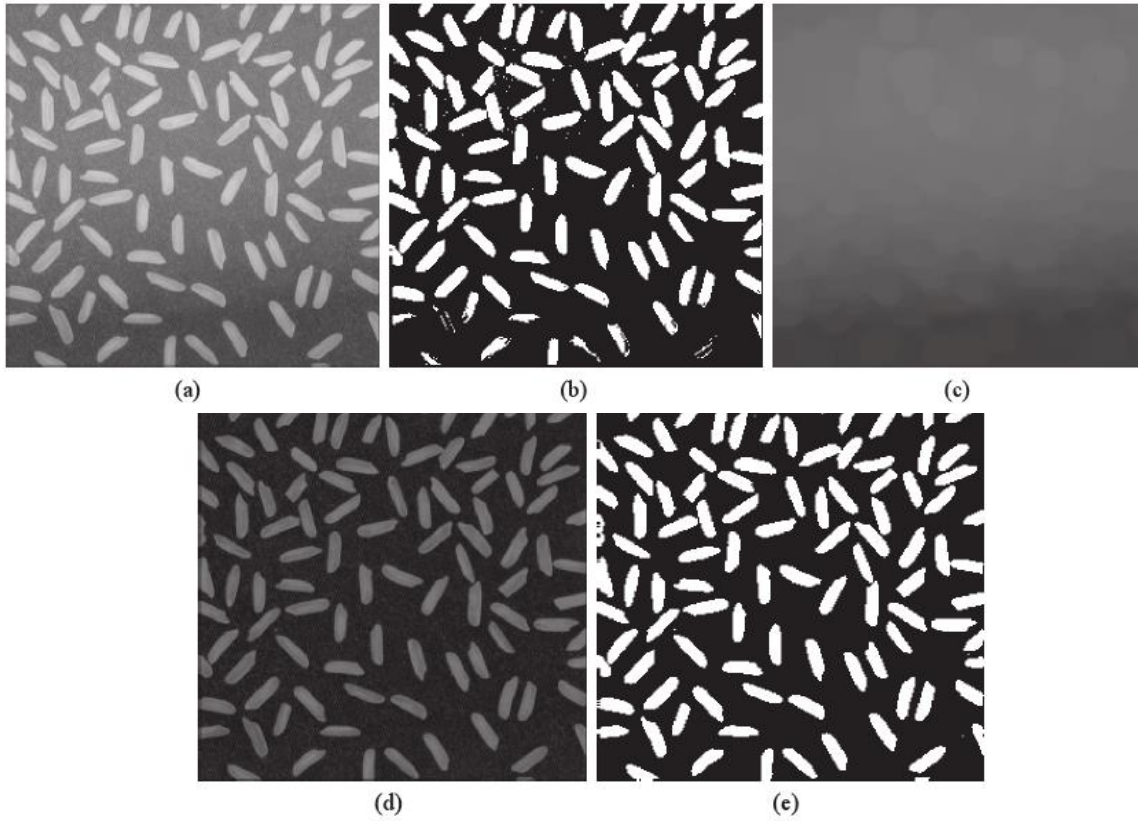


圖9.26：對不均勻照度進行補償處理(a) 原始影像；(b) 取臨界值後影像；(c) 顯示估測背景的斷開影像；(d) 原始影像減去估測背景後的結果；(e) 影像(d) 臨界值法後的結果 (原始影像由 MathWorks 公司提供)

灰階形態學

- 頂帽轉換(tophat transformation)
  - 使用原始影像減去經過斷開運算的影像
  - 工具箱函數imtophat

- 底帽轉換(bottomhat transformation)
  - 轉換為經過閉合運算的影像減去原始影像
  - 工具箱函數imbothat

灰階形態學

- 範例9.11粒度測定法(granulometry)

偵測一張影像中粒子的大小分佈是**粒度測定法 (granulometry)** 領域中一個重要的應用。形態學技術可以間接的用於測量粒子的大小分佈，也就是不需要明確的辨識與測量每個粒子。對於有規則形狀且比背景還要亮的粒子而言，基本的處理方法是運用逐漸遞增大小的結構元素進行形態學斷開運算。在每次的斷開運算中，計算所有在斷開中像素值之和，這個總和有時被稱為影像的**表面積 (surface area)**。以下的指令使用半徑 0 到 35 的碟型結構元素對圖 9.25(a) 中的影像進行斷開運算：

```
>> f = imread('plugs.jpg');
>> sumpixels = zeros(1, 36);
>> for k = 0:35
       se = strel('disk', k);
       fo = imopen(f, se);
       sumpixels(k + 1) = sum(fo(:));
end

>> plot(0:35, sumpixels), xlabel('k'), ylabel('Surface area')
```

圖 9.27(a) 顯示函數 sumpixels 對變數 k 所畫出的分佈圖形，更有趣的是連續斷開運算間，表面積將逐漸減少：

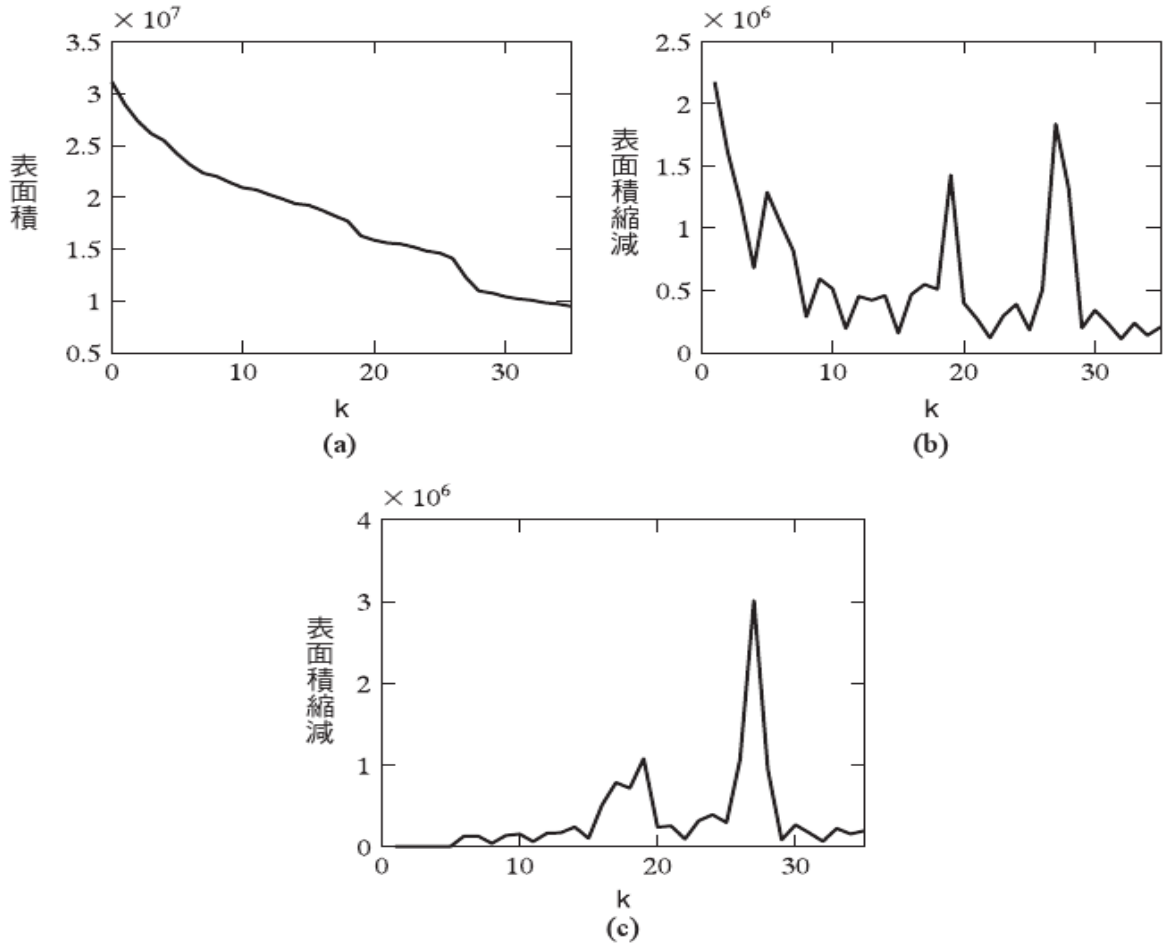
```
>> plot(-diff(sumpixels))
>> xlabel('k')
>> ylabel('Surface area reduction')
```

在圖 9.27(b) 中分佈圖的尖峰，表示有該半徑大小的物件出現。因為此分佈圖相當受雜訊干擾，因此我們使用圖 9.25(d) 中木塞影像的平滑化版本重覆此過程。結果呈現於圖 9.27(c) 中，更清楚的表示有兩個不同大小的物件出現在原始影像中。

如果 v 是一個向量則指令 diff(v) 會傳回一個比向量 v 少一個元素的向量，用來表示向量 v 中相鄰元素的差值。如果 X 是一個矩陣，則指令 diff(X) 會傳回一個以矩陣 X 各列差值所構成的矩陣。[X(2:end,:)-X(1:end-1,:)]

灰階形態學

- 範例9.11粒度測定法(granulometry)

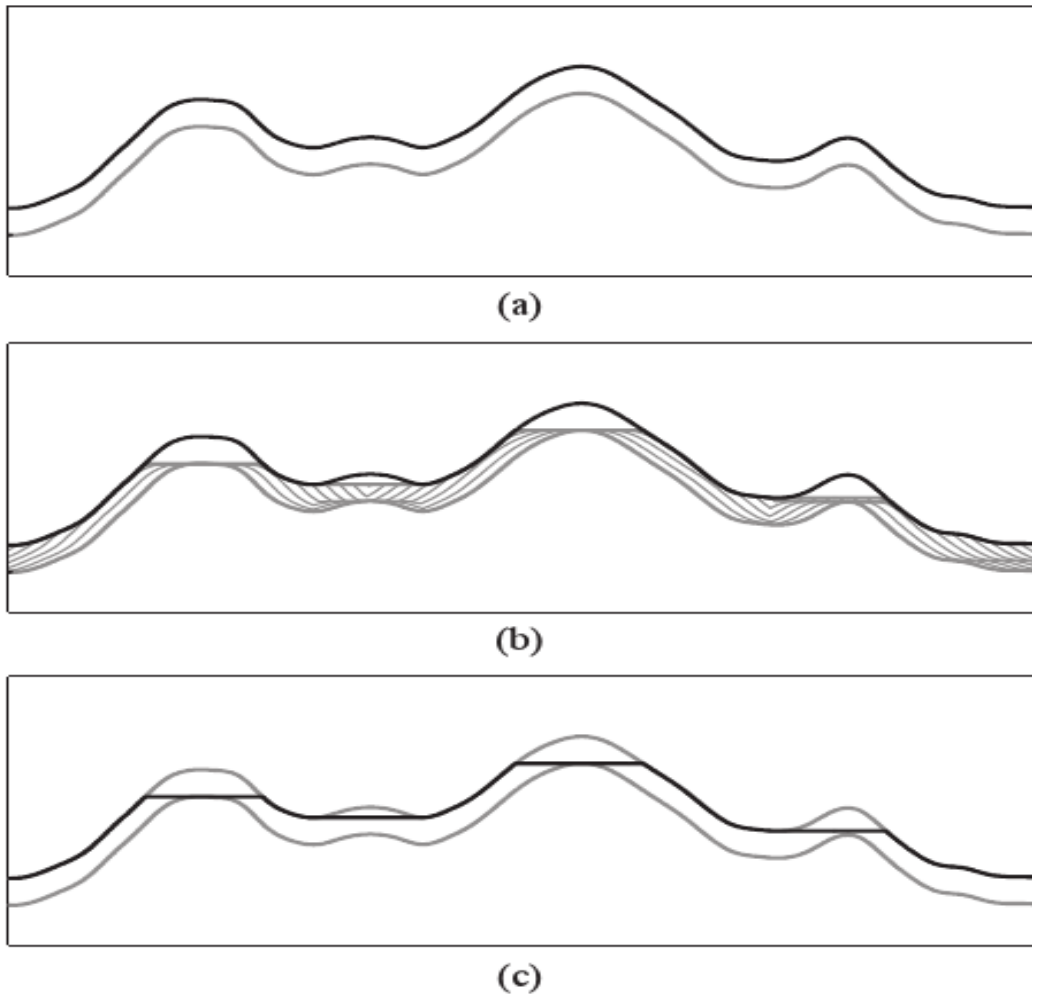


9.27

粒度測定法 **(a)** 表面積與結構元素半徑大小間的關係；**(b)** 表面積縮減與結構元素半徑大小間的關係；**(c)** 對平滑影像，表面積縮減與結構元素半徑大小間的關係

灰階形態學

- 圖9.28 一維的灰階形態學重建



9.28

一維的灰階形態學重建

(a) 遮罩（上方）和標記曲線

(b) 重建的遞迴計算

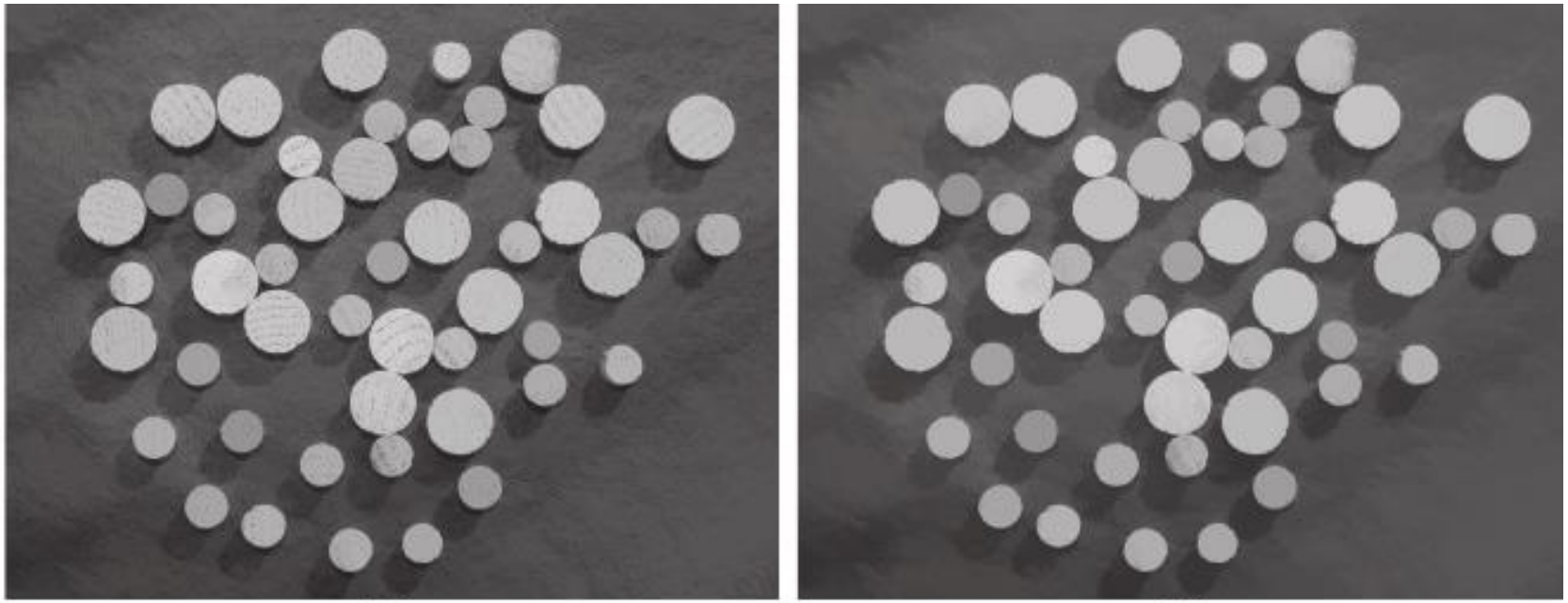
(c) 重建的結果（黑色曲線）

灰階形態學

- 使用重建來進行斷開運算(openingby-reconstruction)
  - 影像先進行侵蝕運算
  - 進行閉合運算
- 使用重建來進行閉合運算(closing-byreconstruction）
  - 取得影像的補集
  - 重建進行斷開運算
  - 取得結果的補集

灰階形態學

圖9.29 (a) 使用 重建進行斷開運算；(b)使用 重建進行斷開運算後再使用 重建進行閉合運算



灰階形態學

- 範例9.12使用灰階重建移除複雜背景

我們最後總結的範例是在幾個步驟中使用灰階重建，目的在於將顯示於圖 9.30(a) 中的計算機按鍵影像中的文字分離出來。第一個步驟為抑制在每個按鍵頂端的水平反射。為完成此工作，我們利用這些反射會比影像中任一個文字的字體寬的現象，我們使用一個長水平線作為結構元素，使用重建完成斷開運算：

```
>> f = imread('calculator.jpg');
>> f_obr = imreconstruct(imerode(f, ones(1, 71)), f);
>> f_o = imopen(f, ones(1, 71));  % 此為用來作比較的影像。
```

使用重建來進行斷開運算 (f_obr) 的結果顯示於圖 9.30(b)。為了比較，圖 9.30(c) 顯示標準斷開運算 (f_o) 的結果。使用重建來進行斷開運算在水平相鄰按鍵間的背景抽取有較好的成果，原始影像減去使用重建進行斷開運算之結果影像稱為**使用重建進行頂帽轉換 (tophat-by-reconstruction)**，其結果顯示於圖 9.30(d) 中：

```
>> f_thr = f - f_obr;
>> f_th = f - f_o;          % 或是指令imtophat(f,one(1,71))。
```

圖 9.30(e) 顯示標準的頂帽轉換（也就是 f_th）之結果。

灰階形態學

- 範例9.12使用灰階重建移除複雜背景

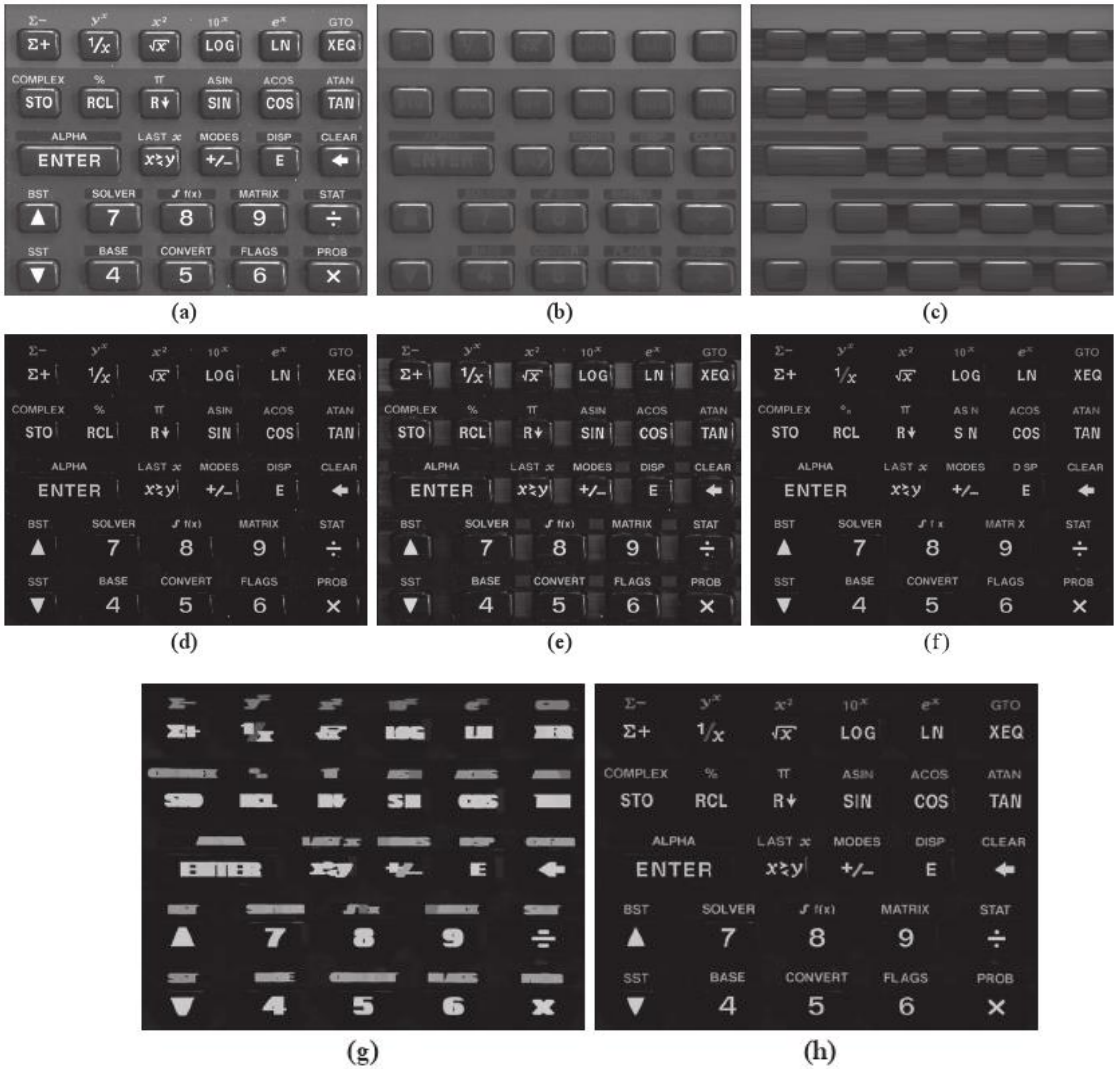


圖9.30：一個灰階重建的應用(a) 原始影像；(b) 使用重建進行斷開運算；(c) 斷開運算；(d) 使用重建來進行頂帽運算；(e) 頂帽運算；(f) 對影像(d) 使用水平線作為結構元素並使用重建進行斷開運算；(g) 對影像(f)使用水平線作為結構元素進行膨脹運算；(h) 最後的重建結果

灰階形態學

- 範例9.12使用灰階重建移除複雜背景

接著，我們抑制在圖 9.30(d) 中按鍵右邊緣的垂直反射。這個工作可以使用一個短垂直線作為結構元素，使用重建完成斷開運算來完成：

```
>> g_obr = imreconstruct(imerode(f_thr, ones(1, 11)), f_thr);
```

在圖 9.30(f) 的運算結果中，垂直反射消失了，但是字元中較細的垂直筆劃也消失了，例如百分比符號中的斜線和單字 ASIN 中的字元 I 都消失了，我們利用被錯誤抑制的字元在空間上會非常接近其它仍然存在的字元這個現象，先進行膨脹運算如圖 9.30(g) 所示

```
>> g_obrd = imdilate(g_obr, ones(1, 21));
```

接著以影像 f_thr 為遮罩影像，影像 (g_obrd, f_thr) 為標記影像進行最後的重建運算：

```
>> f2 = imreconstruct(min(g_obrd, f_thr), f_thr);
```

圖 9.30(h) 顯示最後的結果，注意影像中在背景跟按鍵上的陰影與反射都順利的移除了。