# 《数字图像处理》
# 第10讲 形态学图像处理

冯建江

清华大学 自动化系

2017.11.30

# 形态学图像处理

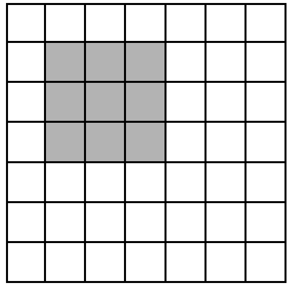
- 抽取图像中区域的形状特征，如边界、骨骼和凸壳等，也经常用于图像的预处理和后处理，如形态学滤波、细化和修剪等。
- 道德经：道生一，一生二，二生三，三生万物
- 形态学也给人这种感觉

# 内　容

- 数学基础
- 形态学基本运算
- 形态学算法
- 灰度形态学

# 集合的概念

- 形态学图像处理主要关注图像中的区域（例如物体），区域可表示为像素坐标的集合

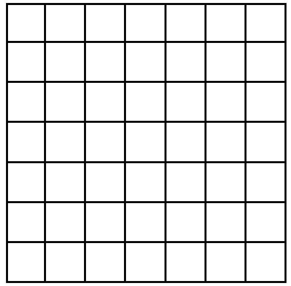


集合$A$

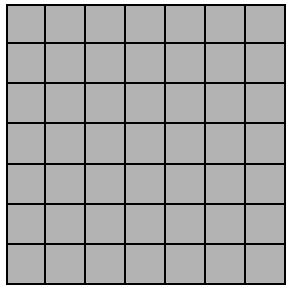
- 定义集合的方法：罗列元素或者用表达式

$$A = \{(2,2),(2,3),(2,4),(3,2),(3,3),(3,4),(4,2),(4,3),(4,4)\}$$
$$A = \{(x,y)|2 \le x \le 4, 2 \le y \le 4\}$$

- 两个特殊的集合
  - 空集$\emptyset$（empty set）：没有元素的集合
  - 全集$U$（universe set）：图像域中全部像素坐标的集合



空集

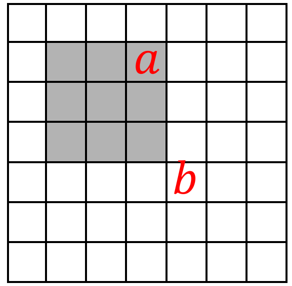


全集

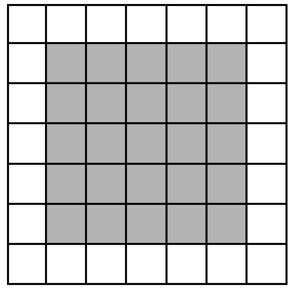
SetOperations.m

# 集合的概念

- 如果$a=(x,y)$是$A$的一个元素，称$a$属于$A$，记$a\in A$。例如(4,2)
- 如果$b$不是$A$的元素，称$b$不属于$A$，记$b\notin A$。例如(5,5)
- 子集：如果集合$A$的所有元素也属于集合$B$，则称$A$是$B$的子集，记为$A\subseteq B$

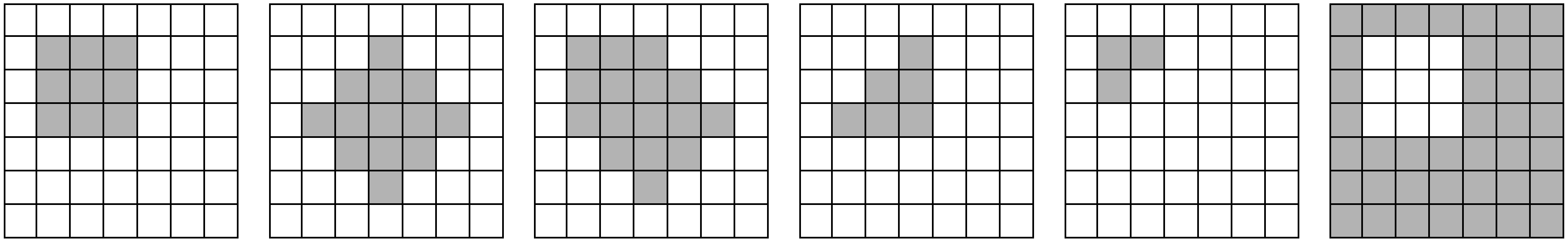


集合$A$


集合$B$

# 集合的运算

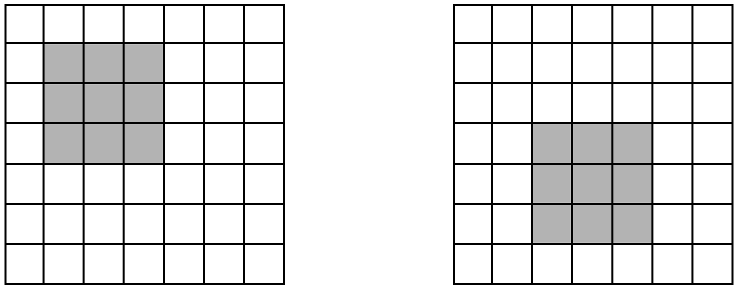
- $A$和$B$的并集（union）：$C = A \cup B$
- $A$和$B$的交集（intersection）：$D = A \cap B$
- 不相连（互斥）：$A \cap B = \emptyset$
- $A$和$B$的差集：$A - B = \{w | w \in A, w \notin B\} = A \cap B^c$
- $A$的补集：$A^c = \{w | w \notin A\} = U - A$



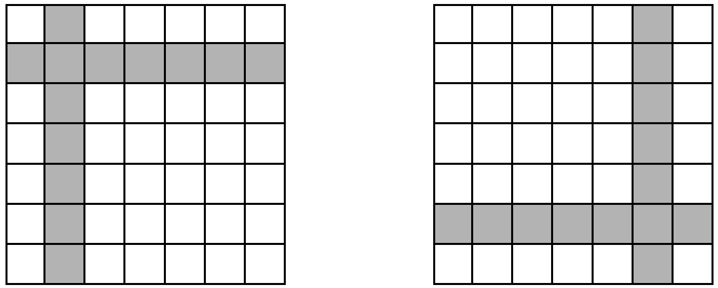
集合$A$　集合$B$　$A \cup B$　$A \cap B$　$A - B$　$A^c$

# 集合的运算

- 形态学运算需要将小图（结构元素）在另一幅大图上移动。集合的平移：$(A)_z$
- 有的形态学运算还需要将结构元素旋转180度（类似空域卷积）。集合的反射（旋转180度）：$\hat{B}$



平移



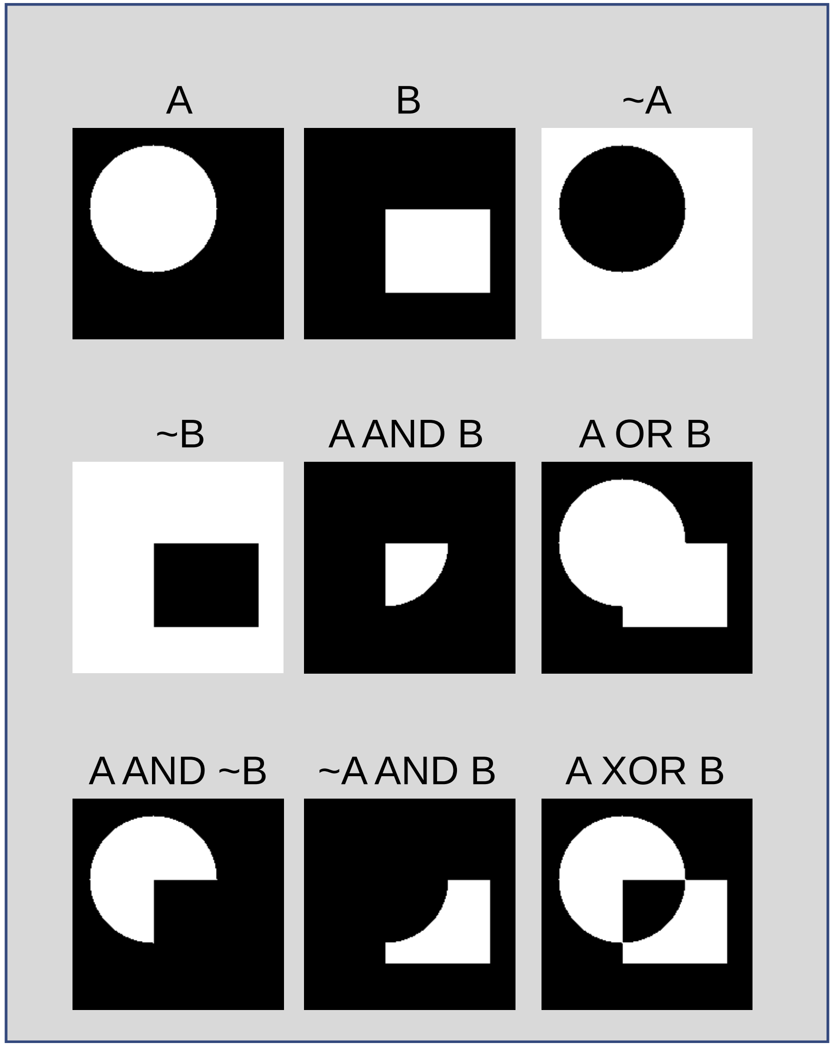
反射

# 逻辑运算

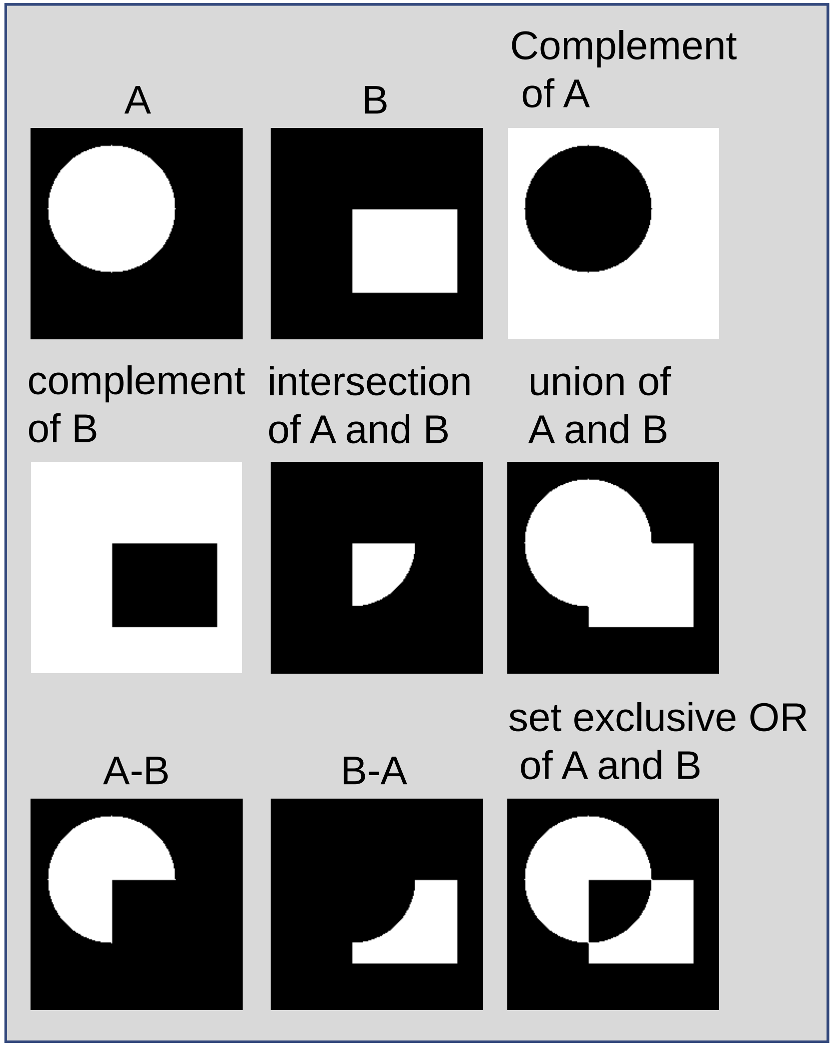
形态学是针对二值图像
三种最基本的逻辑运算：与、或、非（补）

| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

# 对二值图像，逻辑运算与集合运算等价




逻辑运算




集合运算

SetAndLogic.m

# 内　容

- 数学基础
- 形态学基本运算
- 形态学算法
- 灰度形态学

# 形态学基本运算
# （Basic Morphological Operations）

- 腐蚀 (erosion)
- 膨胀 (dilation)
- 开和闭 (opening and closing)
- 击中与否变换 (hit-or-miss transformation)

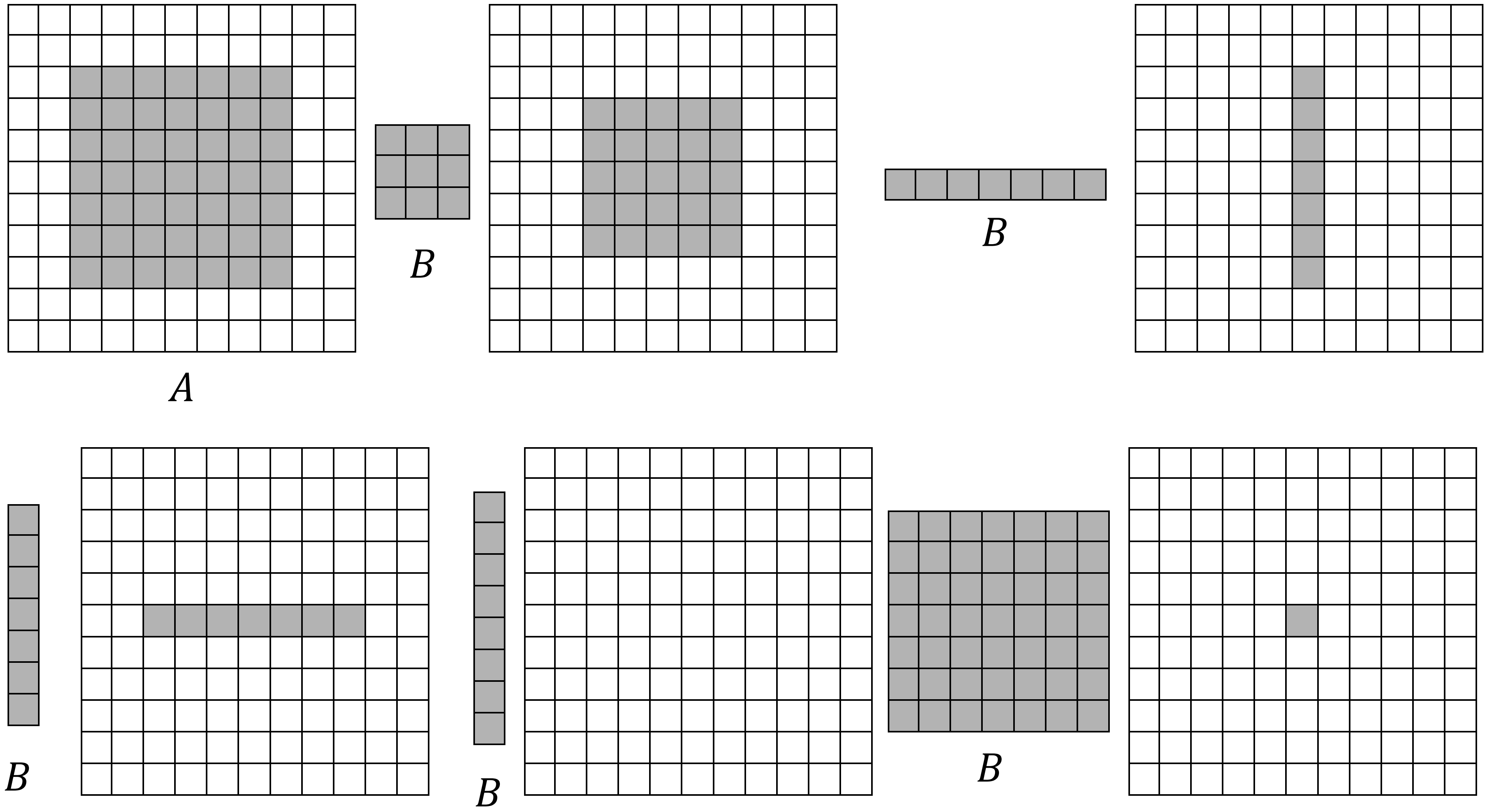
# 腐蚀

- $A$和$B$是$Z^2$（二维整数坐标系）上的两个集合，用结构元素$B$对$A$进行腐蚀的定义为：

$$A \ominus B = \{z|(B)_z \subseteq A\}$$

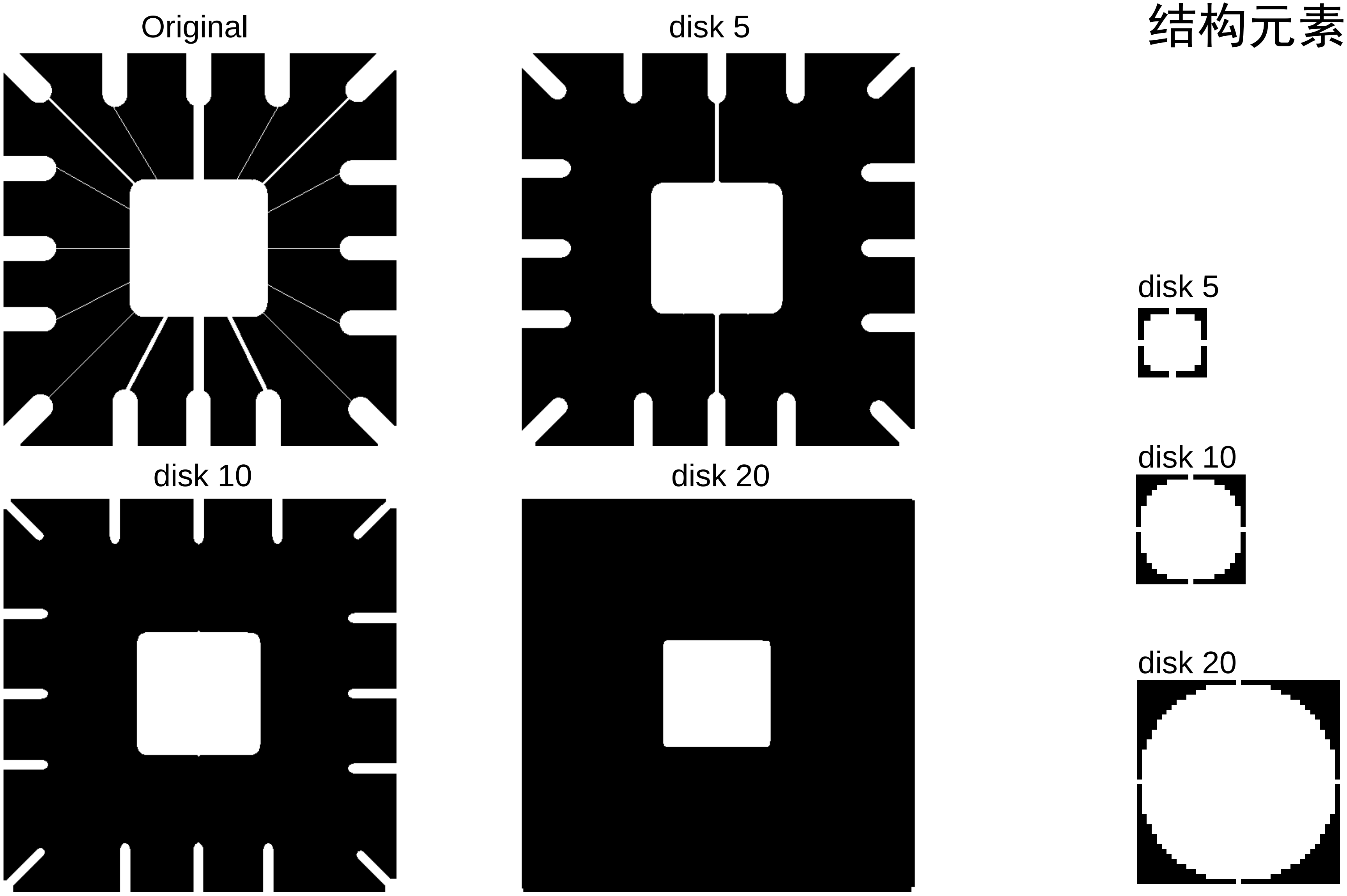
- 含义：将$B$平移$z$后包含于$A$。这些$z$构成的集合。
- 等价于：$A \ominus B = \{z|(B)_z \cap A^c = \emptyset\}$
- 上述定义基于数学公式，准确、简洁，但不够直观。
- 用图像来解释更为直观。

# 腐蚀示例



```matlab
% ErosionDemo
N = 11;
[X, Y] = meshgrid(1:N);
t = 20; % magnify factor
A = false(N, N);
A(Y>=3 & Y<=9 & X>=3 & X<=9) = 1;
J = MagnifyAndGrid(A, t);
k=0;
k=k+1;
figure(k),imshow(J),set(k,'name','A');
imwrite(J,sprintf('ErosionDemo_%d.bmp',k));
B = ones(3, 3);
J = MagnifyAndGrid(B, t);
k=k+1;
figure(k),imshow(J),set(k,'name','B');
imwrite(J,sprintf('ErosionDemo_%d.bmp',k));
C = imerode(A, B);
J = MagnifyAndGrid(C, t);
k=k+1;
figure(k),imshow(J),set(k,'name','Erosion of A');
imwrite(J,sprintf('ErosionDemo_%d.bmp',k));
```

# 腐蚀



```matlab
% ex0901_erosion
A = imread('..\data\Fig0905(a)(wirebond-mask).tif');
se1 = strel('disk', 5, 0);
A1 = imerode(A, se1);
se2 = strel('disk', 10, 0);
A2 = imerode(A, se2);
se3 = strel('disk', 20, 0);
A3 = imerode(A, se3);
figure(1),
ax(1)=subplot(2,2,1); imshow(A); title('Original');
ax(2)=subplot(2,2,2); imshow(A1); title('disk 5');
ax(3)=subplot(2,2,3); imshow(A2); title('disk 10');
ax(4)=subplot(2,2,4); imshow(A3); title('disk 20');
linkaxes(ax);

figure(2),
ax2(1)=subplot(1,3,1); imshow(getnhood(se1)); title('disk 5');
ax2(1)=subplot(1,3,2); imshow(getnhood(se2)); title('disk 10');
ax2(1)=subplot(1,3,3); imshow(getnhood(se3)); title('disk 20');
linkaxes(ax2);
```

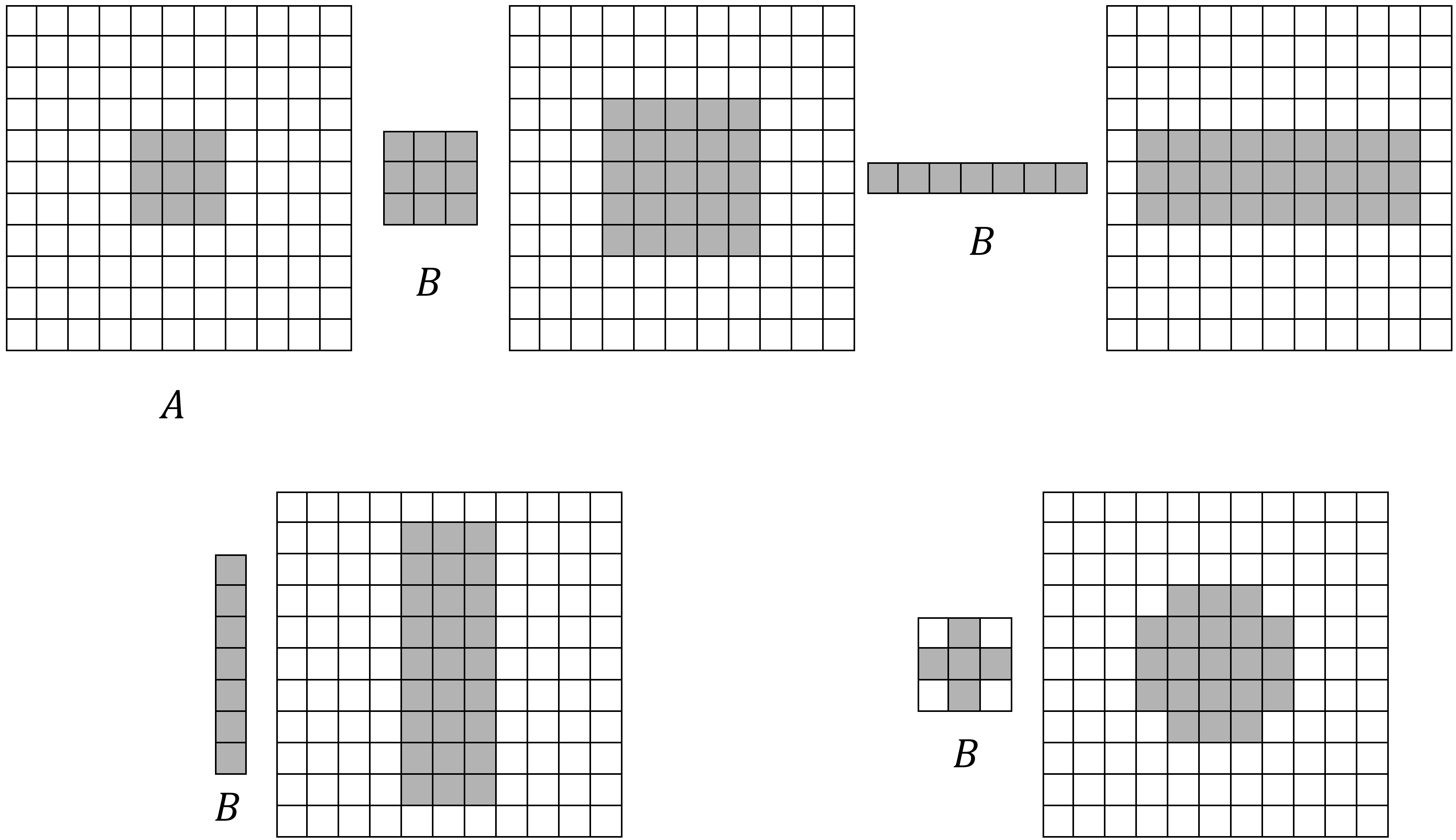
# 膨胀

- $A$和$B$是$Z^2$上的两个集合，用结构元素$B$对$A$进行膨胀的定义为：

$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\}$$

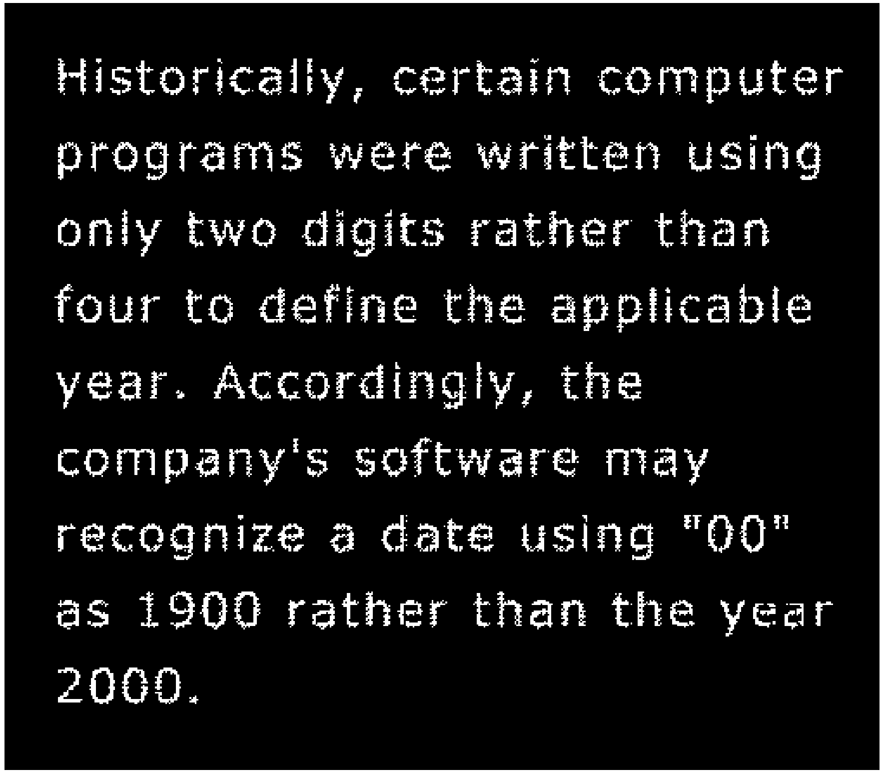
- 含义：将$B$围绕其原点旋转180度（即$\hat{B}$），再将其平移$z$（即$(\hat{B})_z$），与$A$的交集不为空。这些$z$组成的集合。
- 与空域卷积过程很类似
- 上述定义基于数学公式，准确、简洁，但不够直观。下面用图像来解释。

# 膨胀示例

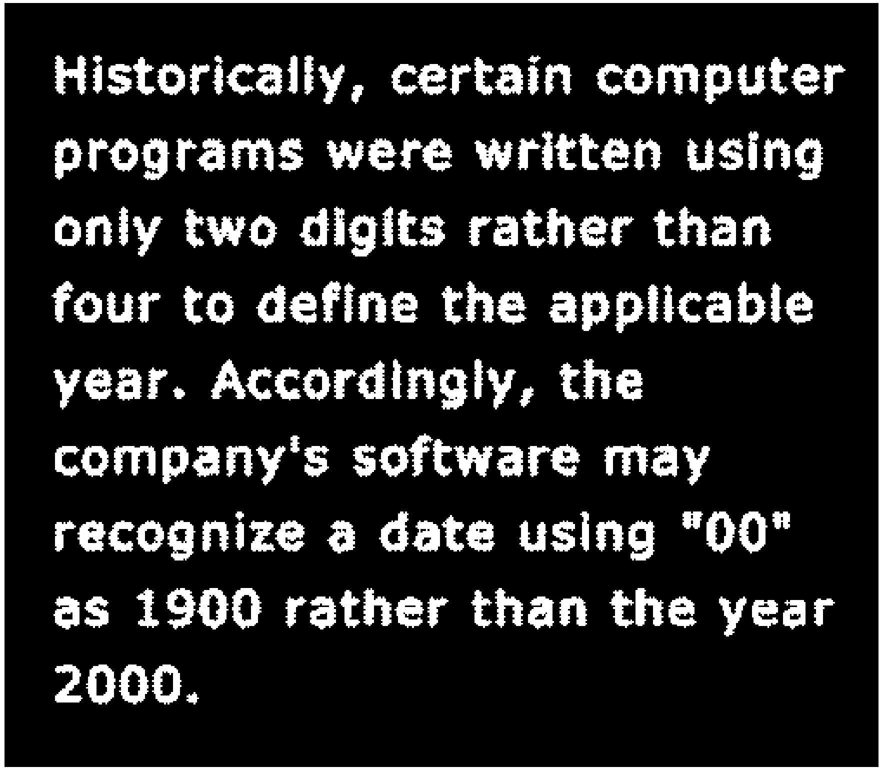


```matlab
% DilationDemo
close all
N = 11;
[X, Y] = meshgrid(1:N);
t = 20; % magnify factor

A = false(N, N);
A(Y>=5 & Y<=7 & X>=5 & X<=7) = 1;
J = MagnifyAndGrid(A, t);
k=0;
k=k+1;
figure(k),imshow(J),set(k,'name','A');
imwrite(J,sprintf('DilationDemo_%d.bmp',k));

B = [0 1 0; 1 1 1; 0 1 0];
J = MagnifyAndGrid(B, t);
k=k+1;
figure(k),imshow(J),set(k,'name','B');
imwrite(J,sprintf('DilationDemo_%d.bmp',k));

C = imdilate(A, B);
J = MagnifyAndGrid(C, t);
k=k+1;
figure(k),imshow(J),set(k,'name','Dilation of A');
imwrite(J,sprintf('DilationDemo_%d.bmp',k));
```

# 膨胀应用举例：连接断裂字符



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

结构元素

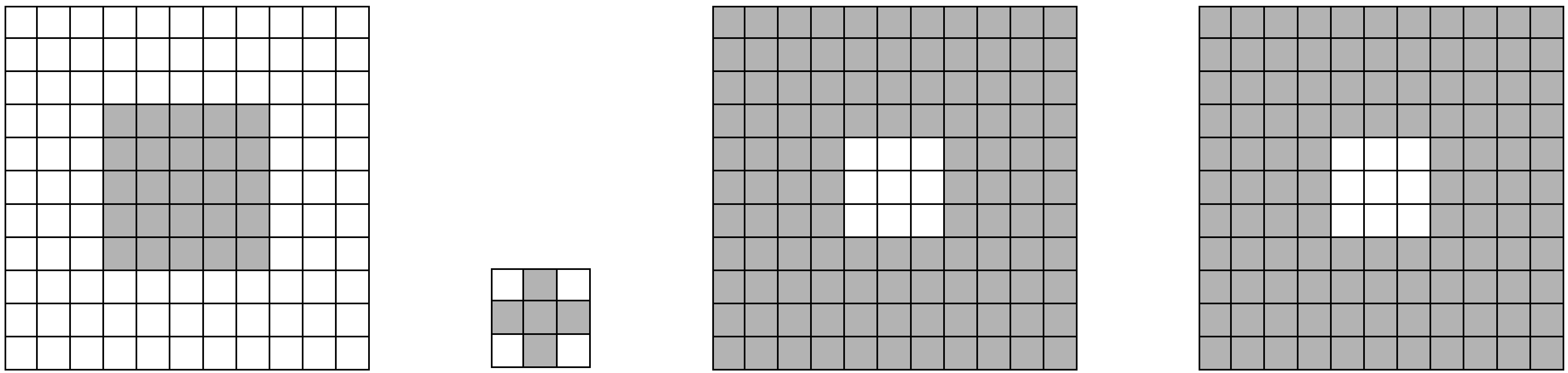
| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

```
% ex0902_dilation
A = imread('..\data\Fig0907(a)(text_gaps_1_and_2_pixels).tif');
B = [0 1 0; 1 1 1; 0 1 0];
A2 = imdilate(A, B);
figure(1),
ax(1)=subplot(1,2,1); imshow(A);
ax(2)=subplot(1,2,2); imshow(A2);
linkaxes(ax);
```

# 腐蚀与膨胀的对偶性

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$



$A$　　$B$　　$(A \ominus B)^c$　　$A^c \oplus \hat{B}$

# 开与闭

- 用结构元素$B$对$A$进行开运算的定义为：

$$A \circ B = (A \ominus B) \oplus B$$

- 相当于先用结构元素$B$对$A$腐蚀，再对腐蚀结果用同样的结构元素进行膨胀。
- 用结构元素$B$对$A$进行闭运算的定义为：

$$A \cdot B = (A \oplus B) \ominus B$$

- 相当于先用结构元素$B$对$A$膨胀，再对膨胀结果用同样的结构元素进行腐蚀，过程与开运算正好相反。
- 注意：不同文献用的运算符号可能不同，但运算相同
- 下面做直观的解释

# 开运算的几何解释

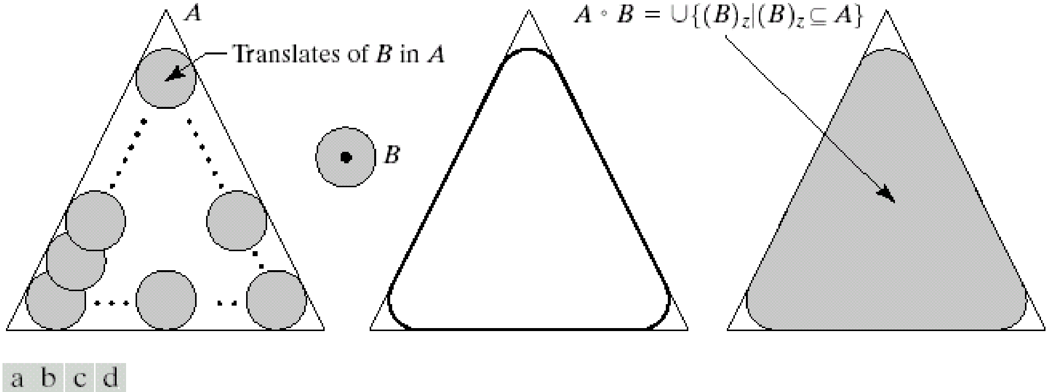



**FIGURE 9.8** (a) Structuring element $B$ "rolling" along the inner boundary of $A$ (the dot indicates the origin of $B$). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).

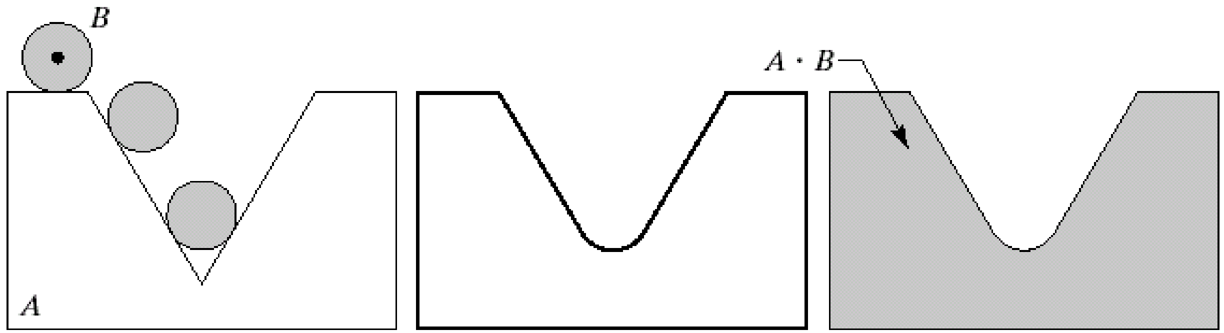
开运算：当$B$在$A$内部滚动时，$B$所能覆盖到的$A$内所有点集合。
开运算也可以通过下面的式子来表示：

$$A \circ B = \bigcup\{(B)_z|(B)_z \subseteq A\}$$

基本属性：

- 对$A$开的结果是$A$的子集；
- 如$C$是$D$的子集，则$C$与$B$开的结果是$D$与$B$开运算结果的子集；
- 对同样的$A$，做多次开运算的结果与做一次是一样的。

# 闭运算的几何解释




a b c

**FIGURE 9.9** (a) Structuring element $B$ "rolling" on the outer boundary of set $A$. (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

闭运算：结构元素$B$在$A$外边界滚动，到达的最远处为闭运算的边界。

基本属性：

- $A$是对$A$闭运算结果的子集；
- 如$C$是$D$的子集，则$C$与$B$闭作用的结果是$D$与$B$闭运算结果的子集；
- 对$A$做多次闭运算的结果与做一次是一样的。

# 开、闭运算的基本作用

- 从开、闭运算的基本定义和运行过程可以看出，这两种集合操作的效果如下：
- 开运算通常对图像轮廓进行平滑，使狭窄的“地峡”形状断开，去掉细的突起、小的物体。
- 闭运算也能平滑图像的轮廓，但和开运算相反，它一般使窄的断开部位、细长的裂缝、小的洞合并。

# 开运算应用举例：消除不相关细节

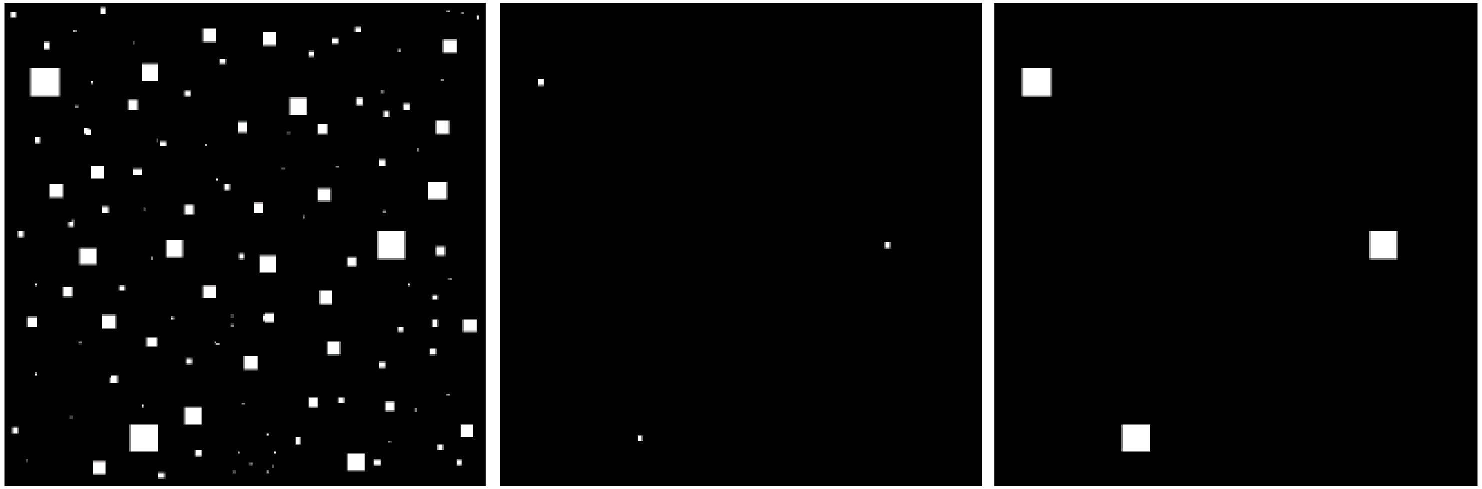
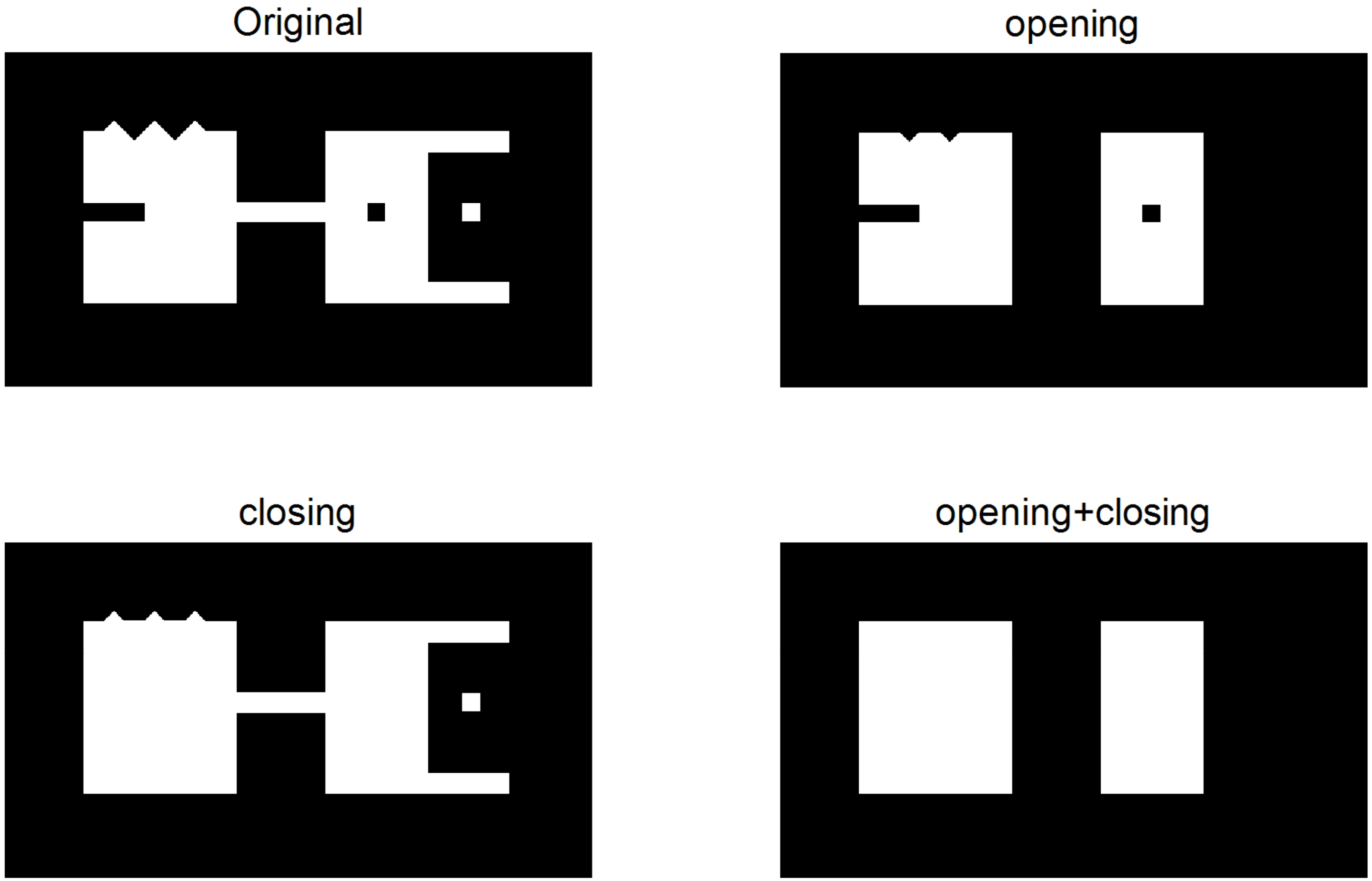

a b c

**FIGURE 9.7** (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

# 开、闭运算的应用

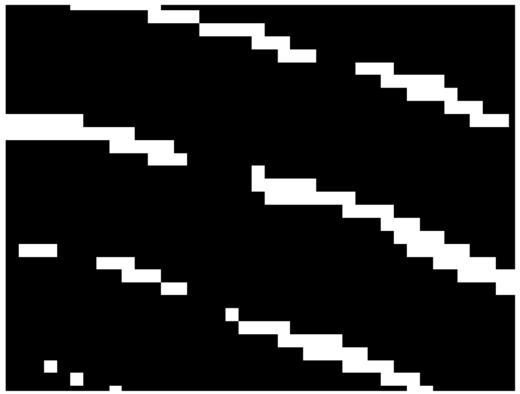


```matlab
% fig0910_OpenClose
f = imread('..\data\Fig0910(a)(shapes).tif');
se = strel('square', 20);
fo = imopen(f, se);
fc = imclose(f, se);
foc = imclose(fo, se);
figure(1),
ax(1)=subplot(2,2,1); imshow(f); title('Original');
ax(2)=subplot(2,2,2); imshow(fo); title('opening');
ax(3)=subplot(2,2,3); imshow(fc); title('closing');
ax(4)=subplot(2,2,4); imshow(foc);
title('opening+closing');
linkaxes(ax);
```

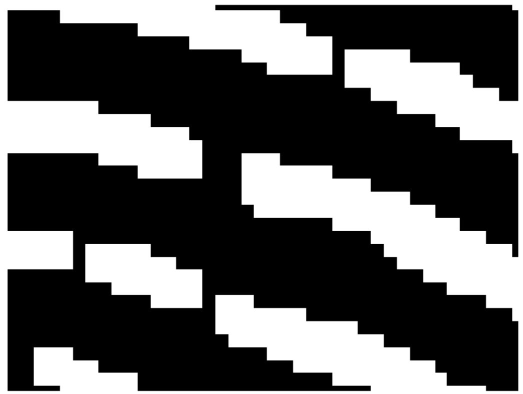
# 开、闭运算的应用

Original

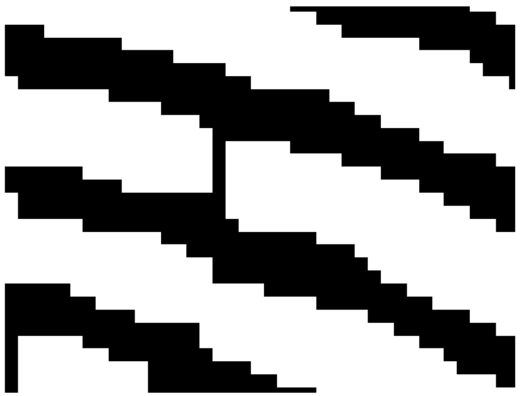
erosion of original



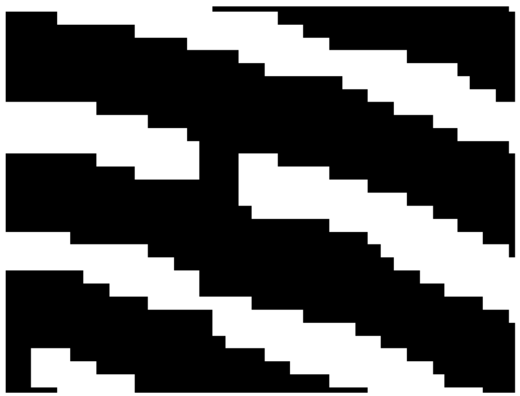
opening of original



dilation of opening



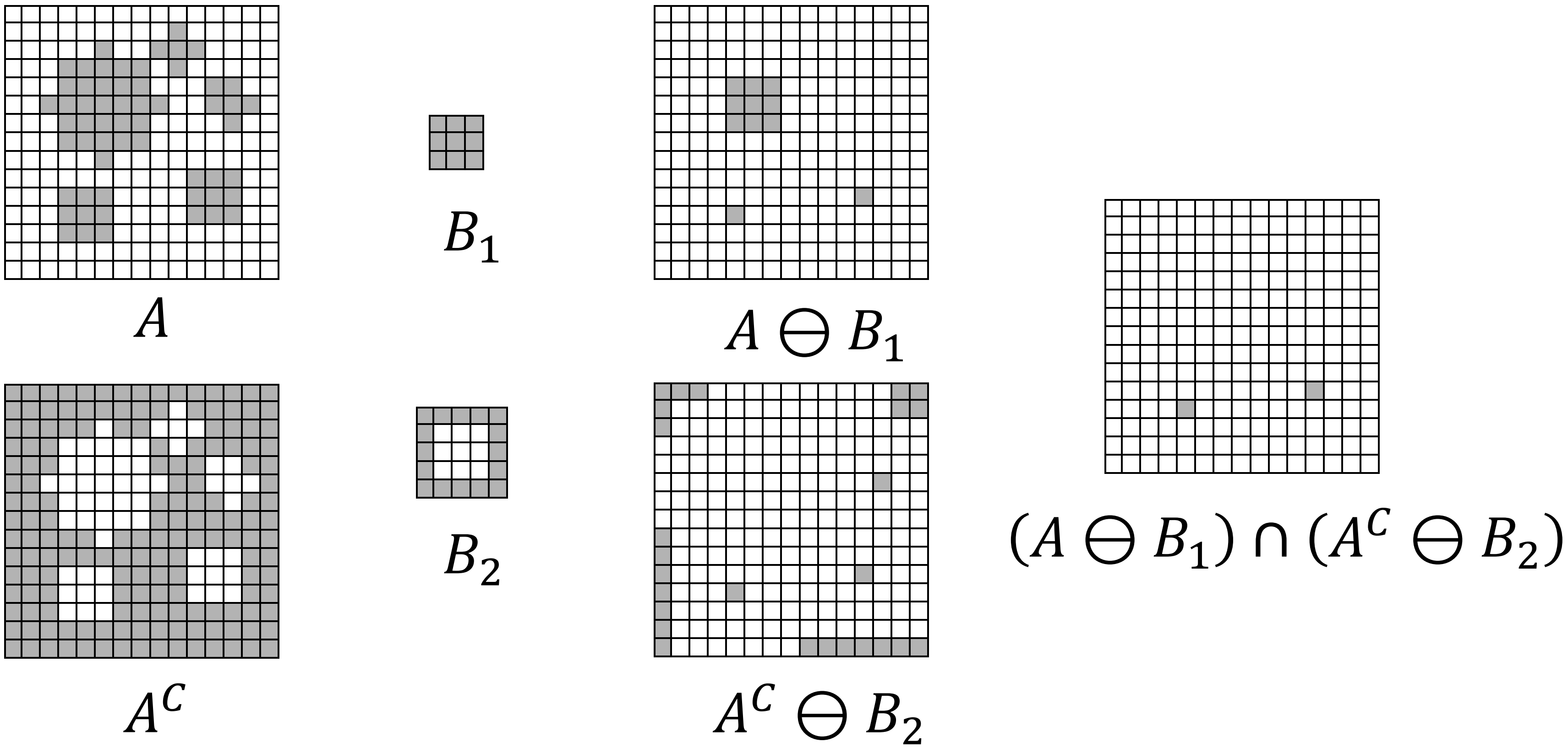
closing of opening



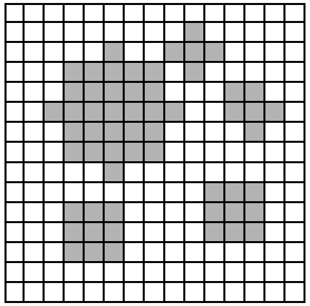
```
% ex0904_OpenClose
f = imread('..\data\Fig0911(a)(noisy_fingerprint).tif');
se = strel('square', 3);
fe = imerode(f, se);
fo = imdilate(fe, se);
fod = imdilate(fo, se);
foc = imerode(fod, se);
figure(1),
ax(1)=subplot(2,3,1); imshow(f); title('Original');
ax(2)=subplot(2,3,2); imshow(fe); title('erosion of
original');
ax(3)=subplot(2,3,3); imshow(fo); title('opening of
original');
ax(4)=subplot(2,3,4); imshow(fod); title('dilation of
opening');
ax(5)=subplot(2,3,5); imshow(foc); title('closing of
opening');
linkaxes(ax);
```

# 击中与否变换（Hit-or-miss）

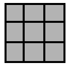
- 击中与否变换是形状检测的基本工具。
- 从图$A$中检测感兴趣形状。输出图像中，感兴趣形状的原点为1，其他处均为0
- 用2个结构元素：$B_1$（形状前景）和$B_2$（形状背景）



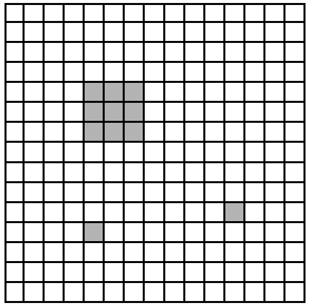
# 击中与否变换



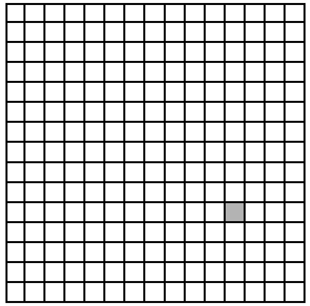
$A$



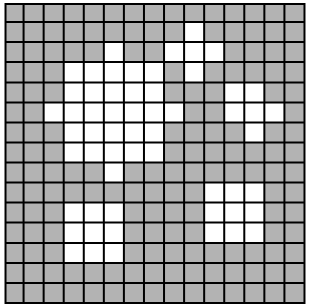
$B_1$



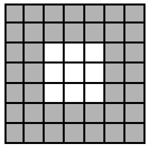
$A \ominus B_1$



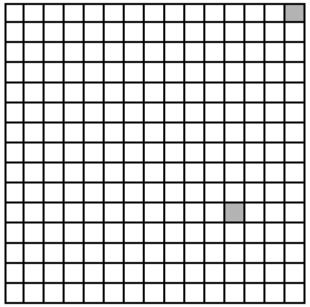
$(A \ominus B_1) \cap (A^C \ominus B_2)$



$A^C$

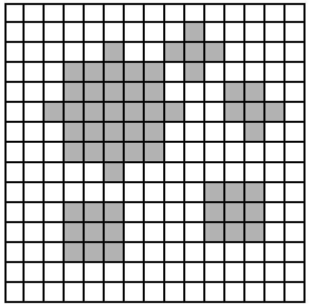


$B_2$

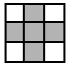


$A^C \ominus B_2$

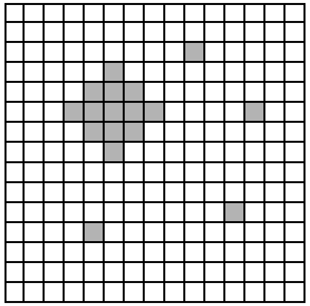
# 击中与否变换



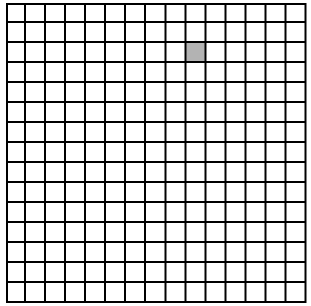
$A$



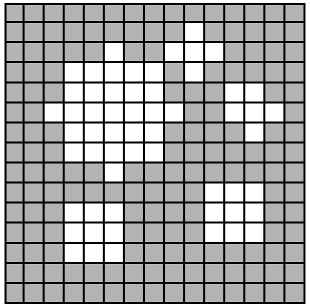
$B_1$



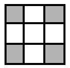
$A \ominus B_1$



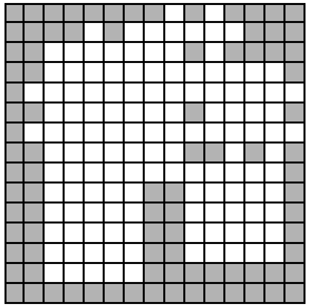
$(A \ominus B_1) \cap (A^C \ominus B_2)$



$A^C$

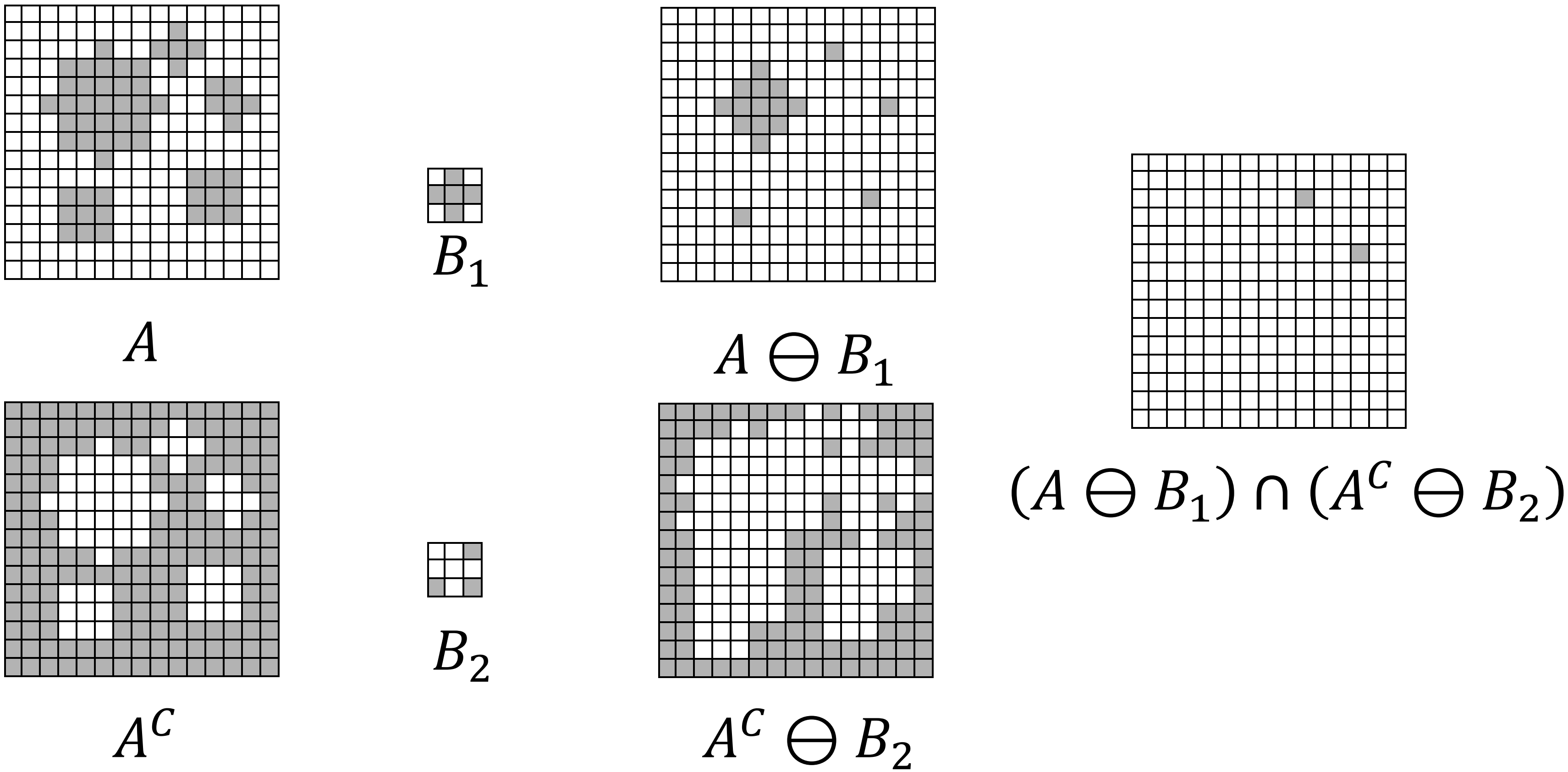


$B_2$



$A^C \ominus B_2$

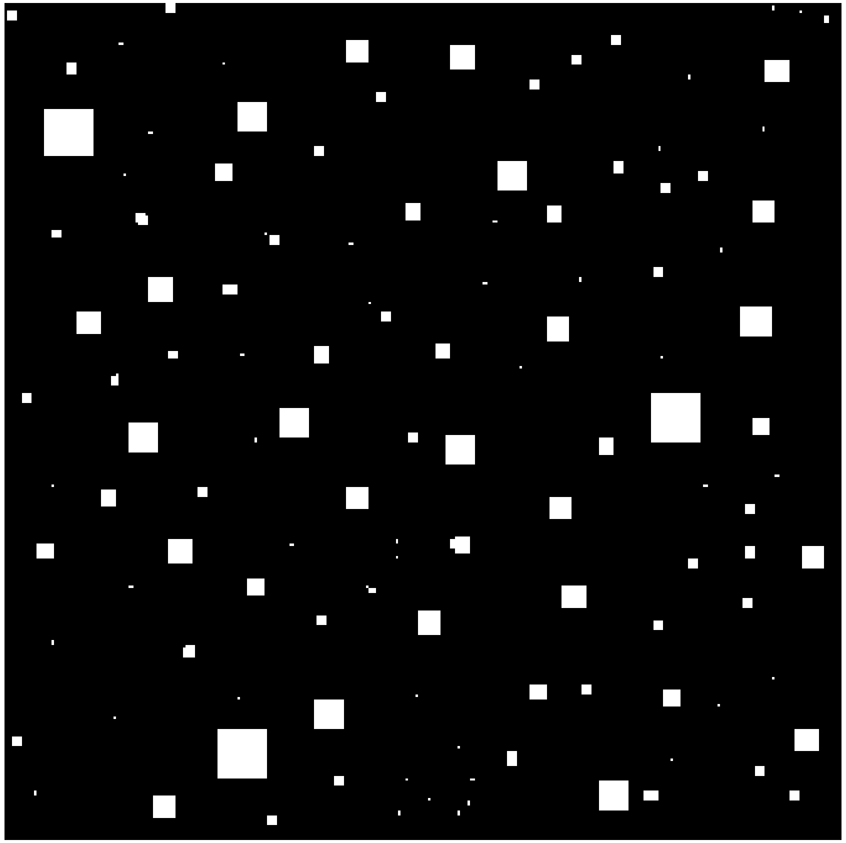
# 击中与否变换



```matlab
% hitmiss_demo2
N = 15;
[X, Y] = meshgrid(1:N);
A = false(N, N);
mask_circle = sqrt((X-6).^2+(Y-6).^2)<=3;
A(mask_circle) = 1;
A(2,10) = 1;
A(3,9:11) = 1;
A(4,10) = 1;
A(5,12:13) = 1;
A(6,12:14) = 1;
A(7,13) = 1;
mask_box1 = X>3 & X<=6 & Y>10 & Y<=13;
A(mask_box1) = 1;
mask_box2 = X>10 & X<=13 & Y>9 & Y<=12;
A(mask_box2) = 1;
close all
t = 10;% magnify
figure, imshow(MagnifyAndGrid(A, t))
switch 4
    case 1
        b1 = true(3,3); B1 = strel(b1);
        b2 = true(5,5); b2(2:end-1,2:end-1) = 0; B2 = strel(b2);
    case 2
        b1 = true(3,3); B1 = strel(b1);
        b2 = true(7,7); b2(3:end-2,3:end-2) = 0; B2 = strel(b2);
    case 3
        b1 = [0 1 0; 1 1 1; 0 1 0]; B1 = strel(b1);
        b2 = 1-b1; B2 = strel(b2);
    case 4
        b1 = [0 1 0; 1 1 1; 0 1 0]; B1 = strel(b1);
        b2 = 1-b1; b2(1,1) = 0; B2 = strel(b2);
end
figure, imshow(MagnifyAndGrid(b1, t))
figure, imshow(MagnifyAndGrid(b2, t))
Ae = imerode(A, B1);
figure, imshow(MagnifyAndGrid(Ae, t))
Ace = imerode(~A, B2);
figure, imshow(MagnifyAndGrid(~A, t))
figure, imshow(MagnifyAndGrid(Ace, t))
figure, imshow(MagnifyAndGrid(Ace&Ae, t))
```

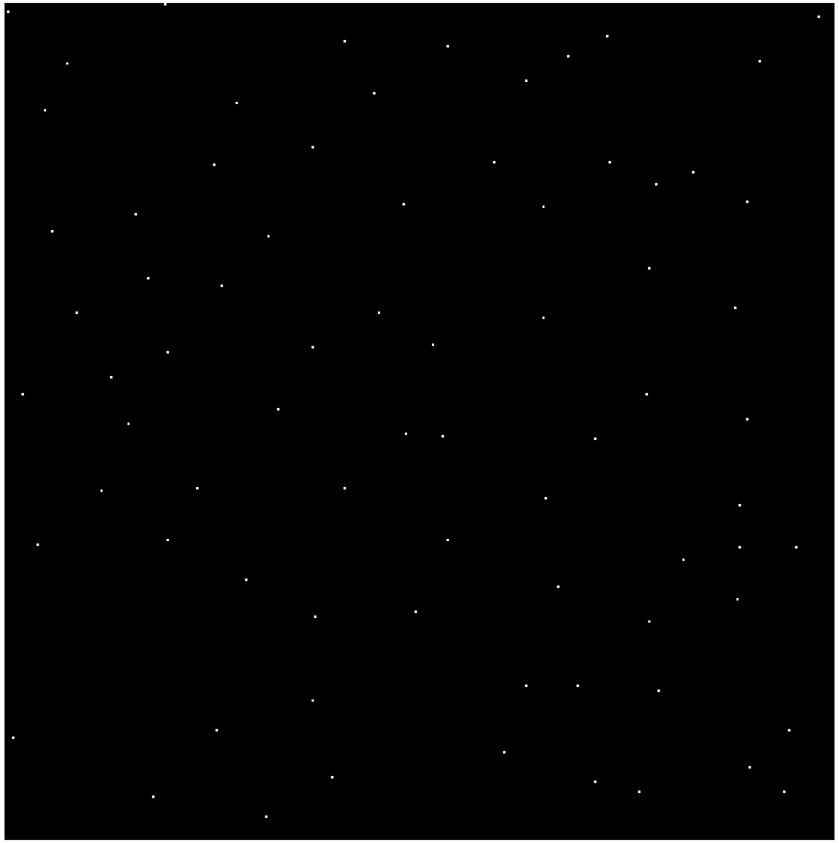
# 击中与否的应用

Original



检测白块的左上角

Corner



```matlab
% ex0905_CornerDetection
% DIPUM, p352

f = imread('..\data\FigP0918(left).tif');
B1 = strel([0 0 0; 0 1 1; 0 1 0]);
B2 = strel([1 1 1; 1 0 0; 1 0 0]);
g = bwhitmiss(f, B1, B2);

rgb = zeros(size(f,1), size(f,2), 3);
rgb(:,:,1) = f; rgb(:,:,2) = f; rgb(:,:,3) = f;
idx = find(g);
rgb(idx+numel(f)) = 0;
rgb(idx+2*numel(f)) = 0;

figure(1),
ax(1)=subplot(2,2,1); imshow(f); title('Original');
ax(2)=subplot(2,2,2); imshow(g); title('Corner');
ax(3)=subplot(2,2,3); imshow(rgb); title('Corner overlaid on Original');
linkaxes(ax);
```

# 内　容

- 数学基础
- 形态学基本运算
- 形态学算法
- 灰度形态学

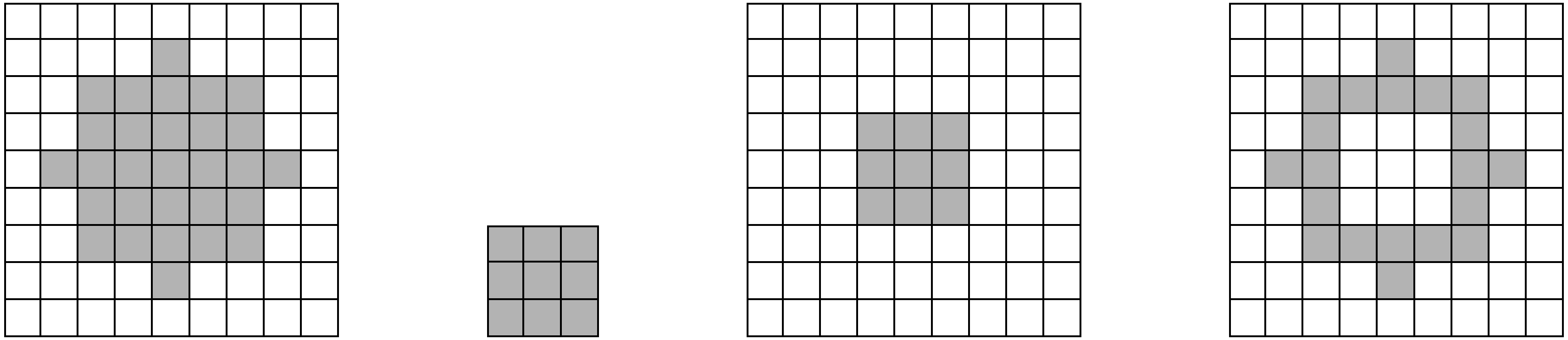
# 形态学算法

- 边界提取 (boundary extraction)
- 形态学重建 (morphological reconstruction)
- 连通分量提取 (extraction of connected components)
- 区域填充 (region filling)
- 凸包 (convex hull)
- 细化 (thinning)
- 粗化 (thickening)
- 提取骨架 (extraction of skeleton)
- 修剪 (pruning)

# 边界提取

提取集合$A$的边界$\beta(A)$：用合适的结构元素$B$对$A$腐蚀，然后用$A$减去腐蚀结果

$$\beta(A) = A - (A \ominus B)$$



$A$　　$B$　　$A \ominus B$　　$\beta(A)$

当结构元素大小为$3 \times 3$时，边界宽度为1像素

增大结构元素，边界变宽

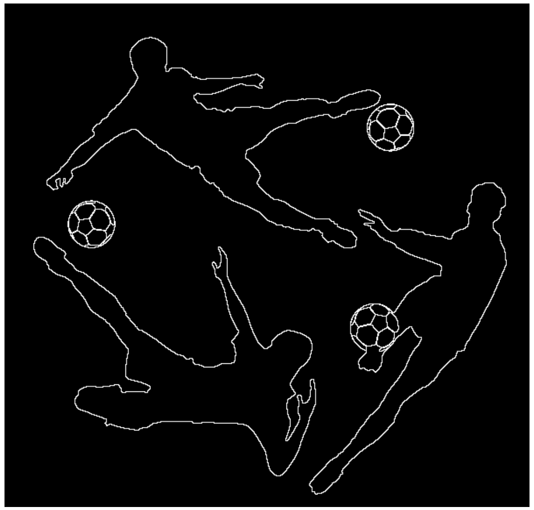
```matlab
% BoundaryDemo
N = 9;
[X, Y] = meshgrid(1:N);
A = false(N, N);
mask = sqrt((X-5).^2+(Y-5).^2)<=3;
A(mask) = 1;
B = ones(5, 5);
Ae = imerode(A, B);
Ab = A & ~Ae;
```

# 边界提取

Original

erosion of original

boundary



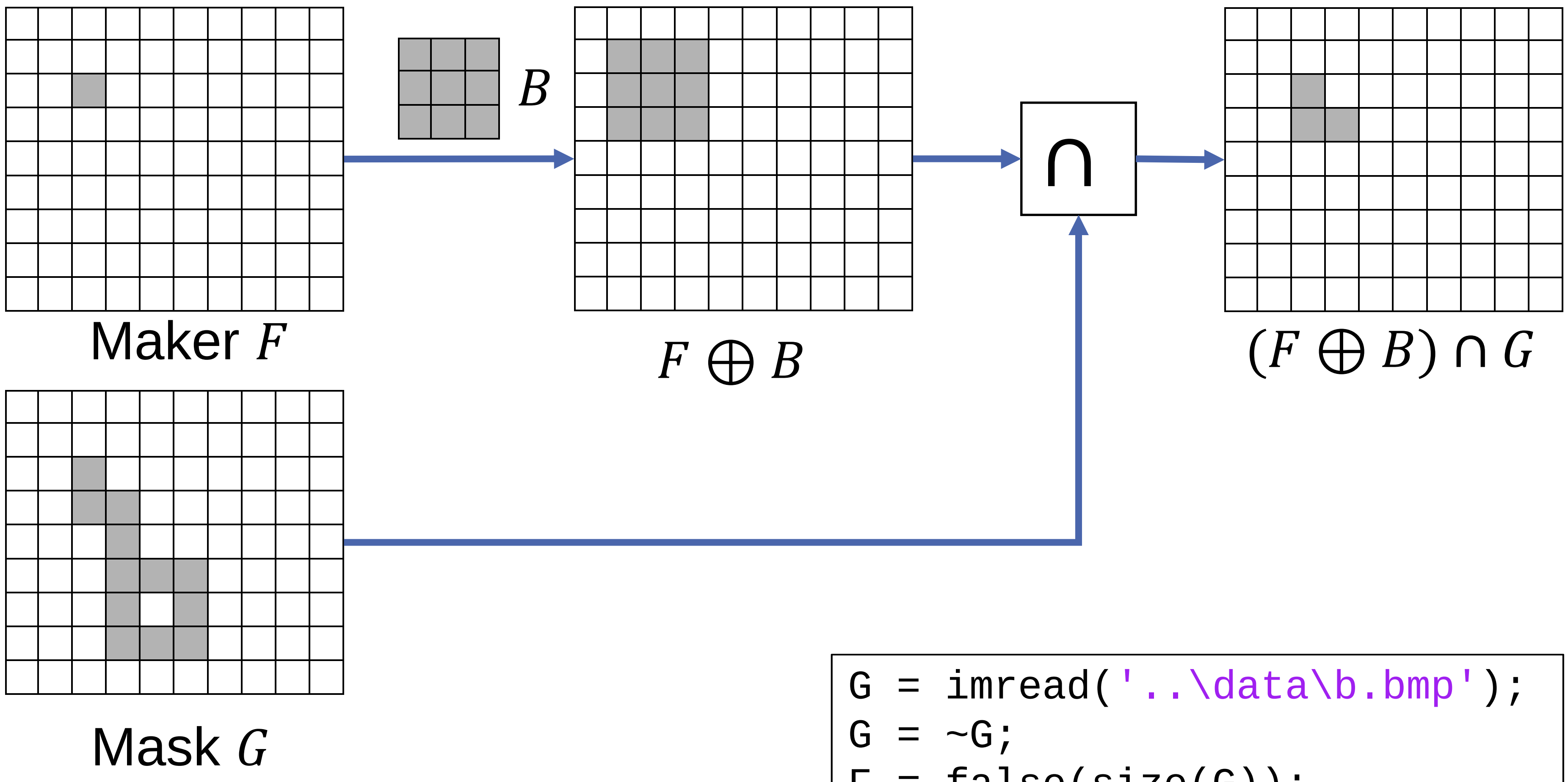
boundary overlaid on original

```matlab
function ex0914_Boundary()

f = imread('..\data\players.bmp');
fe = imerode(f, ones(3, 3));
fb = f & ~fe;

rgb = zeros(size(f,1), size(f,2), 3);
rgb(:,:,1) = f; rgb(:,:,2) = f; rgb(:,:,3) = f;
idx = find(fb);
rgb(idx) = 0;
rgb(idx+2*numel(f)) = 0;

figure(1),
ax(1)=subplot(2,2,1); imshow(f); title('Original');
ax(2)=subplot(2,2,2); imshow(fe); title('erosion of original');
ax(3)=subplot(2,2,3); imshow(fb); title('boundary');
ax(4)=subplot(2,2,4); imshow(rgb); title('boundary overlaid on original');
linkaxes(ax);
```

# 形态学重建（Morphological Reconstruction）

- 形态学重建需要2幅图像（marker $F$， mask $G$ ）和结构元素B
- 形态学重建的核心是测地学膨胀（geodesic dilation）
- 1次测地学膨胀$D_G^{(1)}(F) = (F \oplus B) \cap G$
- $n$次测地学膨胀$D_G^{(n)}(F) = D_G^{(1)}\left[D_G^{(n-1)}(F)\right]$

# 测地学膨胀

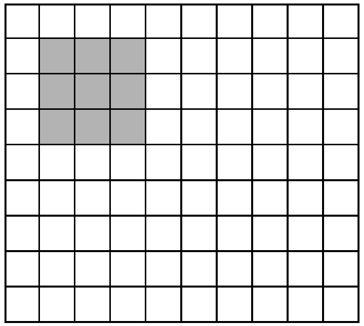



```
G = imread('..\data\b.bmp');
G = ~G;
F = false(size(G));
F(3,3) = 1;
B = ones(3,3);
Fd = imdilate(F, B);
Fgd = Fd & G;
```

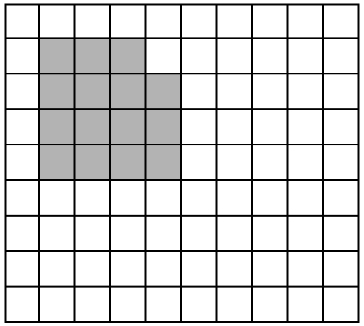
# 形态学重建

由于交集运算，测地学膨胀会最终收敛。

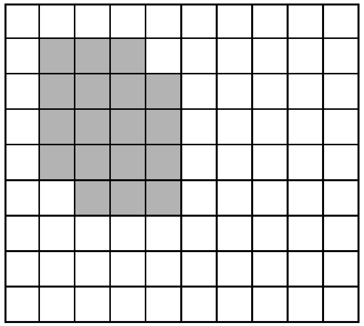
收敛时的结果就是其形态学重建结果$R_G^D(F) = D_G^{(k)}(F)$



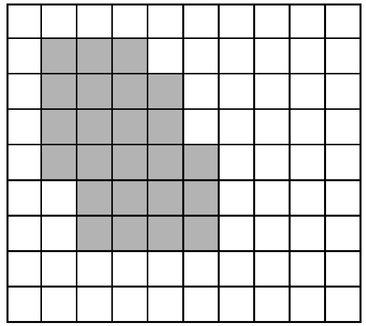
$F \oplus B$



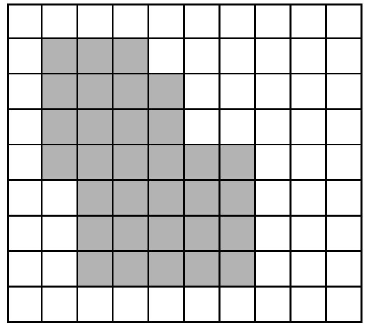
$D_G^{(1)}(F) \oplus B$



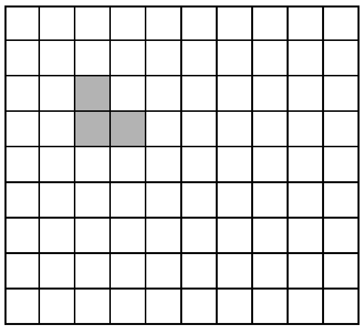
$D_G^{(2)}(F) \oplus B$



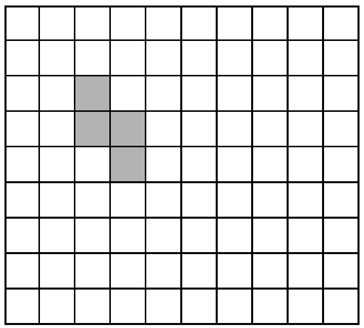
$D_G^{(3)}(F) \oplus B$



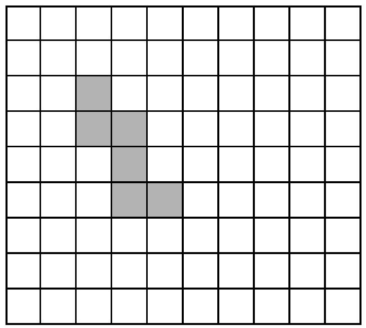
$D_G^{(4)}(F) \oplus B$



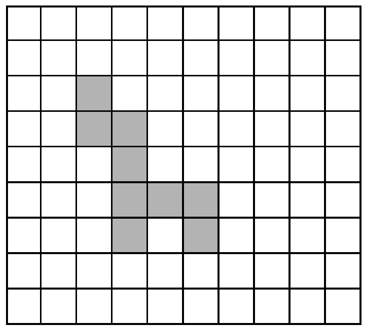
$D_G^{(1)}(F)$



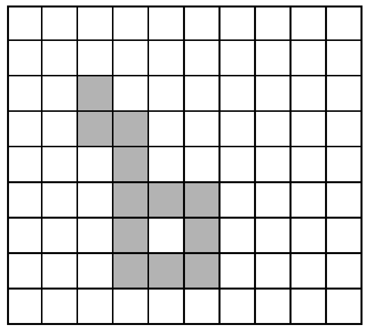
$D_G^{(2)}(F)$



$D_G^{(3)}(F)$



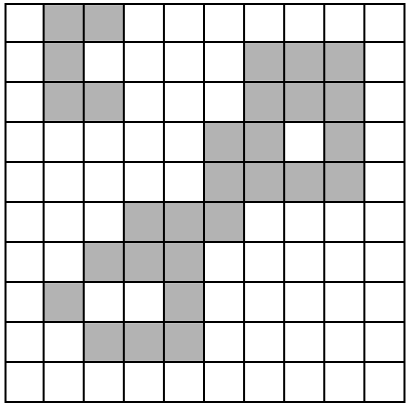
$D_G^{(4)}(F)$



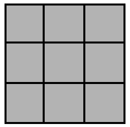
$D_G^{(5)}(F) = D_G^{(6)}(F)$

# 连通成分提取

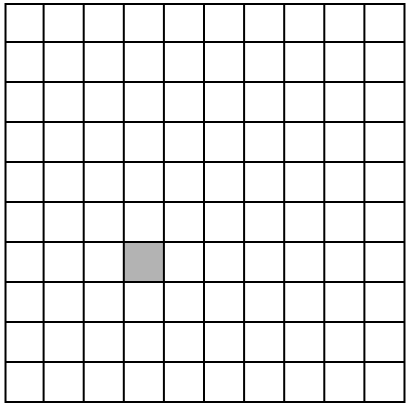
- 提取连通成分在许多图像分析任务中的重要步骤。
- 集合$A$有2个连通成分（8连通）或者3个连通成分（4连通）。
- 给定初始点，用形态学重建可提取包含该点的连通成分。



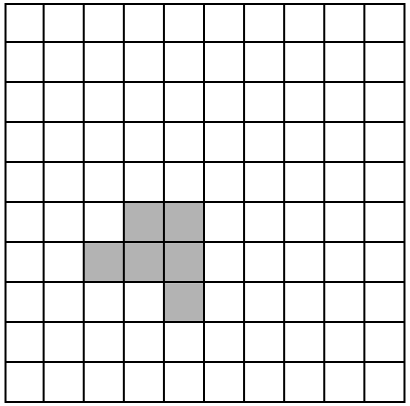
$A$（mask $G$）



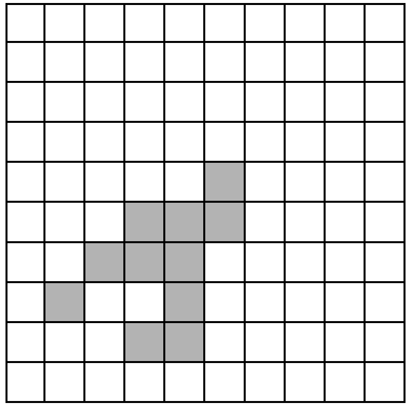
$B$



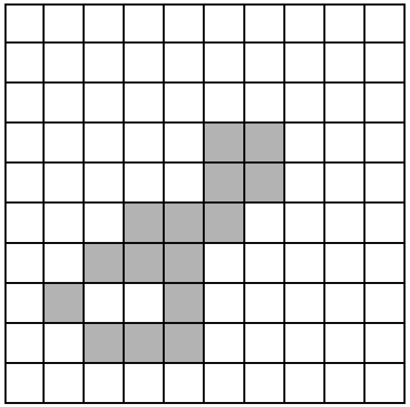
初始点（marker $F$）



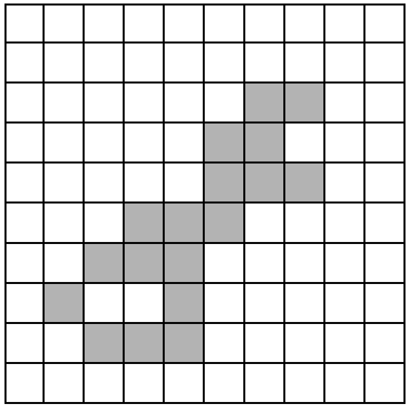
$D_G^{(1)}(F)$



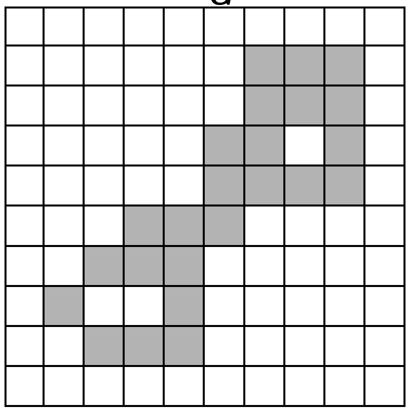
$D_G^{(2)}(F)$



$D_G^{(3)}(F)$

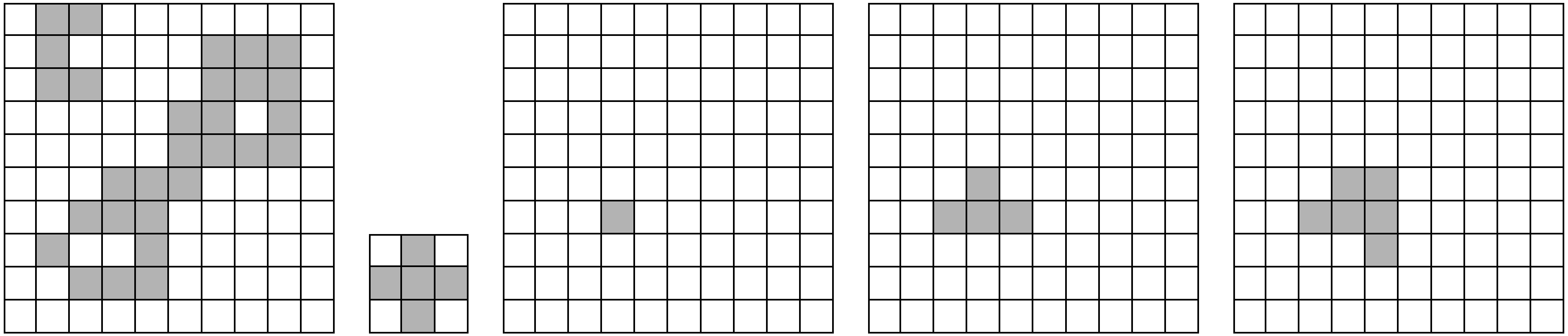


$D_G^{(4)}(F)$

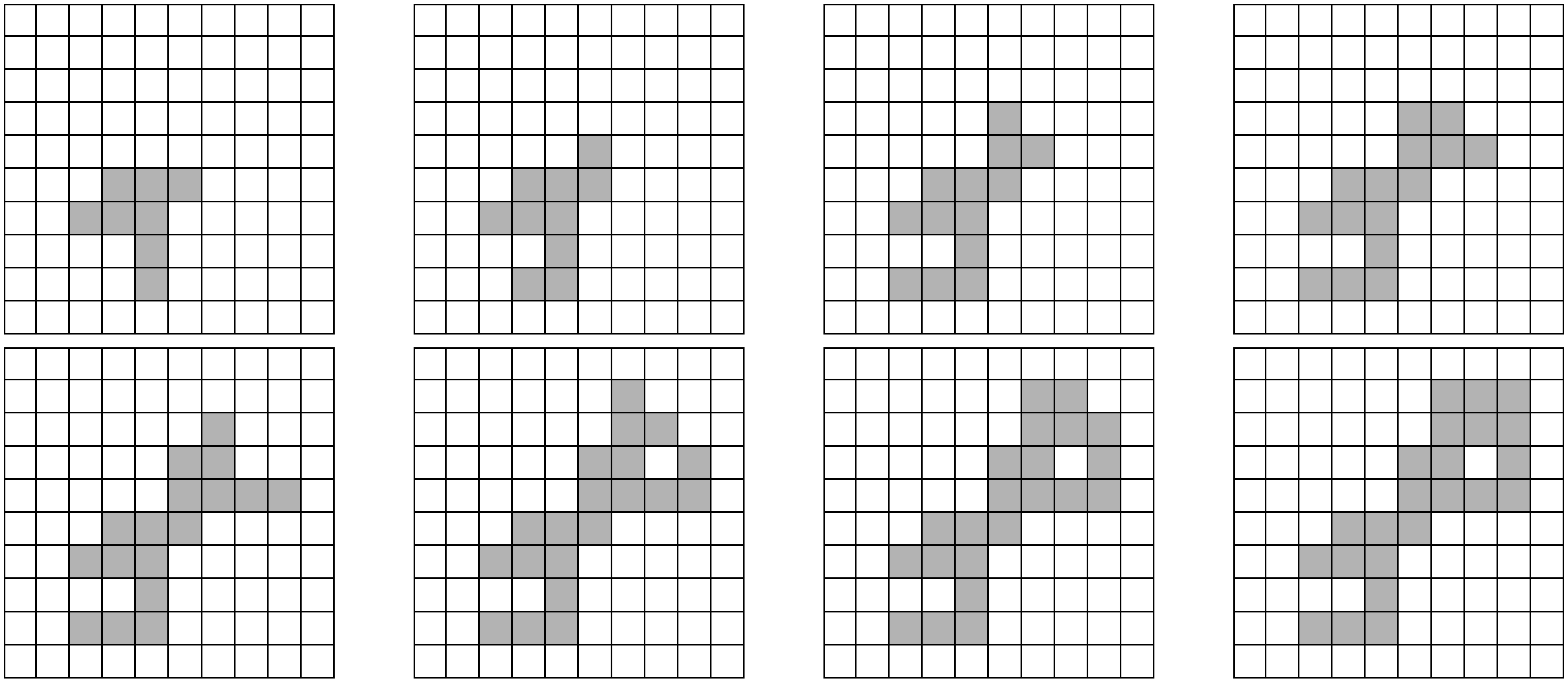


$D_G^{(5)}(F)$

# 提取4连通成分

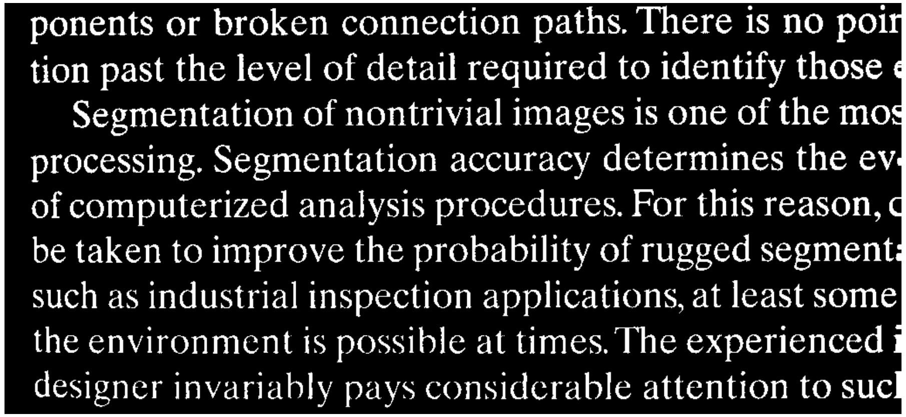


$A$（mask $G$）　$B$　初始点（marker $F$）



# 删除边界物体

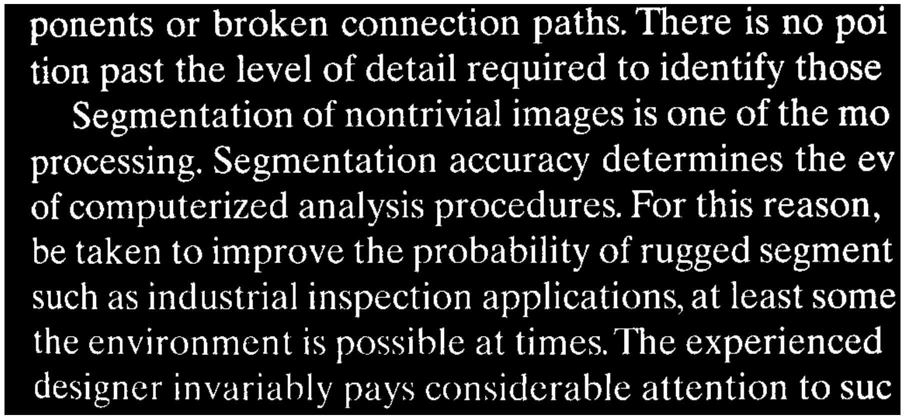
- 这其实是个连通成分提取的问题。
- 前面例子只有一个初始像素，只提取一个连通成分。而下面的算法可提取多个连通成分。


ponents or broken connection paths. There is no poir
tion past the level of detail required to identify those e
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason, c
be taken to improve the probability of rugged segment:
such as industrial inspection applications, at least some
the environment is possible at times. The experienced i
designer invariably pays considerable attention to sucl

Original

Marker

Border letters


ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mo
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason,
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced
designer invariably pays considerable attention to suc

Border cleared

```matlab
% fig0932_BorderClear
% DIP, P683

f = imread('..\data\Fig0929(a)(text_image).tif');
marker = f;
marker(2:end-1,2:end-1) = 0;
g = imreconstruct(marker, f);

close all
figure,
ax(1) = subplot(2,2,1); imshow(f); title('Original');
ax(3) = subplot(2,2,2); imshow(marker); title('Marker');
ax(3) = subplot(2,2,3); imshow(g); title('Border letters');
ax(4) = subplot(2,2,4); imshow(f & ~g); title('Border
cleared');
linkaxes(ax);
```

# 提取全部连通成分

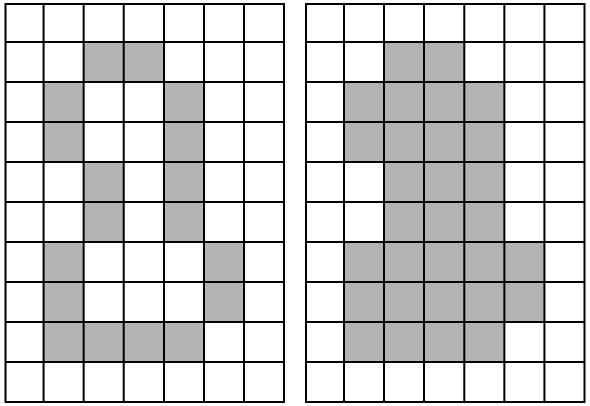
a
b
c d

**FIGURE 9.18** (a) X-ray image of chicken filet with bone fragments. (b) Thresholded image. (c) Image eroded with a 5 × 5 structuring element of 1's. (d) Number of pixels in the connected components of (c). (Image courtesy of NTB Elektronische Geraete GmbH, Diepholz, Germany, www.ntbxray.com.)

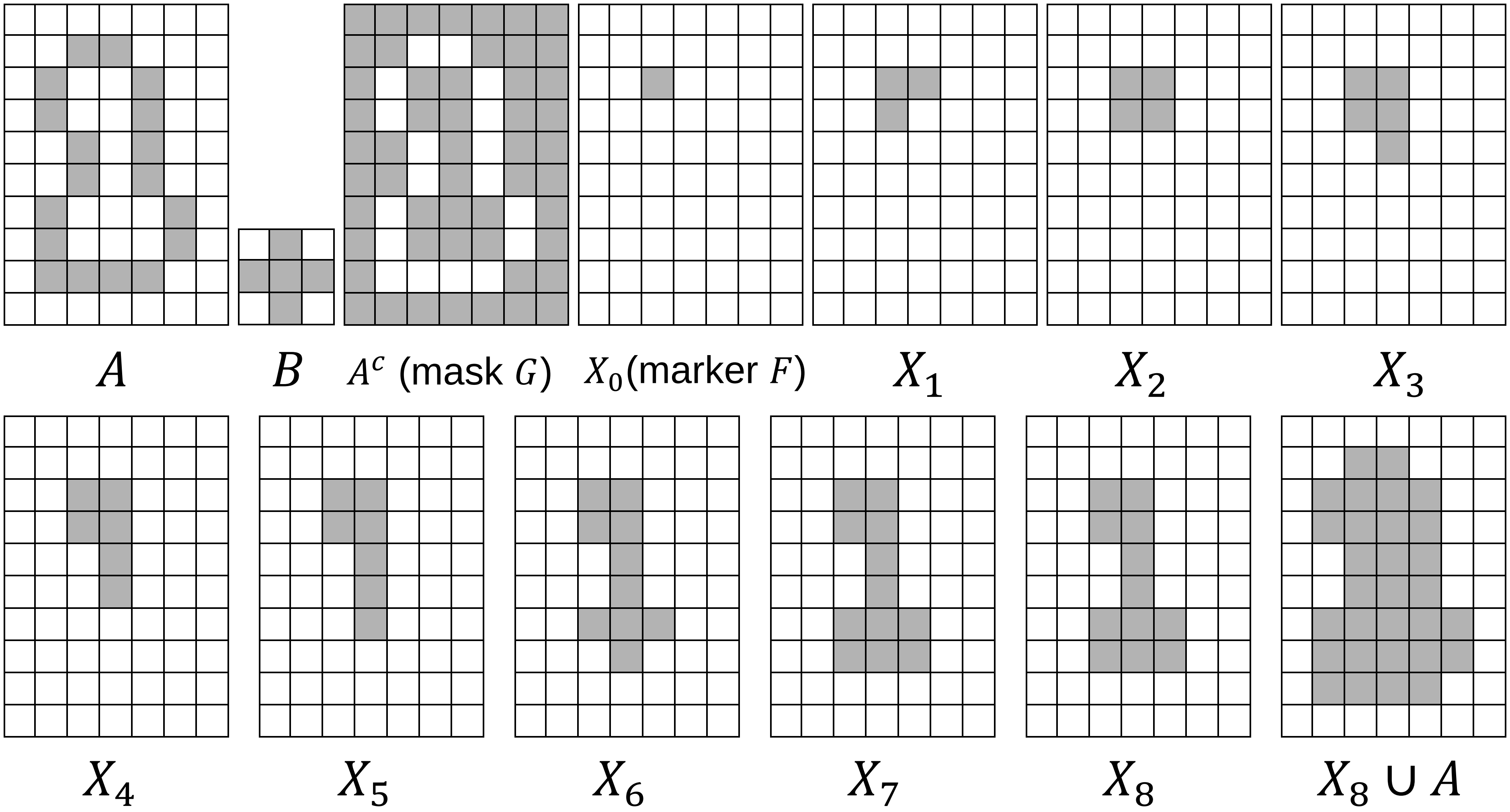
| Connected component | No. of pixels in connected comp |
|---|---|
| 01 | 11 |
| 02 | 9 |
| 03 | 9 |
| 04 | 39 |
| 05 | 133 |
| 06 | 1 |
| 07 | 1 |
| 08 | 743 |
| 09 | 7 |
| 10 | 11 |
| 11 | 11 |
| 12 | 9 |
| 13 | 9 |
| 14 | 674 |
| 15 | 85 |

# 区域填充

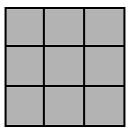
- 区域填充类似于提取反色图像中的连通成分



# 区域填充



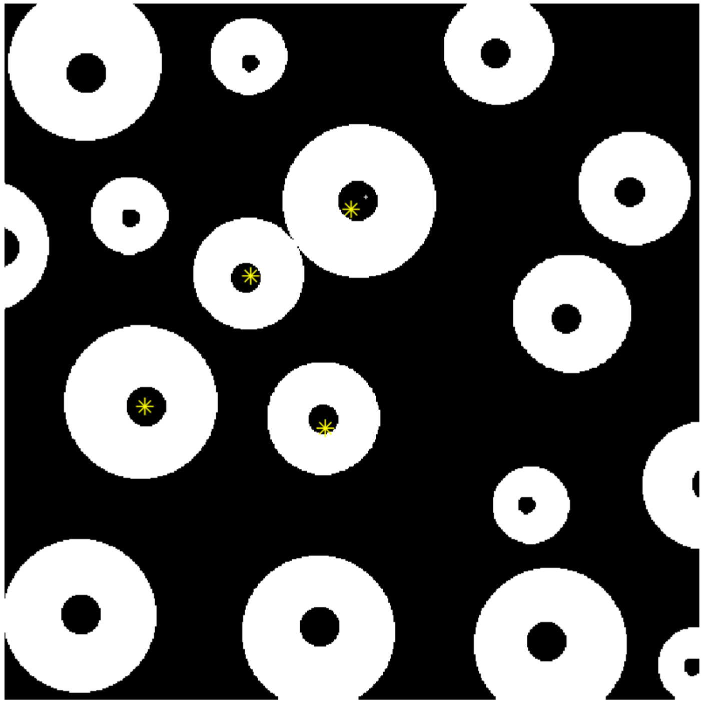

迭代收敛

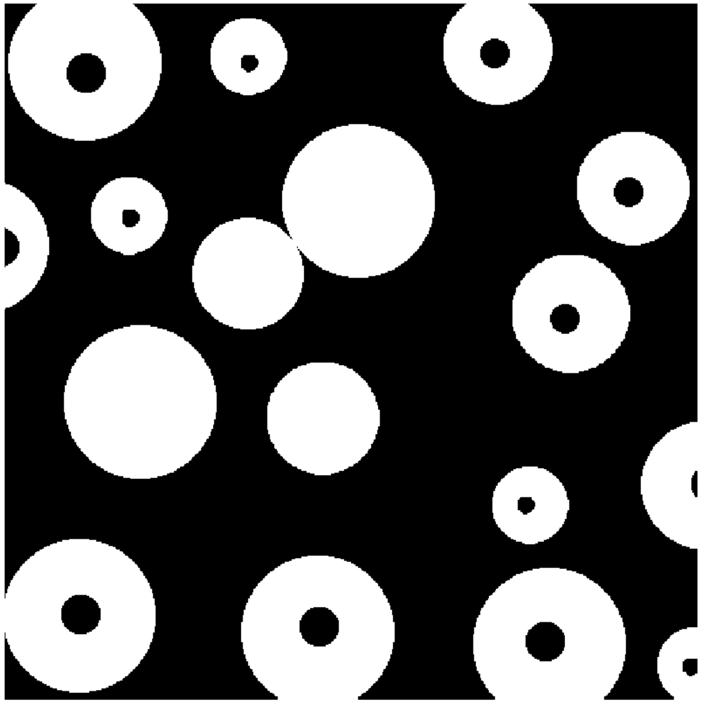
如果结构元素$B$用 ，结果如何？

```
function HoleFillDemo()
A = false(10, 7);
A_set = [2 3; 2 4; 3 2; 3 5; 4 2; 4 5; 5 3; 5 5; 6 3; 6 5; ...
    7 2; 7 6; 8 2; 8 6; 9 2; 9 3; 9 4; 9 5];
A(sub2ind(size(A),A_set(:,1),A_set(:,2))) = 1;
p = [3 3];
X = HoleFill(A, p);
Af = A | X;
```

# 区域填充

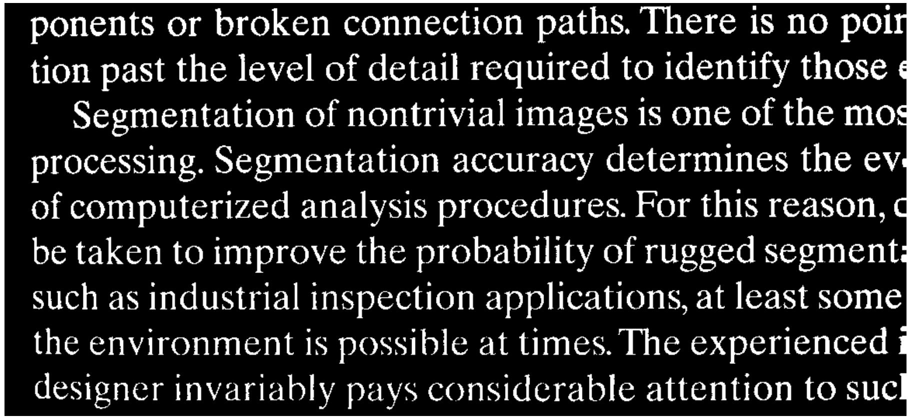
消除球体二值扫描图像中心由于镜面反射造成的中心黑色区域



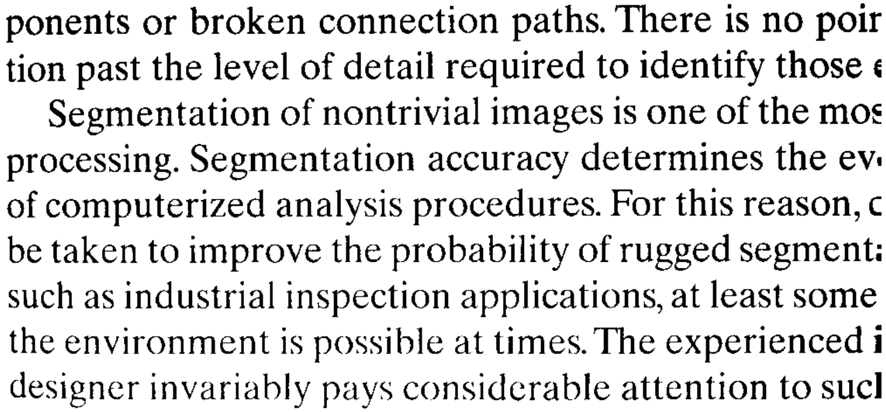


```
function HoleFillExample()

A = imread('..\data\Fig0916(a)(region-filling-reflections).tif');
figure(1),imshow(A);
[x, y] = ginput;
figure(1),hold on,plot(x,y,'y*');
X = HoleFill(A, round([y x]));
figure(2),imshow(X);
figure(3),imshow(X|A);
```

# 自动区域填充

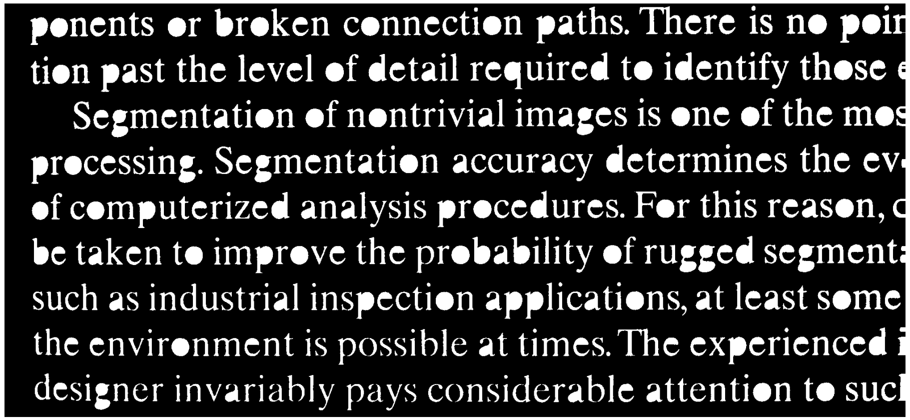



Original




Mask

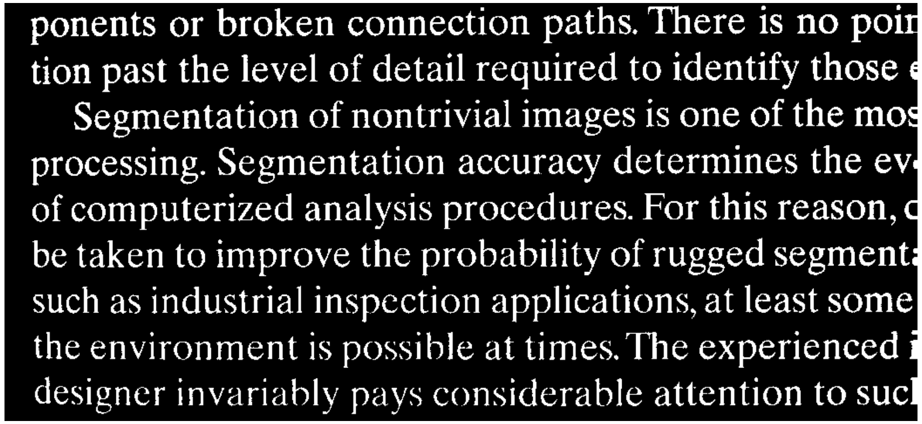
Marker



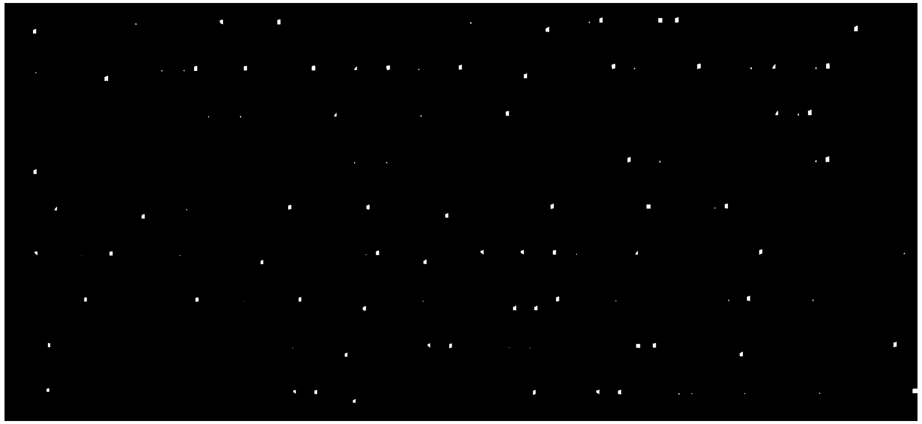

Holes filled

```matlab
% fig0931_FillHoles
% DIP, P683

f = imread('..\data\Fig0931(a)(text_image).tif');
marker = ~f;
marker(2:end-1,2:end-1) = 0;
g = imreconstruct(marker, ~f);

close all
figure,
ax(1) = subplot(2,2,1); imshow(f); title('Original');
ax(2) = subplot(2,2,2); imshow(~f); title('Mask');
ax(3) = subplot(2,2,3); imshow(marker); title('Marker');
ax(4) = subplot(2,2,4); imshow(~g); title('Holes filled');
linkaxes(ax);
```

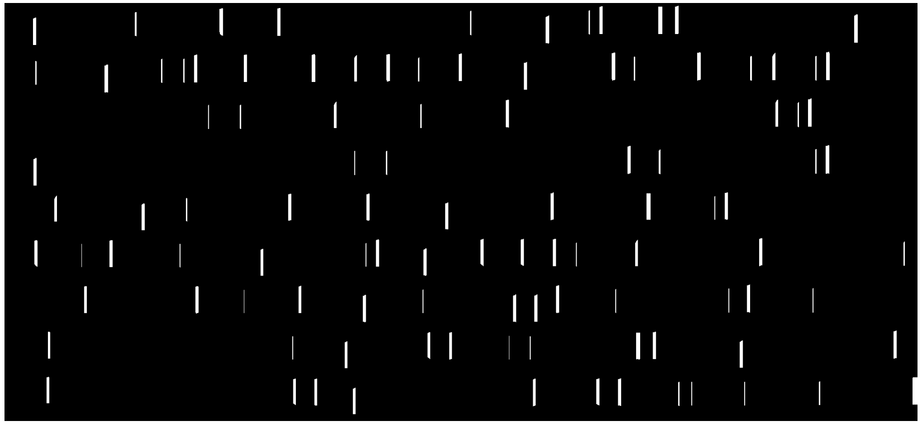
# 重建开（Opening by reconstruction）



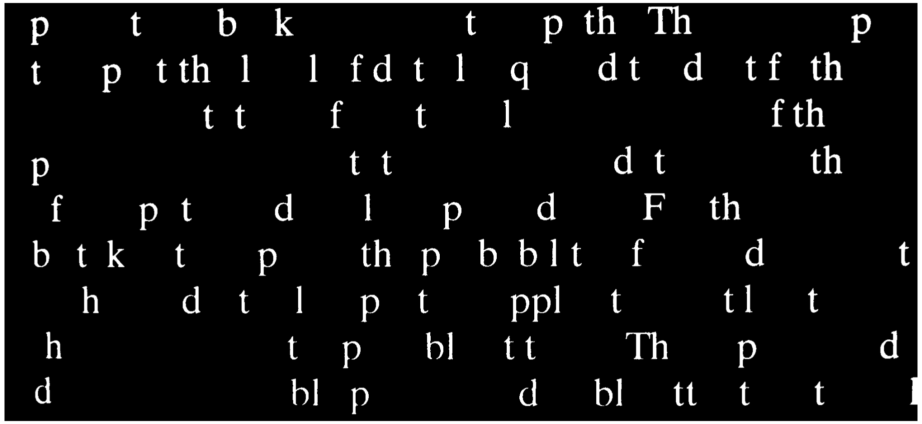

Original



Erosion



Opening

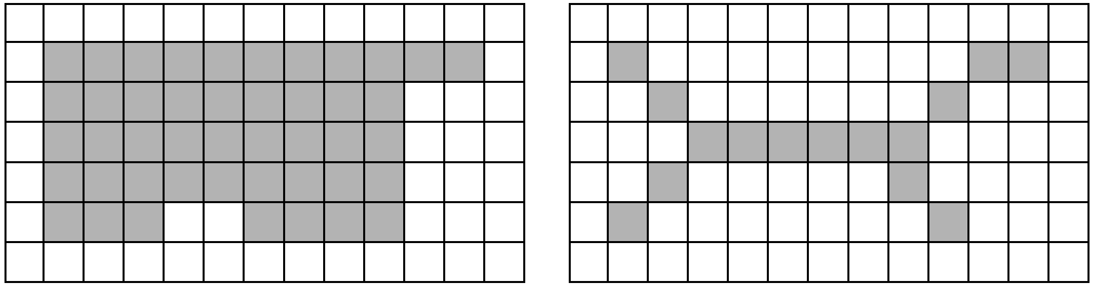


Opening by reconstruction

结构元素为51 × 1像素

```matlab
% fig0929_OpenningByReconstruction
% DIP, P681

f = imread('..\data\Fig0929(a)(text_image).tif');
fe = imerode(f, ones(51, 1));
fo = imopen(f, ones(51, 1));
fobr = imreconstruct(fe, f);

close all
figure,
ax(1) = subplot(2,2,1); imshow(f); title('Original');
ax(2) = subplot(2,2,2); imshow(fe); title('Erosion');
ax(3) = subplot(2,2,3); imshow(fo); title('Opening');
ax(4) = subplot(2,2,4); imshow(fobr); title('Opening by reconstruction');
linkaxes(ax);
```

# 细化算法



使用结构元素$B$对集合$A$进行细化，记为$A \otimes B$，定义如下：

$$A \otimes B = A - (A \circledast B)$$

其中$A \circledast B$是hit-or-miss运算。

实际中，对$A$细化是由一系列结构元素$\{B^1, B^2, \dots, B^n\}$完成的。

$$A \otimes \{\boldsymbol{B}\} = ((\dots((A \otimes B^1) \otimes B^2) \otimes B^3) \dots) \otimes B^n$$

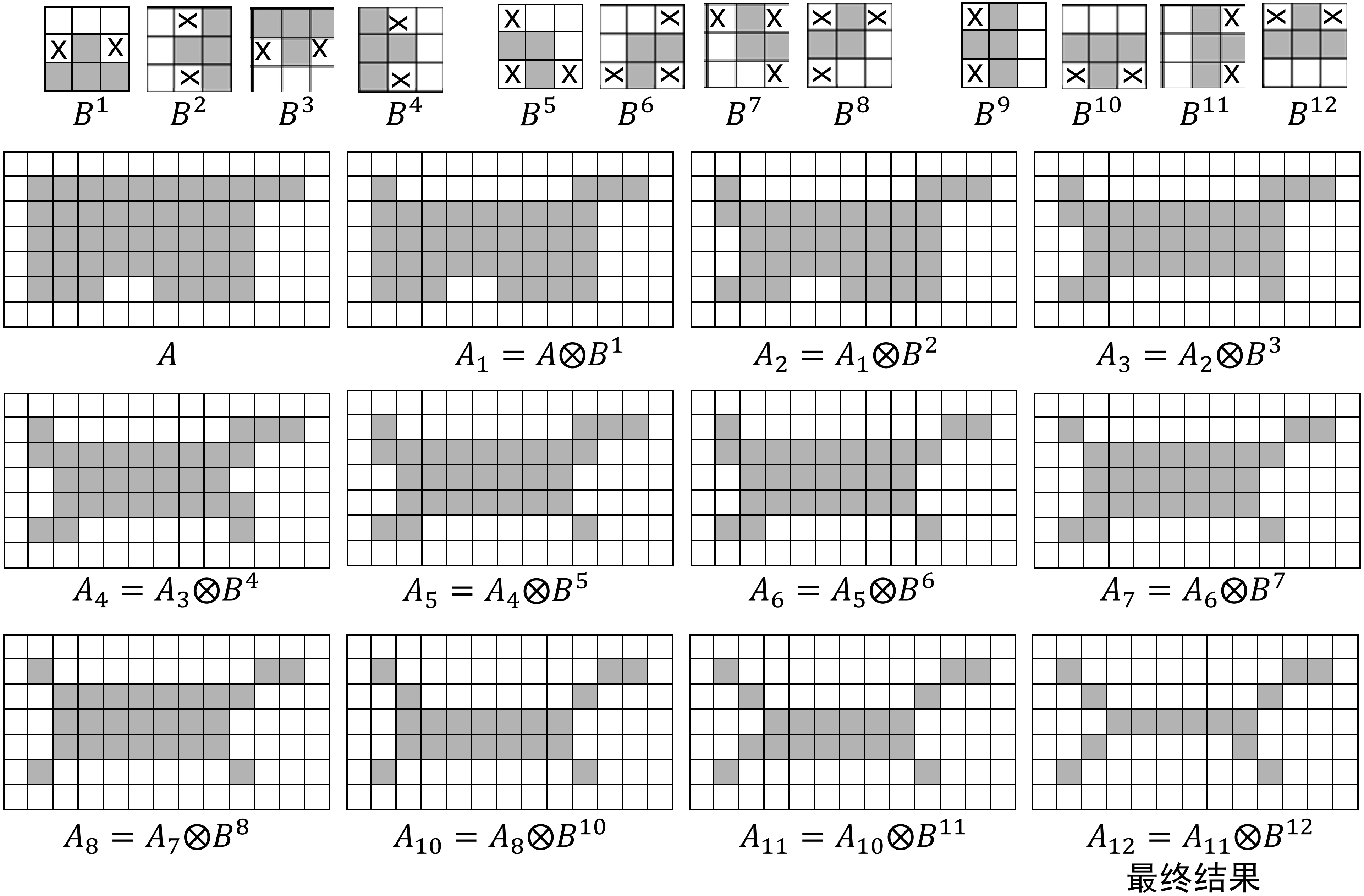
上述运算为一次迭代。

如果某次迭代，集合都没有发生变化，那么算法收敛。

注意：在用某一个结构元素对集合$A$做细化时，不能立刻删除元素。需要做完hit-or-miss运算，检测出所有元素后，再一并从$A$中删除。

这就像空域滤波中的邻域运算，不能直接修改输入图像。否则相邻像素的运算影响。

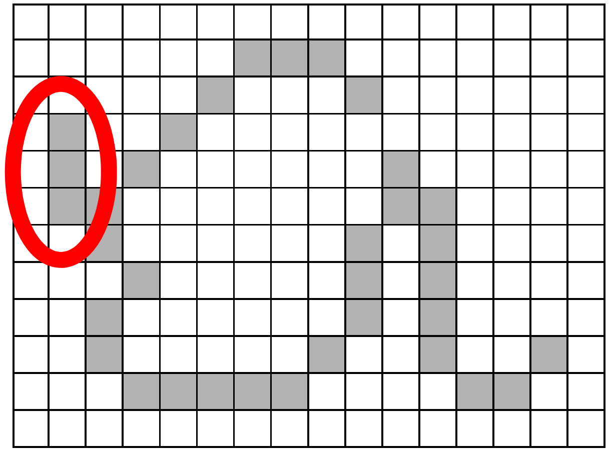
# 细化过程


$B^1$ $B^2$ $B^3$ $B^4$ $B^5$ $B^6$ $B^7$ $B^8$ $B^9$ $B^{10}$ $B^{11}$ $B^{12}$
$A$
$A_1 = A \otimes B^1$
$A_2 = A_1 \otimes B^2$
$A_3 = A_2 \otimes B^3$
$A_4 = A_3 \otimes B^4$
$A_5 = A_4 \otimes B^5$
$A_6 = A_5 \otimes B^6$
$A_7 = A_6 \otimes B^7$
$A_8 = A_7 \otimes B^8$
$A_{10} = A_8 \otimes B^{10}$
$A_{11} = A_{10} \otimes B^{11}$
$A_{12} = A_{11} \otimes B^{12}$
最终结果

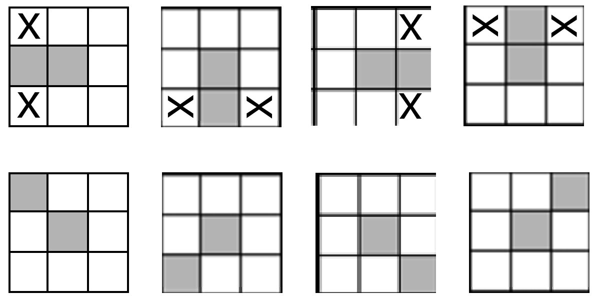
# 细化算法

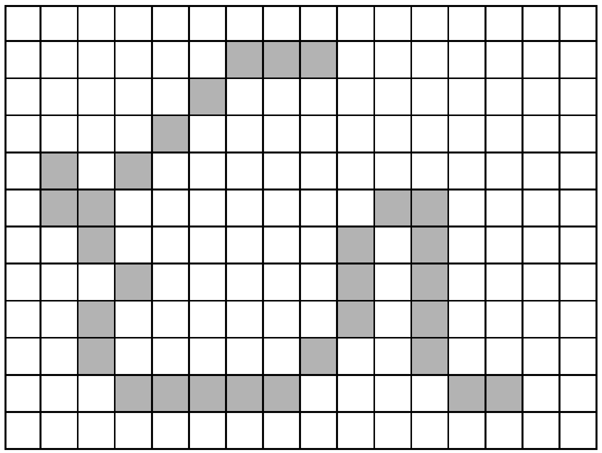
- 前面讲的与Gonzalez书上的不一样，与MATLAB图像处理工具箱的算法也不一样
- 如果结构元素不同或顺序不同，细化的结果未必一样
- Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. "Thinning methodologies-a comprehensive survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.9 (1992): 869-885.
- 这篇细化算法的综述论文引用了上百篇论文
- 没有一个最优的细化算法，可以到处通用

# 修剪算法

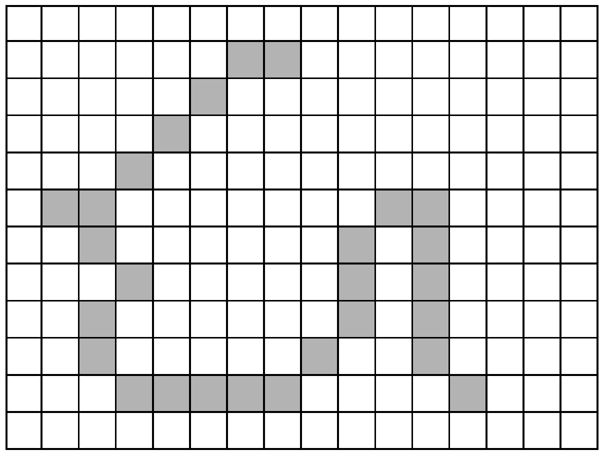


- 细化算法通常会产生一些毛刺，影响后续的分析算法
- 可用修剪算法去除毛刺
- 基本假设：毛刺的长度不超过$L$像素
- 方法：迭代删除端点；迭代$L$次
- 删除端点通过细化操作实现：$X_1 = A \otimes \{\boldsymbol{B}\}$，其中$\{\boldsymbol{B}\}$是检测端点的结构元素集合，共8个结构元素

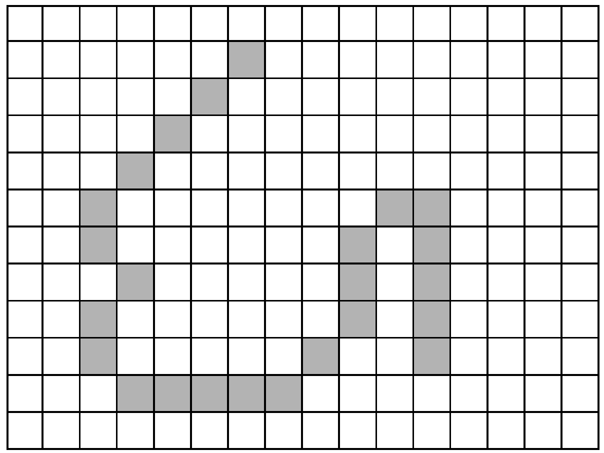




细化1次

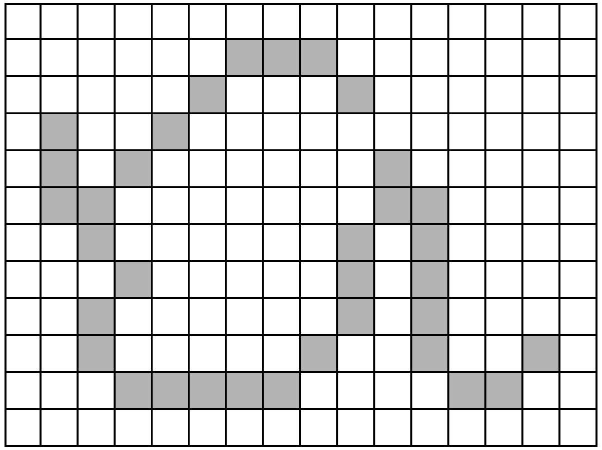


细化2次

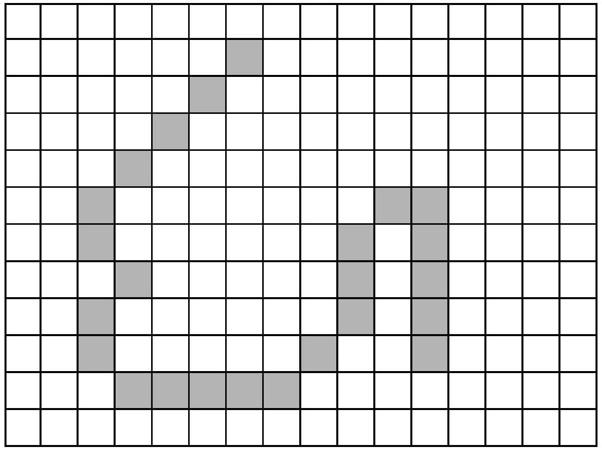


细化3次

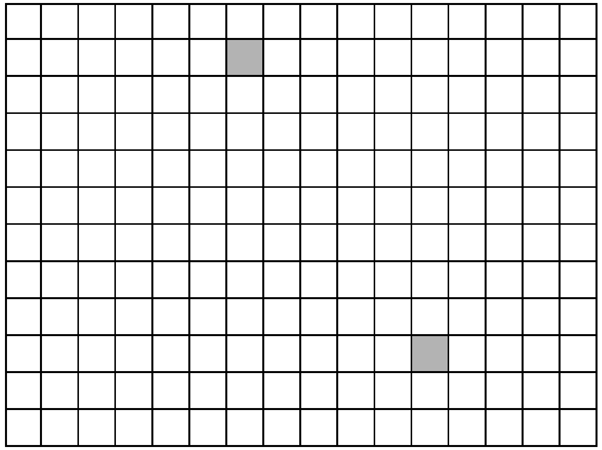
# 修剪算法



原始图像$A$

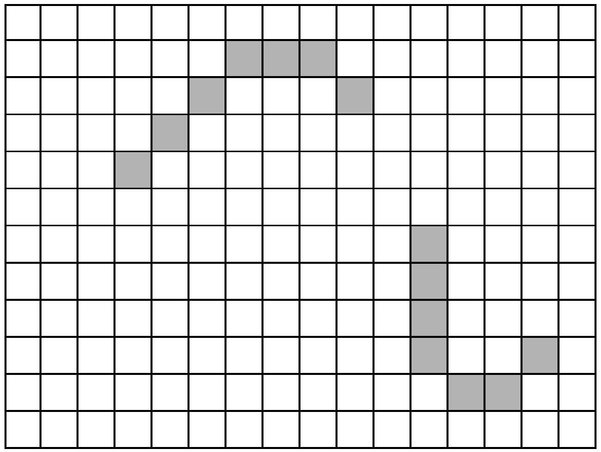


$X_1$



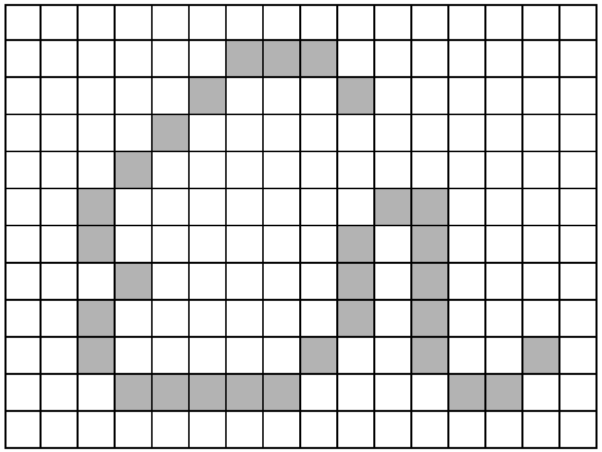
$X_1$的端点

$$X_2=\bigcup_{k=1}^{8}\left(X_1 \circledast B^k\right)$$



对$X_2$条件膨胀3次

$$X_3=(X_2 \oplus H)\cap A$$



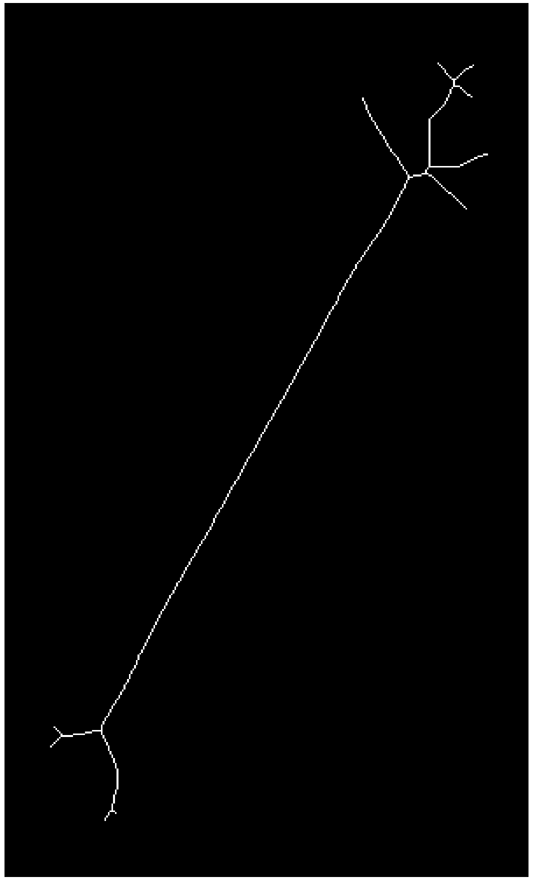
$$X_4 = X_1 \cup X_3$$

# 查表法加速（Lookup Table, LUT）

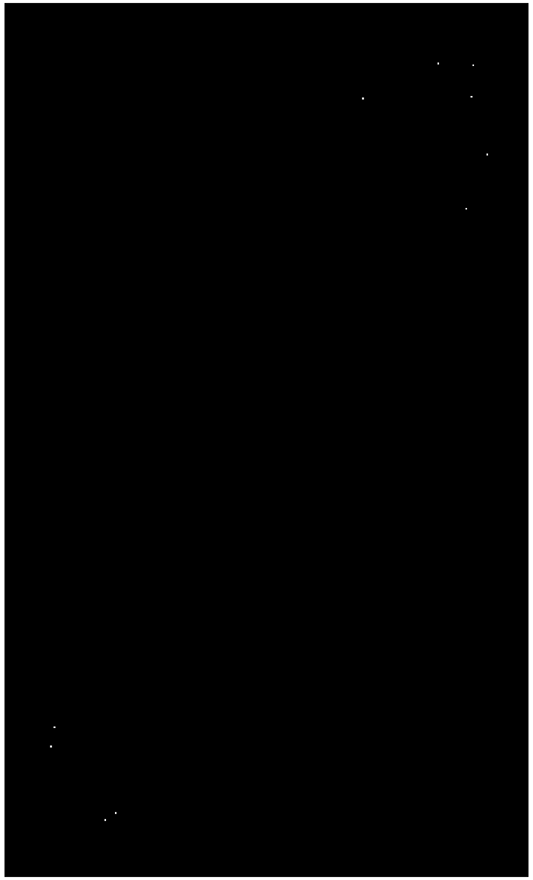
- 前面的多个算法均用到了hit-or-miss运算
- 当结构元素不大时，hit-or-miss运算可用查表法加速
- 例如细化算法、修剪算法中的各种SE是$3 \times 3$的，适合用查表法实现

# 检测端点

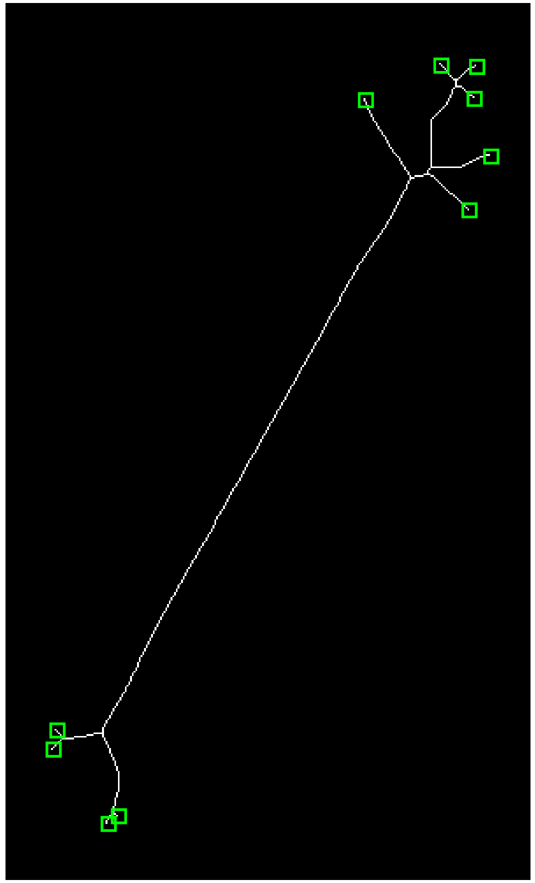
Original



Endpoints



Endpoints overlaid on Original



```matlab
% Endpoints_test
% DIPUM, p354
% Jianjiang Feng
% 2016-11-18
f = imread('..\data\Fig0914(a)(bone-skel).tif');
g = Endpoints(f);
figure(1),
ax(1)=subplot(1,3,1); imshow(f); title('Original');
ax(2)=subplot(1,3,2); imshow(g); title('Endpoints');
ax(3)=subplot(1,3,3); imshow(f); title('Endpoints overlaid on Original');
[row, col] = find(g);
hold on, plot(col,row,'gs','MarkerSize',10)
linkaxes(ax);
```

```matlab
function g = Endpoints(f)
%Endpoints: Detect endpoints in a binary image
% fig0914_Endpoint
% DIPUM, p354
% Jianjiang Feng
% 2016-11-18
persistent lut
if isempty(lut)% perform only once
    lut = makelut(@endpoint_fcn, 3);
end
g = bwlookup(f, lut);

%----------------------
function is_endpoint = endpoint_fcn(nhood)
% nhood is 3x3 binary neighborhood
is_endpoint = nhood(2,2) && (sum(nhood(:))==2);
```

# 内　容

- 数学基础
- 形态学基本运算
- 形态学算法
- 灰度形态学

# 灰度形态学

- 灰度形态学基本运算
  - 灰度膨胀
  - 灰度腐蚀
  - 灰度开与闭
- 灰度形态学算法

# 灰度腐蚀与膨胀

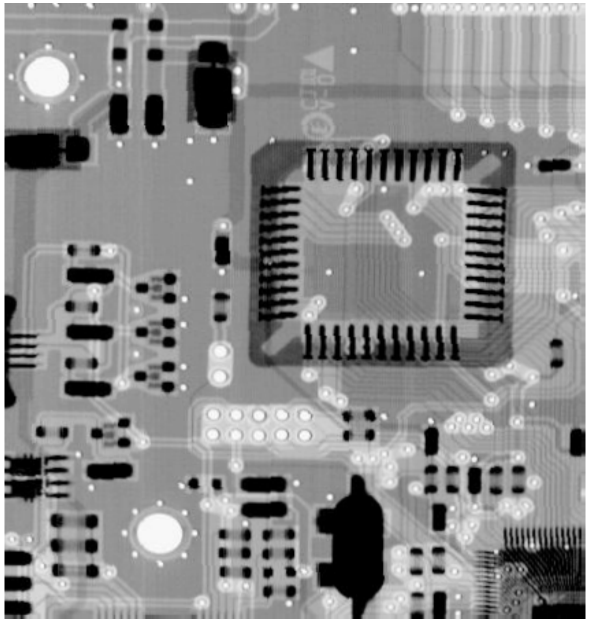
- 以下讨论中，令$f(x,y)$表示图像，$b(x,y)$表示结构元素，$(x,y)$是整数坐标。
- $b$对$f$进行灰度腐蚀可定义为：

$$[f \ominus b](x,y) = \min_{(s,t)\in b}\{f(x+s,y+t)\}$$

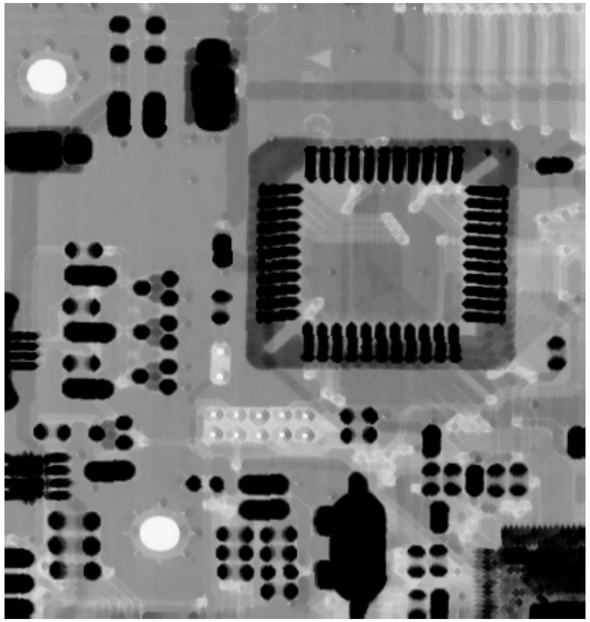
- $b$对$f$进行灰度膨胀可定义为：

$$[f \oplus b](x,y) = \max_{(s,t)\in b}\{f(x-s,y-t)\}$$

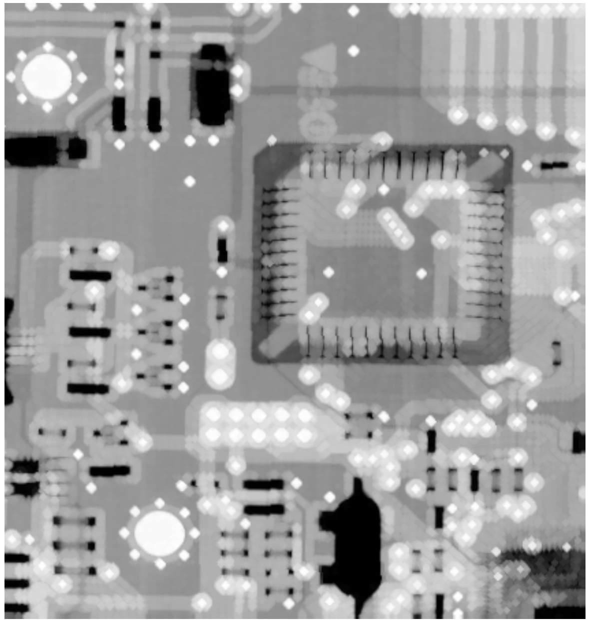
# 灰度腐蚀和膨胀的应用实例



Original



Erosion of original

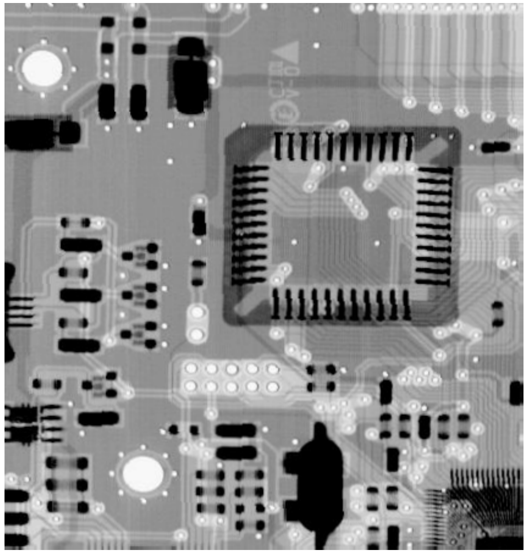


Dilation of original

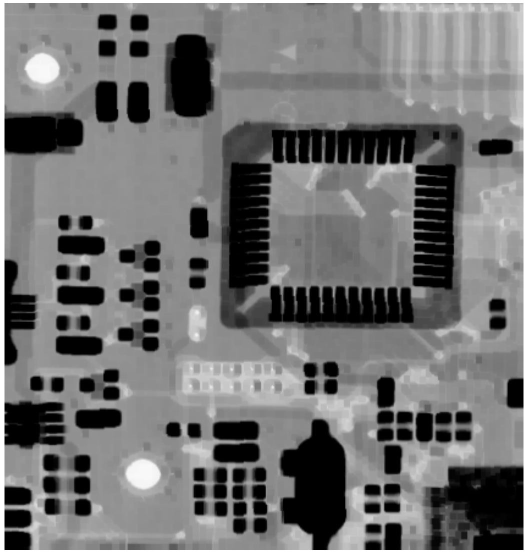
# 灰度开和闭运算

- 灰度开：$f \circ b = (f \ominus b) \oplus b$
- 灰度闭：$f \cdot b = (f \oplus b) \ominus b$
- 开运算通常用于去除小的（相对于结构元素而言）亮细节，而保留总体的灰度及和大的亮的特征不变。因为开始的腐蚀操作消除亮细节的同时也使图像变暗，所以后面的膨胀过程使图像变亮，但不会再引入被去除的细节。
- 闭运算通常用于去除小的暗细节，同时相对保留亮特征不变。因为开始的膨胀操作消除暗细节的同时也使图像变亮，所以后面的腐蚀过程使图像变暗，但不会再引入被去除的细节。

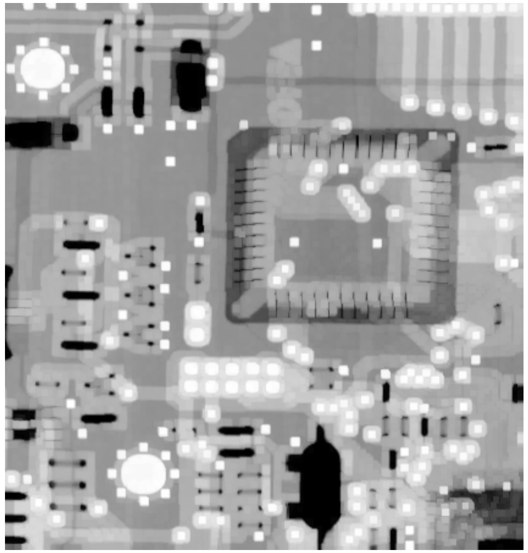
# 开、闭运算应用实例



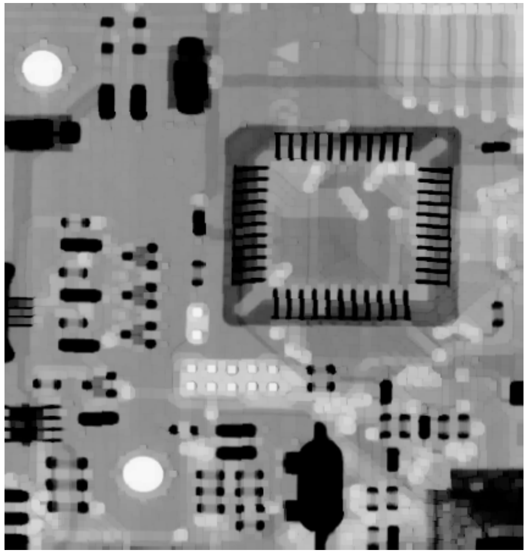
Original



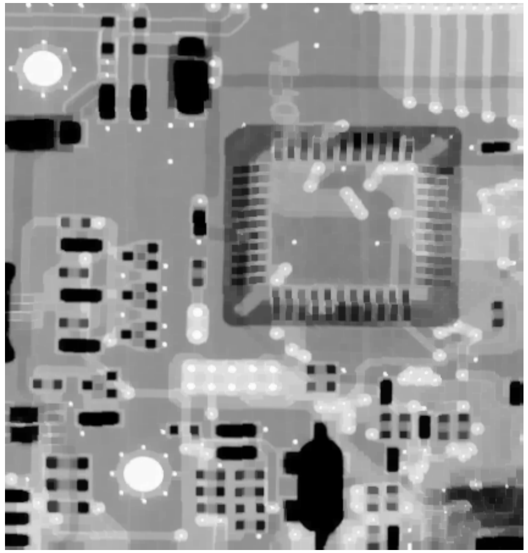
Erosion



Dilation



Openning



Closing

```matlab
% ex0910_GrayOpenClose
% DIP, P692
f = imread('..\data\Fig0937(a)(ckt_board_section).tif');
se = strel('disk', 3);
fe = imerode(f, se);
fd = imdilate(f, se);
fo = imdilate(fe, se);
fc = imerode(fd, se);
figure(1),
ax(1)=subplot(2,3,1); imshow(f); title('Original');
ax(2)=subplot(2,3,2); imshow(fe); title('Erosion');
ax(3)=subplot(2,3,3); imshow(fd); title('Dilation');
ax(4)=subplot(2,3,4); imshow(fo); title('Openning');
ax(5)=subplot(2,3,5); imshow(fc); title('Closing');
linkaxes(ax);
```

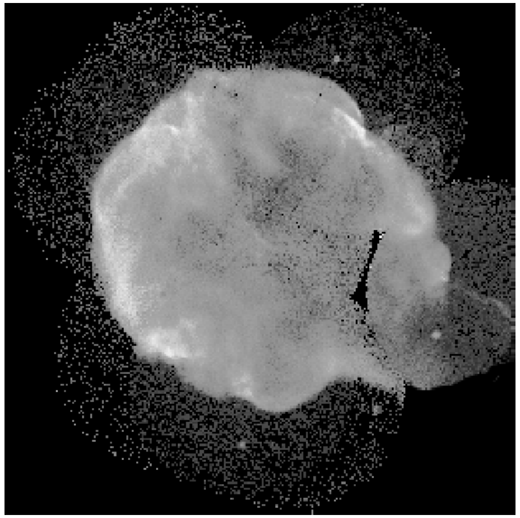
# 灰度形态学算法

- 形态学平滑
- 形态学梯度
- Top-hat 变换
- 纹理分割
- 粒子测度 (granulometry)

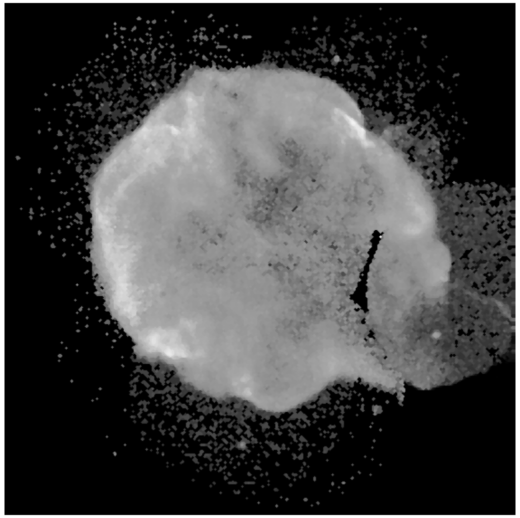
# 形态学平滑

- 开运算去除亮的、比结构元素小的细节
- 闭运算去除暗的、比结构元素小的细节
- 二者组合起来，作为形态学滤波器（morphological filters），用于图像的平滑、去噪

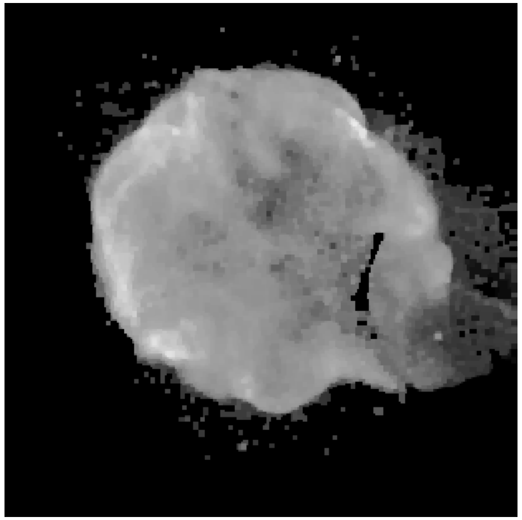
# 形态学平滑



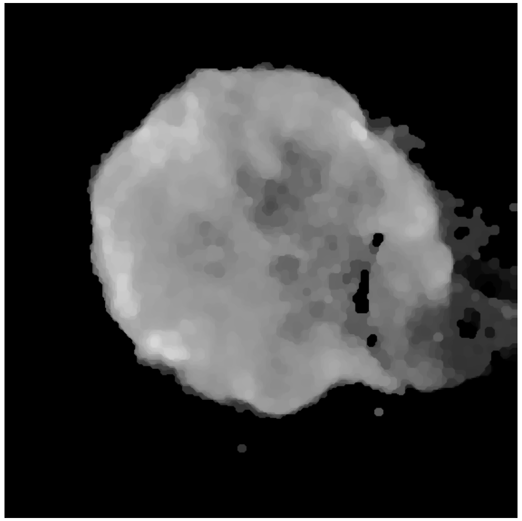
Original



disk radius 1



disk radius 3



disk radius 5

```
% fig0938_GrayMorphSmooth
% DIP, P693
f = imread('..\data\Fig0938(a)(cygnusloop_Xray_original).tif');
se1 = strel('disk', 1);
se2 = strel('disk', 3);
se3 = strel('disk', 5);
g1 = imclose(imopen(f, se1), se1);
g2 = imclose(imopen(f, se2), se2);
g3 = imclose(imopen(f, se3), se3);
figure(1),
ax(1)=subplot(2,2,1); imshow(f); title('Original');
ax(2)=subplot(2,2,2); imshow(g1); title('disk radius 1');
ax(3)=subplot(2,2,3); imshow(g2); title('disk radius 3');
ax(4)=subplot(2,2,4); imshow(g3); title('disk radius 5');
linkaxes(ax);
```

# 形态学梯度

形态学梯度：

$$g = (f \oplus b) - (f \ominus b)$$

结构元素$3 \times 3$

Original

Dilation

Erosion

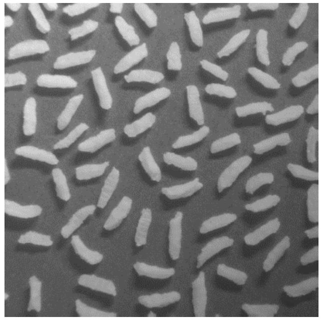
Gradient

```matlab
% fig0939_GrayMorphGradient
% DIP, P693
f = imread('..\data\Fig0939(a)(headCT-Vandy).tif');
fd = imdilate(f, ones(3,3));
fe = imerode(f, ones(3,3));
g = imsubtract(fd, fe);
figure(1),
ax(1)=subplot(2,2,1); imshow(f); title('Original');
ax(2)=subplot(2,2,2); imshow(fd); title('Dilation');
ax(3)=subplot(2,2,3); imshow(fe); title('Erosion');
ax(4)=subplot(2,2,4); imshow(g); title('Gradient');
linkaxes(ax);
```

# 高帽变换（Top-hat ）

- 不均匀光照给物体分割造成困难
- 如何用图像处理方法使光照均匀？
- Top-hat变换：

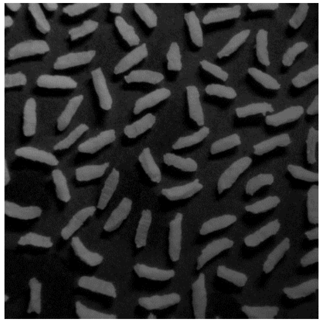
$$T_{\text{hat}}(f) = f - (f \circ b)$$



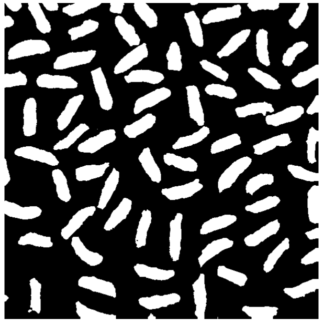
Original

Thresholded image

Openning with a disk of radius 40



Top-hat transformation



Thresholded top-hat image

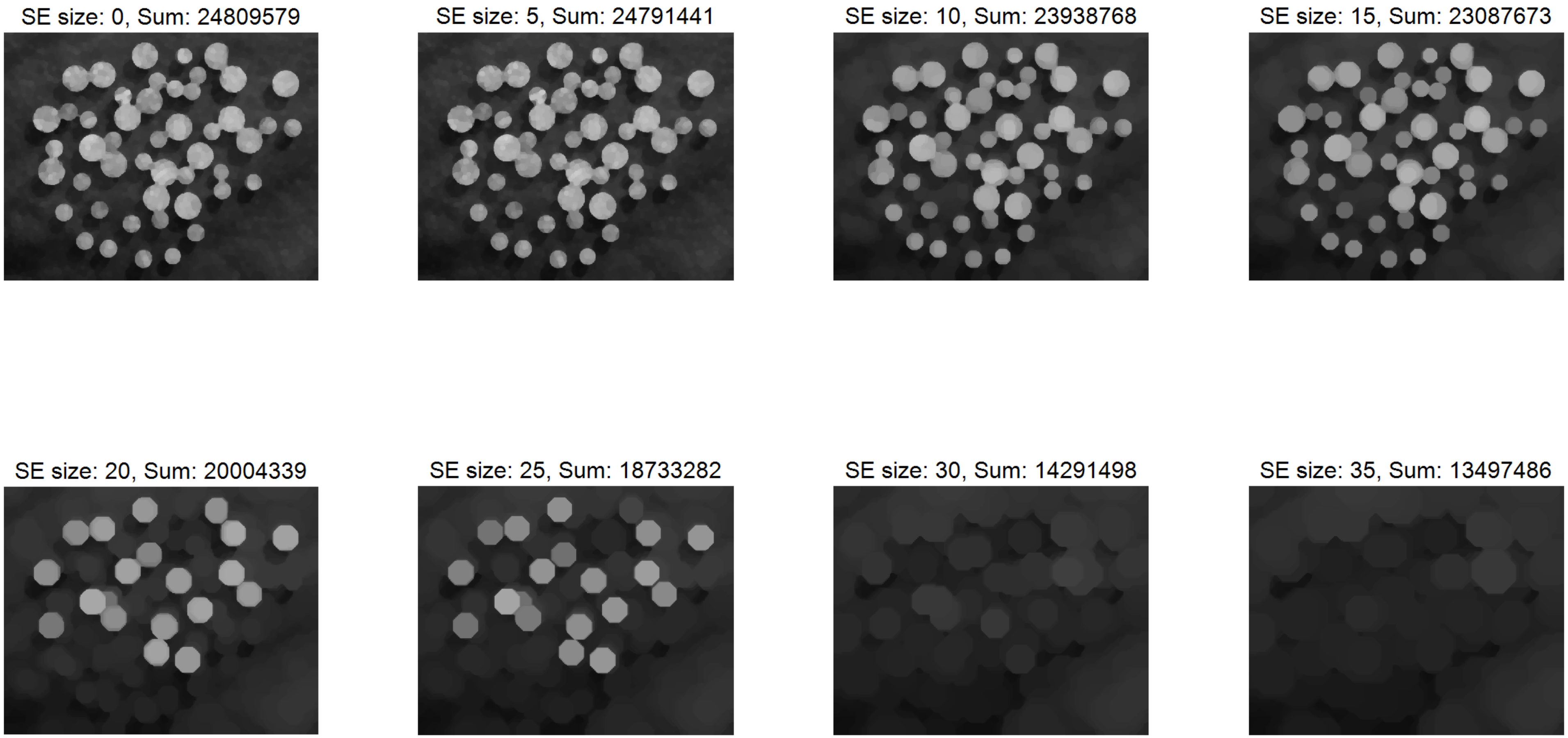
```
% fig0940_Tophat
% DIP, P695
f =
imread('..\data\Fig0940(a)(rice_image_with_intensity_gradient
).tif');
level = graythresh(f);
BW1 = im2bw(f, level);
se = strel('disk', 40);
fo = imopen(f, se);
fth = imsubtract(f, fo);
level = graythresh(fth);
BW2 = im2bw(fth, level);
figure(1),
ax(1)=subplot(2,3,1); imshow(f); title('Original');
ax(2)=subplot(2,3,2); imshow(BW1); title('Thresholded image');
ax(3)=subplot(2,3,3); imshow(fo); title('Openning with a disk
of radius 40');
ax(4)=subplot(2,3,4); imshow(fth); title('Top-hat
transformation');
ax(5)=subplot(2,3,5); imshow(BW2); title('Thresholded top-hat
image');
linkaxes(ax);
```

# 粒子测度

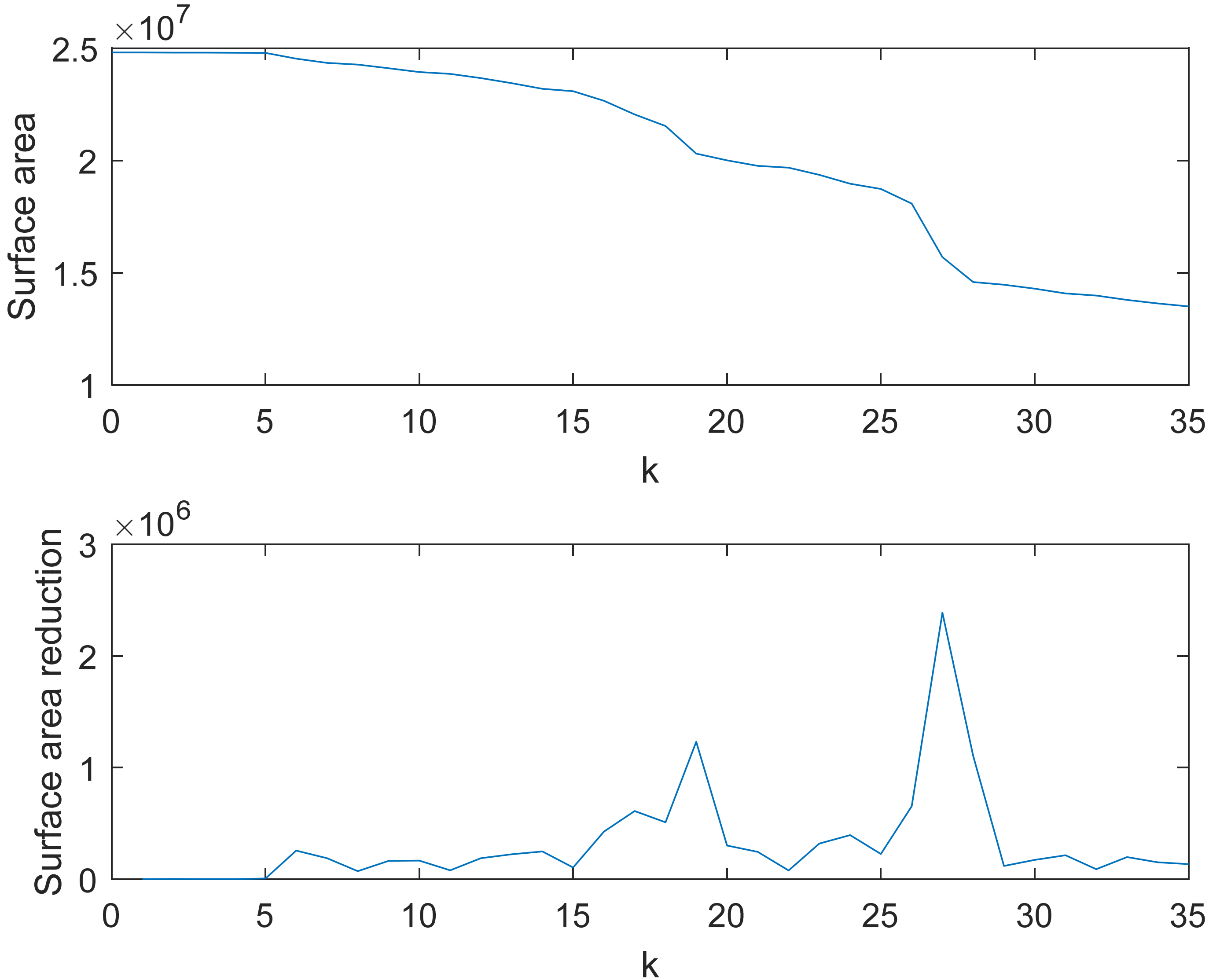
原始图像

- 粒子测度：计算图像中不同尺寸粒子的分布
- 由于粒子相互重叠或者粘连，分割、计数的方法效果不佳
- 利用形态学处理，非直接得到粒子的尺寸分布

# 粒子测度


SE size: 0, Sum: 24809579
SE size: 5, Sum: 24791441
SE size: 10, Sum: 23938768
SE size: 15, Sum: 23087673
SE size: 20, Sum: 20004339
SE size: 25, Sum: 18733282
SE size: 30, Sum: 14291498
SE size: 35, Sum: 13497486

# 粒子测度

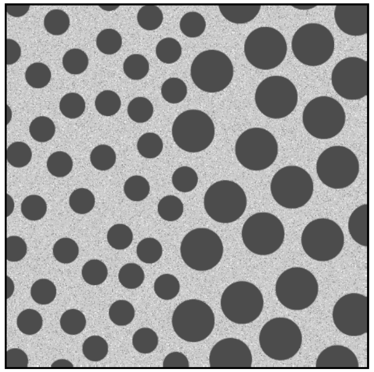


×10^7
2.5
2
1.5
1
Surface area
0
5
10
15
20
25
30
35
k
×10^6
3
2
1
0
Surface area reduction
0
5
10
15
20
25
30
35
k

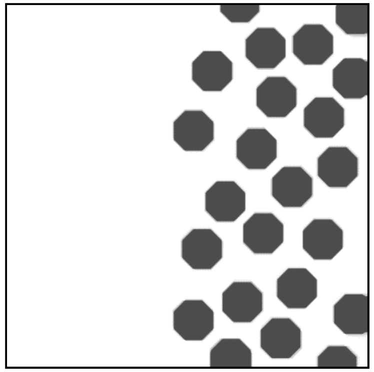
```
% fig0941_granulometry
% DIP, P696
f = imread('..\data\Fig0941(a)(wood_dowels).tif');
se = strel('disk', 5);
fs = imclose(imopen(f, se), se);
close all
figure(1), subplot(1,2,1), imshow(f);
subplot(1,2,2), imshow(fs)
figure(2)
K = 36;
sum_pixels = zeros(1, K);
for k = 0:K-1
    se = strel('disk', k);
    fo = imopen(fs, se);
    sum_pixels(k+1) = sum(fo(:));
    if mod(k,5)==0
        subplot(2,4,k/5+1),imshow(fo);
        title(sprintf('SE size: %d, Sum: %d',k,sum_pixels(k+1)))
    end
end
figure(3), subplot(2,1,1), plot(0:K-1, sum_pixels)
xlabel('k'), ylabel('Surface area');
subplot(2,1,2), plot(1:K-1, -diff(sum_pixels))
xlabel('k'), ylabel('Surface area reduction');
```

# 纹理分割（Texture Segmentation）

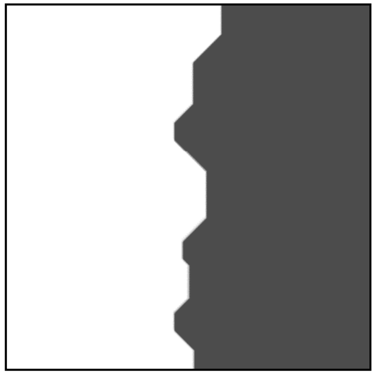
- 用与图像左边球大小相当的结构元素（此时左边的球相当于暗细节）对图像进行闭操作，来消除小球，左边只留下亮背景，右边不变。
- 用大于大球间距离的结构元素对上述结果做开运算，此时图像右边的背景区域（相当于亮细节）被消除，使图像右边全成了黑色。这样就得到左边为白色，右边为黑色的简单图像。
- 用形态学梯度算法得到边界。



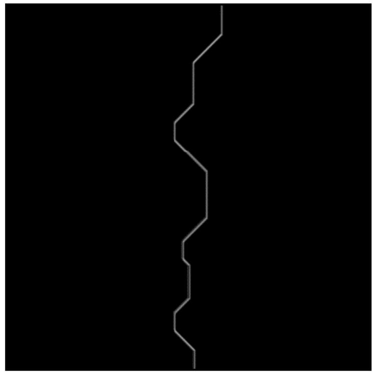
Original



Close with disk of radius 30



Open with disk of radius 60



Boundary

```
% fig0943_TextureSegment
% DIP, P698
f =
imread('..\data\Fig0943(a)(dark_blobs_on_light_background).tif');
se = strel('disk', 30);
fc = imclose(f, se);
close all
figure(1), subplot(2,2,1), imshow(f), title('Original')
subplot(2,2,2), imshow(fc), title('Close with disk of radius 30')

se = strel('disk', 60);
fo = imopen(fc, se);
subplot(2,2,3), imshow(fo), title('Open with disk of radius 60')

% morphological gradient
fd = imdilate(fo, ones(3,3));
fe = imerode(fo, ones(3,3));
g = imsubtract(fd, fe);
subplot(2,2,4), imshow(g), title('Boundary')
```

# 灰度形态学重建（Grayscale Morphological Reconstruction）

- 与二值图像的定义类似
- 1次测地学膨胀$D_g^{(1)}(f) = (f \oplus b) \wedge g$

其中，$f$为marker，$g$为mask，$b$为结构元素，$\wedge$为min运算

- $n$次测地学膨胀$D_g^{(n)}(f) = D_g^{(1)}\left[D_g^{(n-1)}(f)\right]$
- 膨胀形态学重建等于收敛时的测地学膨胀

$$R_g^D(f) = D_g^{(k)}(f)$$
$$D_g^{(k)}(f) = D_g^{(k+1)}(f)$$

- 重建开（opening by reconstruction）：先腐蚀，再做膨胀形态学重建

# 灰度形态学重建

- 1次测地学腐蚀$E_g^{(1)}(f) = (f \ominus b) \vee g$

其中，$f$为marker，$g$为mask，$b$为结构元素，$\vee$为max运算

- $n$次测地学腐蚀$E_g^{(n)}(f) = E_g^{(1)}\left[E_g^{(n-1)}(f)\right]$
- 腐蚀形态学重建等于收敛时的测地学腐蚀

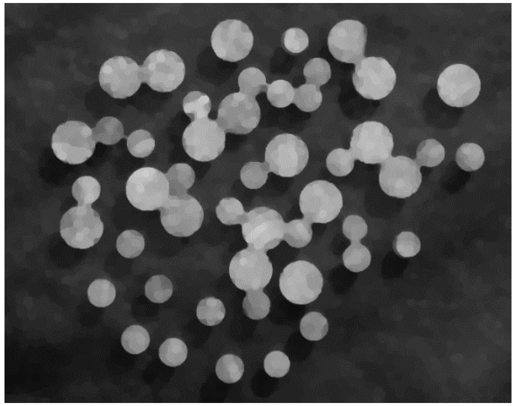
$$R_g^E(f) = E_g^{(k)}(f)$$
$$E_g^{(k)}(f) = E_g^{(k+1)}(f)$$

- 重建闭（closing by reconstruction）：先膨胀，再做腐蚀形态学重建

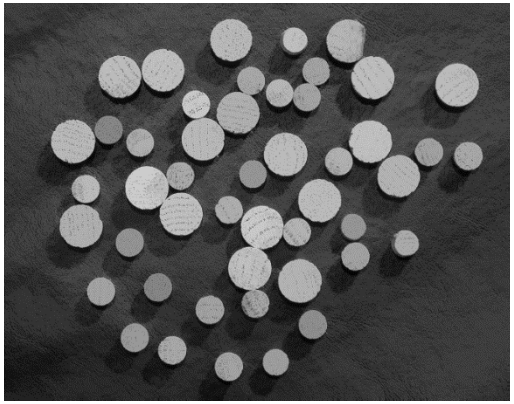
# 重建开和闭

Original

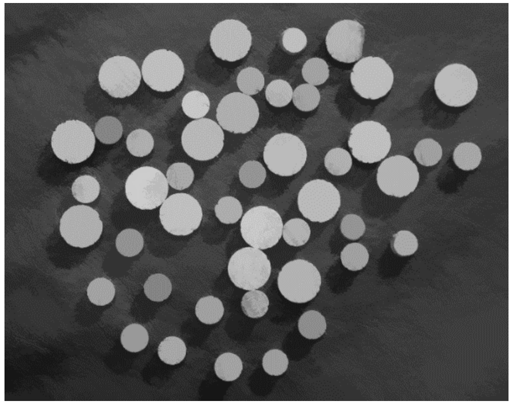
Openning



Closing of openning



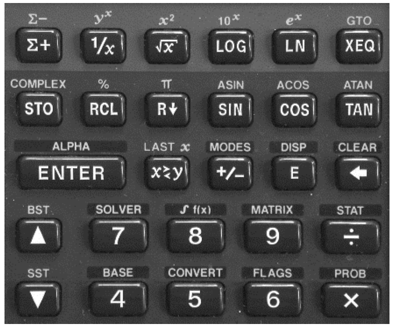
Openning by reconstruction



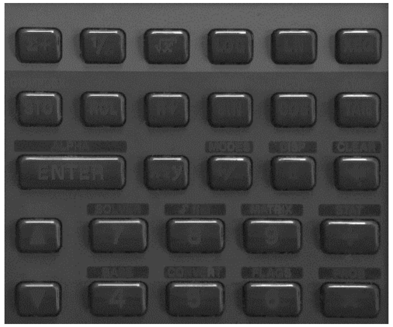
Openning by reconstruction
followed by
closing by reconstruction

```
% fig0929_GrayMorphReconstruction
% DIPUM, P375

f = imread('..\data\Fig0941(a)(wood_dowels).tif');
se = strel('disk', 5);
fe = imerode(f, se);
fo = imdilate(fe, se);
foc = imclose(fo, se);
fobr = imreconstruct(fe, f);
fobrc = imcomplement(fobr);
fobrce = imerode(fobrc, se);
fobrcbr = imcomplement(imreconstruct(fobrce, fobrc));
figure(1), clf
ax(1)=subplot(2,3,1); imshow(f); title('Original');
ax(2)=subplot(2,3,2); imshow(fo); title('Openning');
ax(3)=subplot(2,3,3); imshow(foc); title('Closing of openning');
ax(4)=subplot(2,3,5); imshow(fobr); title('Openning by
reconstruction');
ax(5)=subplot(2,3,6); imshow(fobrcbr);
title('Openning by reconstruction followed by closing by
reconstruction');
linkaxes(ax);
```

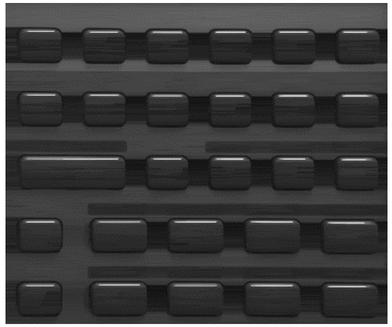
# 去除复杂背景



Original



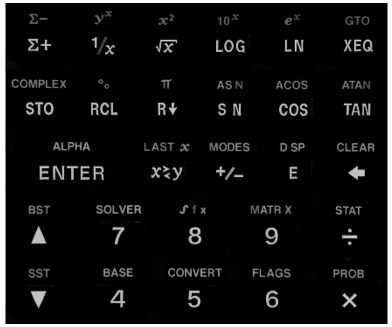
Openning by reconstruction of (1) using a horinontal line 71 pixels long



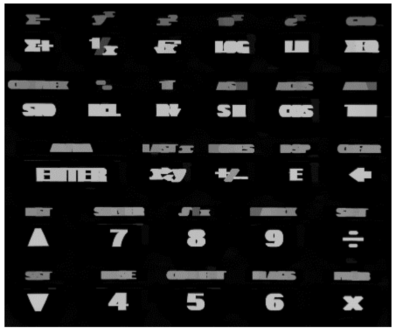
Openning of (1)

Top-hat by reconstruction

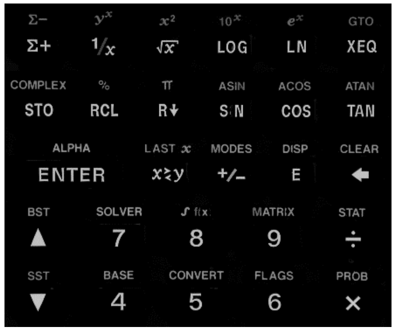
Top-hat



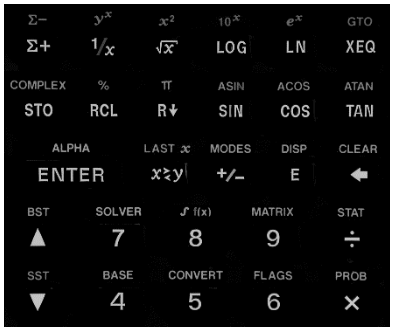
Openning by reconstruction of (4) using a horinontal line 11 pixels long



Dilation of (6) using a horinontal line 21 pixels long



Min of (4) and (7)



Final reconstruction result

```
% ex0911_GrayMorphReconstruction
% DIP, P699
f = imread('..\data\Fig0944(a)(calculator).tif');
fobr = imreconstruct(imerode(f, ones(1, 71)), f);
fo = imopen(f, ones(1, 71));
fthr = imsubtract(f, fobr);
fth = imsubtract(f, fo);
gobr = imreconstruct(imerode(fthr, ones(1,11)), fthr);
gobrd = imdilate(gobr, ones(1, 21));
minf = min(gobrd, fthr);
f2 = imreconstruct(minf, fthr);
figure(1), clf
subplot(3,3,1),imshow(f),title('Original')
subplot(3,3,2),imshow(fobr),title('Openning by reconstruction of (1) using a horinontal line 71 pixels long')
subplot(3,3,3),imshow(fo),title('Openning of (1)')
subplot(3,3,4),imshow(fthr),title('Top-hat by reconstruction')
subplot(3,3,5),imshow(fth),title('Top-hat')
subplot(3,3,6),imshow(gobr),title('Openning by reconstruction of (4) using a horinontal line 11 pixels long')
subplot(3,3,7),imshow(gobrd),title('Dilation of (6) using a horinontal line 21 pixels long')
subplot(3,3,8),imshow(minf),title('Min of (4) and (7)')
subplot(3,3,9),imshow(f2),title('Final reconstruction result')
```