



## 第四章：形态学

---

中国科学技术大学  
电子工程与信息科学系

主讲教师：李厚强 ([lihq@ustc.edu.cn](mailto:lihq@ustc.edu.cn))  
周文罡 ([zhwg@ustc.edu.cn](mailto:zhwg@ustc.edu.cn))  
李 礼 ([lil1@ustc.edu.cn](mailto:lil1@ustc.edu.cn))  
胡 洋 ([eeychu@ustc.edu.cn](mailto:eeychu@ustc.edu.cn))



# 形态学

## □ 形态学

### ■ 二值形态学

- ✓ 基本定义
- ✓ 基本运算
- ✓ 实用算法

### ■ 灰度形态学

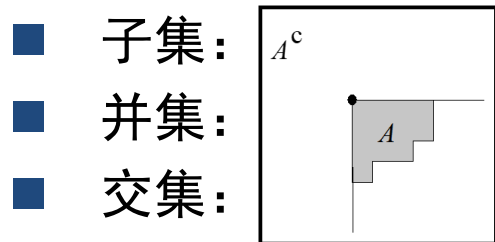
- ✓ 基本运算
- ✓ 实用算法

# 二值形态学

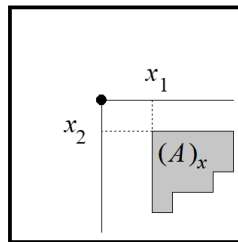
## □ 基本集合定义

■ 集合：用大写字母表示，空集记为 $\emptyset$

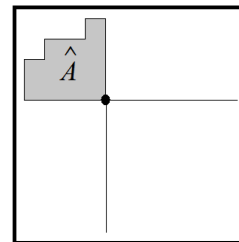
■ 元素：用小写字母表示



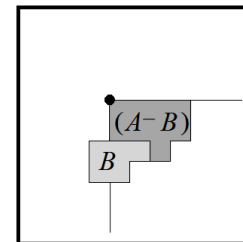
(a)



(b)



(c)



(d)

■ 补集：  $A^c = \{x|x \notin A\}$

■ 位移：  $(A)_x = \{y|y = a + x, a \in A\}$

■ 映像：  $\hat{A} = \{x|x = -a, a \in A\}$

■ 差集：  $A - B = \{x|x \in A, x \notin B\} = A \cap B^c$



# 二值形态学基本运算

## □ 集合运算

- A为图象集合，B 为结构元素（集合）
- 数学形态学运算是用 B 对 A 进行操作
- 结构元素要指定1个原点（参考点）

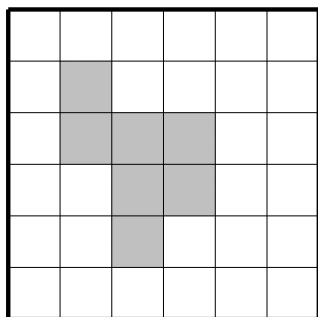
# 膨胀和腐蚀

## □ 膨胀

■ 膨胀的算符为 $\oplus$

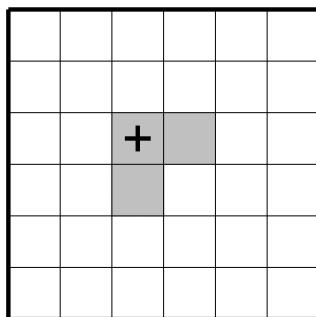
$$A \oplus B = \{x \mid [(\hat{B})_x \cap A] \neq \emptyset\}$$

集合A



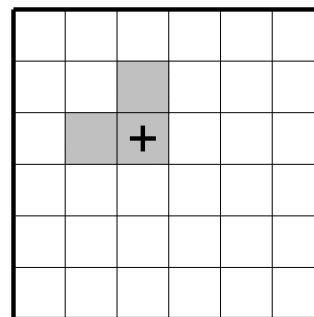
(a)

结构元素B



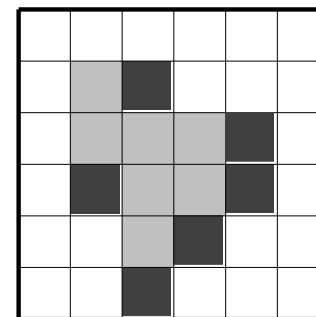
(b)

B的映象



(c)

集合 $A \oplus B$



(d)

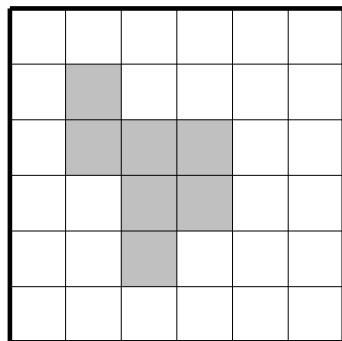
# 膨胀和腐蚀

## □ 腐蚀

- 腐蚀的算符为  $\ominus$

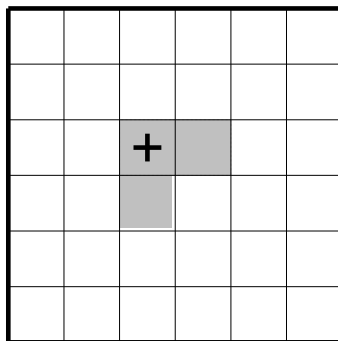
$$A \ominus B = \{x | (B)_x \subseteq A\}$$

集合A



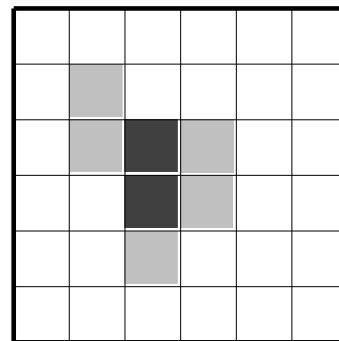
(a)

结构元素B



(b)

集合  $A \ominus B$



(c)

# 膨胀和腐蚀

## □ 原点不包含在结构元素中的膨胀和腐蚀

### ■ 原点包含在结构元素中

✓ 膨胀运算:  $A \subseteq A \oplus B$

✓ 腐蚀运算:  $A \ominus B \subseteq A$

### ■ 原点不包含在结构元素中

✓ 膨胀运算:  $A \not\subseteq A \oplus B$

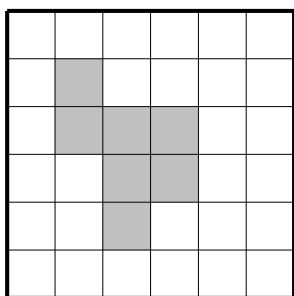
✓ 腐蚀运算:  $A \ominus B \subseteq A$ , 或  $A \ominus B \not\subseteq A$

# 膨胀和腐蚀

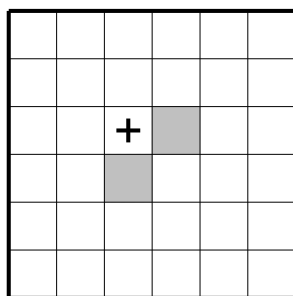
## □ 原点不包含在结构元素中的时的膨胀运算

$$A \not\subset A \oplus B$$

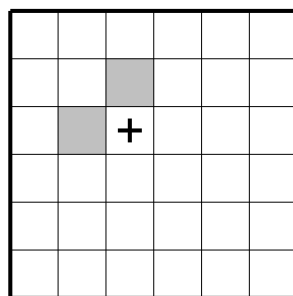
$$A \oplus B = \{x \mid [(\hat{B})_x \cap A] \neq \emptyset\}$$



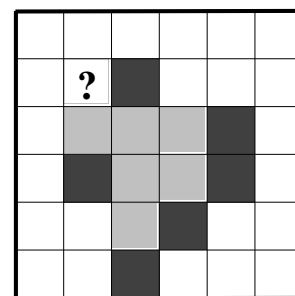
(a)



(b)

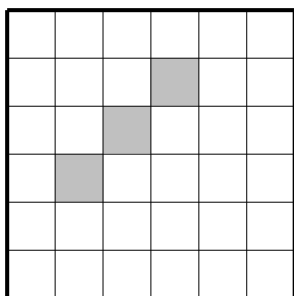


(c)

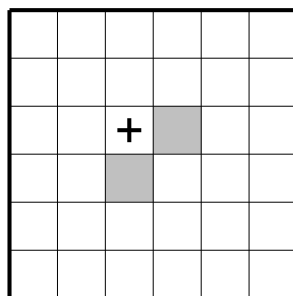


(d)

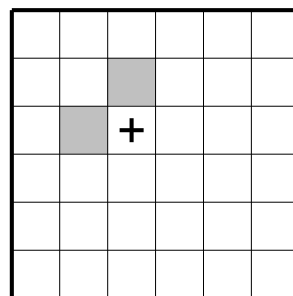
A在膨胀中自身完全消失了



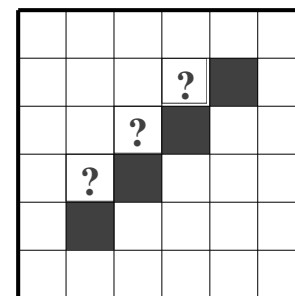
(a)



(b)



(c)



(d)

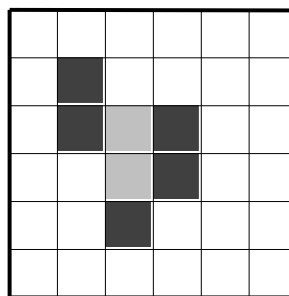
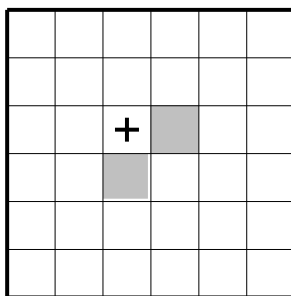
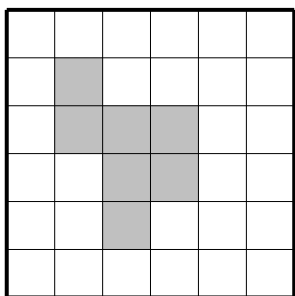


# 膨胀和腐蚀

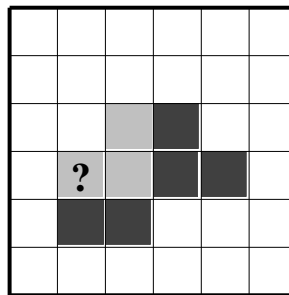
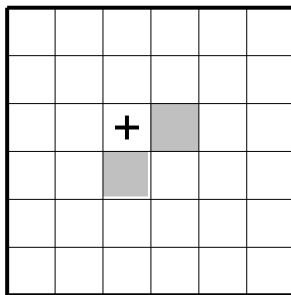
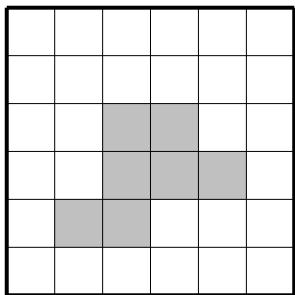
## □ 原点不包含在结构元素中的时的腐蚀运算

$$A \ominus B \subseteq A$$

$$A \ominus B = \{x | (B)_x \subseteq A\}$$



$$A \ominus B \not\subseteq A$$



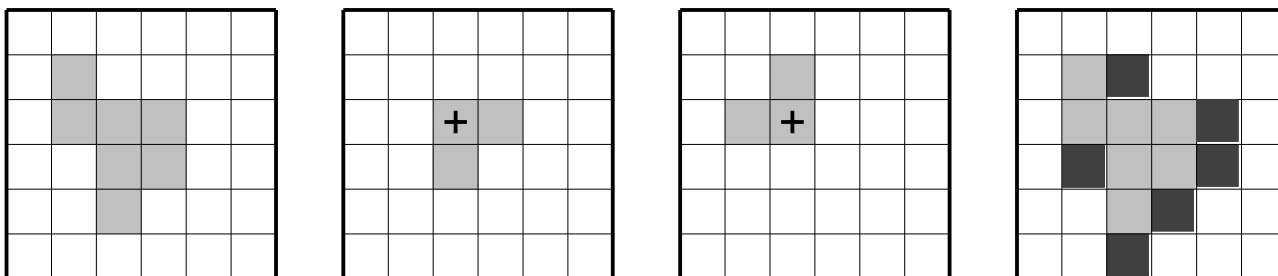
# 膨胀和腐蚀

## □ 用向量运算实现膨胀和腐蚀

$$A \oplus B = \{x | x = a + b, \text{ 对于任意 } a \in A \text{ 和 } b \in B\}$$

$$A = \{(1, 1), (1, 2), (2, 2), (3, 2), (2, 3), (3, 3), (2, 4)\}$$

$$B = \{(0, 0), (1, 0), (0, 1)\}$$



$$A \oplus B = \{(1, 1), (2, 1), (1, 2), (2, 2), (3, 2), (4, 2), \\ (1, 3), (2, 3), (3, 3), (4, 3), (2, 4), (3, 4), (2, 5)\}$$

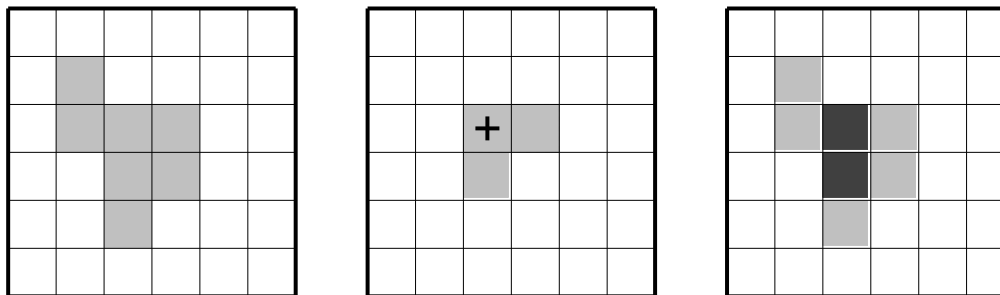
# 膨胀和腐蚀

## □ 用向量运算实现膨胀和腐蚀

$$A \ominus B = \{x | (x + b) \in A \text{ 对每一个 } b \in B\}$$

$$A = \{(1, 1), (1, 2), (2, 2), (3, 2), (2, 3), (3, 3), (2, 4)\}$$

$$B = \{(0, 0), (1, 0), (0, 1)\}$$



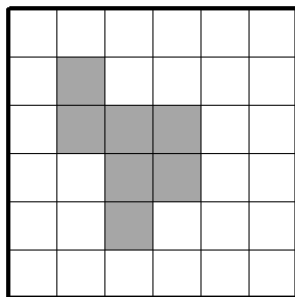
$$A \ominus B = \{(2, 2), (2, 3)\}$$

# 膨胀和腐蚀

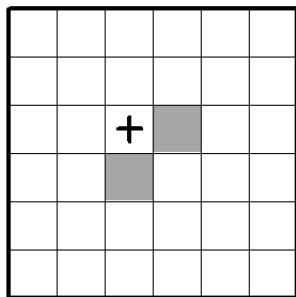
## □ 用位移运算实现膨胀和腐蚀

按每个 $b$ 来位移 $A$ 并把结果或（OR）起来

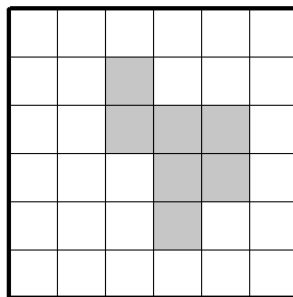
$$A \oplus B = \bigcup_{b \in B} (A)_b$$



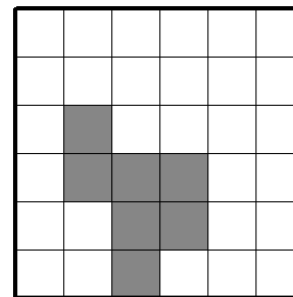
(a)



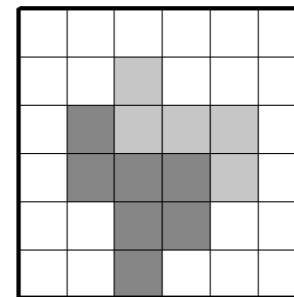
(b)



(c)



(d)



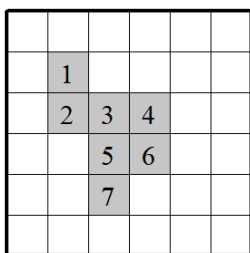
(e)

# 膨胀和腐蚀

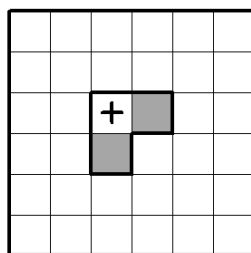
## □ 用位移运算实现膨胀和腐蚀

按每个a来位移B并把结果或（OR）起来

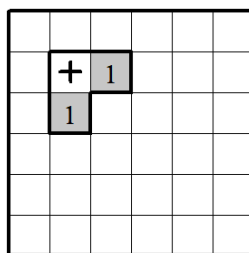
$$A \oplus B = \bigcup_{a \in A} (B)_a$$



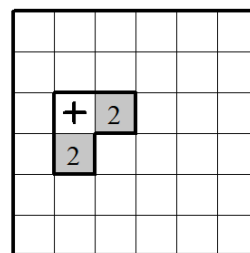
(a)



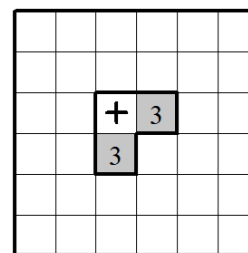
(b)



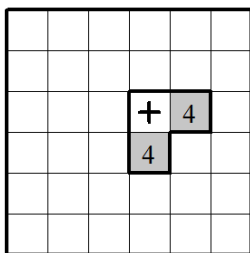
(c)



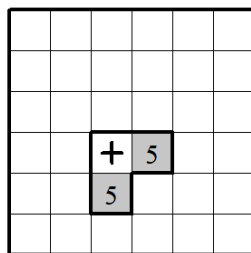
(d)



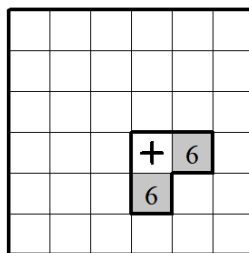
(e)



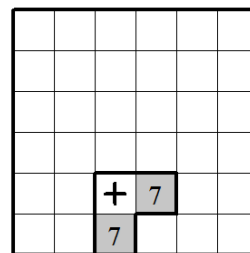
(f)



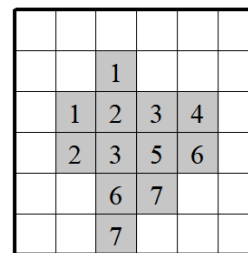
(g)



(h)



(i)



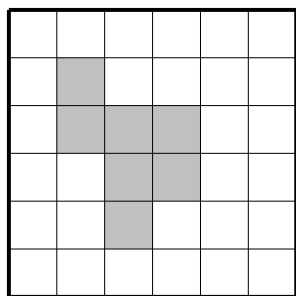
(j)

# 膨胀和腐蚀

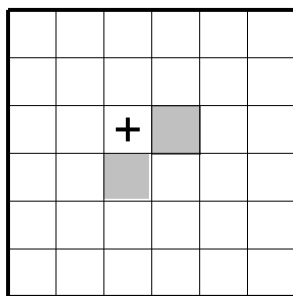
## □ 用位移运算实现膨胀和腐蚀

按每个b来负位移A并把结果交（AND）起来

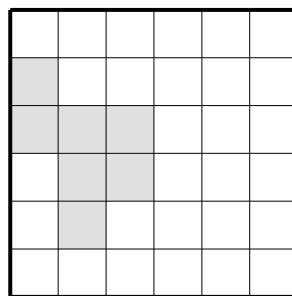
$$A \ominus B = \bigcap_{b \in B} (A)_{-b}$$



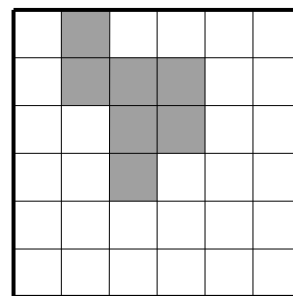
(a)



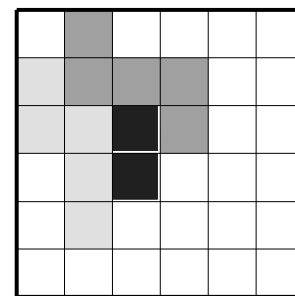
(b)



(c)



(d)

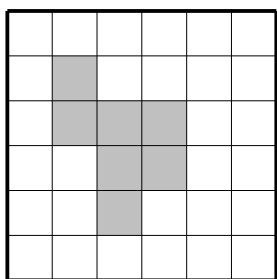


(e)

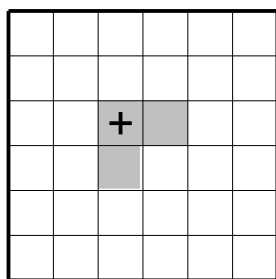
# 膨胀和腐蚀

## □ 膨胀和腐蚀的对偶性

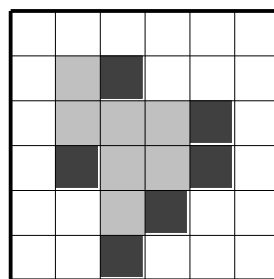
$$(A \oplus B)^c = A^c \ominus \hat{B} \quad (A \ominus B)^c = A^c \oplus \hat{B}$$



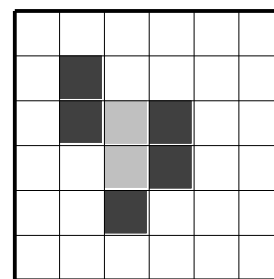
(a)



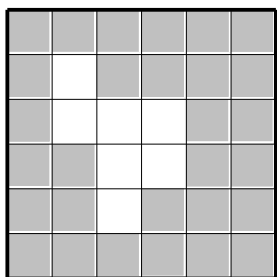
(b)



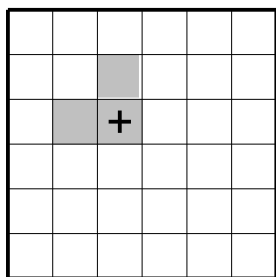
(c)



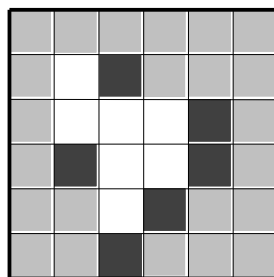
(d)



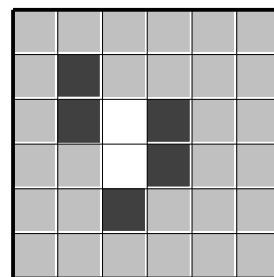
(e)



(f)



(g)



(h)

# 膨胀和腐蚀

## □ 膨胀和腐蚀的对偶性

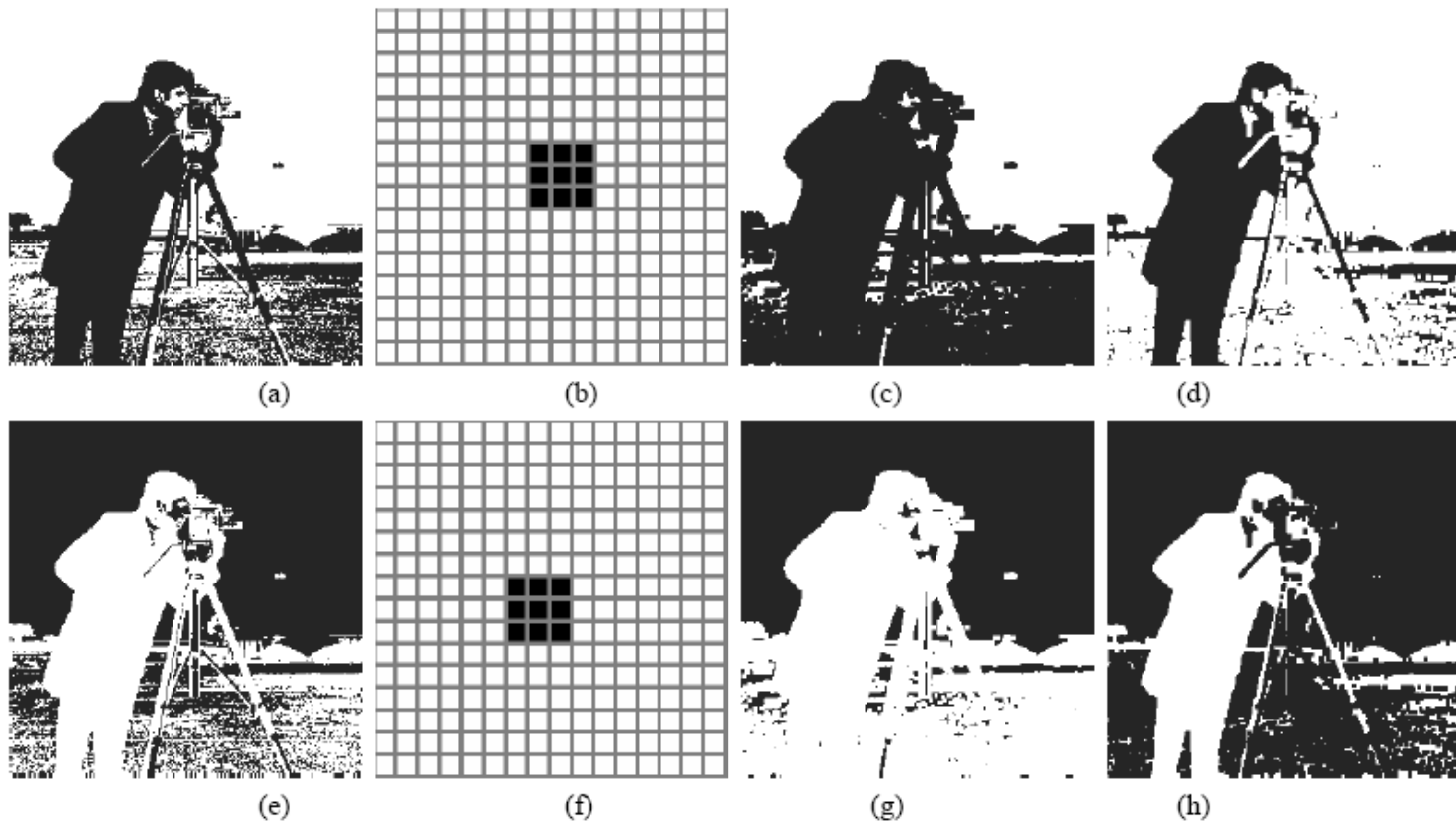


图 14.2.12 膨胀和腐蚀的对偶性验证实例





# 开启和闭合

## □ 开启和闭合定义

- 膨胀和腐蚀并不互为逆运算
- 它们可以级连结合使用
- 开启：先对图象进行腐蚀然后膨胀其结果

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

- 闭合：先对图象进行膨胀然后腐蚀其结果

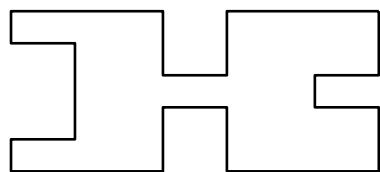
$$A \cdot B = (A \oplus B) \ominus B$$

- 开启和闭合不受原点是否在结构元素之中的影响

# 开启和闭合

## □ 开启和闭合定义

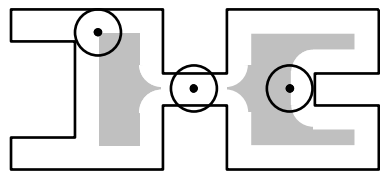
- 开启运算可以把比结构元素小的突刺滤掉
- 闭合运算可以把比结构元素小的缺口或孔填充上



(a)



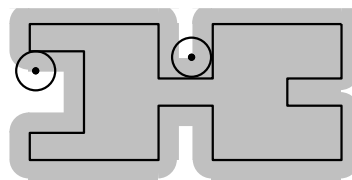
(b)



(c)



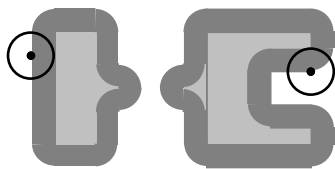
(d)



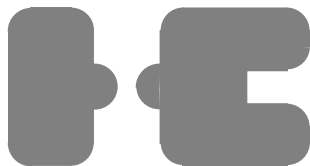
(g)



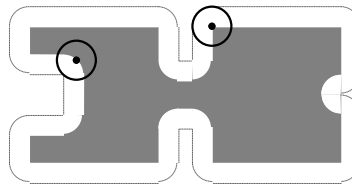
(h)



(e)



(f)



(i)



(j)

# 开启和闭合

## □ 开启和闭合定义



原图



(a)



(b)

图 14.2.14 开启和闭合实例



# 开启和闭合

## □ 开启和闭合的对偶性

- 开启和闭合也具有对偶性

$$(A \circ B)^c = A^c \cdot \hat{B}$$

$$(A \cdot B)^c = A^c \circ \hat{B}$$

$$(A \circ B)^c = [(A \ominus B) \oplus B]^c = (A \ominus B)^c \ominus \hat{B} = A^c \oplus \hat{B} \ominus \hat{B} = A^c \cdot \hat{B}$$

$$(A \cdot B)^c = [(A \oplus B) \ominus B]^c = (A \oplus B)^c \oplus \hat{B} = A^c \ominus \hat{B} \oplus \hat{B} = A^c \circ \hat{B}$$

# 开启和闭合

## □ 开启和闭合与集合的关系

操 作	并 集	交 集
开 启	$\left(\bigcup_{i=1}^n A_i\right) \circ B \supseteq \bigcup_{i=1}^n (A_i \circ B)$	$\left(\bigcap_{i=1}^n A_i\right) \circ B \subseteq \bigcap_{i=1}^n (A_i \circ B)$
闭 合	$\left(\bigcup_{i=1}^n A_i\right) \cdot B \supseteq \bigcup_{i=1}^n (A_i \cdot B)$	$\left(\bigcap_{i=1}^n A_i\right) \cdot B \subseteq \bigcap_{i=1}^n (A_i \cdot B)$

# 开启和闭合

## □ 开启和闭合的几何解释

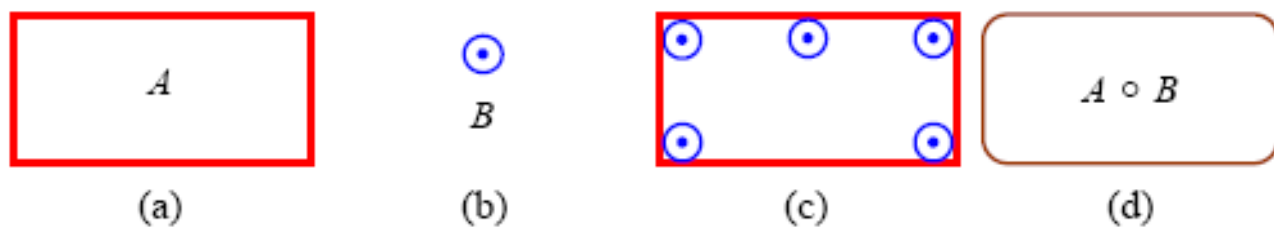


图 14.2.15 开启的填充特性

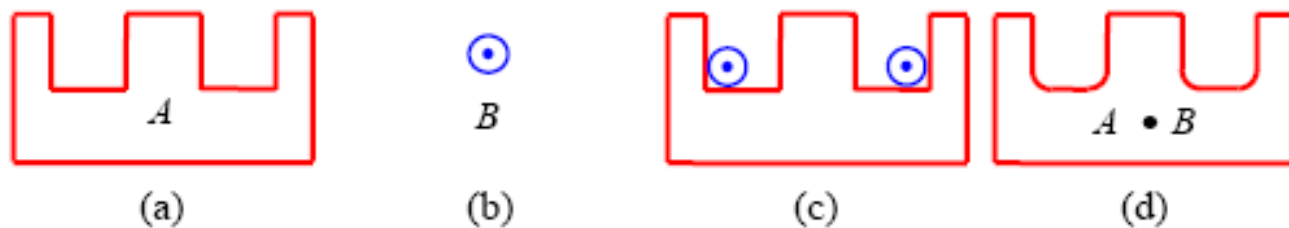


图 14.2.16 闭合的几何解释

# 击中-击不中变换

## □ 击中-击不中变换

- 形状检测的一种基本工具
- 对应两个操作，所以用到两个结构元素
- 设A为原始图象，E和F为一对不重合的集合

$$A \uparrow (E, F) = (A \ominus E) \cap (A^c \ominus F) = (A \ominus E) \cap (A \oplus F)^c$$

$E$ : 击中结构元素

$F$ : 击不中结构元素

# 击中-击不中变换

## □ 击中-击不中变换

$$A \uparrow (E, F) = (A \ominus E) \cap (A^c \ominus F) = (A \ominus E) \cap (A \oplus F)^c$$

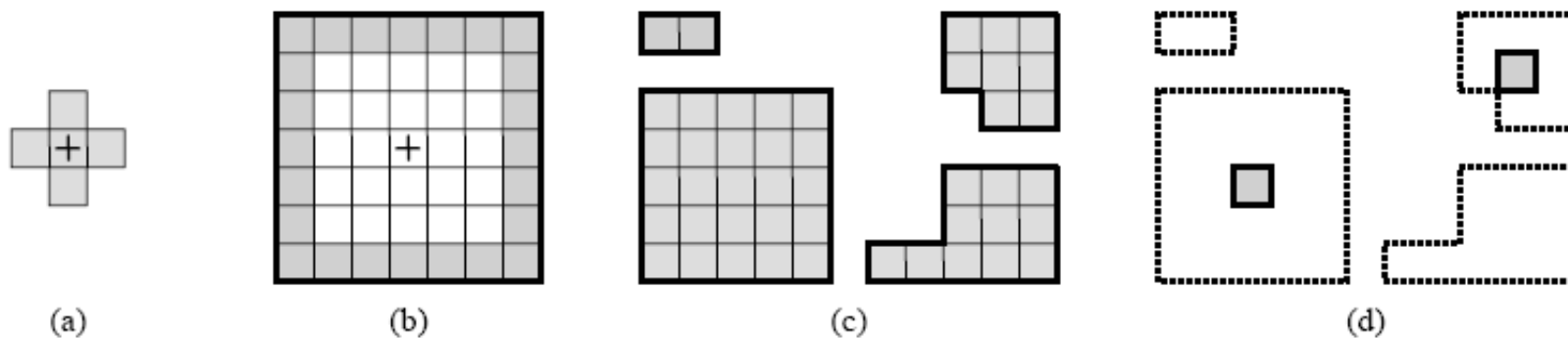


图 14.3.1 击中-击不中变换示例

- (a): 击中结构元素      (b): 击不中结构元素  
(c): 原始图像          (d): 变换结果



# 击中-击不中变换

□ 击中-击不中变换 ( (e)和(f)来自于别的变换)

击中变换:  $[1 \ 1 \ 1]$

击不中变换:  $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

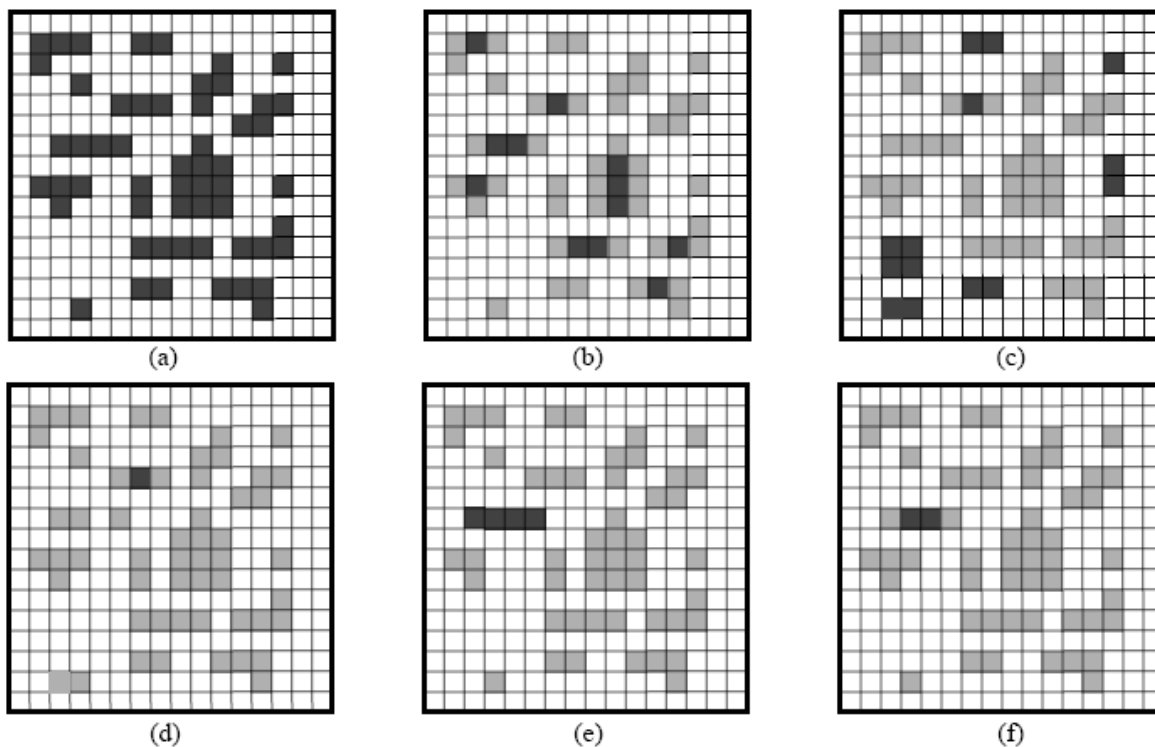


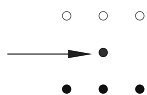
图 14.3.2 利用击中-击不中算子以提取包含水平方向上有连续 3 个像素的线段

# 击中-击不中变换

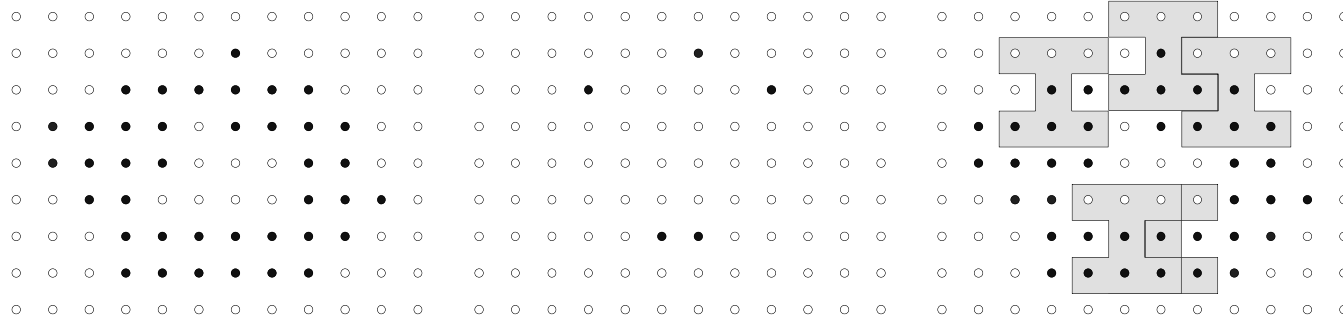
## □ 击中-击不中变换中的结构元素

- $A \uparrow B$ 的结果中仍保留的目标像素对应在 $A$ 中其邻域与结构元素 $B$ 对应的像素

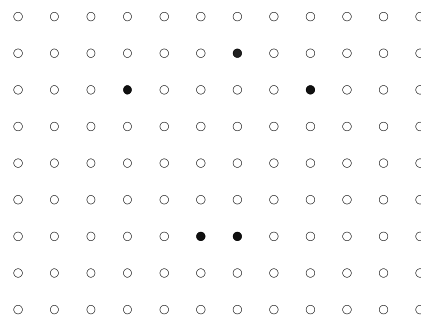
$$A \uparrow B = (A \ominus B_o) \cap (A^c \ominus B_b)$$



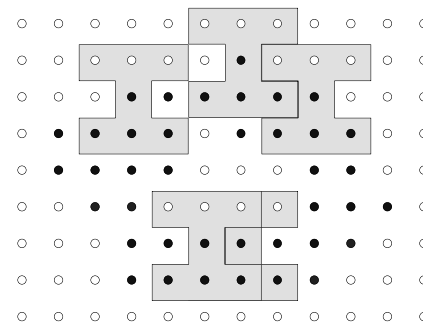
(a)



(b)



(c)



(d)

# 组合运算-I

## □ 区域凸包

- 令  $B_i (i = 1, 2, 3, 4)$  代表4个结构元素,  $X_i^0 = A$ 构造:

$$X_i^k = (X_i^{k-1} \uparrow B_i) \cup A \quad i = 1, 2, 3, 4 \text{ 和 } k = 1, 2, \dots$$

- 令  $D_i = X_i^{conv}$ , 上标 “conv” 表示在  $X_i^k = X_i^{k-1}$  意义下收敛,  $A$ 的凸包可表示为:

$$C(A) = \bigcup_{i=1}^4 D_i$$

# 组合运算-I

## □ 区域凸包 (X表示其值可为任意)

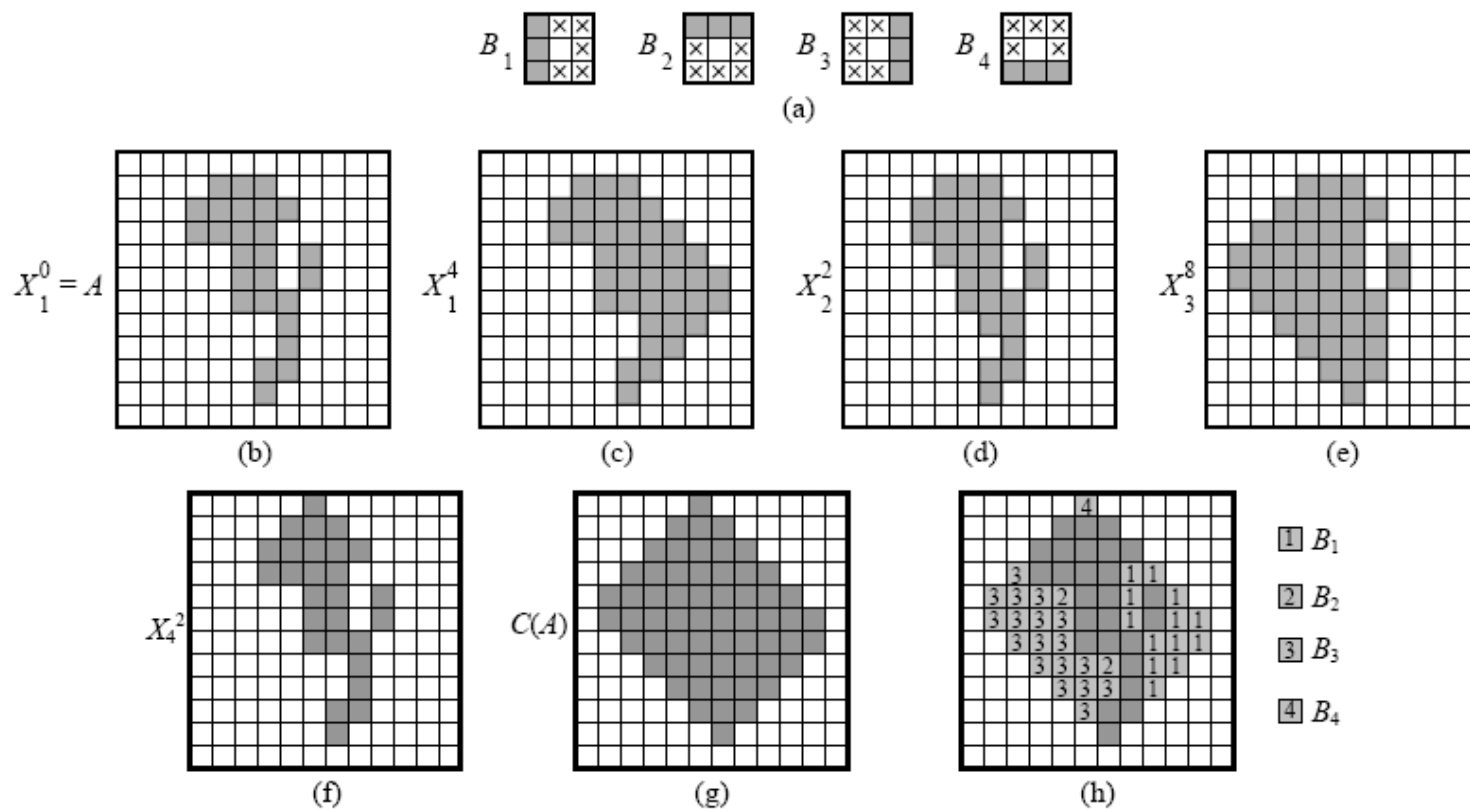


图 14.3.5 构造凸包的示例

# 组合运算-II

## □ 细化

- 用结构元素B细化集合A记作 $A \otimes B$
- 借助击中-击不中变换定义

$$A \otimes B = A - (A \uparrow B) = A \cap (A \uparrow B)^c$$

- 定义一个结构元素系列

$$\{B\} = \{B_1, B_2, \dots, B_n\}$$

$$A \otimes \{B\} = A - ((\dots ((A \otimes B_1) \otimes B_2) \dots) \otimes B_n)$$

# 组合运算-II

## □ 细化

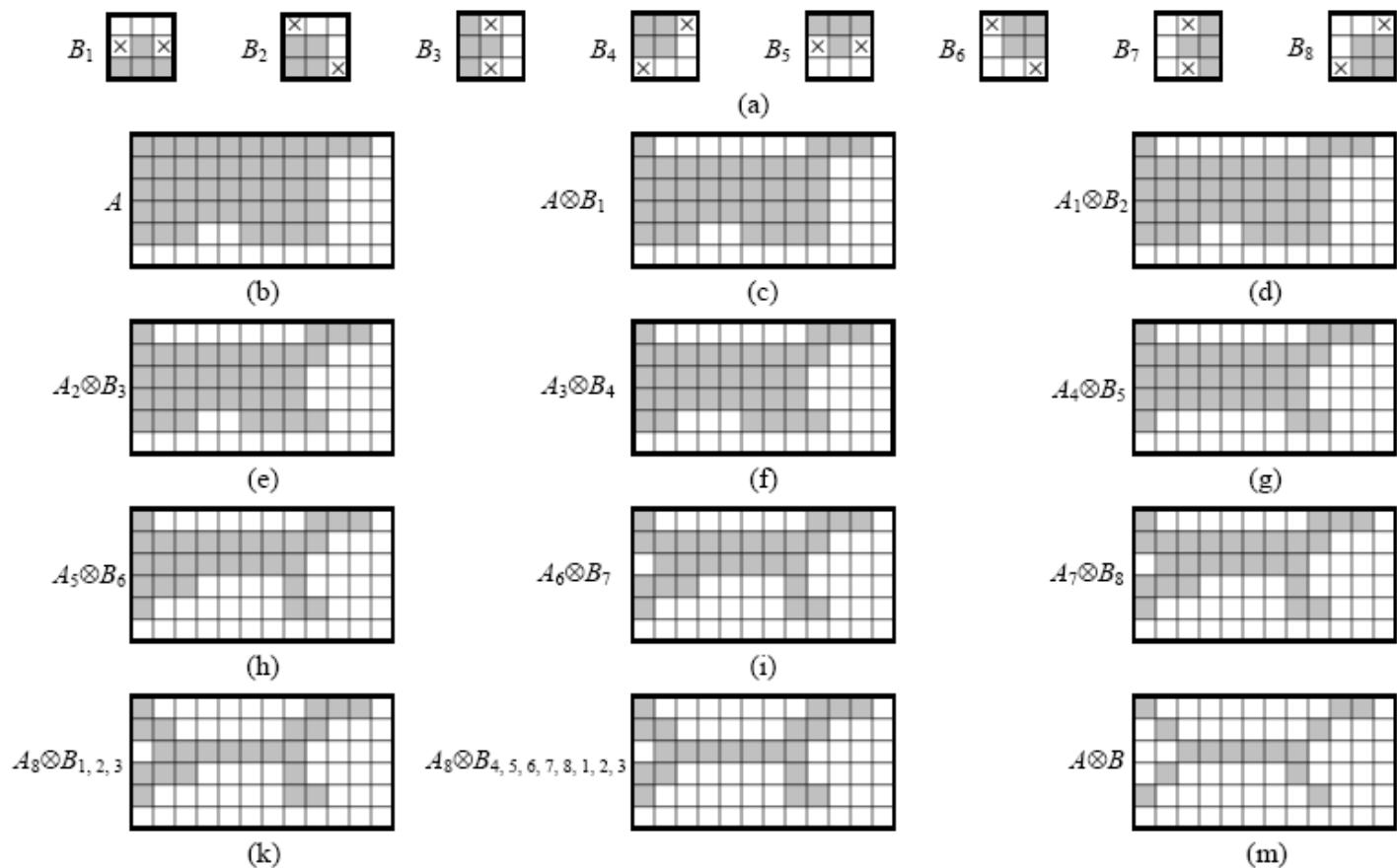


图 14.3.6 细化示例

# 组合运算-III

## □ 粗化

- 用结构元素B粗化集合A记作 $A \oplus B$

$$A \oplus B = A \cup (A \uparrow B)$$

- 定义为一系列操作

$$A \oplus \{B\} = ((\cdots((A \oplus B_1) \oplus B_2) \cdots) \oplus B_n)$$

粗化从形态学角度来说与细化是对应的，实际中可先细化背景然后求补以得到粗化的结果。换句话说，如果要粗化集合A，可先构造 $C = A^c$ ，然后细化C，最后求 $C^c$ 。

# 组合运算-III

## □ 粗化

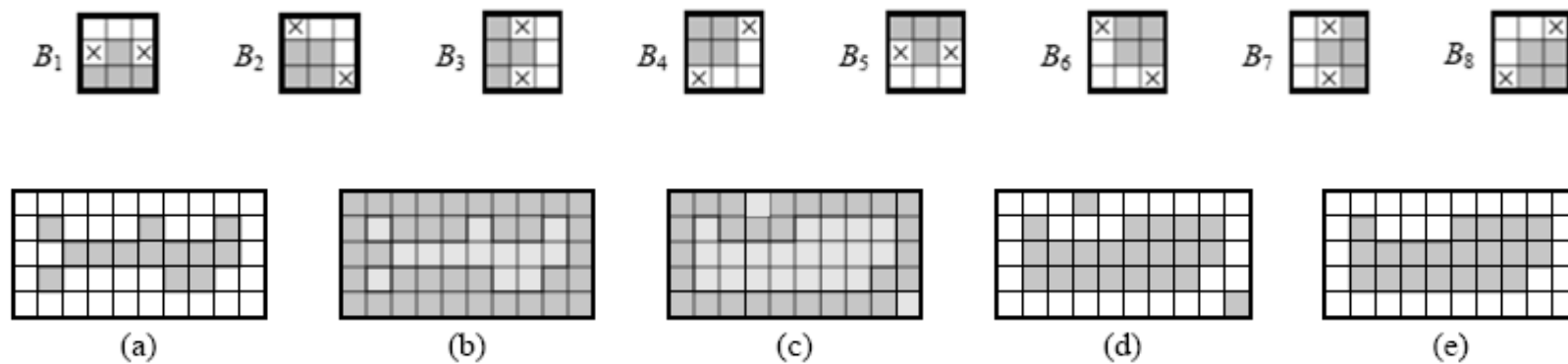


图 14.3.7 利用细化进行粗化





# 二值形态学实用算法

---

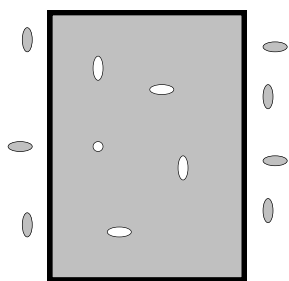
- ☐ 噪声滤除
- ☐ 目标检测
- ☐ 边界提取
- ☐ 区域填充
- ☐ 连通组元提取
- ☐ 区域骨架提取

# 二值形态学实用算法-I

## □ 噪声滤除

### ■ 先开启后闭合

$$\{[(A \ominus B) \oplus B] \oplus B\} \ominus B = (A \circ B) \cdot B$$

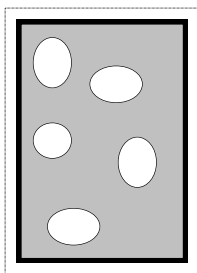


(a)



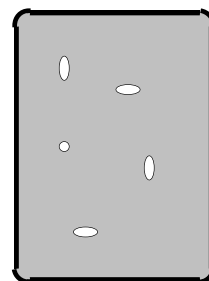
(b)

腐蚀



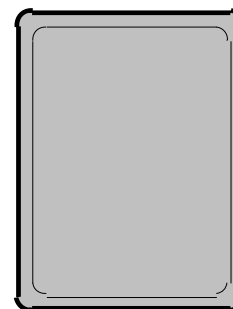
(c)

膨胀



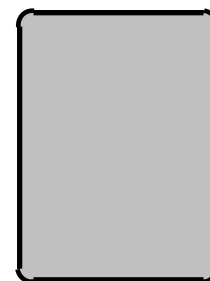
(d)

膨胀



(e)

腐蚀



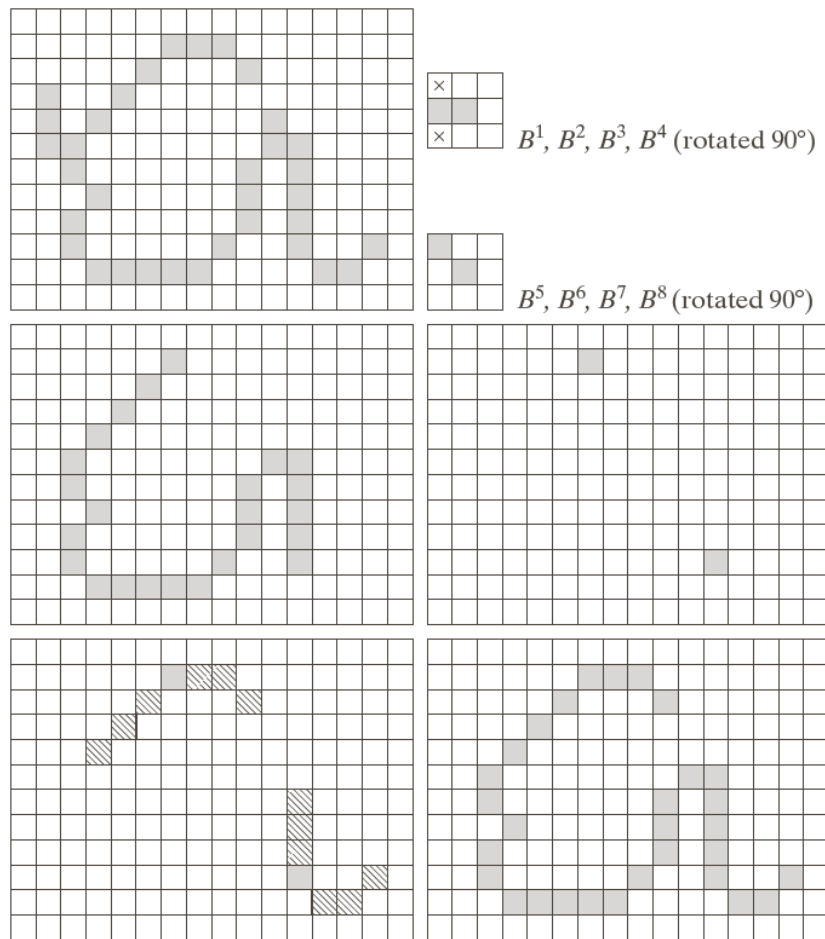
(f)

# 二值形态学实用算法-I

## □ 噪声滤除：裁剪

裁剪方法本质上是对细化和骨架算法的补充，因为这些过程会保留某些寄生成分，因而需要用**后处理**来清除这些寄生成分

在手写字符的自动识别中，通常使用的方法是分析每个字符的骨架形状。这些骨架通常带有许多“毛刺”（寄生成分）。





# 二值形态学实用算法-I

## □ 噪声滤除：裁剪

- 上图显示了手写字符“a”的骨架。该字符最左侧的寄生成分就是我们要消除的部分。使用一系列仅设计用来检测端点的结构元，对输入集合A进行细化可以得到期望的结果。令

$$X_1 = A \otimes \{B\}$$

式中， $\{B\}$ 表示结构元序列。连续对A应用三次，得到集合 $X_1$ 。

- 然后，是将字符“复原”为原来的形状，但要去掉寄生分支。首先需要求形成一个包含 $X_1$ 中所有端点的集合 $X_2$ ：

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

式中， $B^k$ 是相同端点检测子。

- 接着，用集合A作为限定器，对端点进行三次膨胀（条件膨胀）：

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

最后， $X_3$ 和 $X_1$ 的并集就是我们想要的结果：

$$X_4 = X_1 \cup X_3$$

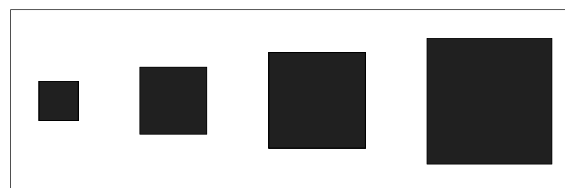
# 二值形态学实用算法-II

## □ 目标检测（击中击不中变换）

■  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$  和  $9 \times 9$  的实心正方形

$3 \times 3$  实心正方形

$9 \times 9$  方框



(a)

(b) : E

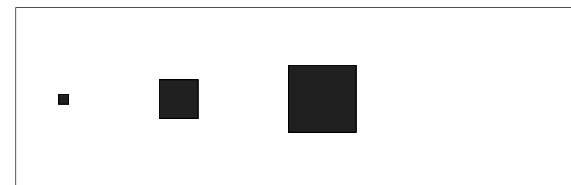
(c) : F



(b)



(c)



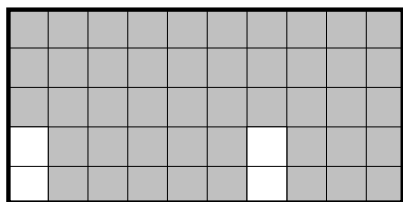
(d)

# 二值形态学实用算法-III

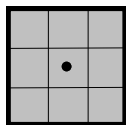
## □ 边界提取

- 先用1个结构元素B腐蚀 A，再求取腐蚀结果和A的差集就可得到边界  $\beta(A)$

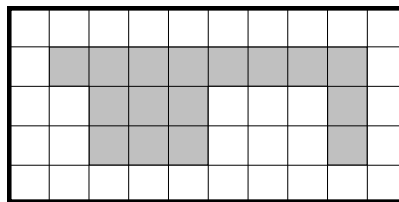
$$\beta(A) = A - (A \ominus B)$$



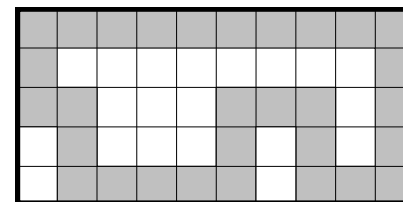
(a)



(b)



(c)



(d)

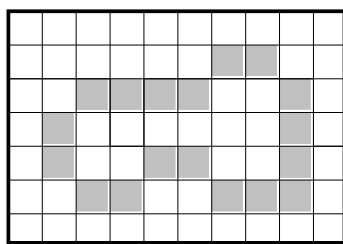
结构元素是8-连通的，而所得到的边界是4-连通的

# 二值形态学实用算法-IV

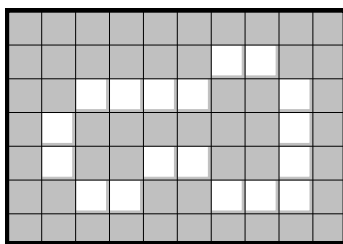
## □ 区域填充

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots$$

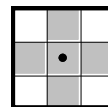
取一个点，按照模板膨胀，取交集，迭代多次；最后与（a）取并集



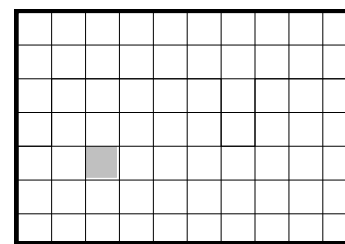
(a)



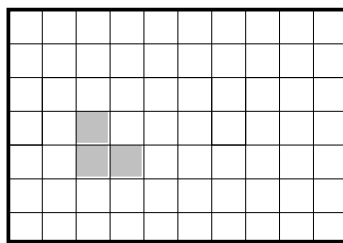
(b)



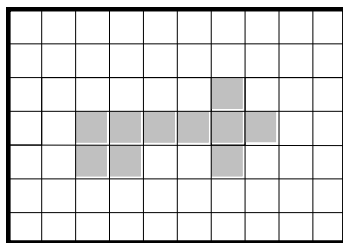
(c)



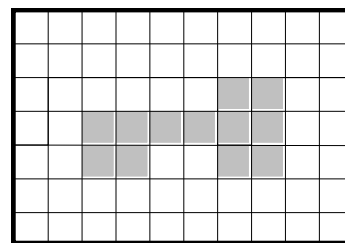
(d)



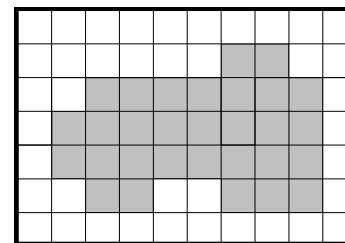
(e)



(f)



(g)



(h)

结构元素是4-连通的，而原填充的边界是8-连通的

# 二值形态学实用算法-V

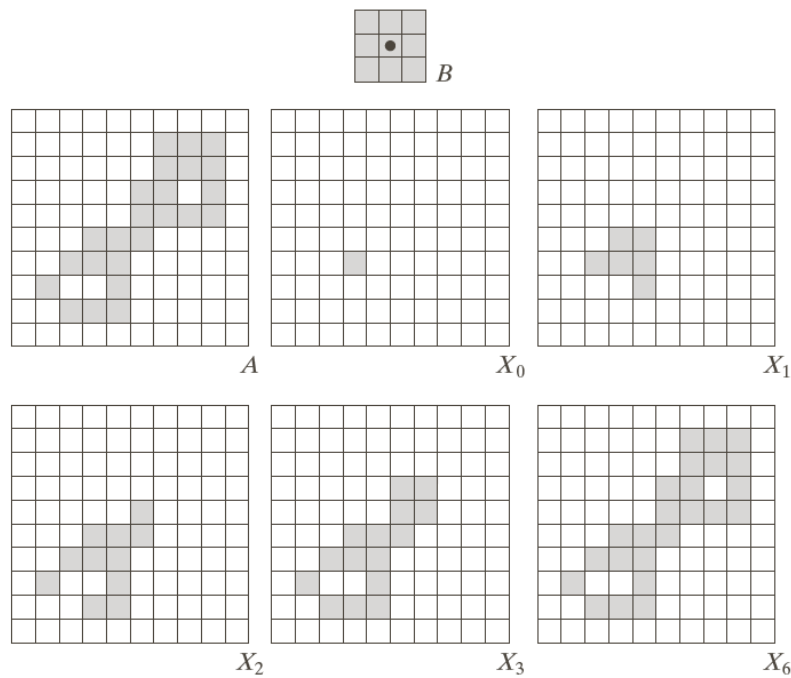
## □ 连通组元提取

- 设  $Y$  为集合  $A$  中的一个连通组元，已知  $Y$  上的一个点记为阵列  $X_0$ 。如下迭代过程可完成这一目的：

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

- 当  $X_k = X_{k-1}$  时，迭代过程结束， $X_k$  包含输入图像中的所有连通分量。

右图说明了此机理， $k=6$  时即可收敛。注意，所用结构元的形状在像素间是基于8连通的





# 二值形态学实用算法-VI

## □ 区域骨架提取

$$S(A) = \bigcup_{k=0}^K S_k(A) \quad S_k(A) = (A \ominus kB) - [(A \ominus kB) \circ B]$$

$$(A \ominus kB) = ((\dots (A \ominus B) \ominus B) \ominus \dots) \ominus B$$

$$K = \max\{k | (A \ominus kB) \neq \emptyset\}$$

## □ 也可以用骨架重构A

$$A = \bigcup_{k=0}^K (S_k(A) \oplus kB)$$

# 二值形态学实用算法-VI

## □ 区域骨架提取

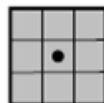
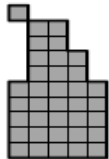
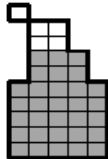
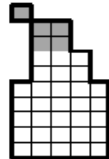
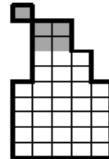
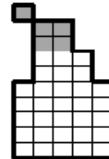
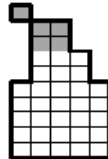
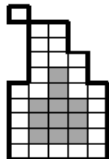
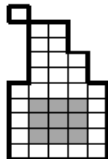
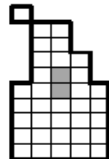
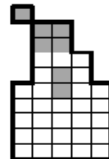
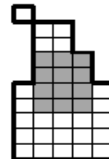
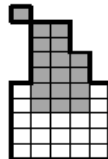
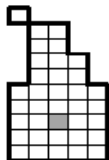
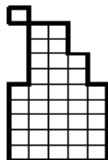
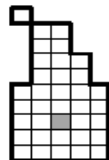
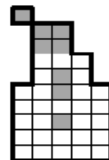
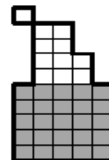
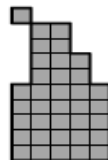


表 14.4.1 区域骨架抽取示例

列	1	2	3	4	5	6	7
运算		$A \ominus kB$	$(A \ominus kB) \circ B$	$S_k(A)$	$\bigcup_{k=0}^K S_k(A)$	$S_k(A) \oplus kB$	$\bigcup_{k=0}^K [S_k(A) \oplus kB]$
$k=0$							
$k=1$							
$k=2$							



# 图像数字化

## □ 数字图像基础

## □ 形态学

### ■ 二值形态学

- ✓ 基本定义
- ✓ 基本运算
- ✓ 实用算法

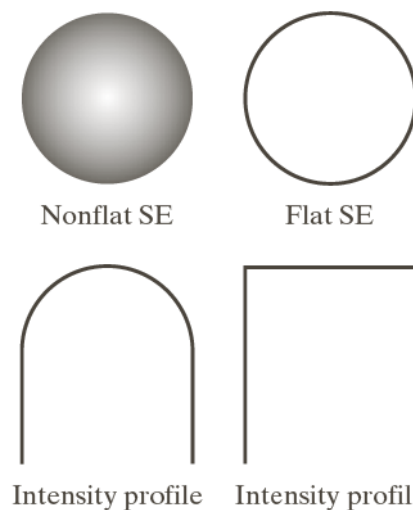
### ■ 灰度形态学

- ✓ 基本运算
- ✓ 实用算法

# 灰度形态学基本定义

- 将把膨胀、腐蚀、开闭操作扩展到灰度级图像
  - 处理形如 $f(x,y)$ 和 $b(x,y)$ 的数字图像，其中 $f(x,y)$ 是一幅灰度级图像，而 $b(x,y)$ 是一个结构元素。
  - $f$ 和 $b$ 是对每个坐标赋以灰度值（来自实数集合 $R$ 的一个实数）的函数
- 灰度级形态学中的结构元素作为一个“探测器”，以明确的特性检验一幅给定的图像
  - 结构元素分为两类：非平坦的和平坦的
  - 实际中灰度级结构元素并不常用
- 结构元的映像的定义为
$$\hat{b}(x,y) = b(-x,-y)$$

本节中所有例子都以高度为1、对称的、平坦的结构元为基础，其原点位于中心处



a b  
c d

**FIGURE 9.34** Nonflat and flat structuring elements, and corresponding horizontal intensity profiles through their center. All examples in this section are based on flat SEs.

# 灰度形态学基本运算

- 当 $b$ 的原点位于 $(x,y)$ 处时，用一个平坦的结构元 $b$ 在 $(x,y)$ 处对图像 $f$ 的腐蚀定义为图像 $f$ 中与 $b$ 重合区域的最小值。公式为：

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

- 为寻求 $b$ 对 $f$ 的腐蚀，我们把结构元的原点放在图像的每个像素的位置。
- 在任何位置的腐蚀由从包含在与 $b$ 重合区域中的 $f$ 的所有值中选取的最小值决定

# 灰度形态学基本运算

- 类似地，当 $\hat{b}$ 的原点位于位置 $(x,y)$ 处时，平坦结构元 $b$ 在任何位置 $(x,y)$ 处对图像 $f$ 的膨胀定义为图像 $f$ 中与 $\hat{b}$ 重合区域的最大值：

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

- 非平坦结构元具有随定义域变化的灰度级， $b_N$ 对图像 $f$ 的腐蚀定义：

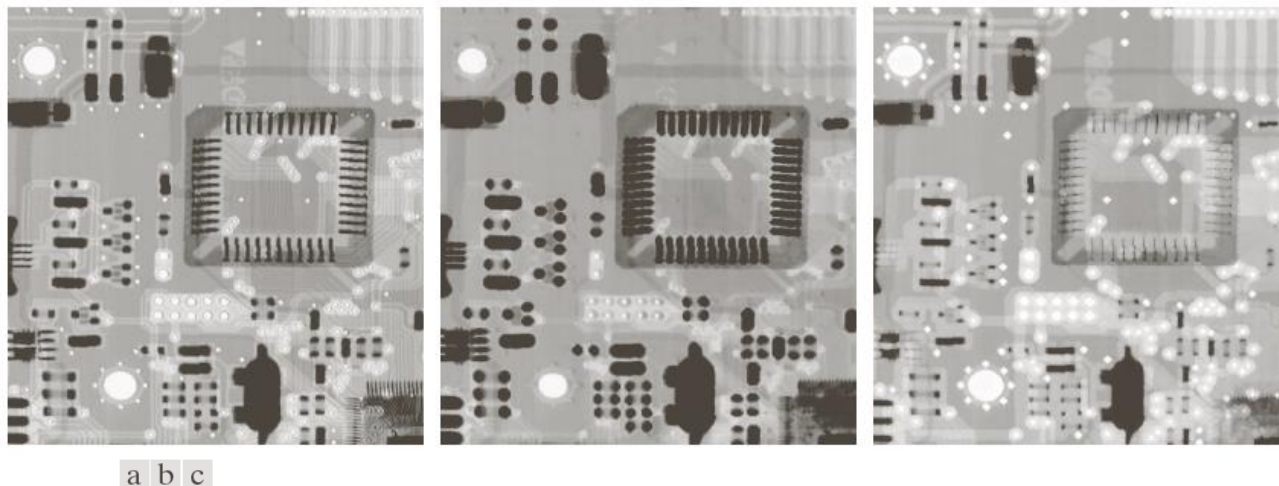
$$[f \ominus b_N](x,y) = \min_{(s,t) \in b_N} \{f(x+s, y+t) - b_N(s,t)\}$$

- 类似地，使用非平坦结构元的膨胀的定义如下：

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

# 灰度形态学腐蚀和膨胀

## □ 灰度级腐蚀和膨胀的说明



**FIGURE 9.35** (a) A gray-scale X-ray image of size  $448 \times 425$  pixels. (b) Erosion using a flat disk SE with a radius of two pixels. (c) Dilation using the same SE. (Original image courtesy of Lixi, Inc.)

如二值情况一样，腐蚀和膨胀是关于函数的补集和反射对偶的：

$$(f \ominus b)^c(x, y) = (f^c \oplus \hat{b})(x, y)$$

式中， $f^c = -f(x, y)$ 。类似地，有：

$$(f \oplus b)^c = (f^c \ominus \hat{b})$$

# 灰度形态学开操作和闭操作

- 灰度级图像的开操作和闭操作的表达式，与二值图像的对应操作具有相同的形式。结构元 $b$ 对图像 $f$ 的开操作表示为 $f \circ b$

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

- 类似地， $b$ 对 $f$ 的闭操作表示为 $f \cdot b$ ，即

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

- 开闭操作关于函数的补集和结构元的反射是对偶的：

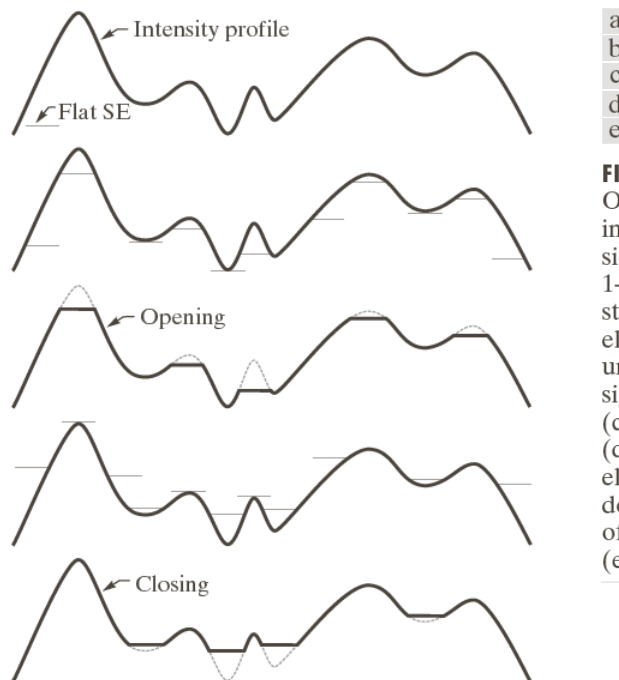
$$(f \cdot b)^c = (f^c \circ \hat{b}) \text{ 和 } (f \circ b)^c = (f^c \cdot \hat{b})$$



# 灰度形态学开操作和闭操作

## □ 几何解释

- 将一个图像函数视为一个三维表面， $b$ 对 $f$ 的开操作可从几何解释为：在 $b$ 的每个原点位置，
  - ✓ **开操作**是当从 $f$ 的下表面向上推动结构元时， $b$ 的任何部分所达到的最高值。
  - ✓ **闭操作**通过寻找结构元的任何部分所能到达的最低点构建。



**FIGURE 9.36**  
Opening and closing in one dimension. (a) Original 1-D signal. (b) Flat structuring element pushed up underneath the signal. (c) Opening. (d) Flat structuring element pushed down along the top of the signal. (e) Closing.

# 灰度形态学开操作和闭操作

- 开操作用于去除**较小的明亮细节**，而**保持整体灰度级和较大的明亮特征**相对不变
- 闭操作**抑制暗细节**
- 灰度级开操作和闭操作的说明
  - 开操作：所有亮特征的灰度都降低了，降低的程度取决于这些特征相对于结构元的尺寸
  - 与腐蚀的结果不同，开操作对图像的暗特征影响可忽略不计
  - 闭操作：亮细节和背景相对来说未受影响，但削弱了暗特征，削弱的程度取决于这些特征相对于结构元的尺寸

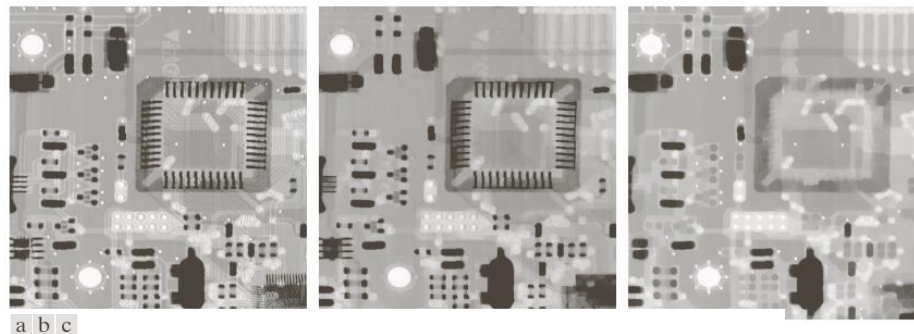
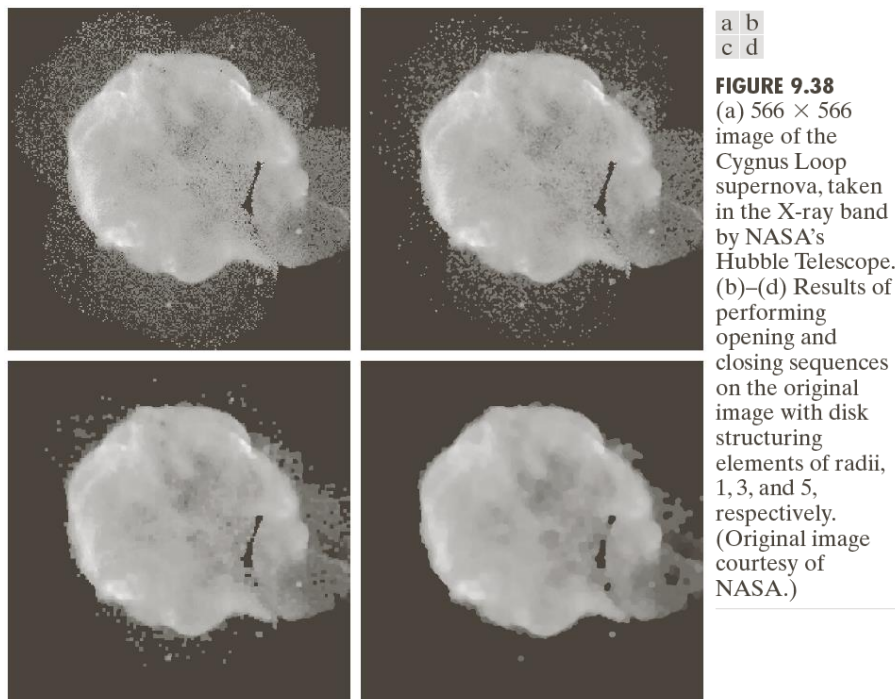


FIGURE 9.37 (a) A gray-scale X-ray image of size  $448 \times 425$  pixels. (b) Opening using a disk SE with a radius of 3 pixels. (c) Closing using an SE of radius 5.

# 灰度形态学实用算法

## □ 形态学平滑

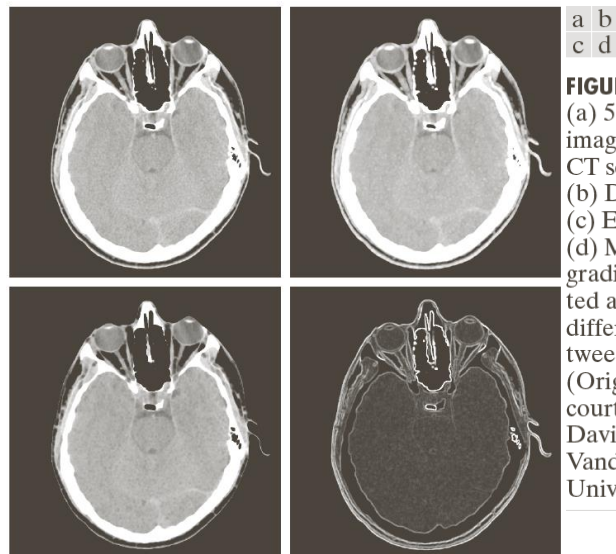
- 因为开操作抑制比结构元小的亮细节，而闭操作抑制暗细节，所以它们常常以**形态滤波**的形式结合起来**平滑图像**和**去除噪声**。
- 假设下图中心的亮区域是我们感兴趣的目标，而较小的分量是噪声。我们的目的是去除噪声。



# 灰度形态学实用算法

## □ 形态学梯度

- 膨胀和腐蚀可与图像相减结合起来得到一幅图像的**形态学梯度**，由g定义： $g = (f \oplus b) - (f \ominus b)$
- 膨胀粗化一幅图像中的区域，而腐蚀则细化它们。**膨胀和腐蚀之差强调区域间的边界**。同质区域不受影响，因此相减操作趋于消除同质区域。
- 最终结果是边缘被增强而同质区域的贡献则被抑制的图像，因此产生了“类似于微分”效果。



**FIGURE 9.39**  
(a)  $512 \times 512$  image of a head CT scan.  
(b) Dilation.  
(c) Erosion.  
(d) Morphological gradient, computed as the difference between (b) and (c).  
(Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

# 灰度形态学实用算法

## □ 顶帽变换和底帽变换

- 图像相减与开操作和闭操作相结合，可产生所谓的**Top-hat（顶帽）**变换和**Bottom-hat（底帽）**变换

- 灰度级图像 $f$ 的**顶帽变换**定义为 $f$ 减去其开操作：

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

- 类似地， $f$ 的**底帽变换**定义为 $f$ 的闭操作减去 $f$ ：

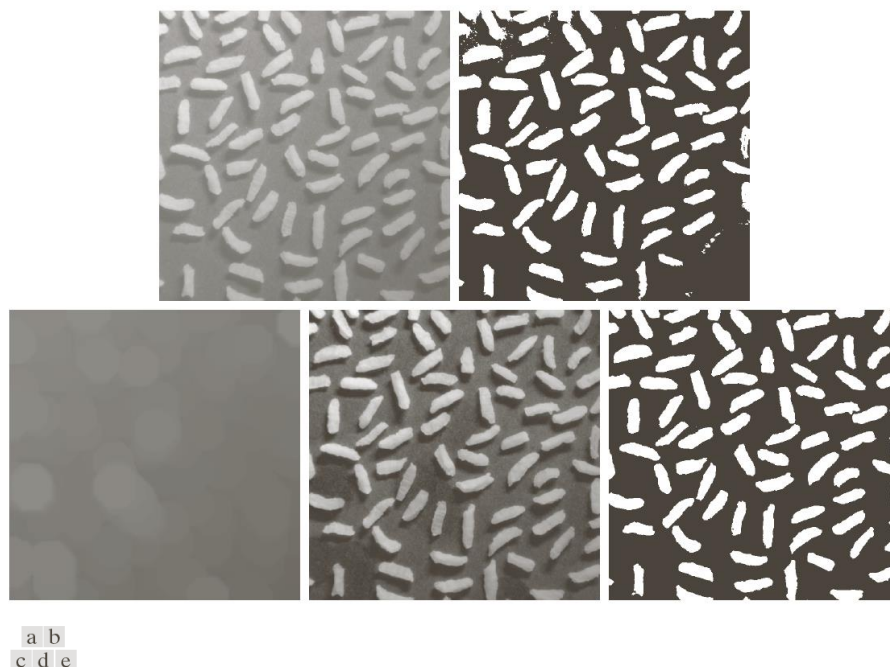
$$B_{hat}(f) = (f \cdot b) - f$$

- 这些变换的主要应用之一是，用一个结构元通过开操作或闭操作从一幅图像中**删除物体**，而不是拟合被删除的物体。然后，差操作得到一幅**仅保留已删除分量**的图像
- 顶帽变换用于暗背景上的亮物体，而底帽变换则用于相反的情况。由于这一原因，这些变换又称为**白顶帽变换**和**黑底帽变换**
- 顶帽变换的一个重要用途是校正不均匀光照的影响。合适（均匀）的光照在从背景中提取目标的处理中扮演核心的角色

# 灰度形态学实用算法

## □ 顶帽变换和底帽变换

- 下图是在非均匀光照下得到的，如图像底部及最右侧的暗色区域就是明证。非均匀光照的最终结果导致了暗区域的分割错误（一些米粒未从背景中提取出来），且在图像的左上角，背景部分被错误地分类了



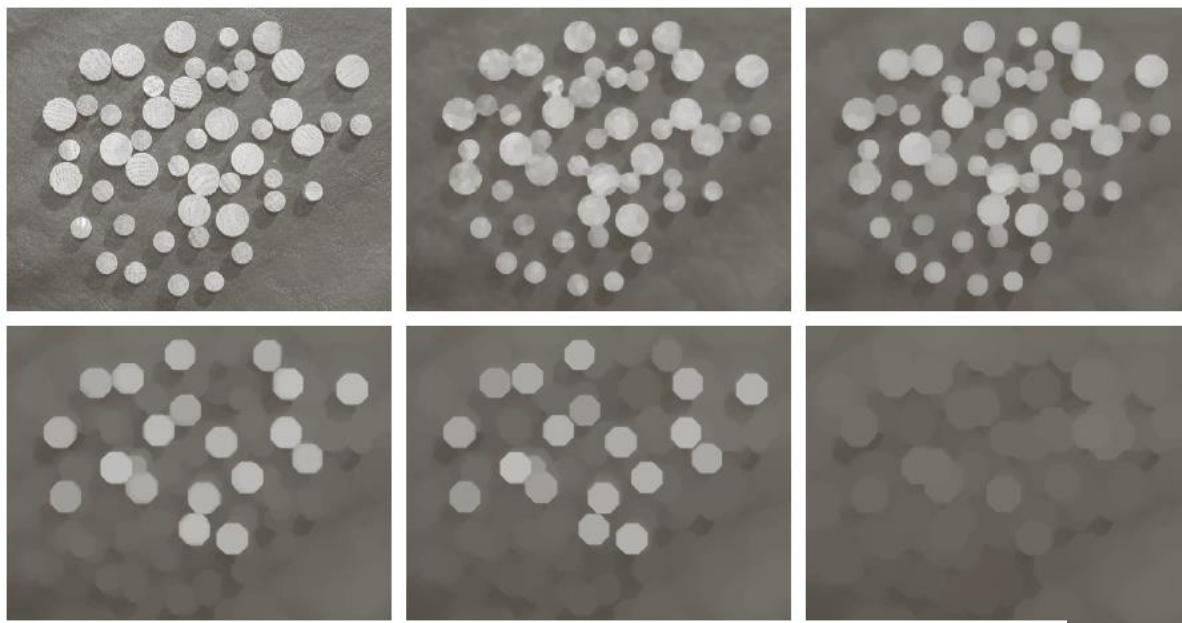
**FIGURE 9.40** Using the top-hat transformation for *shading correction*. (a) Original image of size  $600 \times 600$  pixels. (b) Thresholded image. (c) Image opened using a disk SE of radius 40. (d) Top-hat transformation (the image minus its opening). (e) Thresholded top-hat image.



# 灰度形态学实用算法

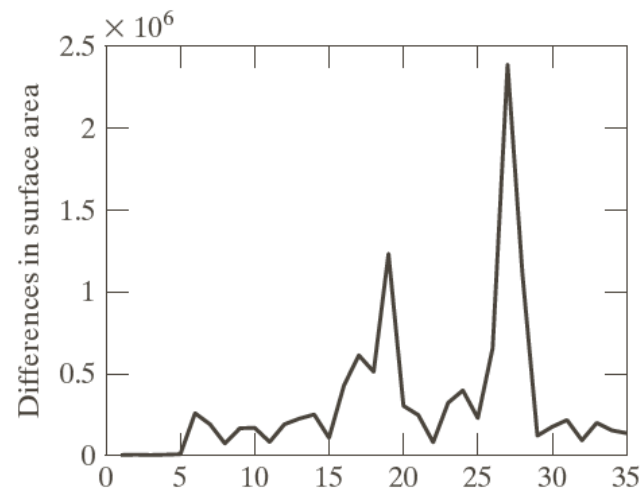
## □ 粒度测定

- 左图是两种不同大小的木钉图像
- 右图显示了该差值阵列的曲线，图中有两个明显的峰值，表面图像存在两种主要的物体尺寸



a b c  
d e f

**FIGURE 9.41** (a)  $531 \times 675$  image of wood dowels. (b) Smoothed image. (c)–(f) Openings of (b) with disks of radii equal to 10, 20, 25, and 30 pixels, respectively. (Original image courtesy of Dr. Steve Eddins, The MathWorks, Inc.)



**FIGURE 9.42** Differences in surface area as a function of SE disk radius,  $r$ . The two peaks are indicative of two dominant particle sizes in the image.

# 灰度形态学实用算法

## □ 纹理分割

- 右图有两个纹理区域：右边大斑点组成的区域和左边小斑点组成的区域。目的是以纹理内容为基础找到两个区域的边界
- 对图执行闭操作后删除了小斑点
- 然后执行开操作后删除了大斑点间的亮间隔区域
- 再对图像执行形态学梯度操作，得到两个区域的边界
- 将边界叠加到原图像上

