形态学图像处理

陈小潘

计算机与信息工程学院

目录

9.1 形态学基础

数学形态学

- ✓ 数学形态学具有一套完整的理论、方法及算法体系，是一种非线性图像处理和分析方法，是法国和德国的科学家在研究岩石结构时建立的一门学科，它摒弃了传统的数学建模和数值分析的观点，从集合的角度来刻画和分析图像。
- ✓ 形态学的用途是获取物体**拓扑和结构信息，**它通过物体和结构元素相互作用的某些运算，获取物体更本质的形态。

9.1 形态学基础

(1) 数学形态学运算的基本思想

- 形态学是建立在严格的数学理论基础上的，其数学基础是集合论。
- 数学形态学的应用可以简化图像数据， 保持它们基本的形状特性， 并除去不相干的结构。

在图像处理中的主要应用：

✓ 利用形态学的基本运算，对图像进行观察和处理，从而达到改善图像质量的目的；

✓ 描述和定义图像的各种几何参数和特征，如面积、周长、 连通度、 颗粒度、 骨架和方向性等。

9.1 形态学基础

(1) 数学形态学运算的基本思想

- 数学形态学中，用集合描述目标图像或感兴趣区域，描述图像各部分之间关系，描述目标的结构特点。
- 基于数学形态学的图像处理方法的基本思想：

用具有一定形态的结构元素（“探针”）收集图像的信息。当探针在图像中不断移动时，便可考察图像各个部分之间的相互关系， 从而了解图像的结构特征，进而达到对图像分析和识别目的。

9.1 形态学基础

(2) 结构元素

- 所有形态学处理都基于填放结构元素的概念。
- 所谓的填放，就是用不同的方法将结构元素放在原图像的内部，在结构元素的填放中引出一系列图像的特性。

- 结构元素定义

在分析目标图像时，需要创建一种几何形态滤波模板，用来收集图像信息，称之为结构元素。

(2) 结构元素

设有两幅图像B和A。若A是被处理的对象，而B是用来处理A的，则称B为结构元素，其又被形象地称做刷子。结构元素通常都是一些比较小的图像。

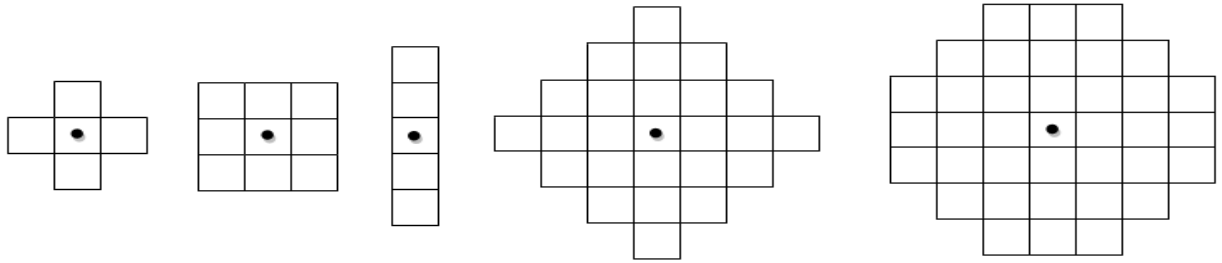
二值形态学中的运算对象是集合。设A为图像集合，S为结构元素，数学形态学运算是用S对A进行操作。

实际上结构元素本身也是一个图像集合。对每个结构元素可以指定一个原点，**它是结构元素参与形态学运算的参考点**。应注意，原点可以包含在结构元素中，也可以不包含在结构元素中，但运算结果会不相同。

9.1 形态学基础

选取结构元素的原则

- ✓ 几何结构要比原图像简单，且有界。
- ✓ 尺寸要明显小于目标图像的尺寸。
- ✓ 形状具有某种凸性，如圆形、十字架、方形等。
- ✓ 需指定一个原点，作为运算的“参考点”。



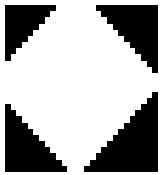
不同形状的结构元素

9.1 形态学基础

SE = strel('diamond',3);%创建一个大小为3的菱形结构元素，太小的话看不来是菱形

GN=getnhood(SE);%获取结构元素的邻域

figure,imshow(GN,[]);



边长为3的
菱形结构元素

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

结构元素矩阵

9.2 二值形态学的基本运算

9.2.1 基本运算

9.2.2 形态滤波

9.2.3 边缘提取

9.2.4 区域填充

9.2.5 击中与否变换

9.2.6 骨架

9.2.7 形态学平滑

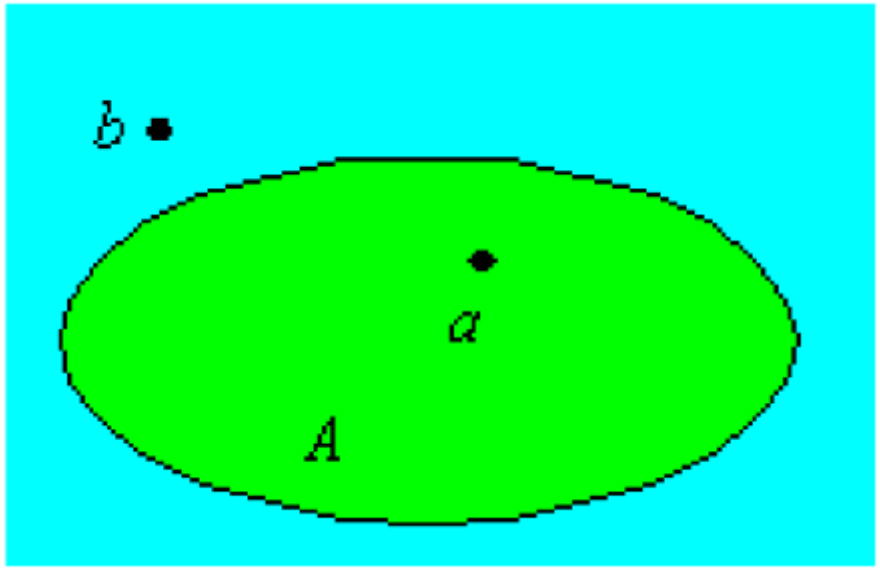
9.2.8 形态学梯度

9.2.9 高帽/低帽变换

9.2.1 基本运算

在数字图像处理的数学形态学运算中， 将一幅图像称为一个集合。

- 对于一幅图像A，如果点a在A的区域以内，那么就说a是A的元素，记为$a \in A$；否则，记作$a \notin A$。



9.2.1 基本运算

B包含于A

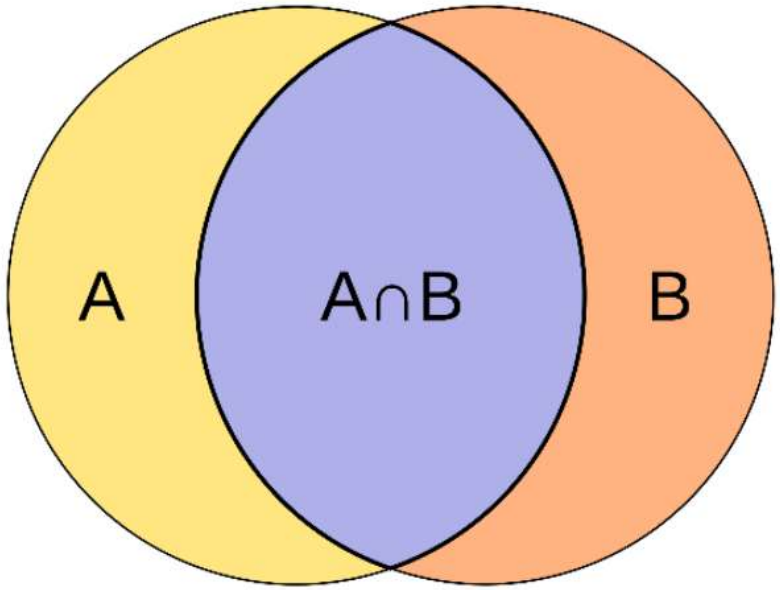
- 设有两幅图像A和B。如果对于B中所有的元素a_i，都有$a_i \in A$，则称B包含于A，记作$B \subset A$。

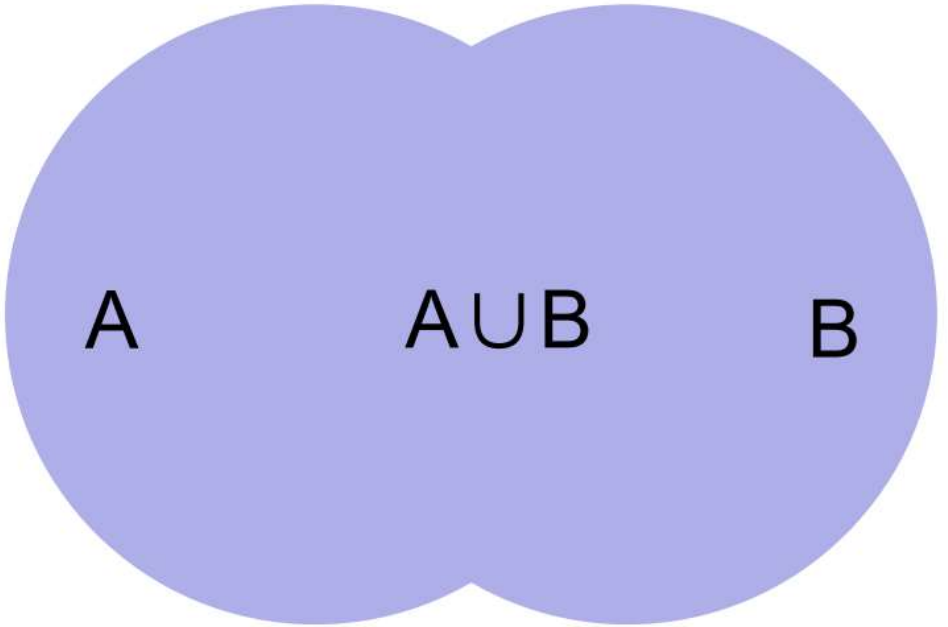
交集和并集

- 两个图像集合A和B的公共点组成的集合称为两个集合的交集，记为$A \cap B$。即，$A \cap B = \{a | a \in A$且$a \in B\}$。
- 两个图像集合A和B的所有元素组成的集合称为两个集合的并集，记为$A \cup B$。即，$A \cup B = \{a | a \in A$或$a \in B\}$。

9.2.1 基本运算

交集和并集

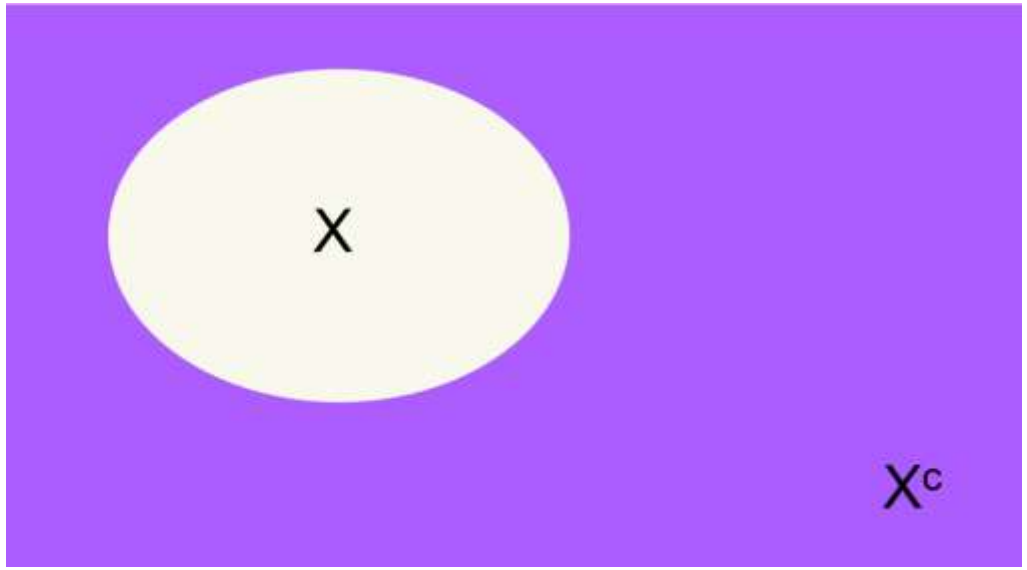




9.2.1 基本运算

补集

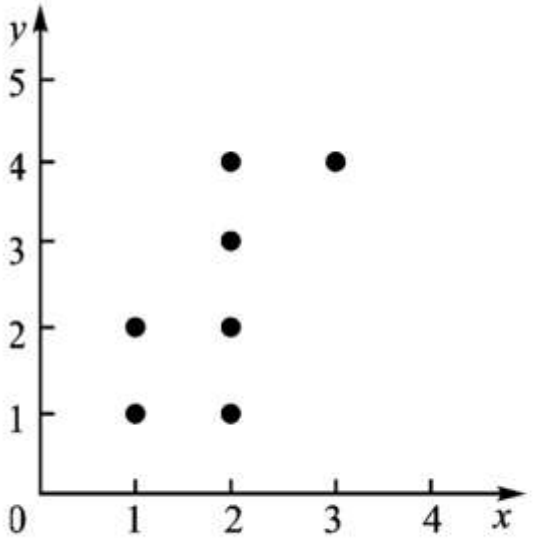
- 设有一幅图像X，所有X区域以外的点构成的集合称为X的补集，记作X^c。
- 显然，如果$B \cap X=\varnothing$，则B在X的补集内。



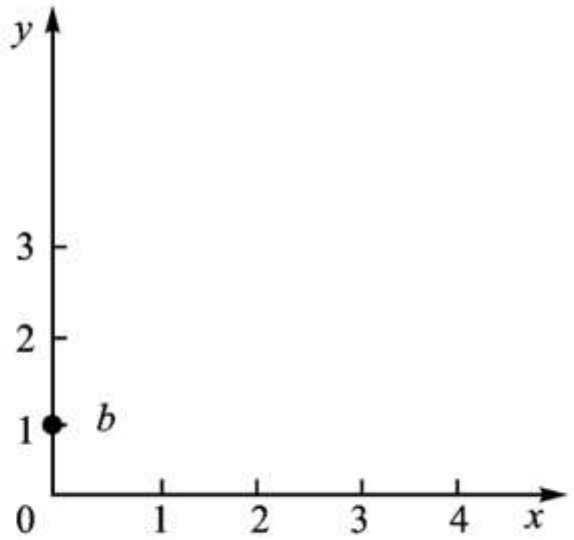
9.2.1 基本运算

平移

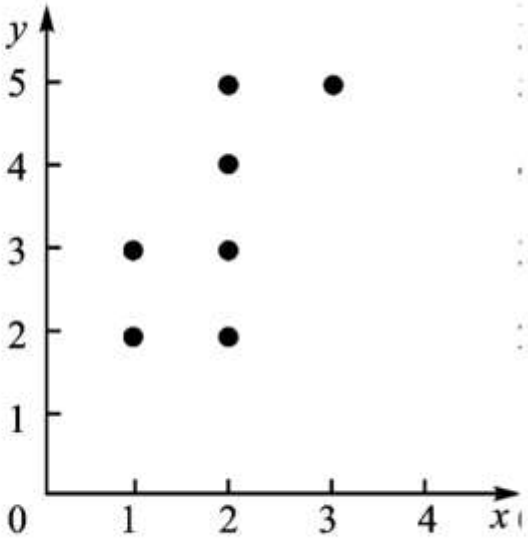
设A是一幅数字图像，b是一个点，那么定义A被b平移后的结果为$A+b=\{a+b|a \in A\}$，即取出A中的每个点a的坐标值，将其与点b的坐标值相加，得到一个新的点的坐标值$a+b$，所有这些新点所构成的图像就是A被b平移的结果，记为$A+b$。



(a) 数字图像A



(b) 结构元素b



(c) A被b平移的结果

9.2.1 基本运算

1、腐蚀运算

腐蚀（Erosion）是一种消除连通域的边界点，使边界向内部收缩的数学形态学运算。

用途：腐蚀运算具有消除图像中比结构元素小的成分的作用，可以去除物体之间的粘连，消除图像中的小颗粒噪声。

9.2.1 基本运算

1、腐蚀运算

设A为目标图像，B为结构元素，则目标图像A被结构元素B腐蚀可定义为：

$$A\Theta B = \{x|(B)_y \subseteq A\}$$

其中，y是一个表示集合平移的位移量。

结构元素B平移y后，形成新的集合$(B)_y$。若$(B)_y$完全包含在X中，记录B的参考点位置，所得即为腐蚀的结果。

9.2.1 基本运算

1、腐蚀运算

腐蚀运算的含义：每当在目标图像A中找到一个与结构元素B**相同**的子图像时，就将该子图像中与B的原点位置对应的那个像素位置标注为1，图像A上标注出的所有这样的像素组成的集合，即为腐蚀运算的结果。

简而言之，腐蚀运算的实质就是在目标图像中标出那些与结构元素相同的子图像的原点位置的像素。

注意：结构元素中的原点位置可以不为1，但要求目标图像中的子图像与结构元素B的原点对应的那个位置的像素值是1。

9.2.1 基本运算

1、腐蚀运算

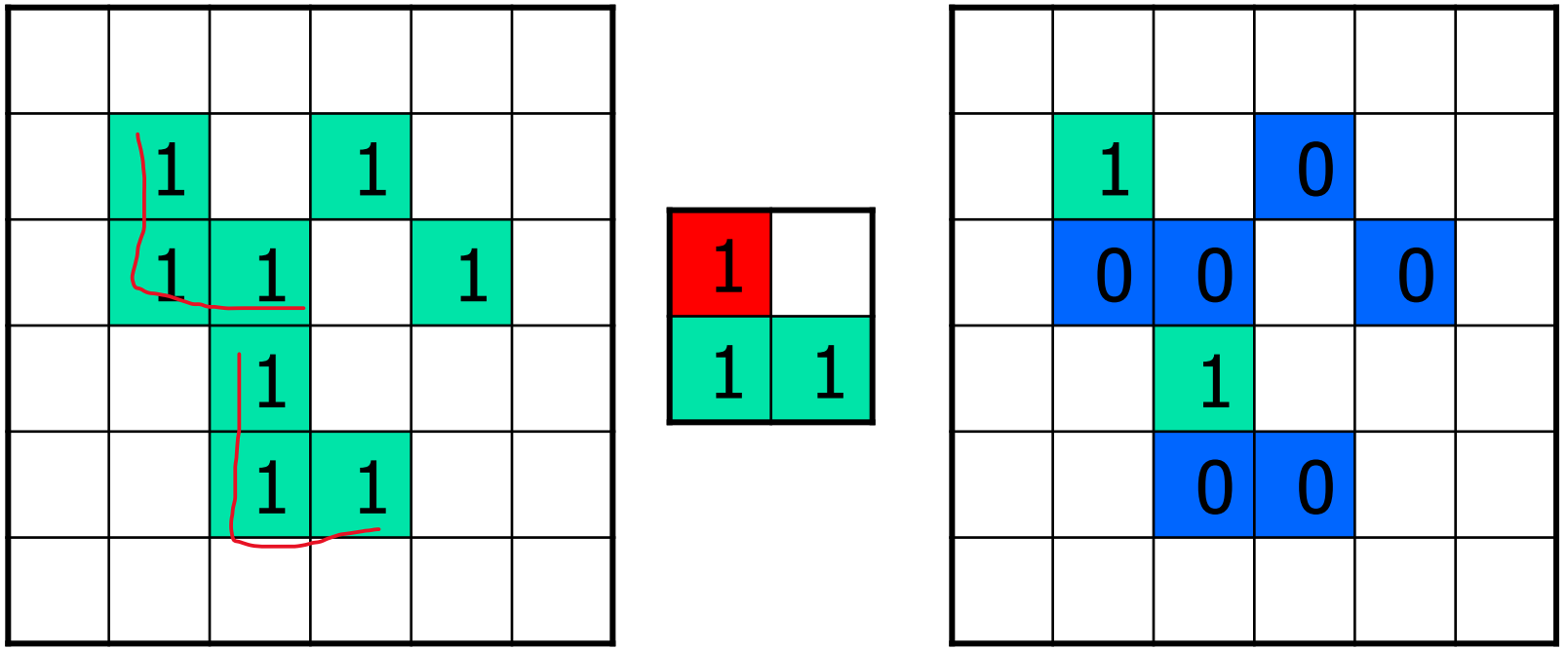
腐蚀运算的基本过程也可以这样理解：将结构元素B看作为一个卷积模板，每当**结构元素的原点及像素值为1的位置平移到与目标图像A中的那些像素值为“1”的位置重合**时，就认为结构元素覆盖的子图像的值与结构元素相应位置的像素值相同，就将目标图像中的那个与原点位置对应的像素位置的值置为“1”，否则置为0。

注意：当结构元素在目标图像上平移时，结构元素中的任何元素不能超出目标图像的范围。

$$A\Theta B = \{x|(B)_y \subseteq A\}$$

9.2.1 基本运算

1、腐蚀运算

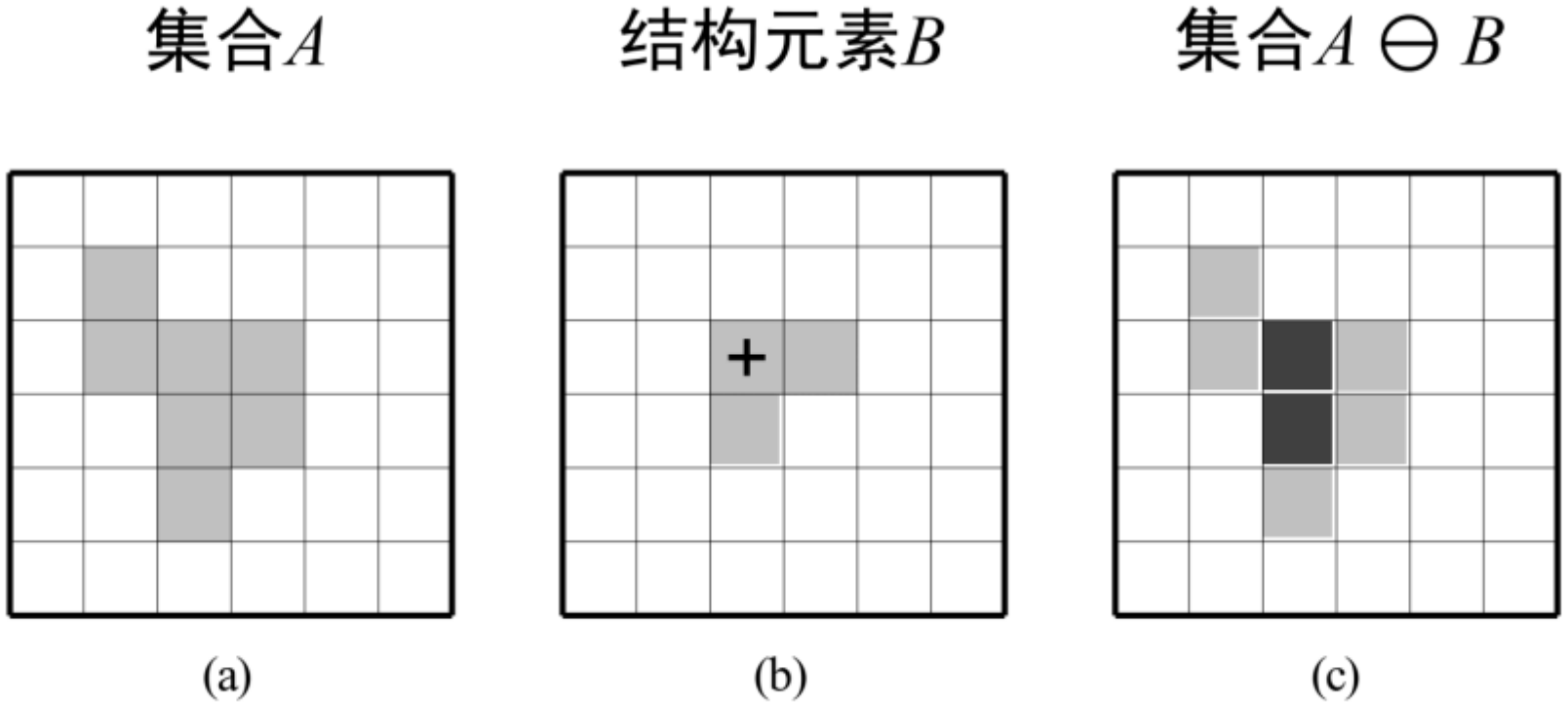


（a）目标图像A　　（b）结构元素B　　（c）腐蚀运算结果图像

图　腐蚀运算实例（红色为原点）

9.2.1 基本运算

1、腐蚀运算



9.2.1 基本运算

1、腐蚀运算

结构元素形状对腐蚀运算结果的影响：腐蚀运算的结果不仅与结构元素的形状(矩形、圆形、菱形等)选取有关，而且还与原点位置的选取有关。

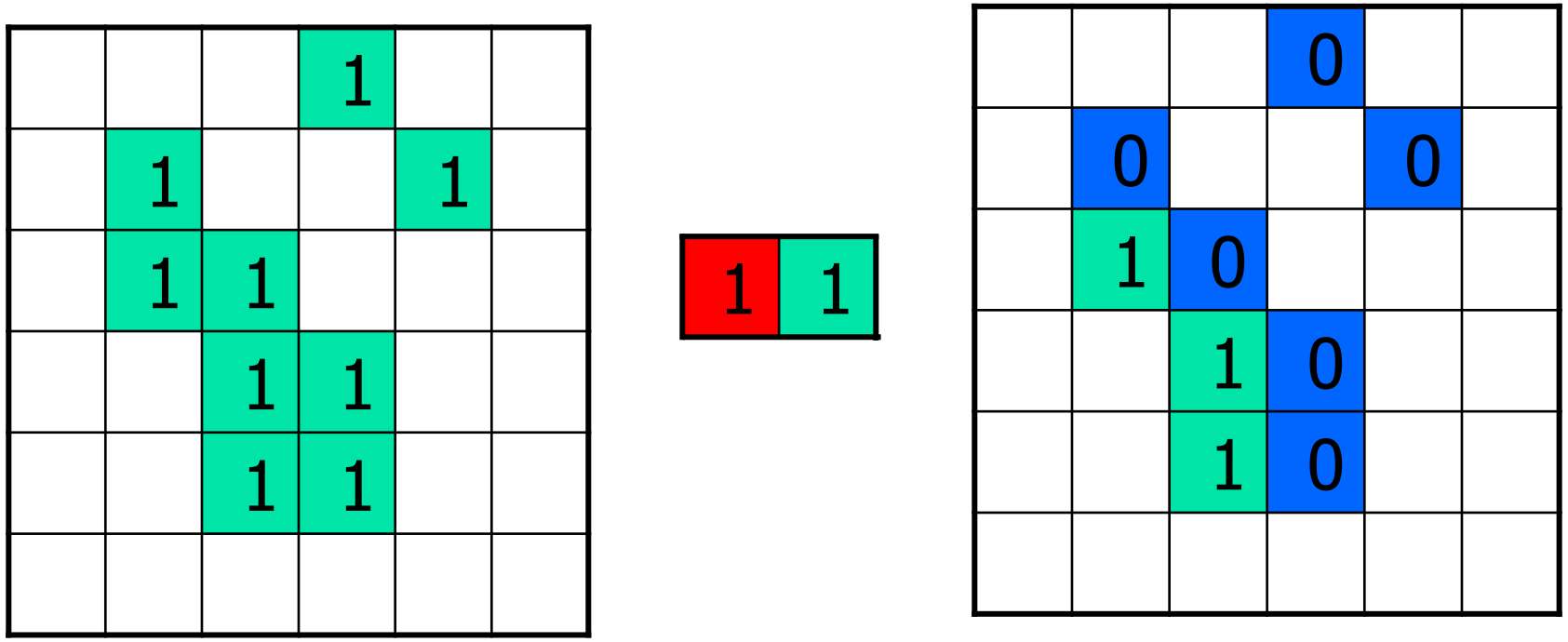


图　结构元素不同时的腐蚀运算实例

9.2.1 基本运算

1、腐蚀运算

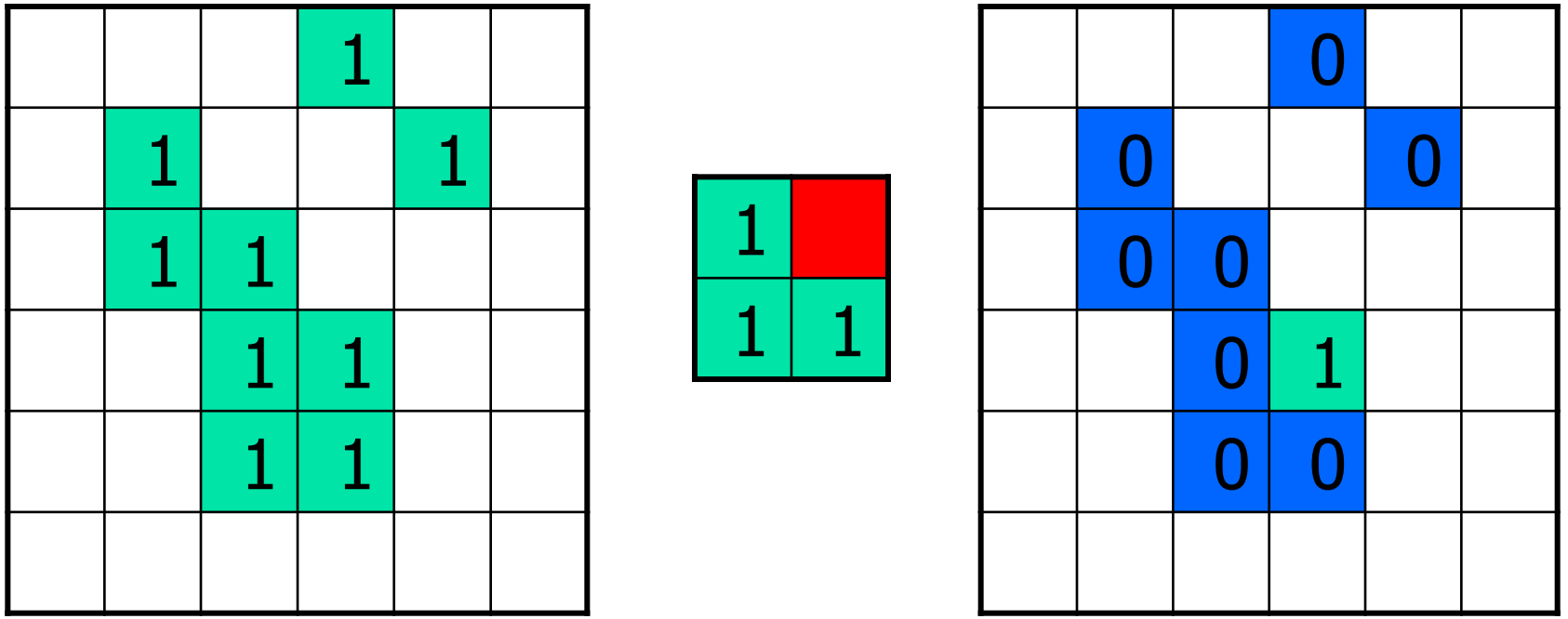


图　结构元素的原点不同时的腐蚀运算实例

9.2.1 基本运算

2、膨胀运算

膨胀（Dilation）是一种将与物体接触的所有背景点合并到物体中，使边界向外部扩张的数学形态学运算。

用途：膨胀运算具有填充图像中比结构元素小的成分的作用，可以连接相邻的物体或目标区域，填充图像中的小孔和狭窄的缝隙。

设A为目标图像，B为结构元素，则目标图像A被结构元素B膨胀可定义为：

$$A \oplus B = \{x|((\hat{B})_y \cap A) \neq \Phi\}$$

其中，$\hat{B}$是B的反射元素，y是一个表示集合平移的位移量。

9.2.1 基本运算

2、膨胀运算

膨胀的含义：先对结构元素B做关于其原点的反射得到反射集合$\hat{B}$，然后再在目标图像A上将$\hat{B}$平移y，则那些$\hat{B}$平移后与目标图像A至少有1个非零公共元素相交时，对应的$\hat{B}$的原点位置所组成的集合，就是膨胀运算的结果。

膨胀运算的基本过程是：

（1）求结构元素B关于其原点的反射集合$\hat{B}$；

（2）每当结构元素$\hat{B}$在目标图像A上平移后，结构元素$\hat{B}$与其覆盖的子图像中至少有一个元素相交时，就将目标图像中与结构元素$\hat{B}$的原点对应的那个位置的像素值置为“1”，否则置为0。

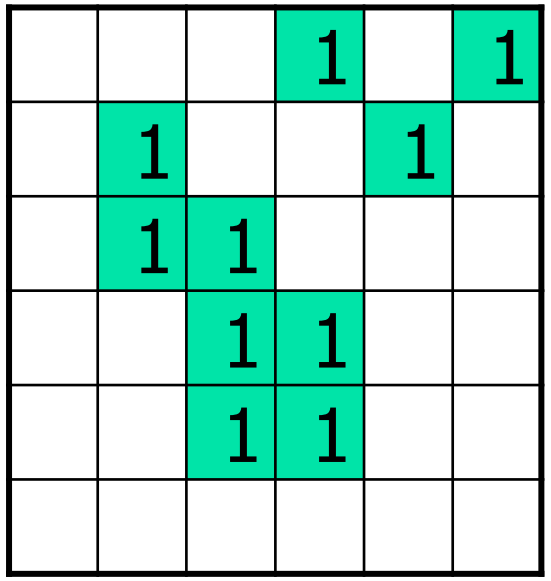
9.2.1 基本运算

2、膨胀运算

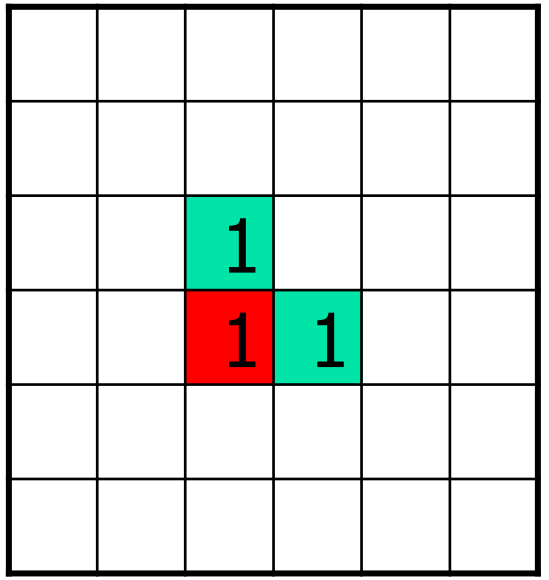
注意：

（1）当结构元素中原点位置的值是0时，仍将它看作是0，而不再将它看作是1。

（2）当结构元素在目标图像上平移时，允许结构元素中的非原点像素超出目标图像范围。



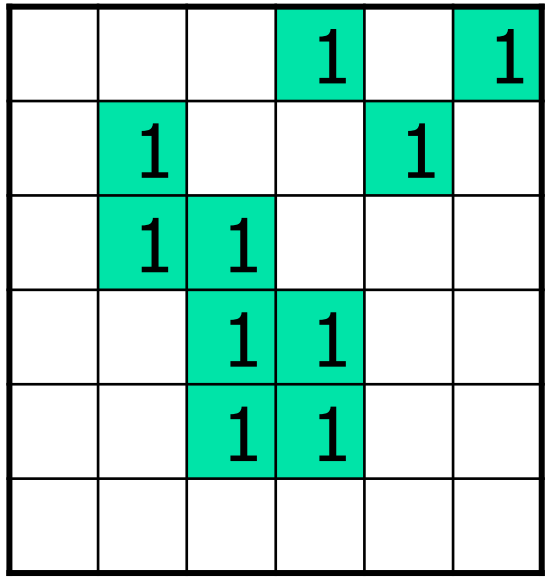
（a）目标图像A



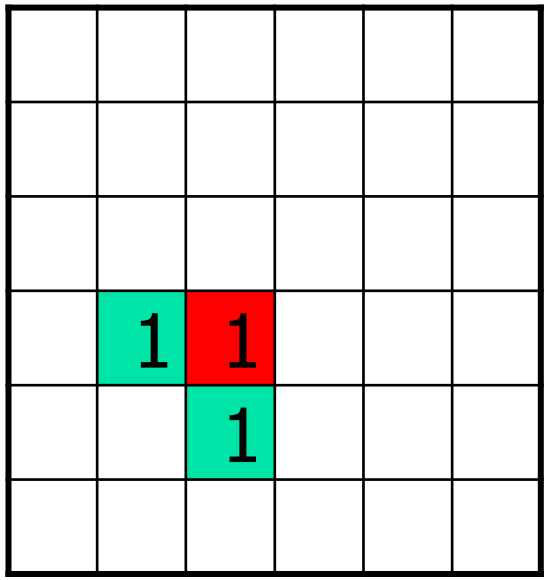
（b）结构元素B

9.2.1 基本运算

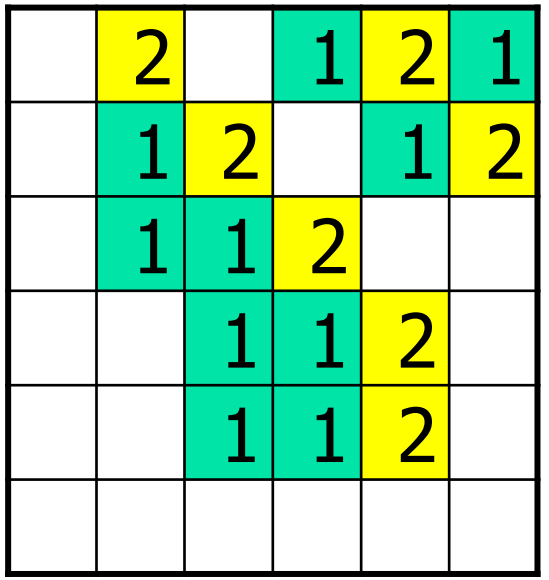
2、膨胀运算



（a）目标图像A



（c）结构元素$\hat{B}$



（d）膨胀运算结果图像

图中“2”表示膨胀结果图像中与原图像相比增加的部分，赋予2仅仅是为了强调被膨胀的部分，实际像素值应为1。

9.2.1 基本运算

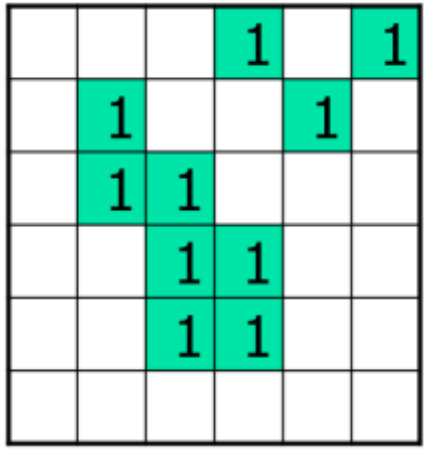
2、膨胀运算

结构元素形状对膨胀运算结果的影响：当目标图像不变，但所给的结构元素的形状改变时；或结构元素的形状不变，而其原点位置改变时，膨胀运算的结果都会发生改变。

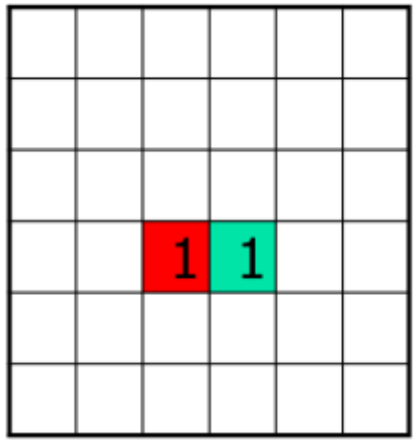
9.2.1 基本运算

2、膨胀运算

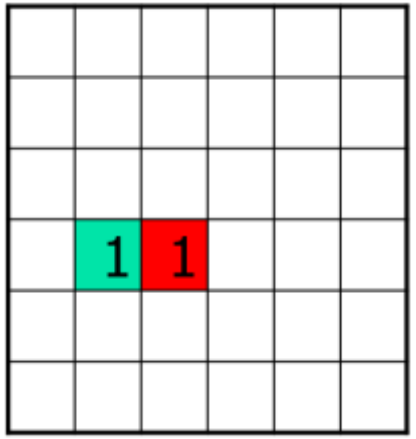
下面给出的是与目标图像相同但结构元素不同时，膨胀运算结果不同的例子。



（a）目标图像A



（b）结构元素B



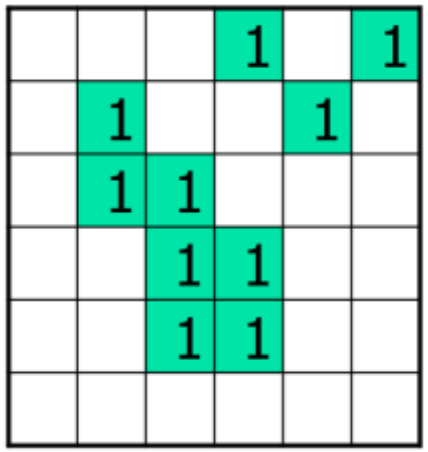
（c）结构元素$\hat{B}$

（d）膨胀运算结果图像

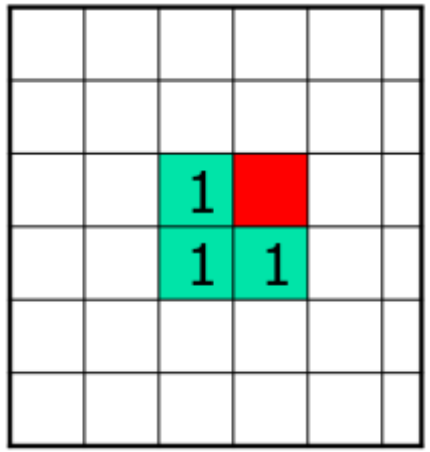
9.2.1 基本运算

2、膨胀运算

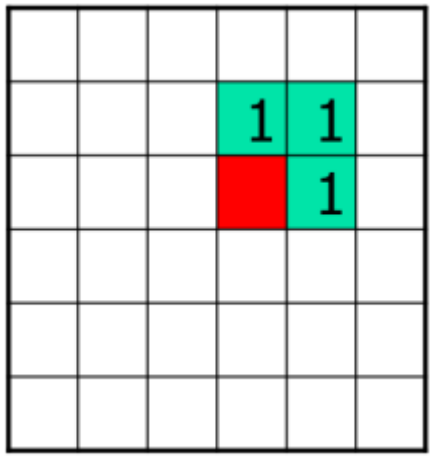
下面给出的是仅结构元素的原点位置改变时，膨胀运算结果不同的例子。



（a）目标图像A



（b）结构元素B



（c）结构元素$\hat{B}$

（d）膨胀运算结果图像

图中标注为0的位置是因为结构元素没有跟图像的区域没有公共的非零元素相交。

9.2.1 基本运算

2、膨胀运算　使用自定义的函数实现

```
clc,clear,close all;
Image=imread('img9_4.bmp');      %打开图像
BW=imbinarize(Image);             %灰度图像转为二值图像
[h,w]=size(BW);                   %获取图像尺寸
result1=zeros(h,w);               %定义输出图像，初始化为0
%执行膨胀运算
for x=2:w-1
   for y=2:h-1                    %扫描图像每一点，即结构元素移动到每一个位置
      for m=-1:1
         for n=-1:1               %当前点周围3×3范围，即结构元素为3×3大小
            if BW(y+n,x+m)        %结构元素所覆盖3×3范围内有像素点为1，即交集不为空
               result1(y,x)=1;  %将参考点记录为前景点
               break;
            end
         end
      end
   end
end
```

9.2.1 基本运算

2、膨胀运算

```
result2=ones(h,w);              %定义腐蚀的输出图像，初始化为1
for x=2:w-1
   for y=2:h-1                  %扫描图像每一点，即结构元素移动到每一个位置
      for m=-1:1
         for n=-1:1             %当前点周围3×3范围，即3×3结构元素所覆盖范围
           if BW(y+n,x+m)==0  %该范围内有像素点为0，即该位置不能完全包含结构元素
              result2(y,x)=0;  %将参考点记录为背景点，即腐蚀掉
              break;
           end
         end
      end
   end
end
subplot(1,3,1),imshow(BW);title('二值图像');
subplot(1,3,2),imshow(result1);title('二值图像膨胀');
subplot(1,3,3),imshow(result2); title('二值图像腐蚀');
```

9.2.1 基本运算

2、膨胀运算

```
%膨胀和腐蚀Matalab自带
clc,clear,close all;
Image=imread('img9_4.bmp');        %打开图像
BW=imbinarize(Image);              %灰度图像转为二值图像
[h,w]=size(BW);                    %获取图像尺寸
SE=strel('square',3);              %创建结构元素
result1=imdilate(BW,SE);           %膨胀运算
subplot(1,3,1),imshow(BW);title('二值图像');
subplot(1,3,2),imshow(result1);title('二值图像imdilate');
result2=imerode(BW,SE);            %腐蚀运算
subplot(1,3,3),imshow(result2);title('二值图像imerode');
```

9.2.1 基本运算

2、膨胀运算

运行结果

9.2.1 基本运算

3、膨胀、腐蚀的性质

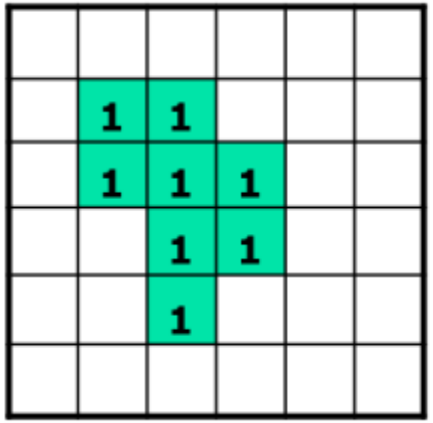
（1）对偶性

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

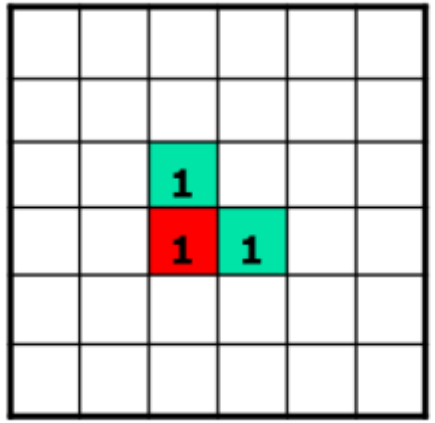
对目标图像的膨胀运算，相当于对图像背景的腐蚀运算操作；对目标图像的腐蚀运算，相当于对图像背景的膨胀运算操作。

9.2.1 基本运算

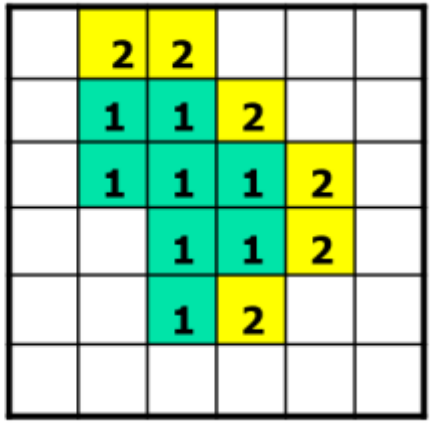
腐蚀运算与膨胀运算的对偶性－示例



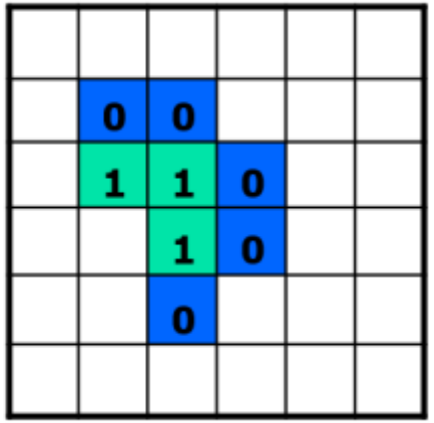
(a) 目标图像A



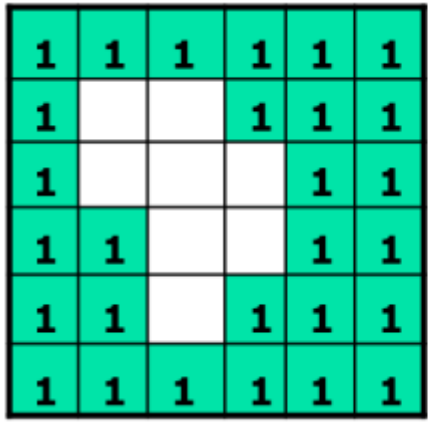
(b) 结构元素B



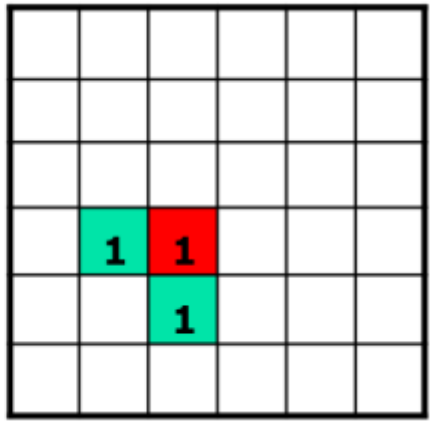
(c) 膨胀$A \oplus B$



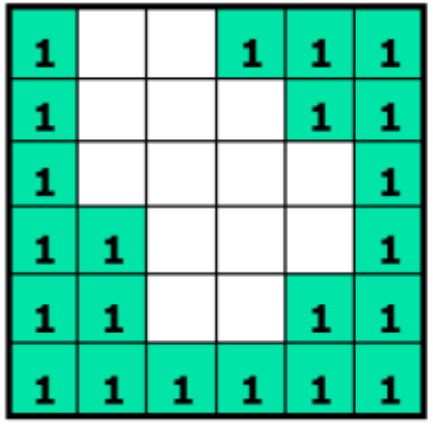
(d) 腐蚀$A \Theta B$



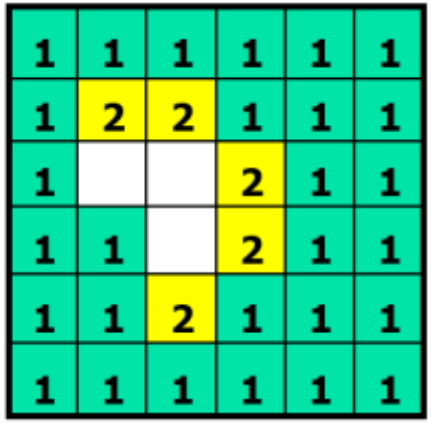
(e) A的补A^c



(f) B的反射$\hat{B}$



(f) 腐蚀$A^c \Theta \hat{B}$



(h) 膨胀$A^c \Theta \hat{B}$

9.2.1 基本运算

3、膨胀、腐蚀的性质

- 互换性

$$X \oplus S = S \oplus X$$

- 结合性

$$X \oplus (S_1 \oplus S_2) = (X \oplus S_1) \oplus S_2$$
$$(X \ominus S_1) \ominus S_2 = X \ominus (S_1 \oplus S_2)$$

- 增长性

$$X \subseteq Y \Rightarrow \begin{cases} (X \oplus S) \subseteq (Y \oplus S) \\ (X \ominus S) \subseteq (Y \ominus S) \end{cases}$$

9.2.1 基本运算

4、开、闭运算

膨胀和腐蚀进行级连结合使用，产生新的形态变换，即开运算(Opening)和闭运算(Closing)。

- **开运算：**用结构元素对图像先腐蚀，再膨胀。

$$X \circ S = (X \ominus S) \oplus S$$

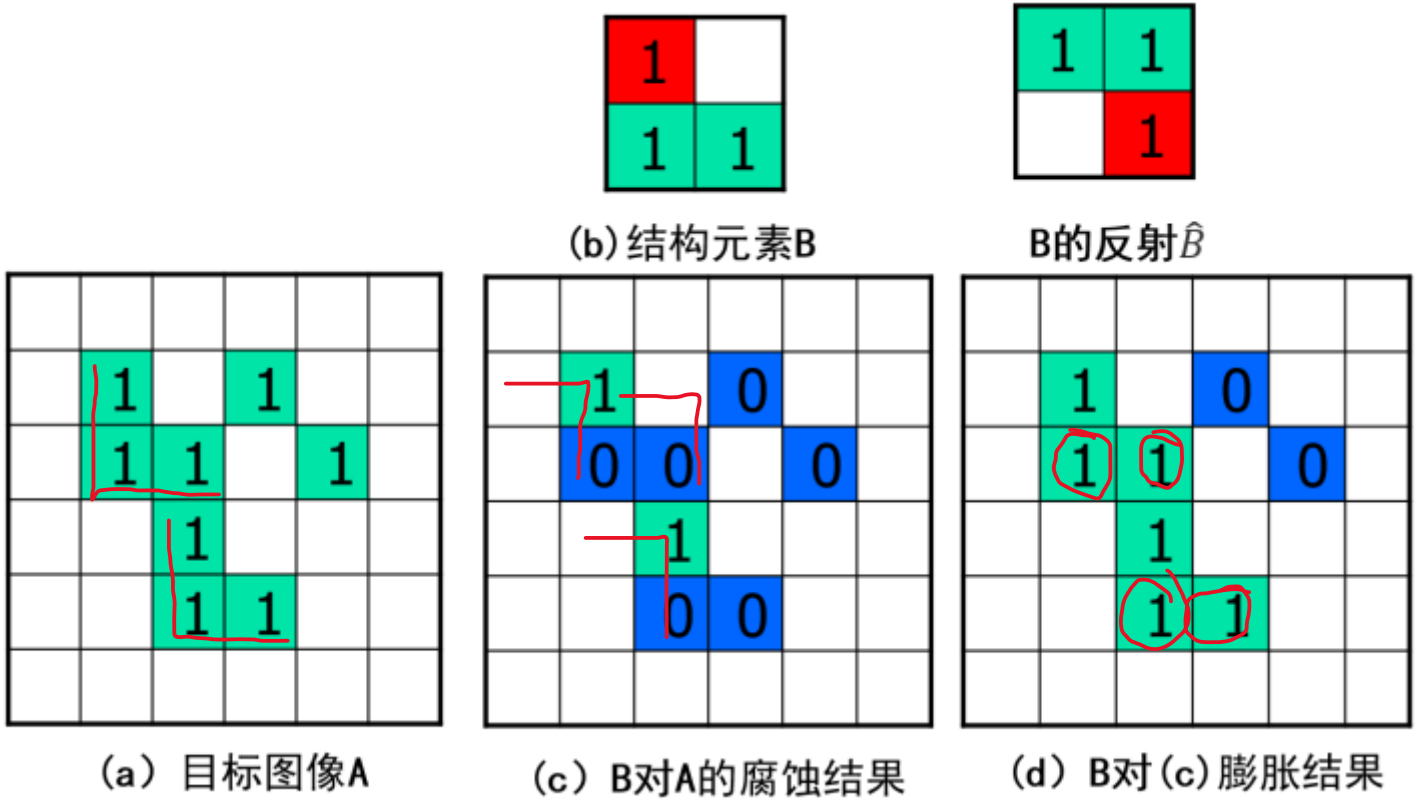
- 闭运算：用结构元素对图像先膨胀，再腐蚀。

$$X \cdot S = (X \oplus S) \ominus S$$

9.2.1 基本运算

4、开、闭运算

开运算举例



9.2.1 基本运算

4、开、闭运算

(a)印刷电路板二值图像　　(b) 对(a)进行开运算的结果图像

图 对含噪声的印刷电路板图像进行开运算实例

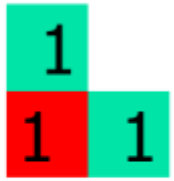
可以看出，开运算有效地平滑了电路的边界，较好地消除了图像中的颗粒噪声，并在短路点将物体进行了分离。实际工作中，通常利用该算法结合形态滤波算法检测电路中的短路点。

9.2.1 基本运算

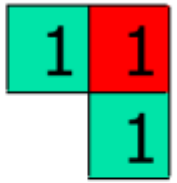
4、开、闭运算

闭运算举例

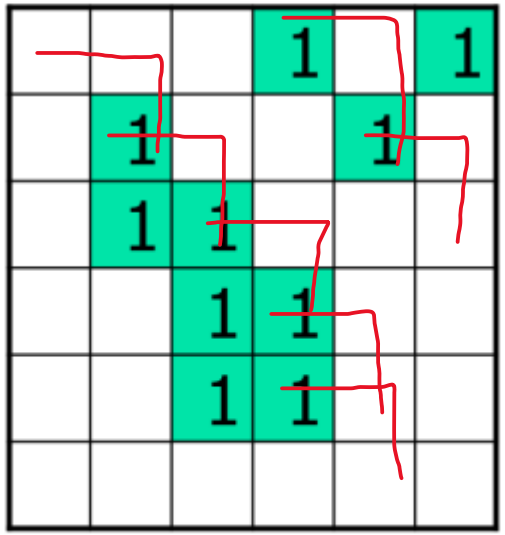
$$X \cdot S = (X \oplus S) \Theta S$$



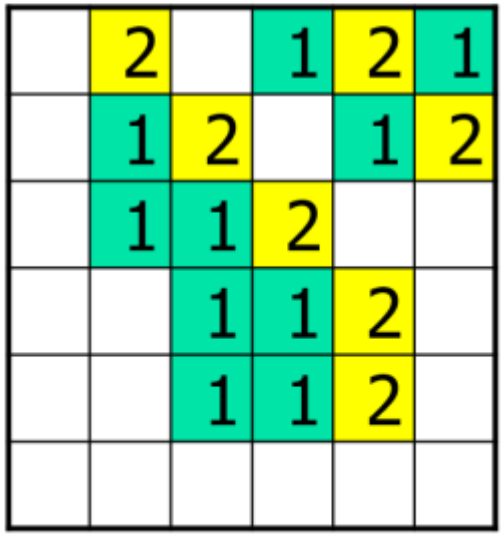
结构元素B



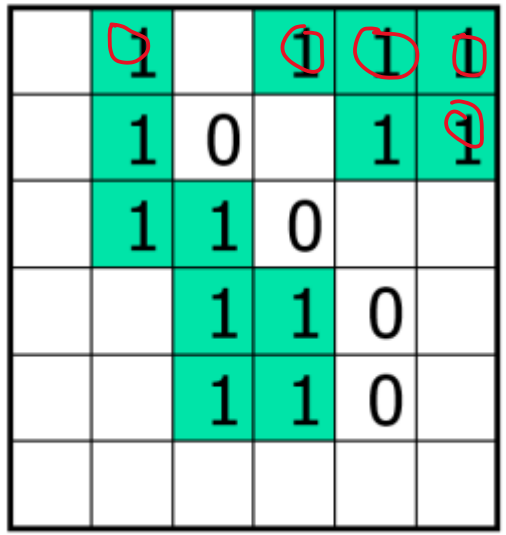
B的反射$\hat{B}$



(a) 目标图像A



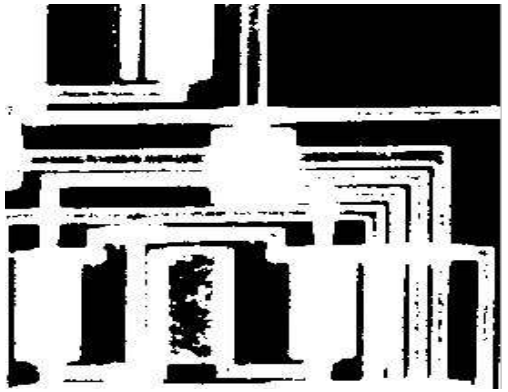
(c) B对A的膨胀结果



(d) B对(c) 腐蚀结果

9.2.1 基本运算

4、开、闭运算



(a)电路板二值图像

（b）对(a)进行闭运算的结果图像

图　电路板二值图像闭运算实例

比较图（a）和图（b）可知：图（a）电路中存在小孔洞和狭窄的间断。在图（b）中已经得到了有效的处理。

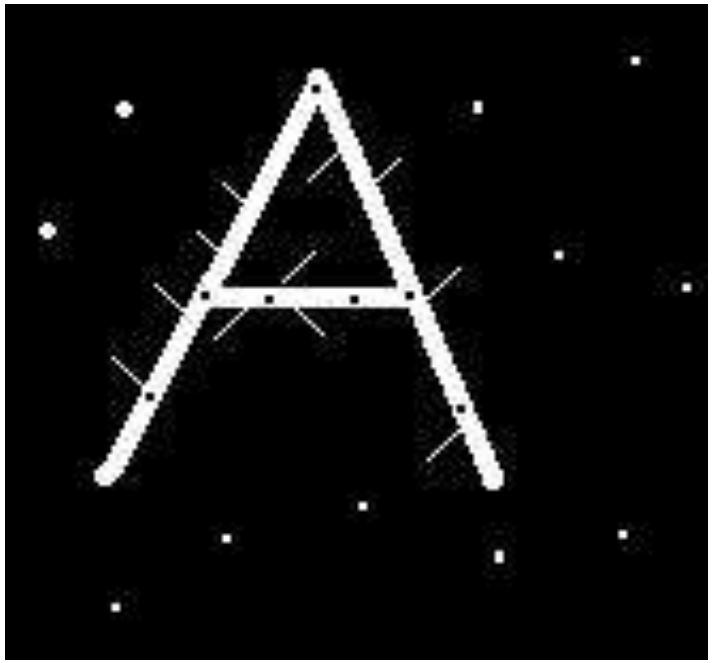
9.2.1 基本运算

4、开、闭运算

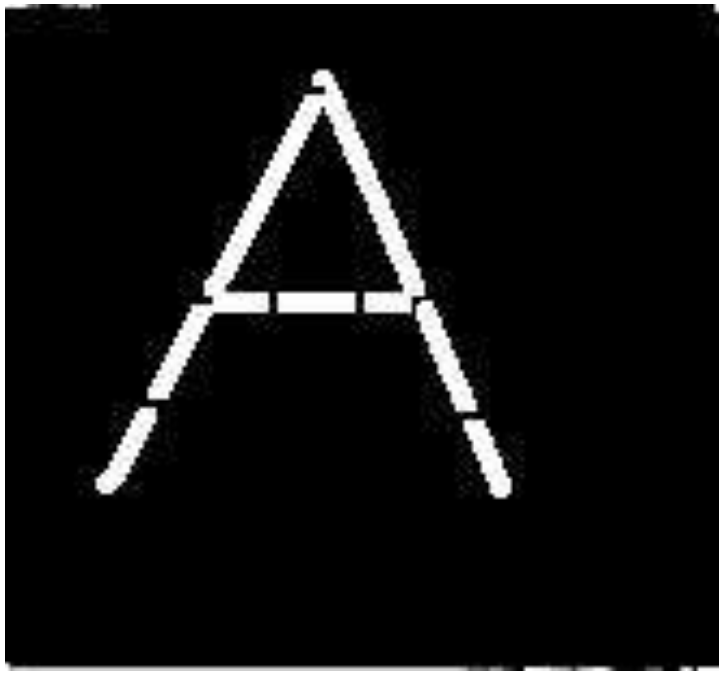
```
clc,clear,close all;
BW=imbinarize(imread('img9_6.bmp'));
SE=strel('square',3);                    %创建方形结构元素
result1=imdilate(imerode(BW,SE),SE);     %先腐蚀后膨胀，即开运算
result2=imerode(imdilate(BW,SE),SE);     %先膨胀后腐蚀，即闭运算
subplot(2,3,1),imshow(BW);title('二值图像');
subplot(2,3,2),imshow(result1);title('开运算');
subplot(2,3,3),imshow(result2);title('闭运算');
SE=strel('square',3);              %创建方形结构元素
result1=imopen(BW,SE);             %用3×3结构元素进行开运算
result2=imclose(BW,SE);            %用3×3结构元素进行闭运算
subplot(2,3,4),imshow(result1);title('开运算imopen');
subplot(2,3,5),imshow(result2);title('闭运算imclose');
```

9.2.1 基本运算

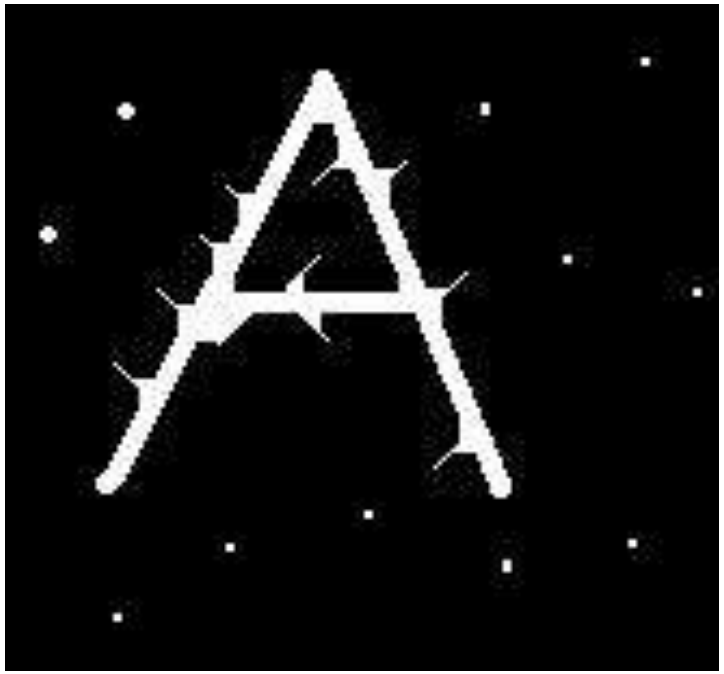
4、开、闭运算



原图



开运算



闭运算

9.2.1 基本运算

4、开、闭运算

■ 开运算的效果

- ✓ 删除小物体(滤掉细长的突起、边缘、毛刺和孤点)；
- ✓ 将物体拆分为小物体(切断细长搭接)起分离作用；
- ✓ 平滑大物体边界而不明显改变它们面积(平滑图像轮廓)。

■ 闭运算的效果

- ✓ 填充物体的裂缝和小洞；
- ✓ 连相近物体(搭接短的间断)起连通补接作用；
- ✓ 平滑大物体边界而不明显改变它们面积(平滑图像轮廓)。

9.2.1 基本运算

4、开、闭运算

开、闭运算的性质

- **增长性**

$$X \subseteq Y \Rightarrow \begin{cases} (X \circ S) \subseteq (Y \circ S) \\ (X \cdot S) \subseteq (Y \cdot S) \end{cases}$$

- **外延性**

$X \circ S \subseteq X$　　$X \subseteq X \cdot S$　　开是非外延的，闭是外延的

- **同前性**

$(X \circ S) \circ S = X \circ S$　　$(X \cdot S) \cdot S = X \cdot S$

同前性表明，对某个集合进行N次连续的开或闭和仅执行一次开或闭的效果一样。

- **对偶性**

$(X \circ S)^C = X^C \cdot \hat{S}$　　$(X \cdot S)^C = X^C \circ \hat{S}$

9.2.2 形态滤波

将开运算和闭运算串起来构建形态滤波器，可以有效地消除目标图像的前景噪声（目标内部的噪声）和背景噪声（目标周围的噪声）

$$(A \circ B) \bullet B = \{[(A \Theta B) \oplus B] \oplus B\} \Theta B$$

- 选择不同形状(如各向同性的圆、十字架、矩形、不同朝向的有向线段等)、不同尺寸的结构元素可以提取图像的不同特征。
- 结构元素的形状和大小直接影响形态滤波结果。

9.2.2 形态滤波

（a）原图像

（b）对(a)进行开运算的结果

（c）形态滤波结果

图 利用圆形结构元素进行形态学滤波示例

9.2.2 形态滤波

思考问题

如果结构元素S的直径缩小一半，能否达到目的？

答：不能，至少效果不好。

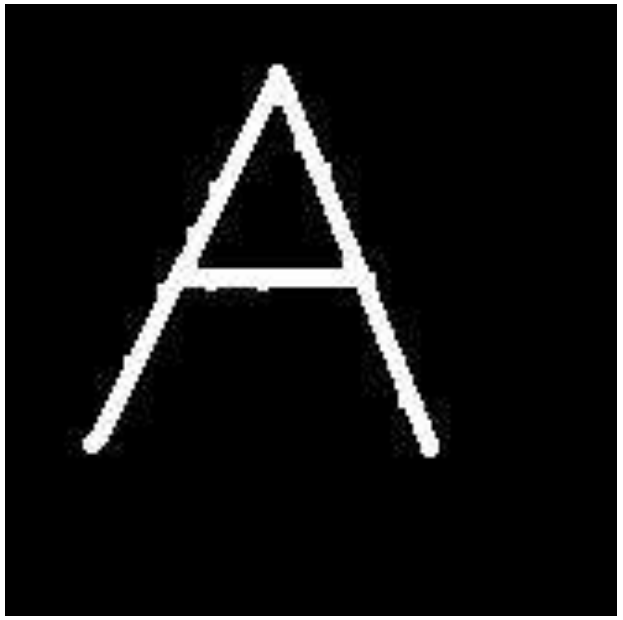
开运算能够去掉外边噪声的关键在于结构元素S的尺寸大于噪声的尺寸，由于噪声不能完全包含结构元素S，在腐蚀时，噪声点被腐蚀掉了；闭运算中，又由于结构元素S的尺寸大于内部孔洞的尺寸，才能在膨胀时融合噪声孔洞。

9.2.2 形态滤波

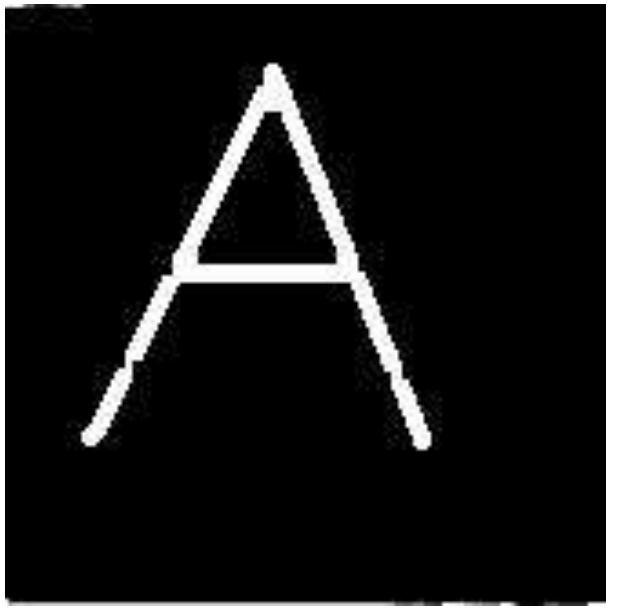
```
clc,clear,close all;
BW=imbinarize(imread('A.bmp'));
SE=strel('square',5);  %创建5×5方形结构元素
result1=imclose(imopen(BW,SE),SE);    %先开后闭
result2=imopen(imclose(BW,SE),SE);    %先闭后开
subplot(1,3,1),imshow(BW);title('二值图像');
subplot(1,3,2),imshow(result1);title('先开后闭平滑处理');
subplot(1,3,3),imshow(result2);title('先闭后开平滑处理');
```

9.2.2 形态滤波

图像A



$(A \cdot S) \circ S$



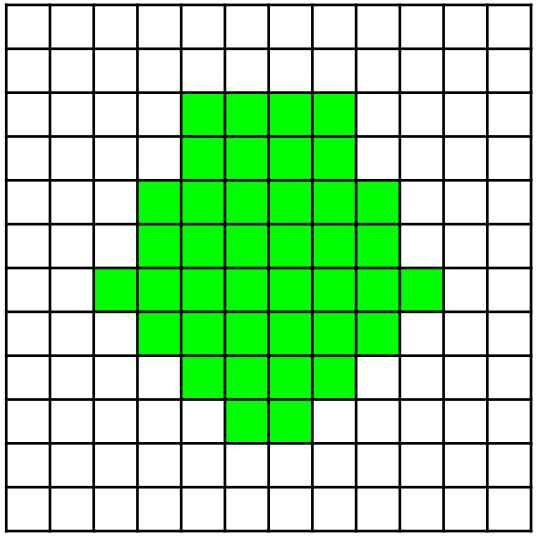
$(A \circ S) \cdot S$

9.2.3 边缘提取

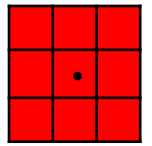
- 图像的边缘或棱线是信息量最为丰富的区域，边缘检测是许多图像处理应用不可缺少的一步。
- 形态学提取边界的基本思想：用一定的结构元素对目标图像进行形态学运算，再将得到的结果与原图像相减。
  - 内边界： $\beta_1(A) = A - (A \Theta B)$
  - 外边界： $\beta_2(A) = (A \oplus B) - A$

9.2.3 边缘提取

提取出图像X的边界？

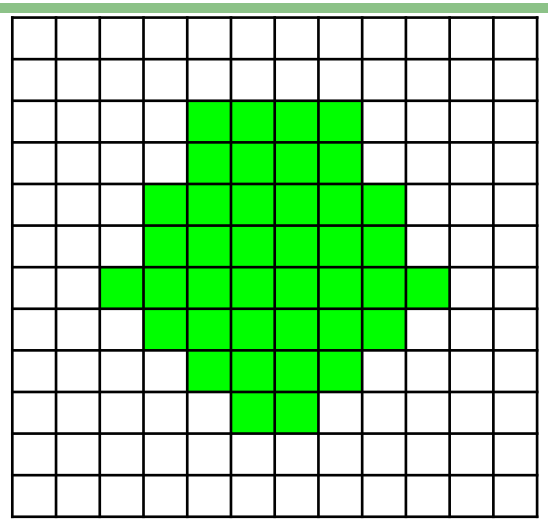


图像 X

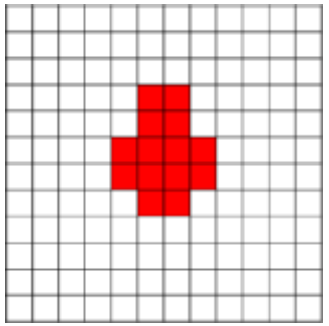


结构元素 S

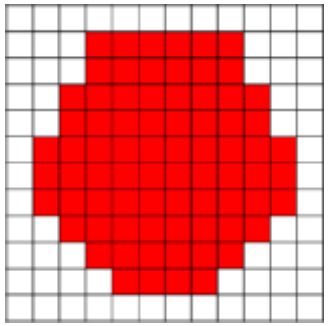
9.2.3 边缘提取



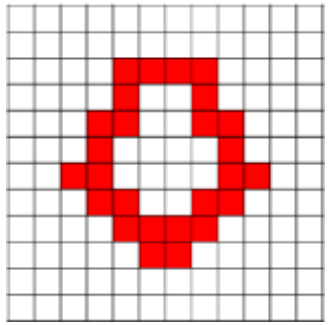
图像 X



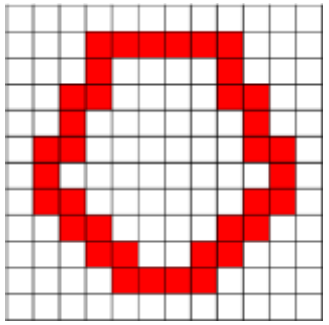
$X \ominus S$



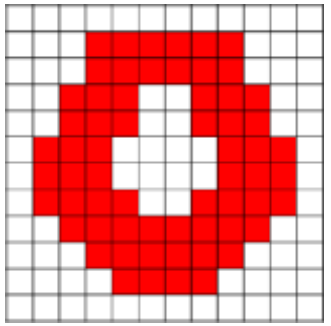
$X \oplus S$



$X - (X \ominus S)$



$(X \oplus S) - X$



$(X \oplus S) - (X \ominus S)$

9.2.3 边缘提取

```
%边缘提取
Image=imread('menu.bmp');
BW=im2bw(Image);
subplot(2,2,1),imshow(result1);title('二值图像');
SE=strel('square',3);%创建方形结构元素
result1=BW-imerode(BW,SE); %提取内边界
result2=imdilate(BW,SE)-BW; %提取外边界
result3=imdilate(BW,SE)-imerode(BW,SE); %形态梯度
subplot(2,2,2),imshow(result1);title('内边界');
subplot(2,2,3),imshow(result2);title('外边界');
subplot(2,2,4),imshow(result3);title('形态梯度');
```

9.2.3 边缘提取

原始图像 X

$X-(X\ominus S)$

$(X\oplus S)-(X\ominus S)$

$(X\oplus S)-X$

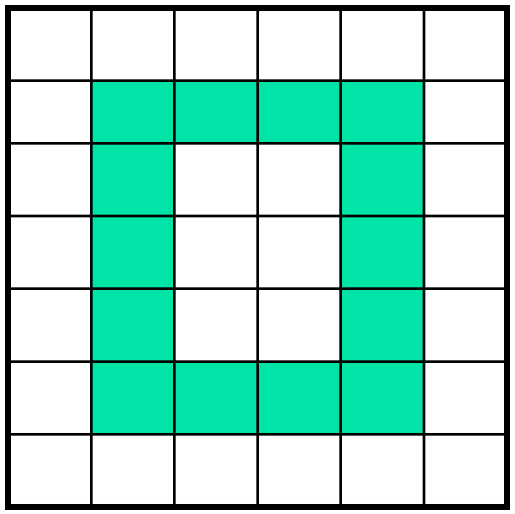
9.2.4 区域填充

在已知区域边界的基础上可以进行区域填充。区域填充是对图像的背景像素进行操作。区域填充一般以图像的膨胀、求补和交集为基础。

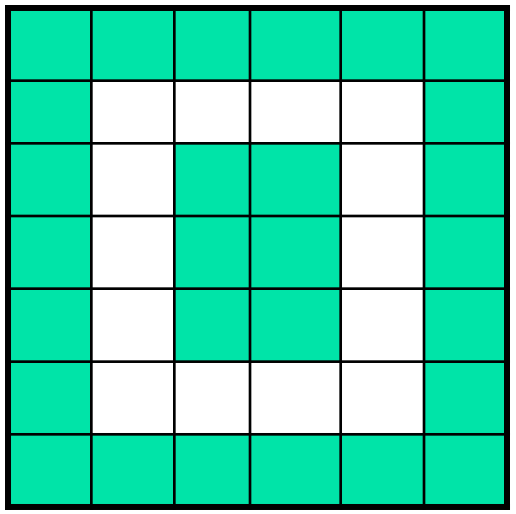
$$X_k = (X_{k-1} \oplus B) \cap A^c$$

- ✓ A表示某区域边界；p是区域内一点（不在边界上）；
- ✓ B是结构元素
- ✓ 取边界内某一点p（$X_0 = p$）为起点，利用上面的公式作迭代运算。
- ✓ 当$X_k = X_{k-1}$时，停止迭代，此时X_k和A的并集为填充集合和它的边界。
- ✓ k是迭代次数

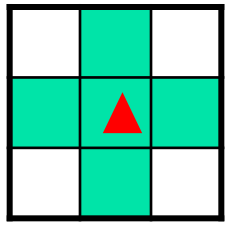
9.2.4 区域填充



（a）边界图像A



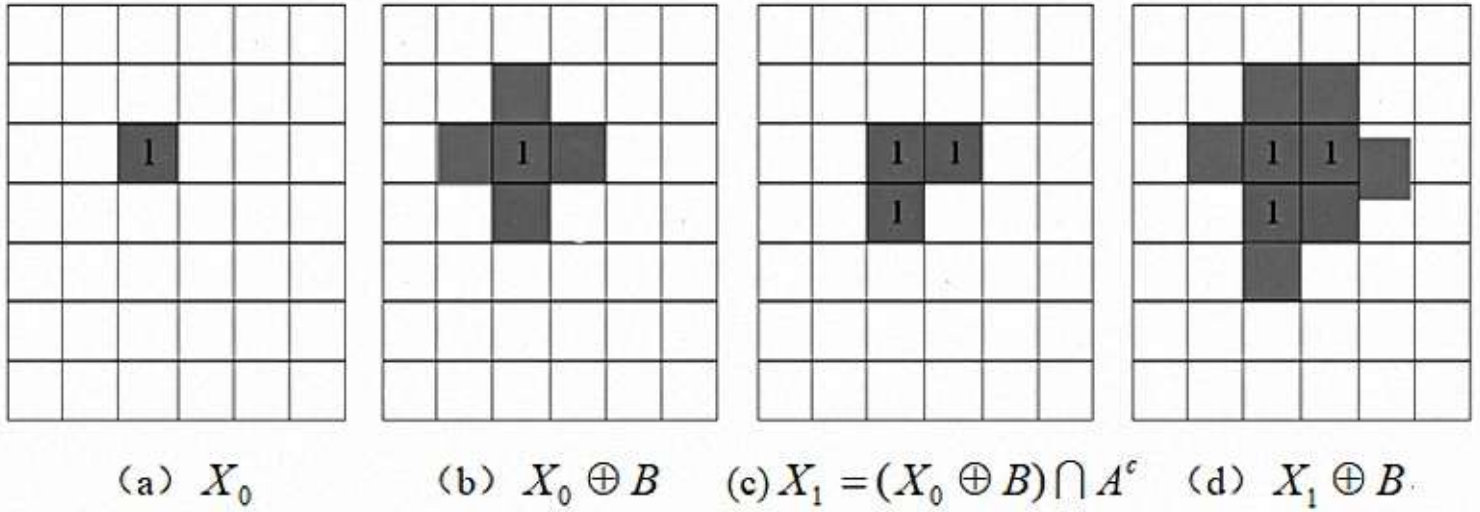
（b）图像A的补集A^c



（c）结构元素B

图　区域填充过程示例用到的边界图像A和结构元素B

9.2.4 区域填充

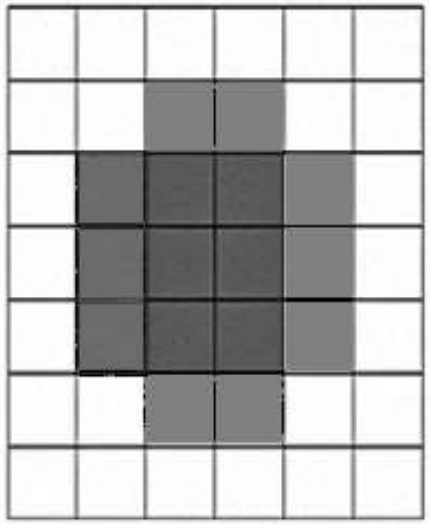


(a) X_0　(b) $X_0 \oplus B$　(c) $X_1 = (X_0 \oplus B) \cap A^c$　(d) $X_1 \oplus B$

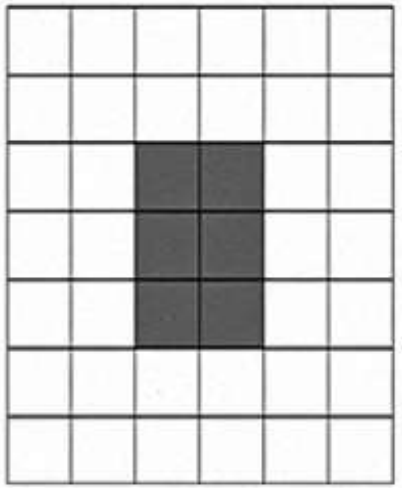
(e) $X_2 = (X_1 \oplus B) \cap A^c$　(f) $X_2 \oplus B$　(g) $X_3 = (X_2 \oplus B) \cap A^c$　(h) $X_3 \cup A$

区域填充过程示意图

9.2.4 区域填充



$X_3 \oplus B$

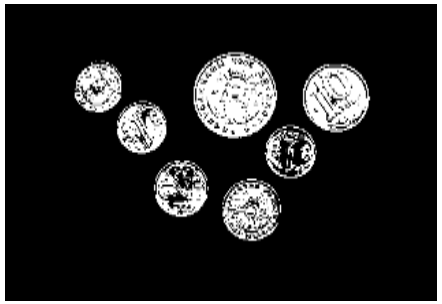


$X_4 = (X_3 \oplus B) \cap A^c$

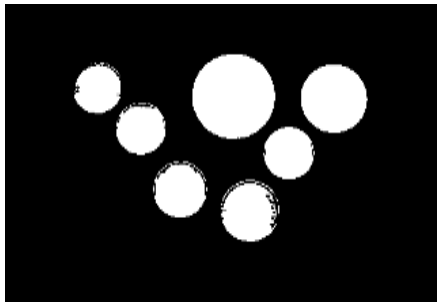
此时X_4=X_3，迭代结束，X_4∪A即是最终结果。

9.2.4 区域填充

```
Image=imread('coin.bmp');
BW=im2bw(Image);
imshow(BW); title('二值图像');
result1=imfill(BW,'holes');
figure,imshow(result1);
title('二值图像的区域填充');
```



原始图像

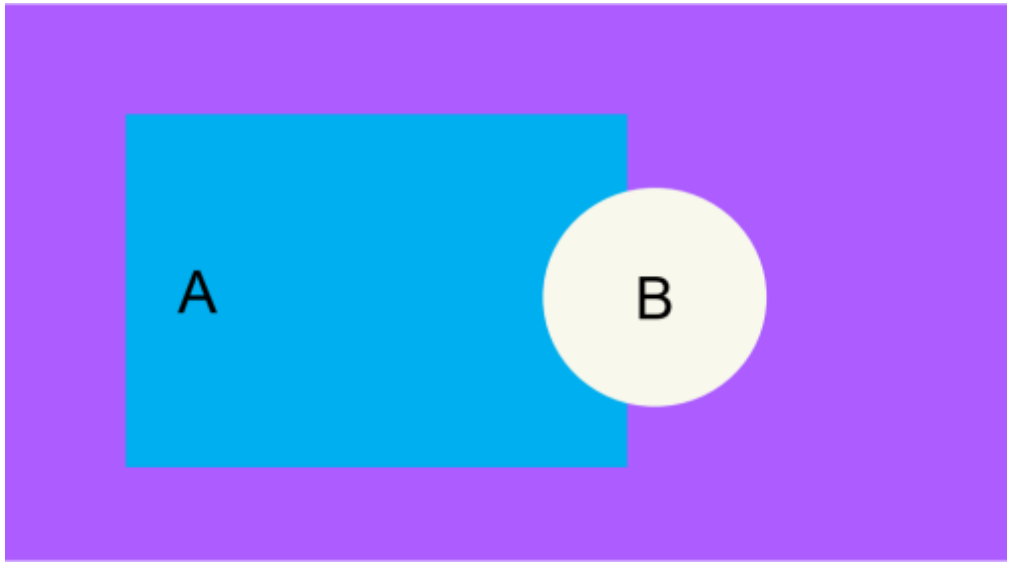


区域填充后图像

9.2.5 击中与否变换

击中

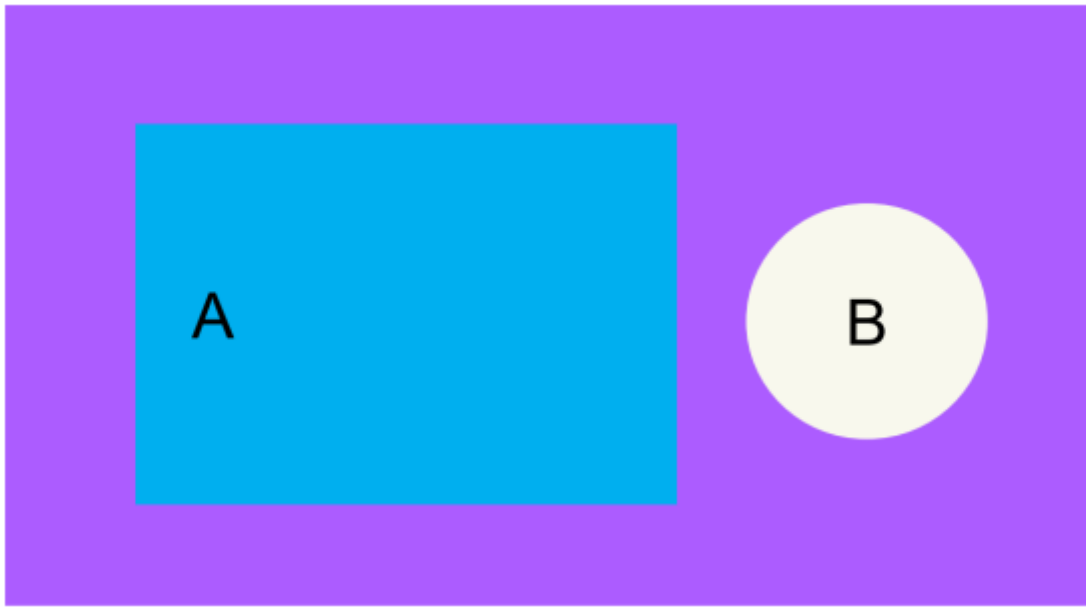
- 设有两幅图像B和A。若存在这样一个点，它既是B的元素，又是A的元素，$A \cap B \neq \emptyset$则称B击中A，记作B个A。



9.2.5 击中与否变换

击不中

- 设有两幅图像B和A。若不存在任何一个点，它既是B的元素，又是A的元素，即B和A的交集是空，则称B不击中A，记作$A \cap B \neq \varnothing$



9.2.5 击中与否变换

- 击中与否变换也称为击中/不击中变换，是在感兴趣区域中探测目标。
- 要进行目标探测，则需要确定目标的位置，可以利用腐蚀特性——腐蚀的过程相当于对可以填入结构元素的位置作标记的过程。因此，可以利用腐蚀运算来确定目标的位置。
- 进行目标检测，既要检测到目标的内部，也要检测到外部，即在一次运算中同时捕获内外标记。

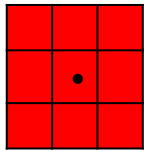
9.2.5 击中与否变换

- 设X为被研究的图像集合，需要采用两个结构元素S_1、S_2，构成一个结构元素对 $S=(S_1, S_2)$，S_1是与目标内部相关的S元素的集合，S_2是与背景（目标外部）相关的S元素的集合，且 $S_1 \cap S_2 = \varnothing$。图像 集合$X$ 用结构元素 $S=(S_1, S_2)$ 进行击中与否变换，记为$X * S$，定义为
- $X * S = (X \ominus S_1) \cap (X^C \ominus S_2)$
- $X * S = (X \ominus S_1) - (X \oplus \hat{S}_2)$
- $X * S = \{x | S_1 + x \subseteq X \text{ 且} S_2 + x \subseteq X^C\}$

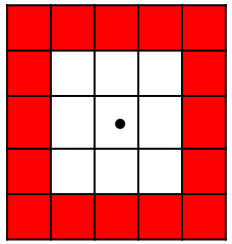
含义：当且仅当S_1平移到某一点可填入X的内部，S_2平移到该点可填入X的外部时，该点才在击中击不中变换的输出中。

9.2.5 击中与否变换

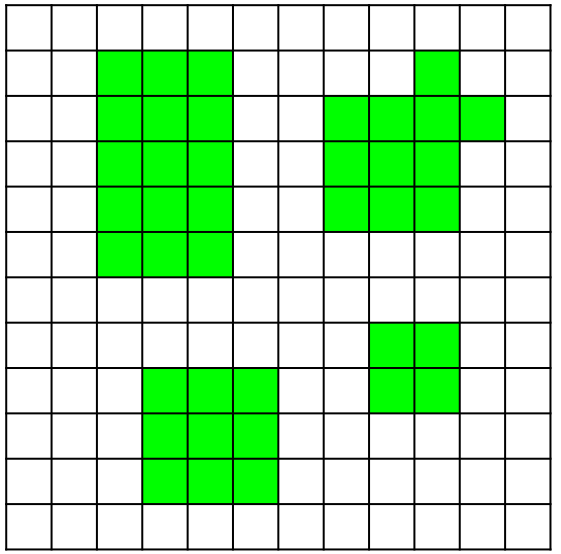
（1）确定结构元素$S=(S_1, S_2)$，选取 S_1为要识别的大方形的形状。



S_1



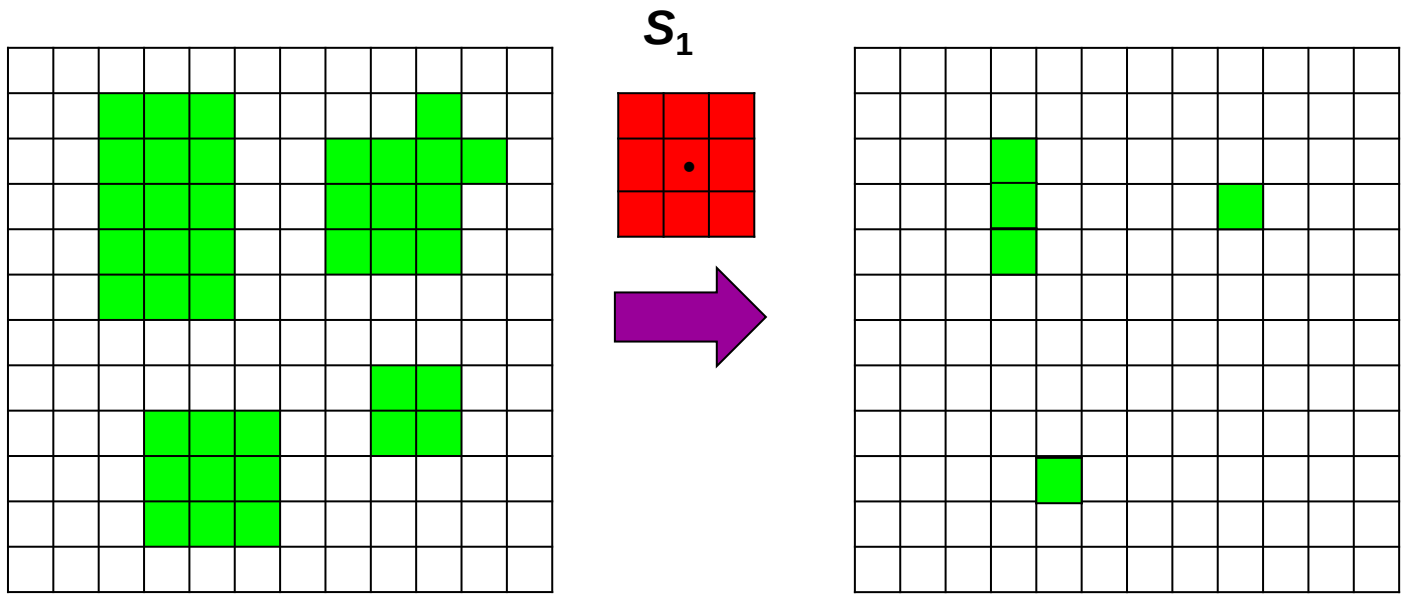
S_2



X

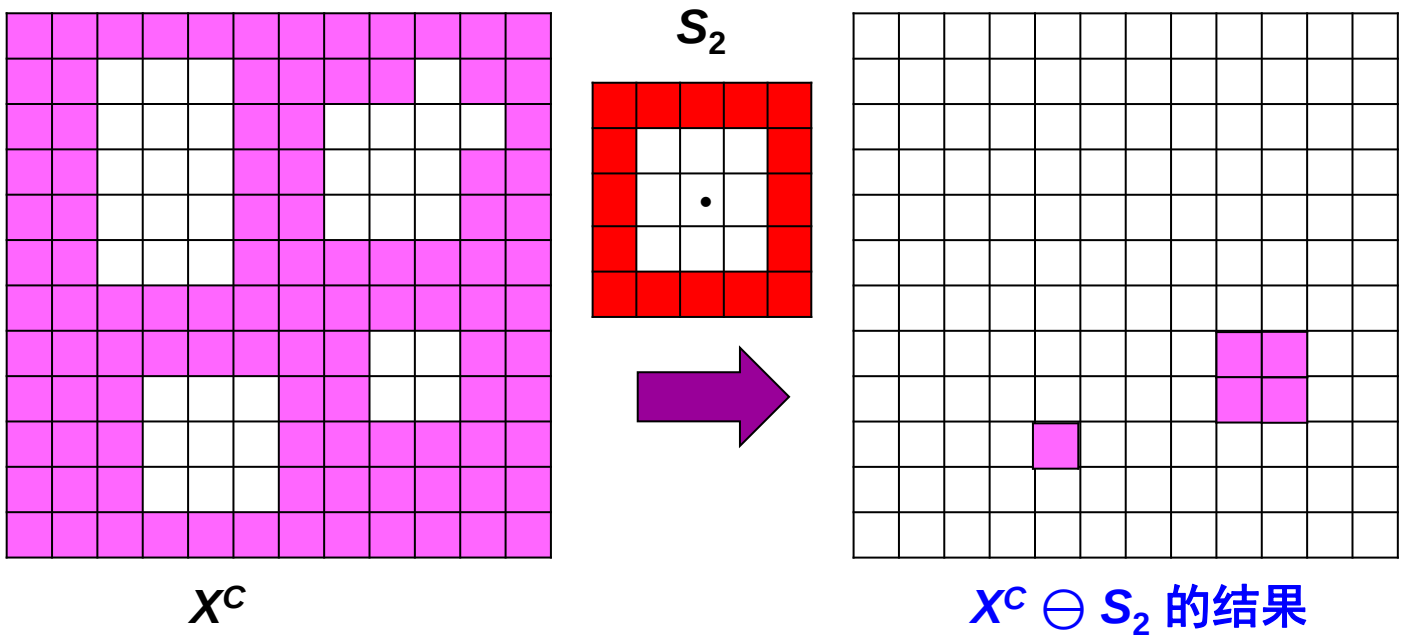
9.2.5 击中与否变换

（2）求$X \ominus S_1$



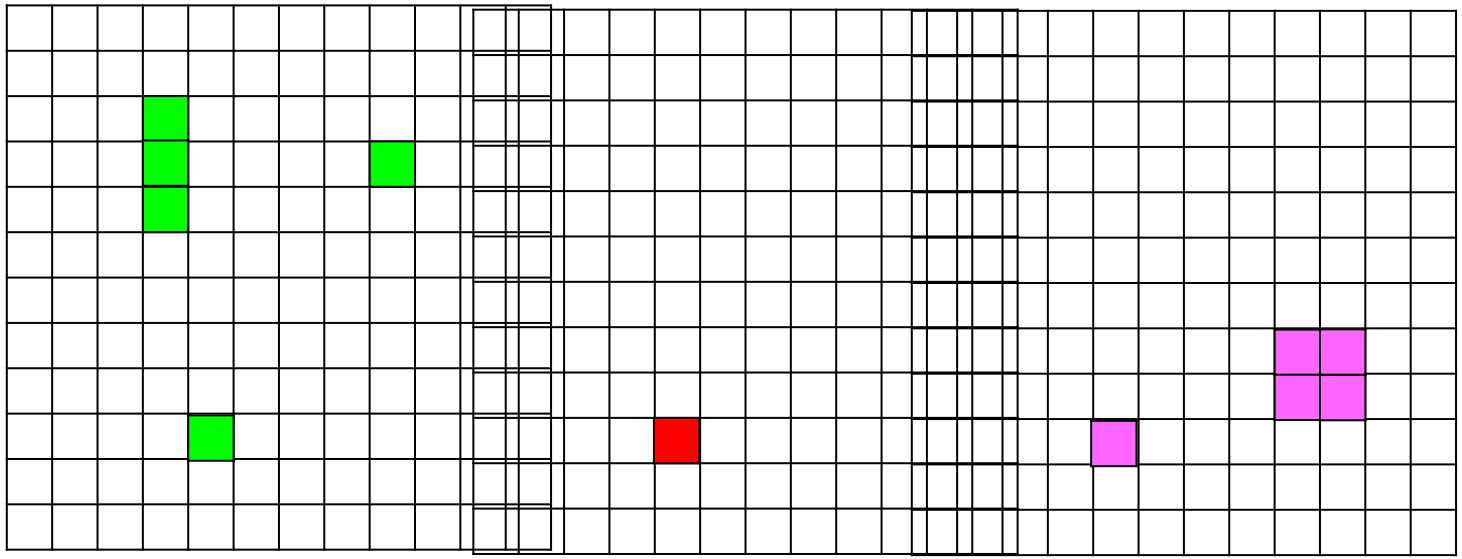
9.2.5 击中与否变换

（3）求 $X^C \Theta S_2$



9.2.5 击中与否变换

（4）求$X * S = (X \Theta S_1) \cap (X^C \Theta S_2)$



$X \Theta S_1$ 的结果　　$X^C \Theta S_2$ 的结果

9.2.5 击中与否变换

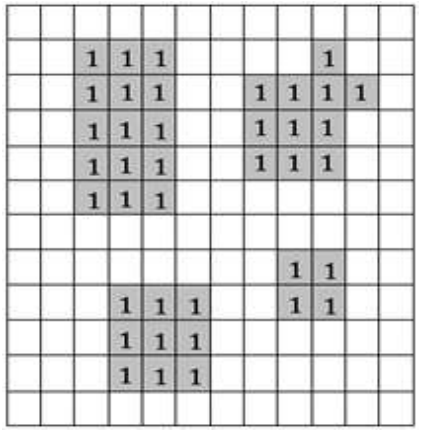
- 程序分析一

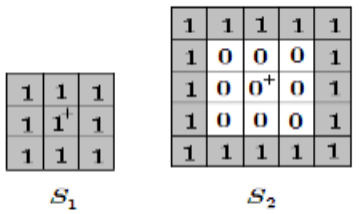
```
%定义图像Image
Image=zeros(12,12);
Image(2:6,3:5)=1;
Image(9:11,4:6)=1;
Image(3:5,8:10)=1;
Image(8:9,9:10)=1;
Image(2,10)=1;
Image(3,11)=1;
```

```
%定义结构元素SE1，SE2
SE1=[0 0 0 0 0
     0 1 1 1 0
     0 1 1 1 0
     0 1 1 1 0
     0 0 0 0 0];

SE2=[1 1 1 1 1
     1 0 0 0 1
     1 0 0 0 1
     1 0 0 0 1
     1 1 1 1 1];
```

二值形态学处理





9.2.5 击中与否变换

- **程序分析一**

%结构元素SE1探测图像内部
result1=imerode(Image,SE1);

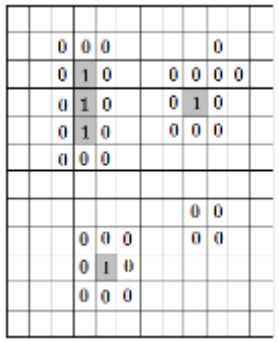
%目标图像Image求补
Image1=~Image;

%结构元素SE2检测图像外部
result2=imerode(Image1,SE2);

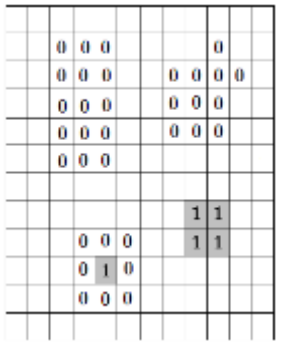
%求出击中与否变换的结果
result=result1 & result2;

figure,imshow(result); title('击中与否');

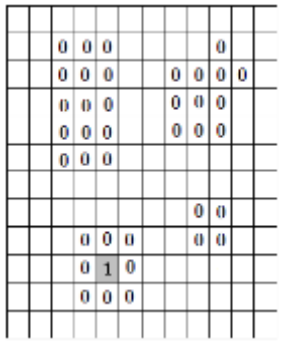
二值形态学处理



$X \Theta S_1$



$X^C \Theta S_2$

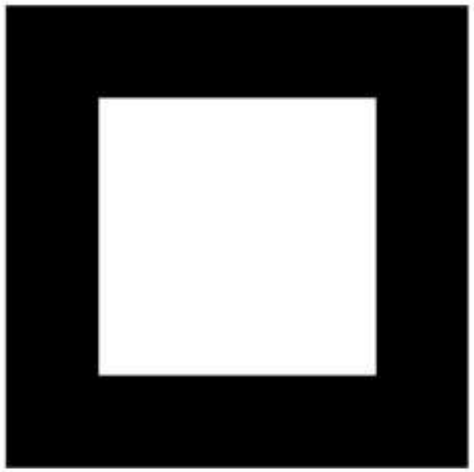


$X * S$

- IM2 = imerode(IM,SE)对灰度图像或二值图像IM进行腐蚀操作，返回结果图像IM2。SE为由strel函数生成的结构元素对象。

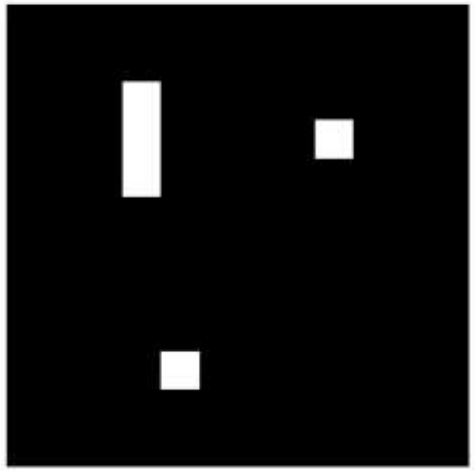
9.2.5 击中与否变换

X



S_1

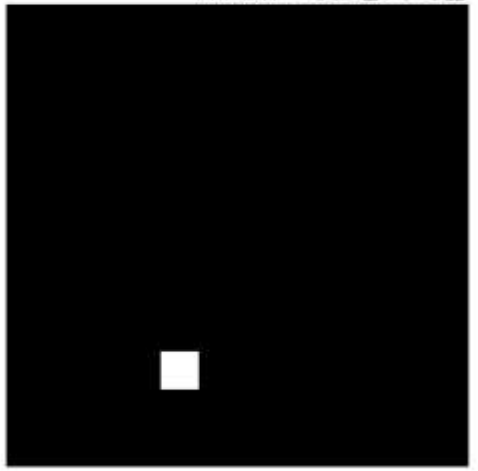
S_2



$X \Theta S_1$

$X^C \Theta S_2$

击中与否



$X * S$

9.2.6 骨架

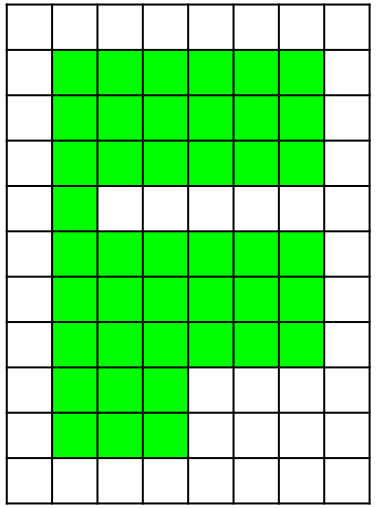
- 骨架化结构是目标图像的重要拓扑描述。
- 对目标图像进行细化，就是求图像的中央骨架的过程。是将图像上的文字、曲线、直线等几何元素的线条沿着其中心轴线将其细化成一个像素宽的线条的处理过程。
- 基于数学形态学变换的细化算法为：

$$X \odot S = X - (X * S)$$

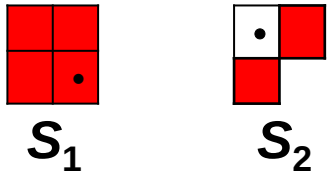
细化实际为从集合X中去掉被结构元素S击中的结果。

9.2.6 骨架

- 示例



图像X



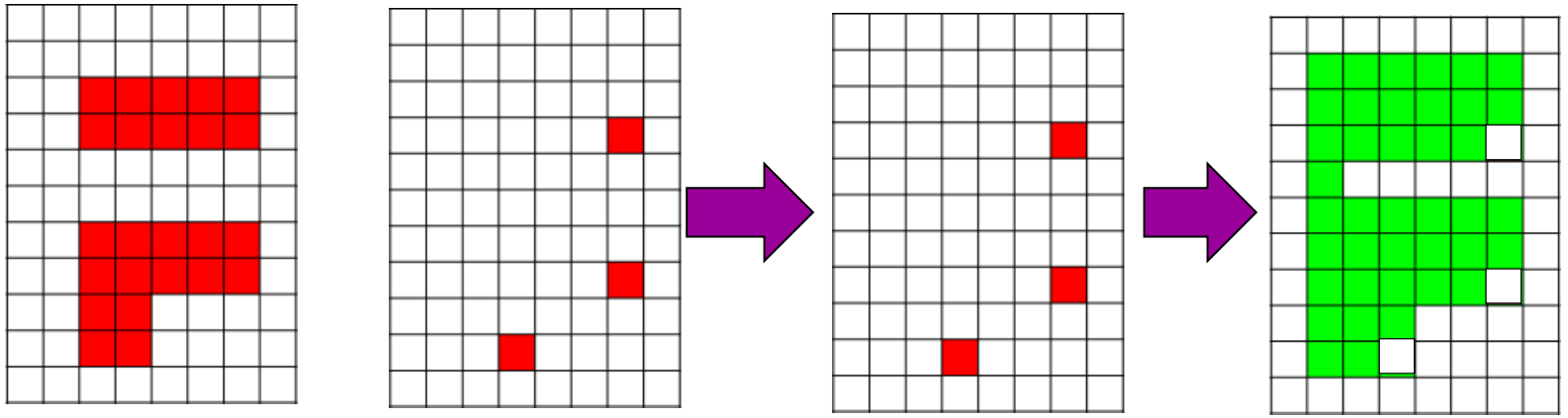
结构元素

$S=(S_1, S_2)$

求：利用结构元素S对图像X进行骨架提取？

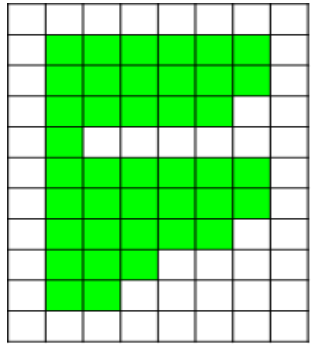
9.2.6 骨架

解：求 $X_1 = X \odot S$

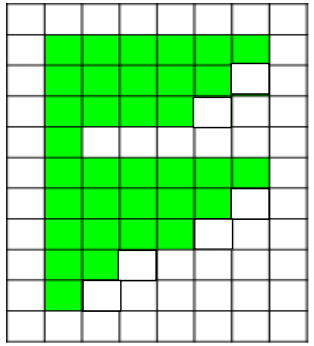


- **采用细化方法：$X_1 = X \odot S, X_2 = X_1 \odot S, \cdots\cdots, X_n = X_{n-1} \odot S$**

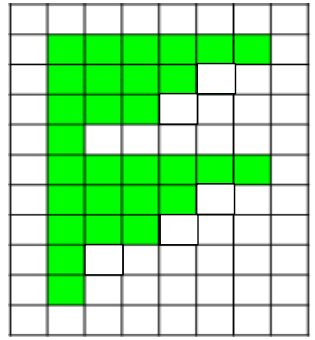
9.2.6 骨架



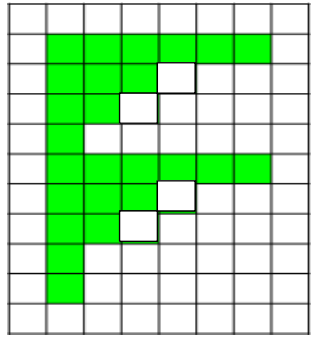
$X_1 = X \odot S$



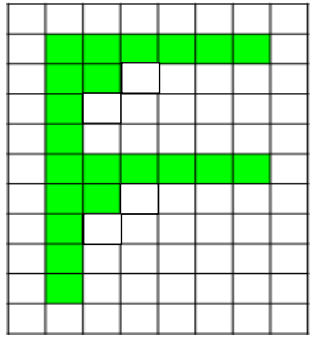
$X_2 = X_1 \odot S$



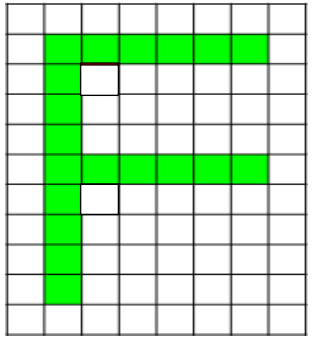
$X_3 = X_2 \odot S$



$X_4 = X_3 \odot S$



$X_5 = X_4 \odot S$



$X_6 = X_5 \odot S$

9.2.6 骨架

```
Image=imread('menu.bmp');
BW=im2bw(Image);
result1=bwmorph(BW,'thin',1);
result2=bwmorph(BW,'thin',Inf);
figure,imshow(result1);title('细化一次');
figure,imshow(result2);title('细化一像素宽');
```

原始图像

细化一次

细化到仅一个像素宽

9.2.7 形态学平滑

形态学平滑是一种通过形态学操作来实现图像平滑的技术。与传统的线性滤波器不同，形态学平滑能够保留图像的边缘信息，同时有效地去除图像中的噪声。

形态学平滑主要基于腐蚀和膨胀两种形态学操作：

1. **腐蚀（Erosion）**：腐蚀操作会使图像中的边缘变得模糊，并且会减小图像中较亮区域的大小。在去除图像中小尺度的噪声或者平滑图像过渡区域方面有很好的效果。
2. **膨胀（Dilation）**：膨胀操作会使图像中的边缘增强，同时也会增大图像中较亮区域的大小。膨胀操作常常用于填充图像中的孔洞或者连接图像中断裂的边缘。
3. 上述两种的组合。

9.2.7 形态学平滑

形态学平滑在图像处理中的应用：

- 去除噪声：特别是在噪声点的大小与图像中目标物体的大小相近的情形下。
- 平滑过渡区域：可以用来平滑图像中的过渡区域，使得图像更加自然，减少过渡的锯齿状效果。
- 孔洞填充：膨胀操作可以用来填充图像中的孔洞，使得图像中的目标物更加完整。
- 边缘保留：与传统的线性滤波器相比，形态学平滑能够更好地保留图像的边缘信息，适用于需要保留边缘的图像处理任务。

9.2.7 形态学平滑

- 灰度开、闭运算的串行结合构成形态学平滑滤波器

$$(f \circ b) \cdot b \quad \text{或} \quad (f \cdot b) \circ b$$

- 经灰度形态学平滑滤波后，可以去掉或减弱图像中特别亮的小亮斑和特别暗的小暗斑，缺点是图像会变得模糊。

9.2.7 形态学平滑

```
%形态学平滑
Image=imread ('lena.tif');
figure,subplot(131);imshow(Image);
Image1=imnoise(Image,'salt & pepper',0.04);
se=strel('disk',2);
imshow(Image1);
result1=imopen(imclose(Image1,se),se);
result2=imclose(imopen(Image1,se),se);
subplot(132);imshow(result1);title('先闭后开图像');
subplot(133);imshow(result2);title('先开后闭图像');
```

9.2.7 形态学平滑

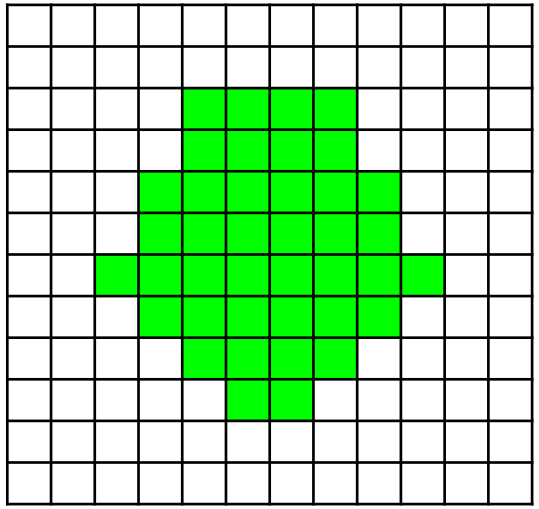
9.2.8 形态学梯度

由于灰度图像中**边缘附近的灰度有较大的梯度**，因此可以利用求图像的形态学梯度的方法来检测图像的边缘。

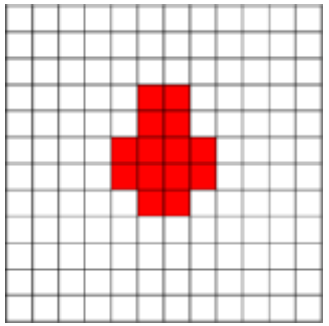
设灰度图像的形态学梯度用g表示，则形态学梯度定义为膨胀减腐蚀：

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

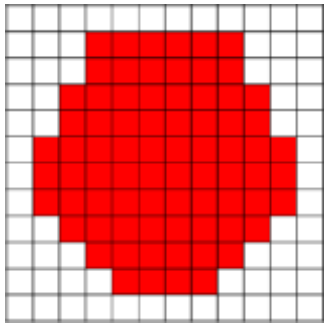
9.2.8 形态学梯度



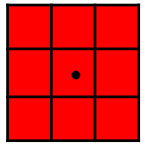
图像 X



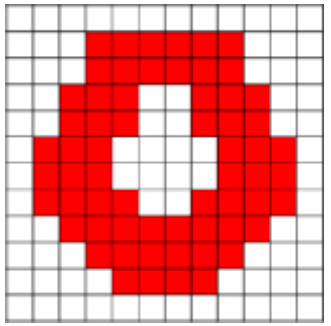
$X \ominus S$



$X \oplus S$



结构元素 S



$(X \oplus S) - (X \ominus S)$

9.2.8 形态学梯度

- ✓ 在图像明暗交界处，暗区域靠近亮区域的像素灰度被膨胀提升较多；
- ✓ 亮区域靠近暗区域的像素灰度被腐蚀降低得较多；
- ✓ 因此，形态学梯度在边缘上形成更大的相对落差，凸显了图像边缘。

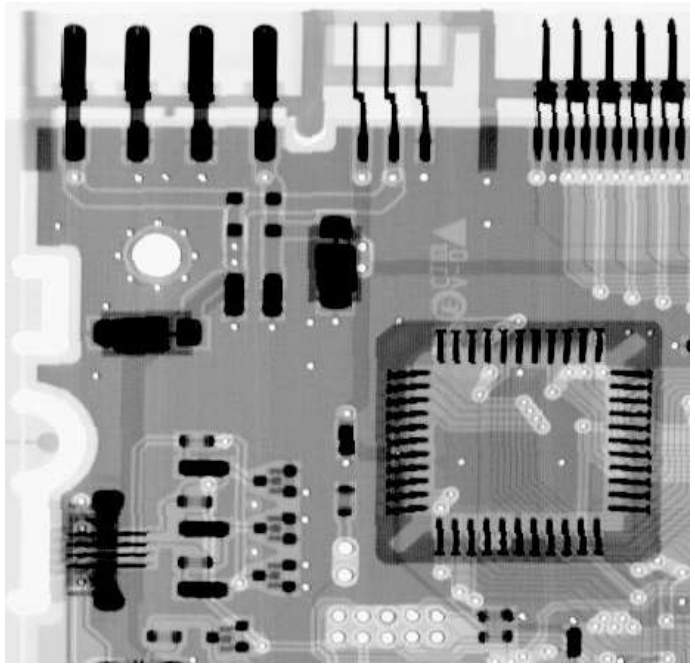
原图

形态学梯度

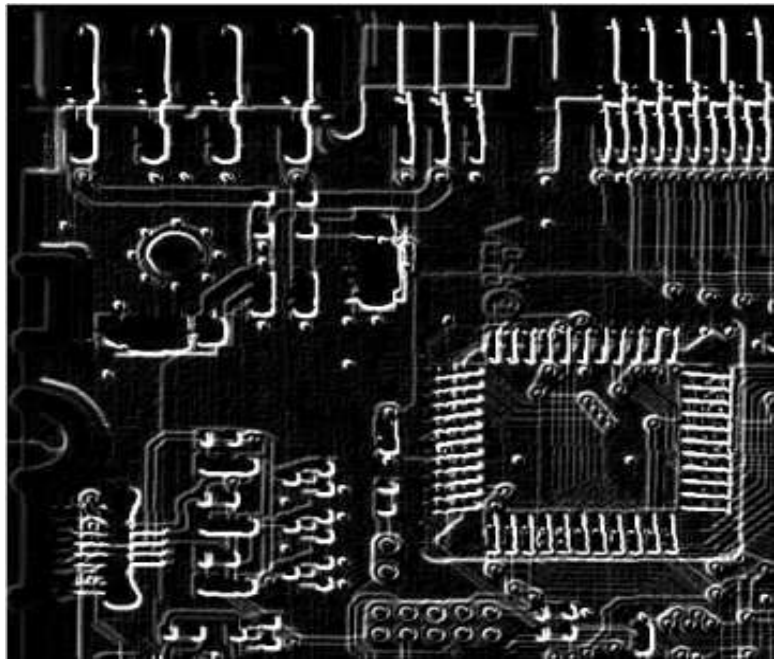
9.2.8 形态学梯度

```
%形态学梯度检测
Image=imread('img9_25.jpg');
BW=edge(Image,'sobel');
SE=strel('disk',1);
result=imdilate(Image,SE)-imerode(Image,SE);
BW1=im2bw(result,0.15);
subplot(1,3,1),imshow(Image);title('原图','fontsize',14);
subplot(1,3,2),imshow(BW),title('Sobel边缘检测','fontsize',14);
subplot(1,3,3),imshow(BW1),title('形态学梯度边缘检测','fontsize',14);
```

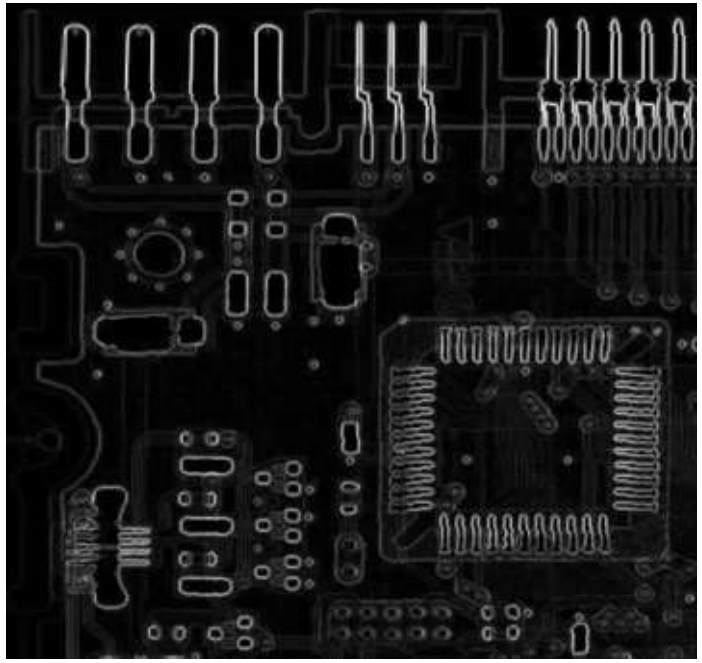
9.2.8 形态学梯度



原图



Sobel算子检测



形态学梯度

利用形态学梯度算子检测的边缘更具有实用性，灰度变化更加尖锐，且采用对称结构元素获得的形态学梯度比利用Sobel算子较少受边缘方向影响。

9.2.9 Top-hat和Bottom-hat变换

- **Top-hat（高帽）变换：**原图像与进行形态学开运算后的图像的差图像。

$$TPH(f) = f - (f \circ b)$$

- **Bottom-hat（低帽）变换：**进行形态学闭运算后的图像与原图像的差图像。

$$BPH(f) = (f \cdot b) - f$$

其中b为结构元素。两个变换都可以检测到图像中变化较大的地方，得到图像中一些重要的标记点。

- 为了使高帽或者低帽处理后的效果更佳明显，对变换后的图像可以做阈值处理。

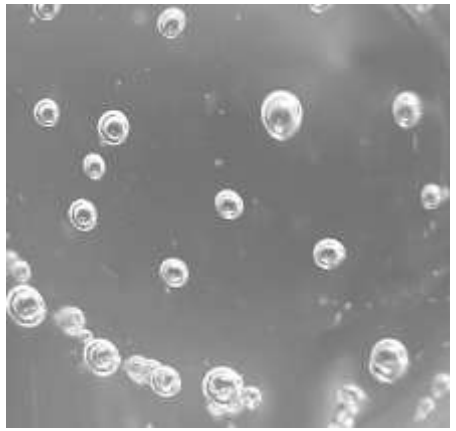
9.2.9 Top-hat和Bottom-hat变换

- Top-hat变换对在较暗的背景中求亮的像素聚集体（颗粒）非常有效，称波峰检测器。
- Bottom-hat变换对在较亮的背景中求暗的像素聚集体（颗粒）非常有效，称波谷检测器。

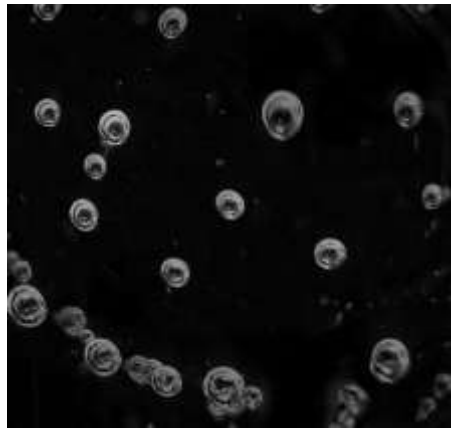
9.2.9 Top-hat和Bottom-hat变换

```
clc,clear,close all;
Image1=imread('img9_26_1.jpg');
Image2=imread('img9_26_2.bmp');%读取灰度图像
se=strel('disk',23);                %选取半径为23的圆盘结构元素
result1=imtophat(Image1,se);        %对图像Image1进行Top-hat变换
result2=imbothat(Image2,se);        %对图像Image2进行Bottom-hat变换
rr1=imadjust(result1);              %进行灰度线性拉伸
rr2=imadjust(result2);              %进行灰度线性拉伸
subplot(2,3,1),imshow(Image1);title('原始cell图像');
subplot(2,3,4),imshow(Image2);title('原始clock图像');
subplot(2,3,2),imshow(result1);title('Top-hat变换');
subplot(2,3,5),imshow(result2);title('Bottom-hat变换');
subplot(2,3,3),imshow(rr1);title('基于Top-hat的对比度增强');
subplot(2,3,6),imshow(rr2);title('基于Bottom-hat的对比度增强');
```

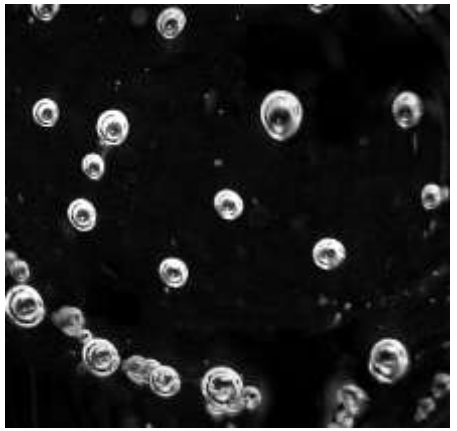
9.2.9 Top-hat和Bottom-hat变换



cell图像



Top-hat变换



结果线性拉伸

clock图像

Bottom-hat变换

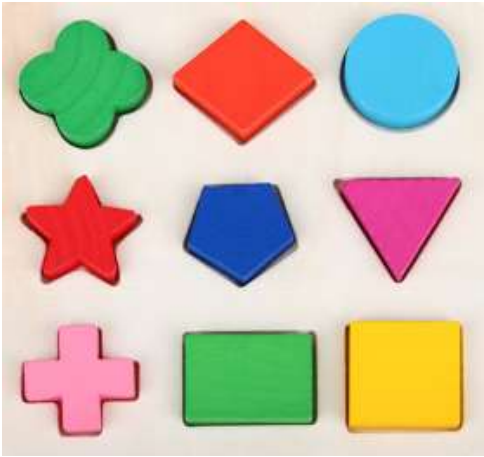
结果线性拉伸

9.3 综合实例

设计程序，采用数学形态学方法实现分割出感兴趣区域

（1）分析

由于要求的图像前景和背景灰度较单一，可以检测目标的边缘，如果能得到闭合边缘，可以通过填充的方法，实现目标提取。



9.3 综合实例

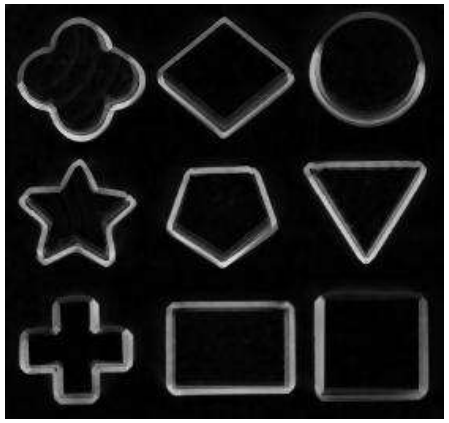
(2) 方案设计

- 将彩色图像灰度化。
- 选取圆盘结构元素，计算形态学梯度。
- 对梯度图像进行对比度增强。
- 实现梯度图像的二值化。
- 将边缘二值化图像进行区域填充，生成目标模板。
- 提取目标区域。

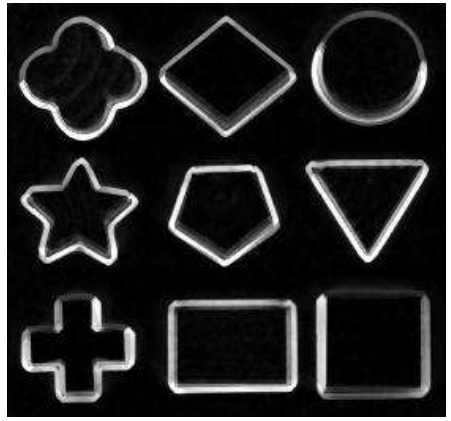
9.3 综合实例

(3) 程序及结果

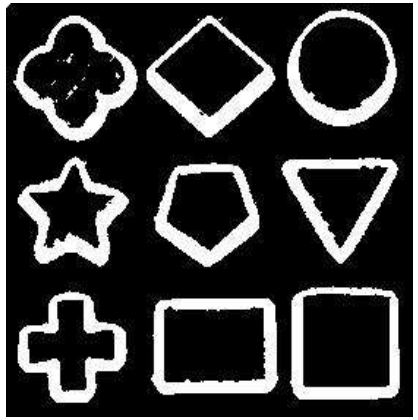
```
clc,clear,close all;
Image=imread('img9_27.png');
gray=im2double(rgb2gray(Image));
se=strel('disk',2);
%形态学梯度
edgeI=imdilate(gray,se)-imerode(gray,se);
enedgeI=imadjust(edgeI); %对比度增强
BW=zeros(size(gray));
BW(enedgeI>0.1)=1; %梯度图像二值化
```



形态学梯度



对比度增强

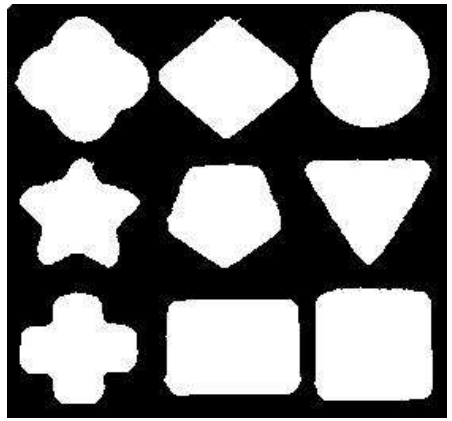


梯度图像二值化

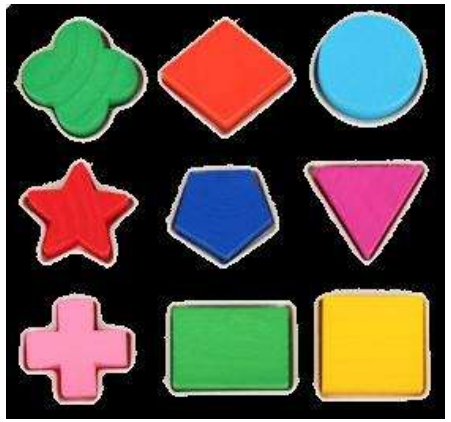
9.3 综合实例

(3) 程序及结果

```
BW1=imclose(BW,se); %闭运算闭合边界
BW2=imfill(BW1,'holes'); %区域填充
template=cat(3,BW2,BW2,BW2);
result=template.*im2double(Image);   %目标提取
subplot(231),imshow(Image),title('原图');
subplot(232),imshow(edgeI),title('形态梯度图像');
subplot(233),imshow(enedgeI),title('对比度增强');
subplot(234),imshow(BW),title('梯度图像二值化');
subplot(235),imshow(BW2),title('目标模板');
subplot(236),imshow(result),title('目标提取');
```



区域填充



目标提取

9.3 综合实例

(4) 效果分析

- 程序设计方案比较简单，**仅适用于背景和前景灰度比较单一的图像**；提取目标的效果依赖于边缘图像的二值化后能否获得封闭的轮廓，提取效果受二值化的阈值限制；

- 图像形态学处理算法**依赖于结构元素的选择**，但在对图像没有先验知识的情况下，结构元素的形状和尺寸选择有很大的随意性。例如，拟采用闭运算闭合轮廓缺口，若结构元素尺寸小，则未必能够实现，在实际应用中需要注意。

案例1 焊缝提取

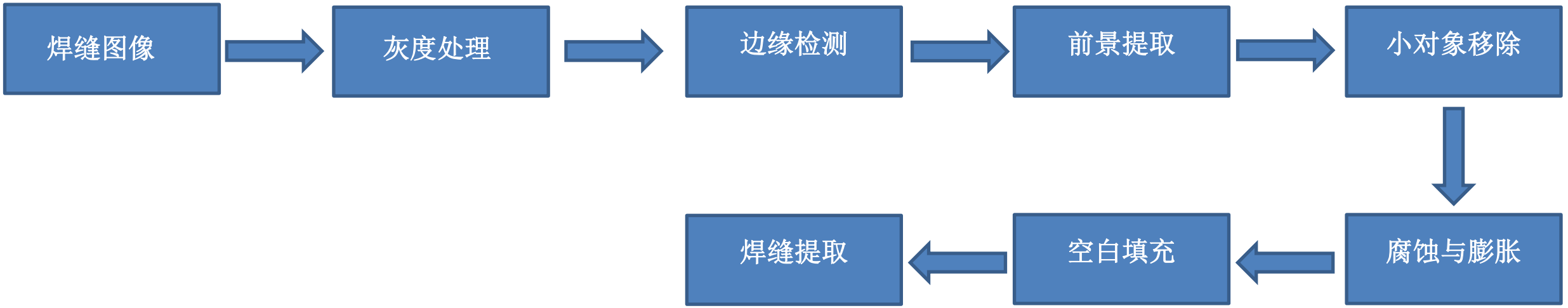
案例背景

基于机器视觉的焊缝跟踪技术因其信息直观、与工件无接触、测量精度高等优点得到了国内外焊接研究工作者的关注。而精确快速的图像处理是实现这一跟踪技术的基础与关键，其中焊缝图像边缘提取的精确性直接决定了最终焊缝位置信息提取的结果。

案例1 焊缝提取

理论基础

焊缝提取采用图像处理与背景分割技术，首先将焊缝图像灰度处理，采用Canny边缘检测提取焊缝的边缘特征，然后将边缘检测后的图像进行腐蚀与膨胀的处理，减少不必要的边缘特征，最终将处理后的边缘进行连接，形成一个封闭区域，将封闭区域填白并与原图像进行点乘，实现焊缝的提取。其技术路线如下所示。



案例1 焊缝提取

边缘检测

在图像中，边缘是指图像中对象的边界，即反映图像中像素值剧烈变化的曲线。对于canny边缘检测算子，其常见用法为：

BW=edge(I,‘canny’)

BW=edge(I,‘canny’,‘thresh’)

BW=edge(I,‘canny’,‘thresh’,sigma)

thresh指阈值，若为两个元素的向量，则第一个元素为低阈值，第二个元素为高阈值，若为标量，则为高阈值，低阈值为$0.5 \times thresh$；sigma是指高斯滤波的标准差，默认值是1，滤波器的大小根据sigma的值选择；BW是返回的边缘，为二值图像，取值为1的是边缘。

案例1 焊缝提取

边缘检测

对于梯度算法，如sobel、prewitt、roberts等，阈值根据计算出来的梯度大小来确定。对于zeroscross算法，阈值根据0交叉的数目来确定，即超过0很多的像素值为边界。对于canny方法，使用了两个梯度的阈值，检测的边缘连续性较好，是这几种方法中效果最好的方法。下图所示为不同边缘检测算子提取的效果图。

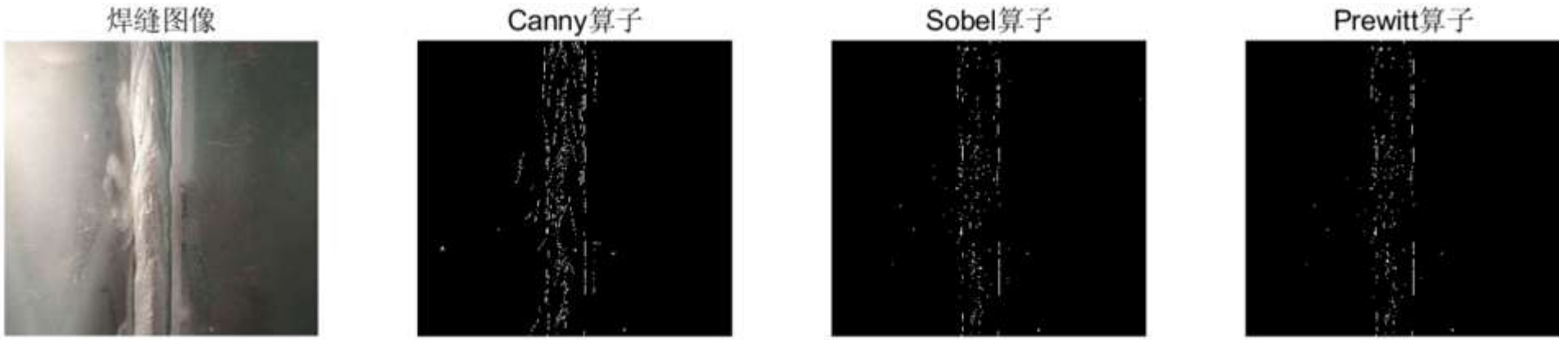


图 不同边缘检测方法结果

案例1 焊缝提取

■ 形态学处理：

（1）前景提取

对于原始图像周围区域，有时可以通过减少不必要的区域来提高边缘检测的效果。在不影响原始图像分割的前提下，往往将目标对象以外的区域像素值赋值为0。

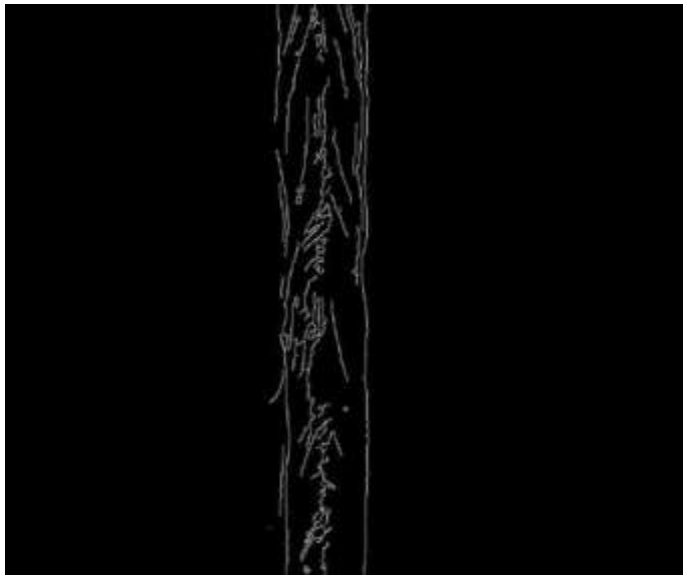


图 前景提取效果图

案例1 焊缝提取

（2）bwareaopen函数

bwareaopen函数的作用是删除二值图像BW中面积小于P的对象，默认情况下conn使用8邻域。下图为删除二值图像BW中像素面积小于120的对象。

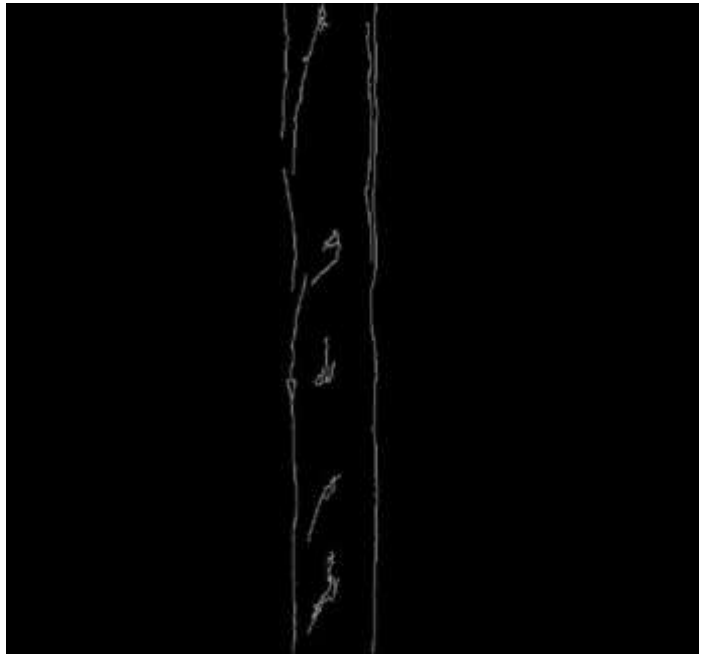


图 小面积移除后效果图

案例1 焊缝提取

（3）腐蚀与膨胀

图 腐蚀与膨胀结果图

案例1 焊缝提取

```
close all;
clear all;
clc;
%读入图片
I=imread('焊缝1.jpg');
im1=rgb2gray(I);
%边缘检测
im1=edge(im1,'canny',0.36);
figure,imshow(im1),title('边缘检测');
%前景提取
im1(1:721,1:300)=0;
im1(1:721,421:769)=0;
figure,imshow(im1);
%小对象移除
im2=bwareaopen(im1,120);
figure,imshow(im2);title('从对象中移除小对象');
%腐蚀与膨胀
se =strel('disk',360,0);
im3=imclose(im2,se);
figure,imshow(im3);title('腐蚀与膨胀结果');
figure,imshow(im5);title('焊缝提取图');
```

```
%区域填充
im4=bwfill(im3,'holes');
figure,imshow(im4),title('填充空白区域')
%焊缝提取
im5=uint8(im4).*I;
figure,imshow(im5);title('焊缝提取图');
```

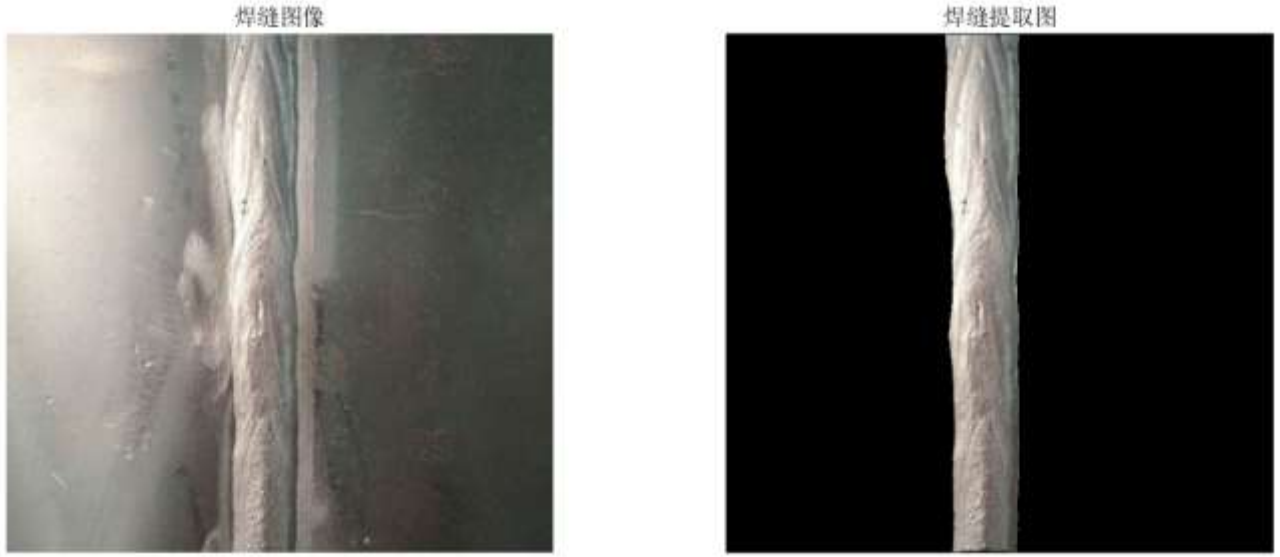


图 焊缝提取结果图

案例2 圆木计数

■ 案例背景

在木材运输过程中，对车辆装载木材的稽查以及查验的方式主要还是以人工方式为主，劳动强度大，工作效率低，易出错。因此对于木材计数的自动识别也得到了越来越多的研究。

■ 理论基础

基于图像识别的木材识别方法，主要是通过对排列比较整齐的木材的图像进行处理来得到木材数量的一种方法。木材计数主要涉及木材图像的采集、木材图像的处理、木材区域分割、木材聚类等步骤。目前对于圆木计数的方法主要有以下几种：

案例2 圆木计数

1)基于Hough变换的圆形检测方法

此方法对于圆形度较好的圆形区域效果较好，对原木的形状提出较高的要求，实用性较差。

2)模板匹配法

模板匹配是一种利用圆木模板，与待测图像进行匹配得到统计结果的方法。模板匹配法的计算量以及存储量要求比较大，而且模板与待处理对象的形状也存在差异问题，适应性不是特别强。

案例2 圆木计数

3)中心聚类的方法

该方法对图像边缘进行中心增强然后对中心进行聚类，此方法对于木材的半径以及圆形度要求较高，同样适应性不是特别的强。

4)中心发散方法

该方法中以棒材为中心，逐层向四周进行检索，并逐步扩散，直到找到所有棒材为止，该方法速度较快，但是对于圆形棒材的直径要求较高，不能偏差太大。

5)阈值分割方法

该方法通过对圆木图像通过阈值分割，并利用腐蚀等操作，分割出每个圆木区域进行统计，但是此方法对图像拍摄要求较高，干扰因素较大。

案例2 圆木计数

本案例对预处理后的图像首先提取背景区域，提取圆木区域二值图像，使用圆形霍夫变换寻找圆进行圆木计数。该方法对圆木形状要求不高，并有效解决了在室外拍摄光照条件不均等情况，具有较好的实用性以及可靠性。其技术路线如图所示。

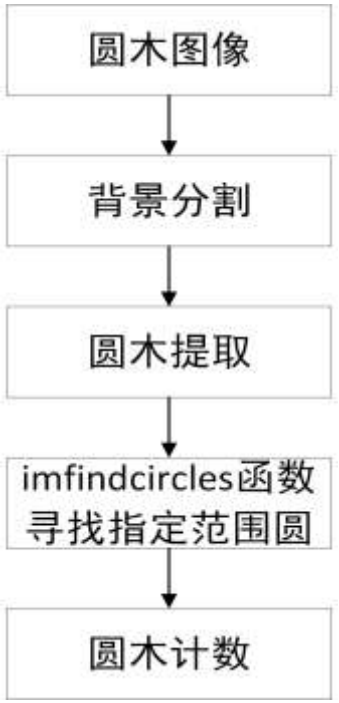


图 圆木计数路线图

案例2 圆木计数

在圆木计数前，首先需要对圆木图像进行背景分割，提取圆木区域，消除无关背景带来的影响。

```
%圆木背景分分割
clear all;
close all;
clc;
I=imread('圆木3.jpg');
figure,subplot(121),imshow(I);title('原始图像')
Ir=I(:,:,1);Ig=I(:,:,2);Ib=I(:,:,3);
%figure,imshow(Ir);
%figure,
%subplot(1,3,1),imshow(Ir),title('R通道图');
%subplot(1,3,2),imshow(Ig),title('G通道图');
%subplot(1,3,3),imshow(Ir),title('B通道图');
%figure,
%subplot(1,3,1),imhist(Ir),title('R通道灰度直方图');
%subplot(1,3,2),imhist(Ig),title('G通道灰度直方图');
%subplot(1,3,3),imhist(Ib),title('B通道灰度直方图');
%imtool(Ib)
Ib_BW=roicolor(Ib,[20,100]);
%figure,subplot(1,4,1),imshow(Ib_BW),title('颜色范围选取');
I_rb=Ir-Ib;
%subplot(1,4,2),imshow(I_rb),title('R-B通道');
I_rb_BW=im2bw(I_rb,graythresh(I_rb));
%subplot(1,4,3),imshow(I_rb_BW),title('通道转差二值图像');
I_rb_BWfill=imfill(I_rb_BW,'holes');
%subplot(1,4,4),imshow(I_rb_BWfill),title('空洞填充二值图像');
Obj=uint8(I_rb_BWfill).*I*1.2;
subplot(122),imshow(Obj);title('圆木提取')
```

案例2 圆木计数

图 圆木提取结果

案例2 圆木计数

imfindcircles函数的功能是使用圆形霍夫变换寻找圆。

- centers = imfindcircles(A, radius)在图像A中找到半径近似等于半径的圆。 中心的输出是一个两列的矩阵，其中包含图像中圆心的（x, y)坐标。
- [centers, radii] = imfindcircles (A, radiusRange)查找半径在radiusRange指定范围内的圆。附加输出参数radii包含与中心中每个圆心相对应的估计半径。
- [centers, radii, metric] = imfindcircles (A, radiusRange)还返回一个列向量metric，其中包含每个圆的累加器阵列峰值的大小（降序排列）。 中心行和半径行对应于公制行。
- [___] = imfindcircles (___, Name, Value)使用以前的任何语法，使用一个或多个Name，Value对参数指定其他选项。

案例2 圆木计数

```
%圆木计数代码
%h = imdistline;%测量圆木直径
[centers,radii] = imfindcircles(Ir,[45 155],'ObjectPolarity','bright',...
'Sensitivity',0.97);
h = viscircles(centers,radii);
num = length(centers)
```

图 圆木计数

THE END

Thank You.