阈值处理

- 基础

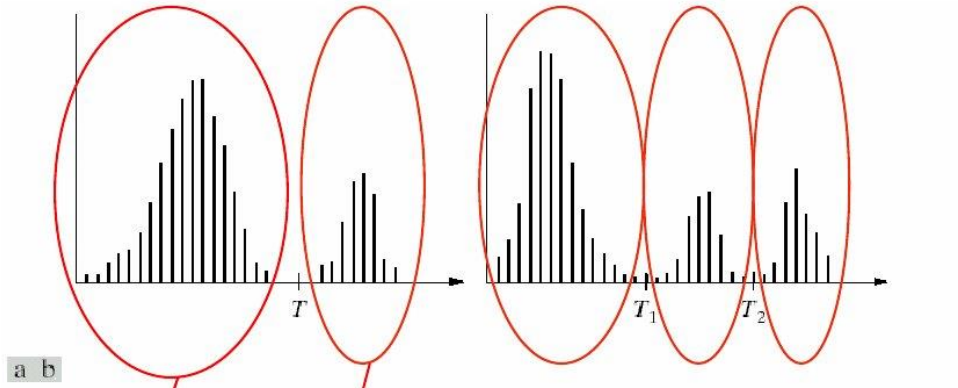


FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

暗的背景：$f(x,y) \leq T$

亮的对象：$f(x,y) > T$

暗的背景：$f(x,y) \leq T1$

亮的一个对象：$T_1 < f(x,y) \leq T_2$

亮的另一个对象：$f(x,y) > T_2$

阈值处理

- 基础
  - ✓ 阈值处理操作

$$T = T[x, y, p(x, y), f(x, y)]$$

f(x,y)是点(x,y)的灰度级，p(x,y)表示该点的局部性质，如以(x,y)为中心的邻域的平均灰度级

  - ✓ 阈值处理后的图像g(x,y)定义为

$$g(x, y) = \begin{cases} 1 & f(x, y) > T \\ 0 & f(x, y) \le T \end{cases}$$

阈值处理

- 基础

$$g(x,y)=\begin{cases}1 & f(x,y)>T\\ 0 & f(x,y)\le T\end{cases}$$

✓ 标记为1的像素对应于对象，标记为0的像素对应于背景

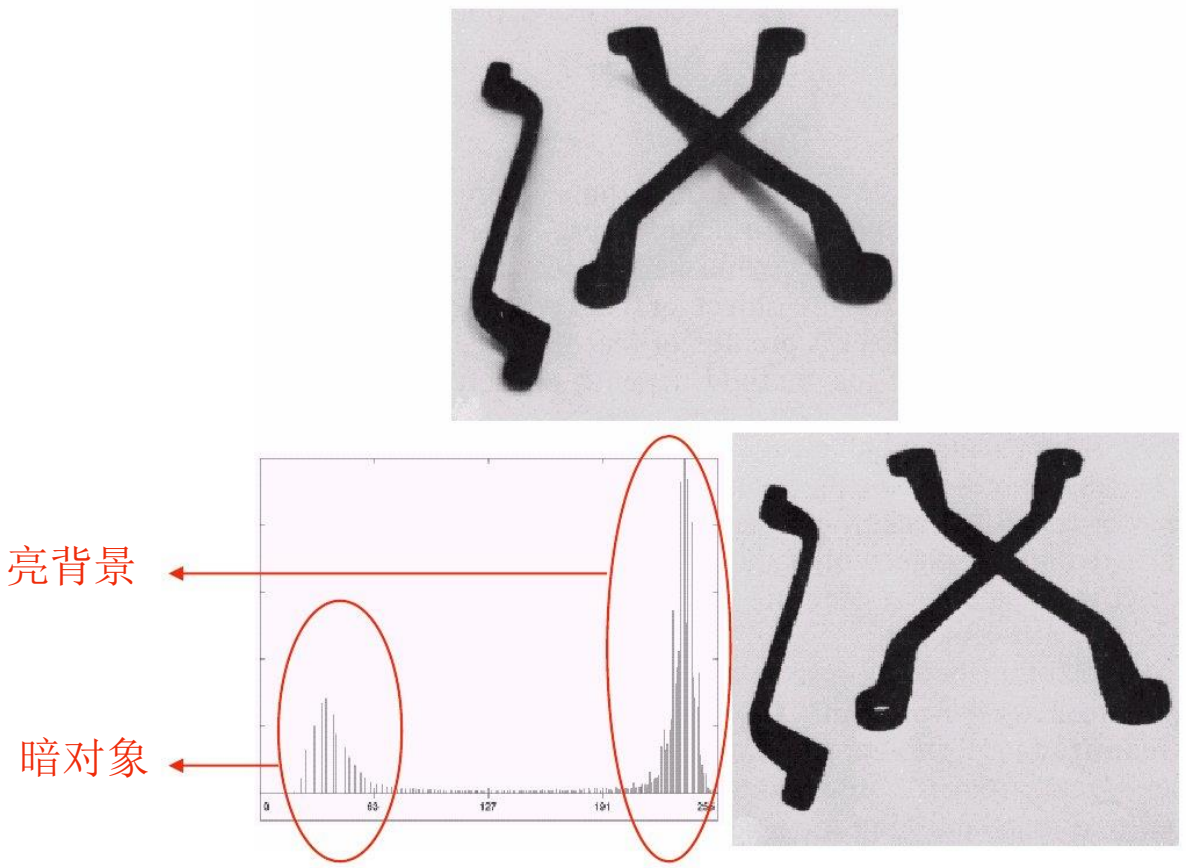
✓ 当T仅取决于f(x,y)，阈值称为全局的

✓ 当T取决于f(x,y)和p(x,y)，阈值是局部的

✓ 当T取决于空间坐标x和y，阈值就是动态的或自适应的

图像分割

- 基本全局阈值例子



a
b c

FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

图像分割

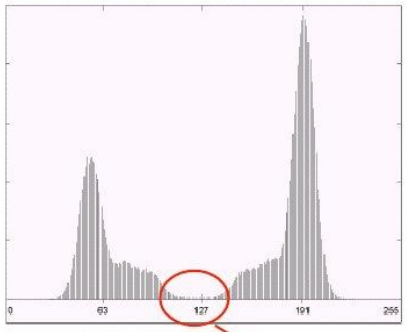
- 计算基本全局阈值算法

1. 选择一个T的初始估计值
2. 用T分割图像，生成两组像素：G_1由所有灰度值大于T的像素组成，而G_2由所有灰度值小于或等于T的像素组成
3. 对区域G_1和G_2中的所有像素计算平均灰度值μ_1和μ_2
4. 计算新的阈值 $T = \frac{1}{2}(\mu_1 + \mu_2)$
5. 重复步骤2到4，直到逐次迭代所得的T值之差小于事先定义的参数T_0

利用基本全局阈值算法的例子

原图

原图的直方图



波谷作为阈值

a b
c

FIGURE 10.29
(a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

基本全局阈值算法处理的结果

T0=0，3次迭代得到值为125.4

最后确定T=125

图像分割

- 基本自适应阈值
  - ✓ 单一全局阈值存在的问题
    - ➢ 不均匀亮度图像无法有效分割
  - ✓ 方法
    - ➢ 将图像进一步细分为子图像，并对不同的子图像使用不同的阈值处理
    - ➢ 解决的关键问题：如何将图像进行细分和如何为得到的子图像估计阈值
    - ➢ 自适应阈值：取决于像素在子图像中的位置

最佳全局和自适应阈值

✓ 假设2个PDF中较大的一个对应背景的灰度级，较小的一个描述了图像中对象的灰度级，则混合PDF是

$$p(z) = P\,p_1(z) + P\,p_2(z)$$

P_1是属于对象像素的概率，P_2是属于背景像素的概率，假设图像只包括对象和背景，则

$$P_1 + P_2 = 1$$

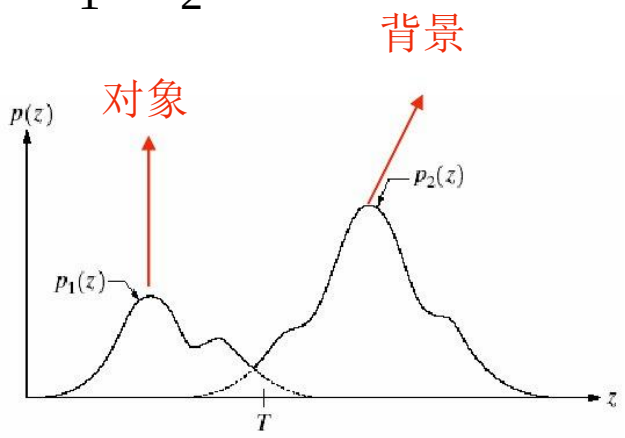


FIGURE 10.32 Gray-level probability density functions of two regions in an image.

最佳全局和自适应阈值

- 在区间[a,b]内取值的随机变量的概率是它的概率密度函数从a到b的积分，即在这两个上下限之间PDF曲线围住的面积，因此，将一个背景点当作对象点进行分类时，错误发生的概率为：

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz$$

这是在曲线$p_2(z)$下方位于阈值左边区域的面积

- 将一个对象点当作背景点进行分类错误发生的概率为

$$E_2(T) = \int_{T}^{\infty} p(z)\,dz$$

这是在曲线$p_1(z)$下方位于阈值右边区域的面积

最佳全局和自适应阈值

- 出错率的整体概率是

$$E(T) = P_2 E_1(T_1) + P_1 E_2(T)$$

- 为了找到出错最少的阈值，使用莱布尼兹法则把E(T)对T求微分并令结果等于0，得到

$$P_2 p_1(T) = P_1 p_2(T)$$

- 上式解出T，即为最佳阈值

- 如果$P_1=P_2$，则最佳阈值位于曲线$p_1(z)$和$p_2(z)$的交点处

最佳全局和自适应阈值

- 高斯密度可以用两个参数均值和方差描述

$$p(z)=\frac{P_1}{\sqrt{2\pi}\,\sigma_1}e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}}+\frac{P_2}{\sqrt{2\pi}\,\sigma_2}e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}}$$

- 出错最少的阈值T的解

$$AT^2+BT+C=0$$

$$A=\sigma_1^2-\sigma_2^2$$

$$B=2\left(\mu_1\sigma_2^2-\mu_2\sigma_1^2\right)$$

$$C=\sigma_1^2\mu_2^2-\sigma_2^2\mu_1^2+2\sigma_1^2\sigma_2^2\ln\left(\sigma_2 P_1/\sigma_1 P_2\right)$$

最佳全局和自适应阈值

- ✓ 如果方差相等 $\sigma_2 = \sigma_{12} = \sigma_2^2$，则得到单一的阈值

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right)$$

- ✓ 如果$P_1=P_2$，最佳阈值是均值的平均数

图像分割

- 通过边界特性选择阈值
  - ✓ 基本思想：
    - ➢ 如果直方图的各个波峰很高、很窄、对称，且被很深的波谷分开时，有利于选择阈值
    - ➢ 为了改善直方图的波峰形状，我们只把区域边缘的像素绘入直方图，而不考虑区域中间的像素
    - ➢ 用微分算子，处理图像，使图像只剩下边界中心两边的值

图像分割

- 通过边界特性选择阈值
  - ✓ 这种方法有以下优点：

    1)在前景和背景所占区域面积差别很大时，不会造成一个灰度级的波峰过高，而另一个过低

    2)边缘上的点在区域内还是区域外的概率是相等的，因此可以增加波峰的对称性

    3)基于梯度和拉普拉斯算子选择的像素，可以增加波峰的高度

图像分割

- 通过边界特性选择阈值
  - ✓ 算法的实现：

    1）对图像进行梯度计算，得到梯度图像。

    2）得到梯度值最大的那一部分（比如10%）的像素直方图

    3）通过直方图的谷底，得到阈值T

  - ✓ 如果用拉普拉斯算子，不通过直方图，直接得到阈值，方法是使用拉普拉斯算子过滤图像，将0跨越点对应的灰度值为阈值T

图像分割

- 基于不同变量的阈值

  - ✓ 在某些情况下，传感器可以产生不止一个在图像中描述每一个像素的可利用的变量，因此，允许进行多谱段阈值处理
  - ✓ 例如一幅有3个变量的图像(RGB分量)，每个像素有16种可能的灰度级，构成16×16×16种灰度级（网格，立方体）
  - ✓ 阈值处理就是在三维空间内寻找点的聚簇的过程。如在直方图中找到有效点簇K，可以对RGB分量值接近某一个簇的像素赋予一个任意值(如白色的值)，对其它像素赋予另一个值（如黑色的值）
  - ✓ 彩色图像处理中的色调和饱和度易于图像分割

图像分割

- 基于区域的分割
  - ✓ 基本公式
  - ✓ 区域生长
  - ✓ 区域分离与合并

图像分割

- 基本概念
  - ✓ 目标：将区域R划分为若干个子区域$R_1,R_2,\ldots,R_n$，这些子区域满足5个条件：

    1)完备性：$\bigcup_{i=1}^{n} R_i = R$

    2)连通性：每个R_i都是一个连通区域

    3)独立性：对于任意$i\neq j$，$R_i \cap R_j = \Phi$

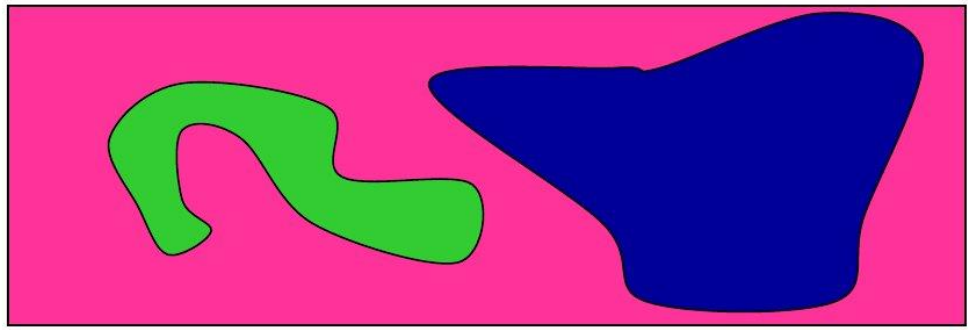
图像分割

- 基本概念

4)单一性：每个区域内的灰度级相等，

$$P(R_i) = TRUE，i = 1,2,\ldots,n$$

5)互斥性：任两个区域的灰度级不等，$P(R_i \cup R_j) = FALSE，i \neq j$



图像分割

- 区域增长的算法实现：

  1）根据图像的不同应用选择一个或一组种子，它或者是最亮或最暗的点，或者是位于点簇中心的点

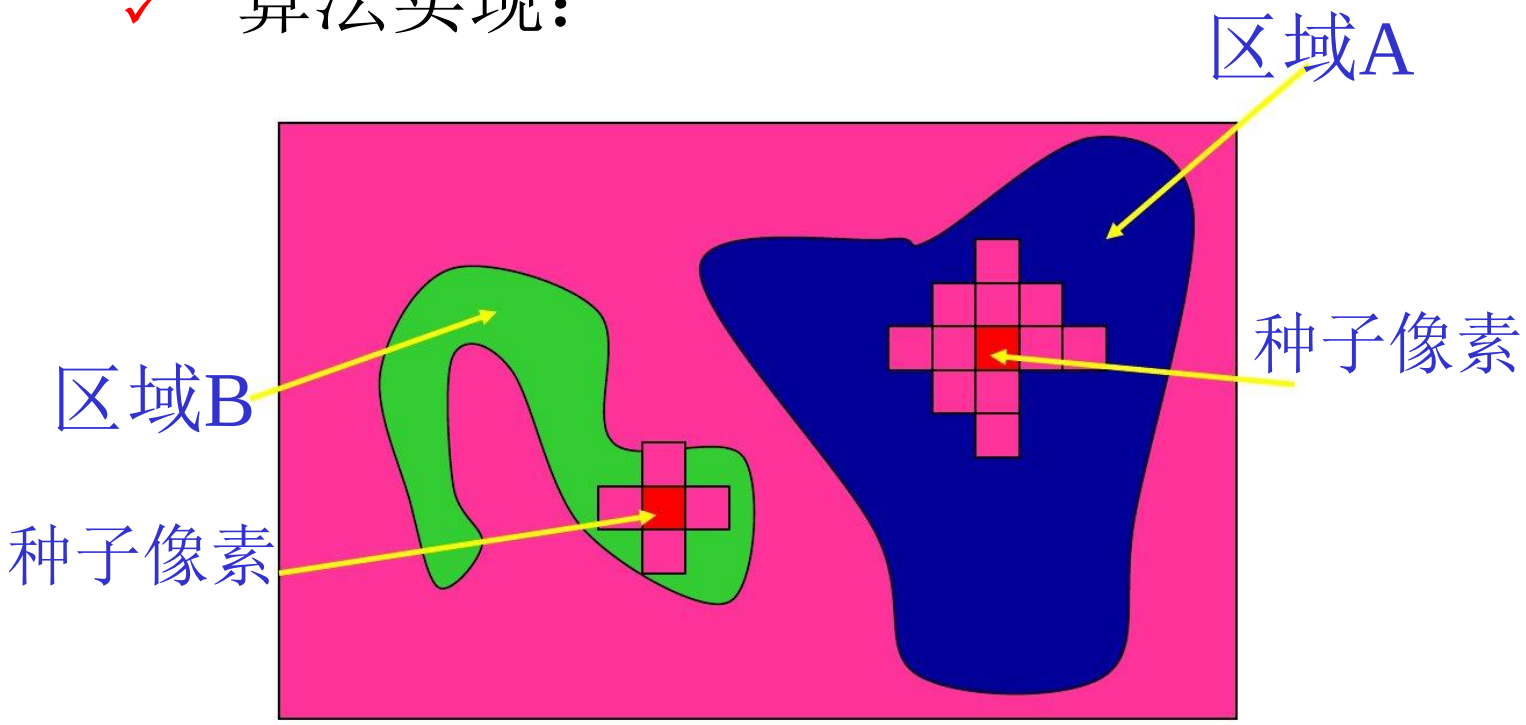
  2）选择一个描述符（条件）

  3）从该种子开始向外扩张，首先把种子像素加入结果集合，然后不断将与集合中各个像素连通、且满足描述符的像素加入集合

  4）上一过程进行到不再有满足条件的新结点加入集合为止

图像分割

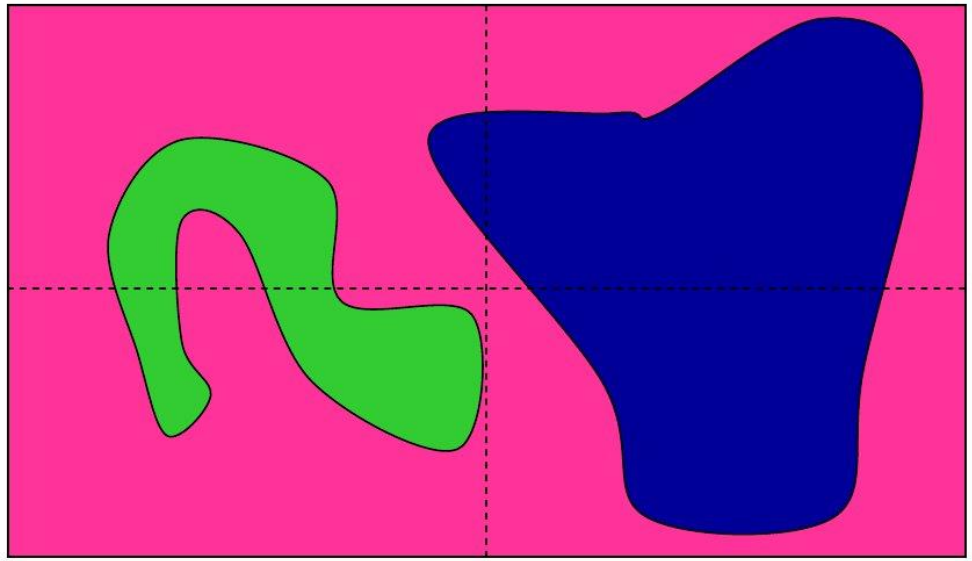
- 通过像素集合的区域增长
  - ✓ 算法实现：



图像分割

- 区域分裂与合并
  - ✓ 算法实现：

    1）对图像中灰度级不同的区域，均分为四个子区域



图像分割

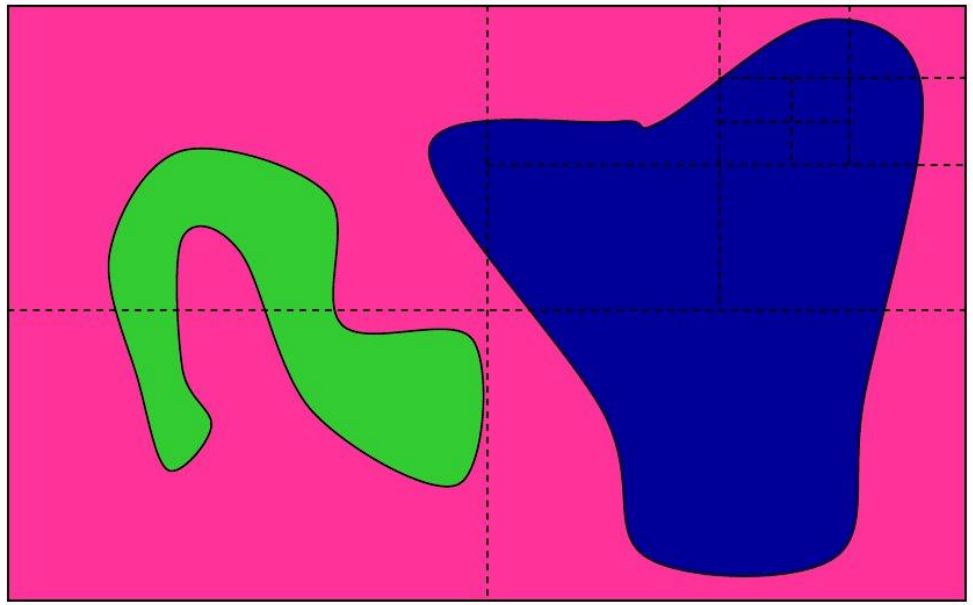
- 区域分裂与合并
  - ✓ 算法实现：

    2）如果相邻的子区域所有像素的灰度级相同，则将其合并

    3）反复进行上两步操作，直至不再有新的分裂与合并为止

图像分割

- 区域分裂与合并
  - ✓ 算法实现：



图像分割

- 区域分裂与合并
  - ✓ 算法实现：实际应用中还可作以下修改：

    P(R_i)的定义为：

    1）区域内多于80%的像素满足不等式

    $$|z_j - m_i| <= 2\sigma_i ,$$

    其中：z_j是区域R_i中第j个点的灰度级，

    m_i是该区域的平均灰度级，

    σ_i是区域的灰度级的标准方差。

    2）当P(R_i)=TRUE时，将区域内所有像素的灰度级置为m_i

图像分割

- 概述
- 间断检测
- 边缘连接和边界检测
- 阈值处理
- 基于区域的分割
- 分割中运动的应用

分割中运动的应用

- 空间技术

✓ 使用两帧图像f(x,y,t_i)和f(x,y,t_j)相减的办法，形成差值图像

$$d_{ij}(x,y)=\begin{cases}1 & \left|f(x,y,t_i)-f(x,y,t_j)\right|>T\\ 0 & \text{其它}\end{cases}$$

✓ 在动态图像处理过程中，d_{ij}中值为1的像素被认为是对象运动的结果

✓ 考虑图像帧序列f(x,y,t1),f(x,y,t2),…,f(x,y,tn),并令f(x,y,t1)为基本图像，一幅累积差异图像(ADI)由基准图像和图像序列的后续图像对比得到

分割中运动的应用

- 空间技术（续）

✓ 令R(x,y)表示基准图像，绝对ADI，正ADI和负ADI定义如下：

$$A_k(x,y)=\begin{cases}A_{k-1}(x,y)+1 & R(x,y)-f(x,y,k)>T\\ A_{k-1}(x,y) & \text{其它}\end{cases}$$

$$P_k(x,y)=\begin{cases}P_{k-1}(x,y)+1 & [R(x,y)-f(x,y,k)]>T\\ P_{k-1}(x,y) & \text{其它}\end{cases}$$

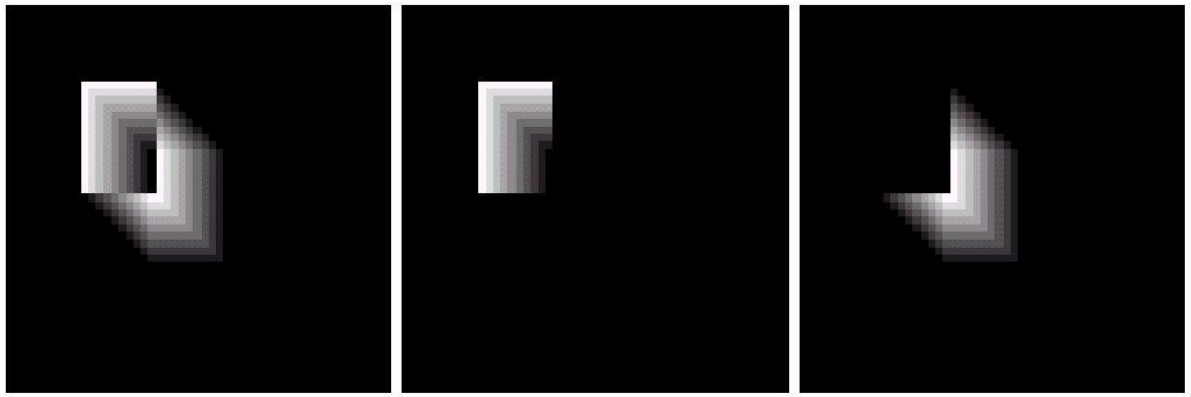
$$N_k(x,y)=\begin{cases}N_{k-1}(x,y)+1 & [R(x,y)-f(x,y,k)]<-T\\ N_{k-1}(x,y) & \text{其它}\end{cases}$$

分割中运动的应用

- 空间技术举例

  向东南方向运动的矩形目标的ADI

绝对ADI　　正ADI　　负ADI



a b c

FIGURE 10.49 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

图像文件格式

多种文件格式：BMP,GIF,TIFF,JPEG等

- 互联网用：GIF、JPG
- 印 刷 用：TIF、JPG、TAG 、PCX
- 国际标准：TIF、JPG

文件类型	说明	1位单色	8位灰度	8位彩色	16位灰度	24位彩色	48位彩色
BMP	微软视窗系统图像文件格式	✓	✓	✓	×	✓	×
GIF	用于网页上的动画、透明	✓	✓	✓	×	×	×
TIFF	可储存多幅图像	✓	✓	✓	✓	✓	✓
PCX	Paintbrush 软件图像格式	✓	✓	✓	×	✓	×
JPG	连续色调静态图像数字压缩	×	✓	×	×	✓	×
PSD	Photoshop 软件图像格式	✓	✓	✓	✓	✓	✓

图像数据格式 ---bmp图像格式

- 文件头定义（14 byte）

名称	类型	字段	说明
文件类型	WORD	bftype	规定文件类型必须是BM
文件大小	DWORD	bfsize	包括文件头在内的文件大小
保留字段1	WORD	bfReserved	必须置0
保留字段2	WORD	bfReserved 2	必须置0
数据偏移	DWORD	bfoffBits	位图数据相对于文件头的偏移字节数

bfoffBits=14(文件头长度）+40（信息头长度）
+ColorNumX4(调色板长度)

图像数据格式 ---bmp图像格式

●位图信息头（40 byte）

名称	类型	说明
biSize	DWORD	文件信息头的字节（40）
biWidth	DWORD	图形宽度（列数）
biHeight	DWORD	图像高度（行数）
biPlanes	WORD	目标设备的位平面数，必须为1
biBitCount	WORD	每个象素所需的位数，1，4，8，或24

●位图信息头数据（cont）

名称	类型	说明
biCompression	DWORD	压缩方式
biSizeImage	DWORD	图象的字节数
bimPelsPerMeter	DWORD	设备水平方向每米长度上的象素数
binPelsPerMeter	DWORD	设备垂直方向每米长度上的象素数
biClrUsed	DWORD	为零时，颜色表中点阵位图实际使用的颜色数 biBitCount=1，4，8时，为2 biBitCount biBitCount=24时，　　不用调色板
biClrImportant	DWORD	给出重要的颜色索引值，“0”表示所以颜色都重要

biSizeImage=（[(biWidth+3)/4]X4）XbiHeight

‘[　]’表示向上取整

图像数据格式 ---bmp图像格式

- 颜色表(每个颜色表占4byte)

名称	类型	说明
rgbBlue	ByTE	象素颜色中蓝色成分
rgbGreen	ByTE	象素颜色中绿色成分
rgbRed	ByTE	象素颜色中红色成分
rgbReserved	ByTE	未使用，必须为0

图像数据格式 ---bmp图像格式

- 位图数据

记录着每点像素在调色板中的索引值，其记录方式也随颜色模式而定：

2色图像每点占1位（1字节表示8个像素）；

16色图像每点占4位（1字节表示2个像素）；

256色图像每点占8位（1字节表示1个像素）；

真彩色图像每点占24位（3字节表示1个像素）；

每一行的字节数必须是4的倍数；

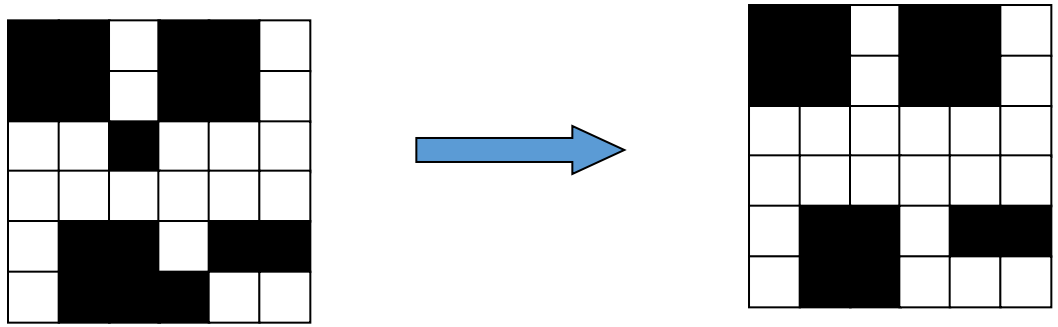
图像数据信息大小=（图像宽度*图像高度*记录像素的位数）/8 （byte）

（[(biWidth+3)/4]X4）

腐蚀 —— 基本概念

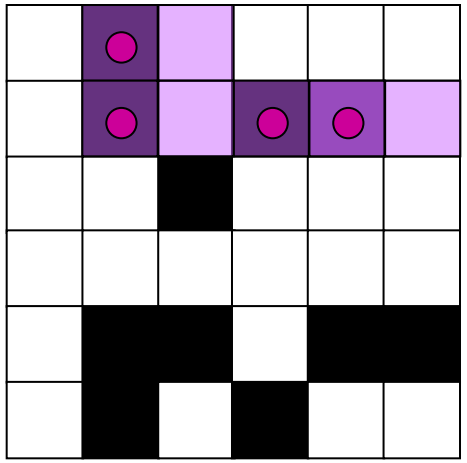
- *腐蚀* 是一种消除连通域的边界点，使边界向内收缩的处理。

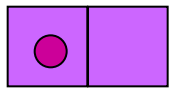
例：



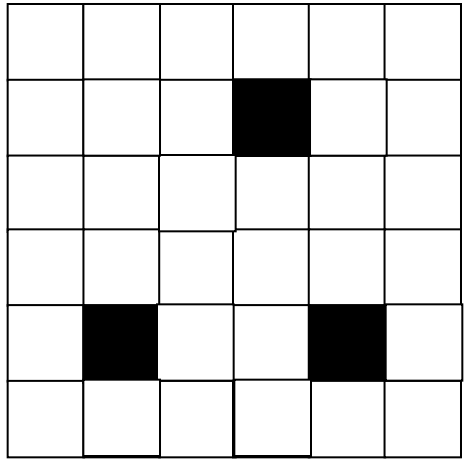
腐蚀 ——设计思想

- **设计一个结构元素，结构元素的原点定位在待处理的*目标像素*上，通过判断是否覆盖，来确定是否该点被腐蚀掉。**





结构元素



腐蚀 —— 算法步骤

1）扫描原图，找到第一个像素值为1的目标点；

2）将预先设定好形状以及原点位置的结构元素的原点移到该点；

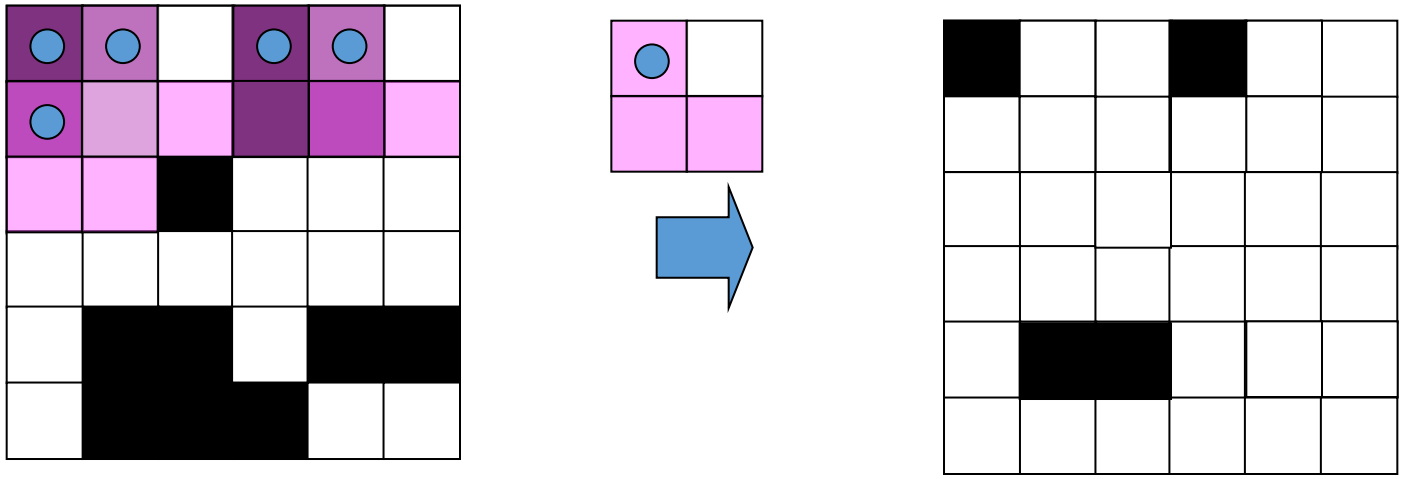
3）判断该结构元素所覆盖的像素值是否全部为1：

如果是，则腐蚀后图像中的相同位置上的像素值为1；

如果不是，则腐蚀后图像中的相同位置上的像素值为0；

4）重复2）和3），直到所有原图中像素处理完成。

腐蚀 —— 例题



注：图像画面上边框处不能被结构元素覆盖的部分可以保持原来的值不变，也可以置为背景。

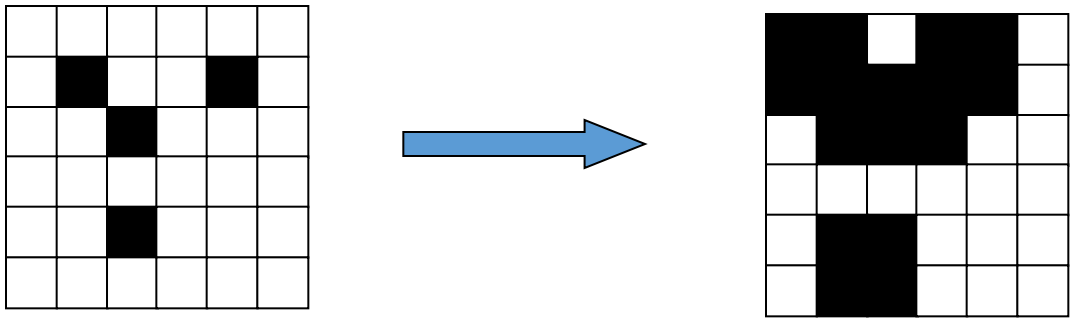
腐蚀 —— 应用

- 腐蚀处理可以将粘连在一起的不同目标物分离，并可以将小的颗粒噪声去除。

膨胀 —— 基本概念

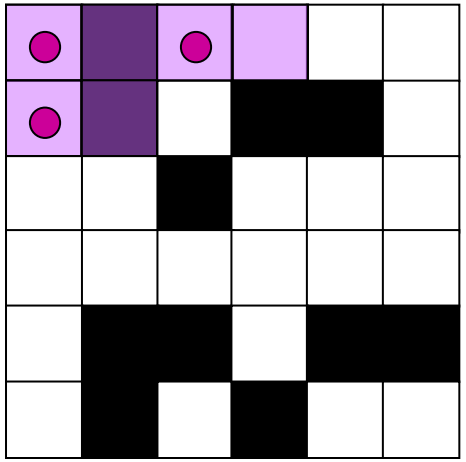
- 膨胀是将与目标区域的背景点合并到该目标物中，使目标物边界向外部扩张的处理。

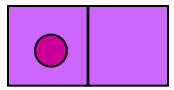
例：



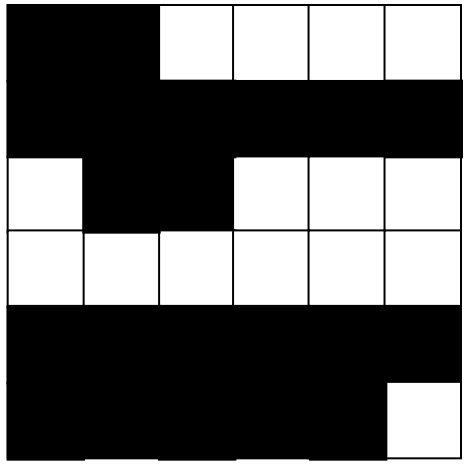
膨胀 —— 设计思想

- 设计一个结构元素，结构元素的原点定位在*背景像素*上，判断是否覆盖有目标点，来确定是否该点被膨胀为目标点。





结构元素



膨胀 —— 算法步骤

1）扫描原图，找到第一个像素值为0的背景点；

2）将预先设定好形状以及原点位置的结构元素的原点移到该点；

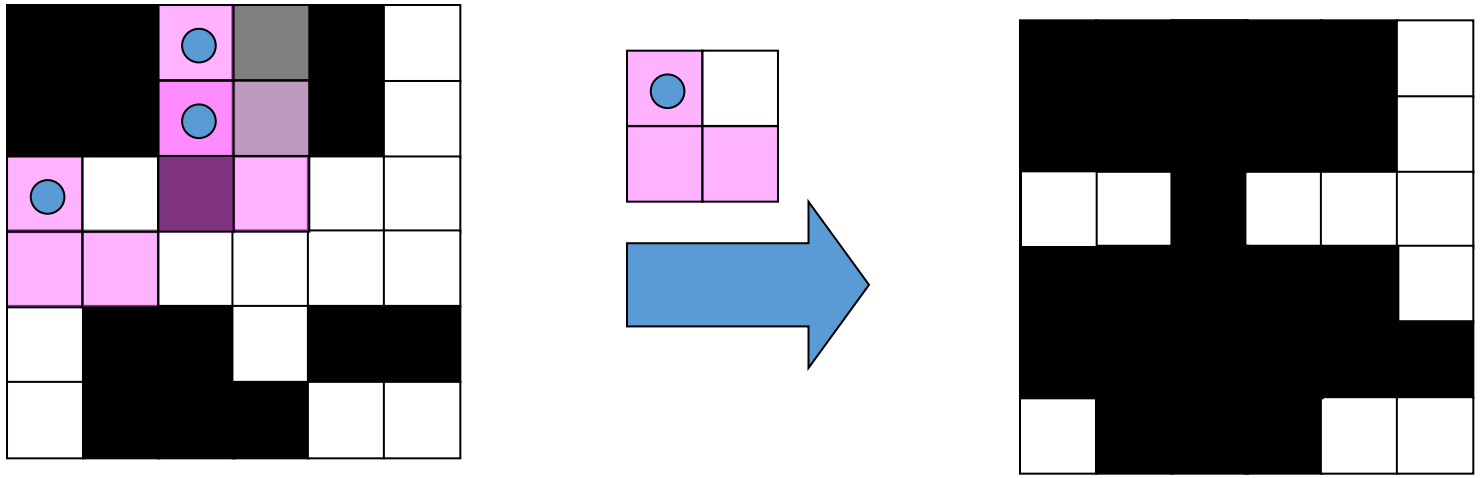
3）判断该结构元素所覆盖的像素值是否存在为1的目标点：

如果是，则膨胀后图像中的相同位置上的像素值为1；

如果不是，则膨胀后图像中的相同位置上的像素值为0；

4）重复2）和3），直到所有原图中像素处理完成。

膨胀 —— 例题



膨胀 —— 应用

- 膨胀处理可以将断裂开的目标物进行合并，便于对其整体的提取。

开运算与闭运算的提出背景

- 前面介绍的膨胀与腐蚀运算，对目标物的后处理有着非常好的作用。但是，腐蚀和膨胀运算的一个缺点是，改变了原目标物的大小。
- 为了解决这一问题，考虑到腐蚀与膨胀是一对逆运算，将膨胀与腐蚀运算同时进行。由此便构成了开运算与闭运算。

- 开运算

  思路：先腐蚀，再膨胀

  - 定义：$B \circ S = (B \otimes S) \oplus S$
  - 结果：

    1）消除细小对象

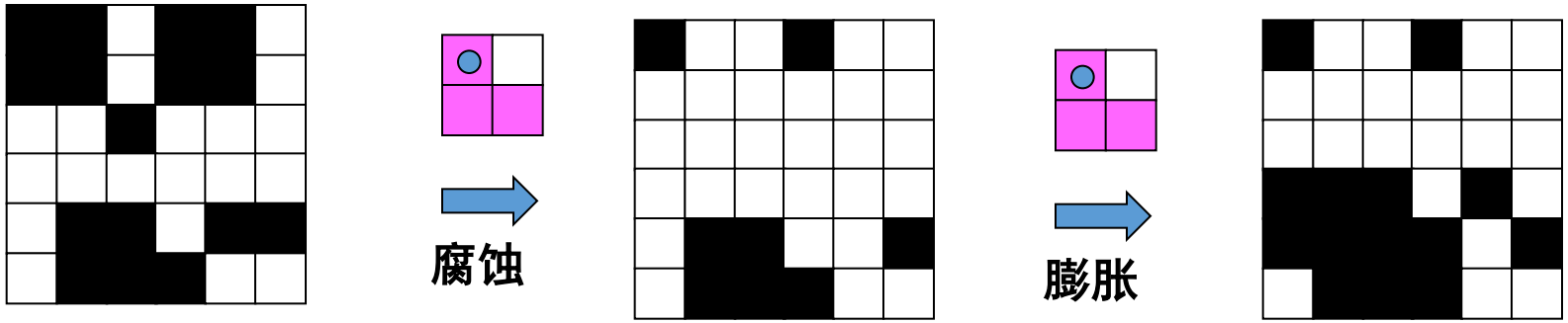
    2）在细小粘连处分离对象

    3）在不改变形状的前提下，平滑对象的边缘

开运算 —— 算法原理

- 开运算是对原图先进行腐蚀处理，后再进行膨胀的处理。
- 开运算可以在分离粘连目标物的同时，基本保持原目标物的大小。

开运算 —— 运算示例



闭运算—— 算法原理

- 闭运算是对原图先进行膨胀处理，后再进行腐蚀的处理。
- 闭运算可以在合并断裂目标物的同时，基本保持原目标物的大小。

- 闭运算

  思路：先膨胀、再腐蚀

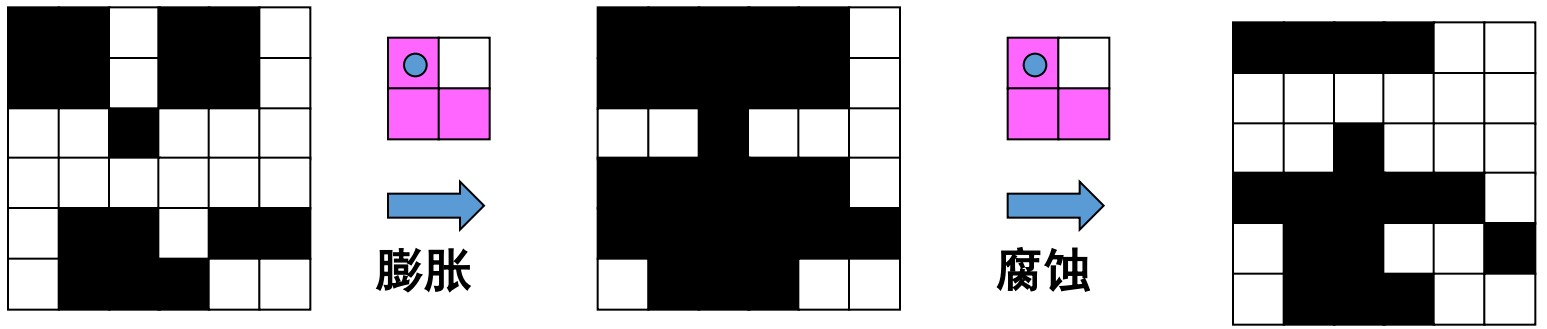
  - 定义：$B \bullet S = (B \oplus S) \otimes S$
  - 结果：

    1）填充对象内细小空洞。

    2）连接邻近对象

    3）在不明显改变面积前提下，平滑对象的边缘

闭运算 —— 运算示例



问题：本例未能将分裂成两个连通域的目标合并，怎么办?

开、闭运算的变形

- 如果当按照常规的开运算不能分离粘连，或者是闭运算不能合并断裂：
- 对于开运算可以先进行N次腐蚀，再进行N次膨胀；
- 对于闭运算可以先进行N次膨胀，再进行N次腐蚀。

变形闭运算的示例