

图 9.2 (a)集合 A 平移到 z , (b)集合 B 的反射,集合 A 和 B 来自图 9.1

9.1.2 二值图像的逻辑运算

很多应用是以本章讨论的形态学概念为基础的,并涉及二值图像。逻辑运算尽管本质上很简单,但对于实现以形态学为基础的图像处理算法是一种有力的补充手段。与掩模有关的逻辑运算在 3.4 节中介绍。在下面的讨论中,我们关注的是涉及二值像素和图像的逻辑运算。

在图像处理中用到的主要逻辑运算是:与、或和非(求补)。表 9.1 中总结了这些运算的性质。这些运算在功能上是完善的。它们可以互相组合形成其他逻辑运算。

表 9.1 三种基本的逻辑运算

p	q	p 与 q ($p \cdot q$)	p 或 q ($p + q$)	非(p) (\bar{p})
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

在两幅或多幅图像的对应像素间逐像素进行逻辑运算(除了“非”运算,此运算只对单一图像的像素进行)。因为只有两个变量都是 1 时,两个二进制变量的“与”运算才为 1,所以只有两幅输入图像的对应像素均为 1 时,“与”运算的结果图像任何位置的结果才是 1。图 9.3 显示了涉及图像逻辑运算的不同例子。这里,黑色表示 1,而白色表示 0[本章中,我们使用两种约定的说法,有时会颠倒暗色(黑色或灰色)和亮色(白色)二值的意义,取决于在给定情况下哪种表达更清楚]。使用表 9.1 中的定义构造其他的逻辑运算很容易。例如,异或运算是当两个像素的值不同时结果为 1,否则为 0。这种运算与“或”运算不同。“或”运算在两个像素有一个为 1 或两个均为 1 时结果为 1。同样,非-与运算可选出在 B 中而不在 A 中的黑色像素。

注意到刚才讨论的逻辑运算与 9.1.1 节中讨论的集合运算有一一对应的关系是很重要的,并且逻辑运算被限制只对二进制变量进行运算,而这通常不是集合运算所处理的情况。因此,比如集合论中的交集运算在运算对象为二进制变量时归为“与”运算。像“相交”和“与”这类术语(甚至它们的符号)经常在各种著述中交替地用于表示一般的或二进制值的集合运算。通常从讨论的上下文中可以清楚地知道它们的意义。

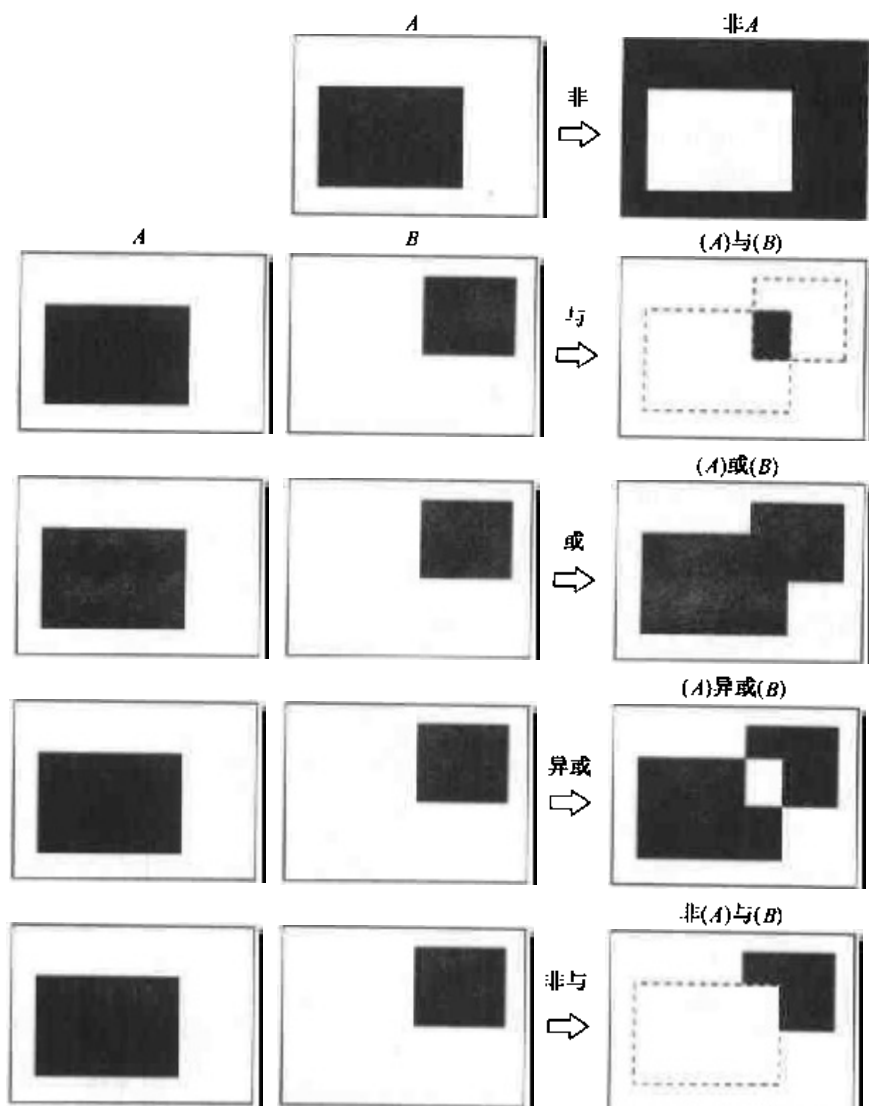


图 9.3 二值图像之间的一些逻辑运算。本例中,黑色表示二进制 1,白色表示二进制 0

9.2 膨胀与腐蚀

我们通过详细探讨两类操作——膨胀和腐蚀——开始我们对形态学上的操作的讨论。这两种操作是形态学处理的基础。实际上,本章中讨论的许多形态学算法都是以这两种原始运算为基础的。

9.2.1 膨胀

由于 A 和 B 是 Z^2 中的集合, A 被 B 膨胀定义为:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \quad (9.2.1)$$

这个公式是以得到 B 的相对于它自身原点的映象并且由 z 对映象进行位移为基础的。 A 被 B 膨胀是所有位移 z 的集合,这样, \hat{B} 和 A 至少有一个元素是重叠的。根据这种解释,式(9.2.1)可以重写为:

$$A \oplus B = \{z \mid [(\hat{B})_z \cap A] \subseteq A\} \quad (9.2.2)$$

与在其他形态学运算中一样,集合 B 通常叫做膨胀的结构元素。

在当前关于形态学方面的著述中式(9.2.1)并不是膨胀的惟一定义形式(见习题 9.10 和习题 9.11 中两种不同但等价的定义形式)。然而,当把结构元素 B 看做一个卷积模板时,式(9.2.1)的定义形式比起其他定义形式更为直观。这是它区别于其他定义形式的突出优点。尽管膨胀是以集合运算为基础,而卷积是以算术运算为基础,但相对于 B 的原点对 B 进行翻转,而后逐步移动 B 以便 B 能滑过集合(图像) A ,这一基本过程与 3.5 节和 4.2.4 节中讨论的卷积过程是相似的。

图 9.4(a)显示了一个简单的集合,图 9.4(b)显示了一个结构元素和它的映像(黑色点表示元素的中心)。此时,结构元素和它的映像是相等的,因为 B 关于它的原点对称。图 9.4(c)中的虚线显示了作为基准的初始集合,实线显示了对 \hat{B} 的原点进一步移动到 z 的限制。超出这个限制会使 \hat{B} 和 A 的交集为空。所以,所有处在这一边界之内的点构成了使用 B 进行的 A 的膨胀。图 9.4(d)显示了一个被设计用来在垂直方向比水平方向进行更多膨胀的结构元素。图 9.4(e)显示了用这个结构元素进行膨胀的结果。

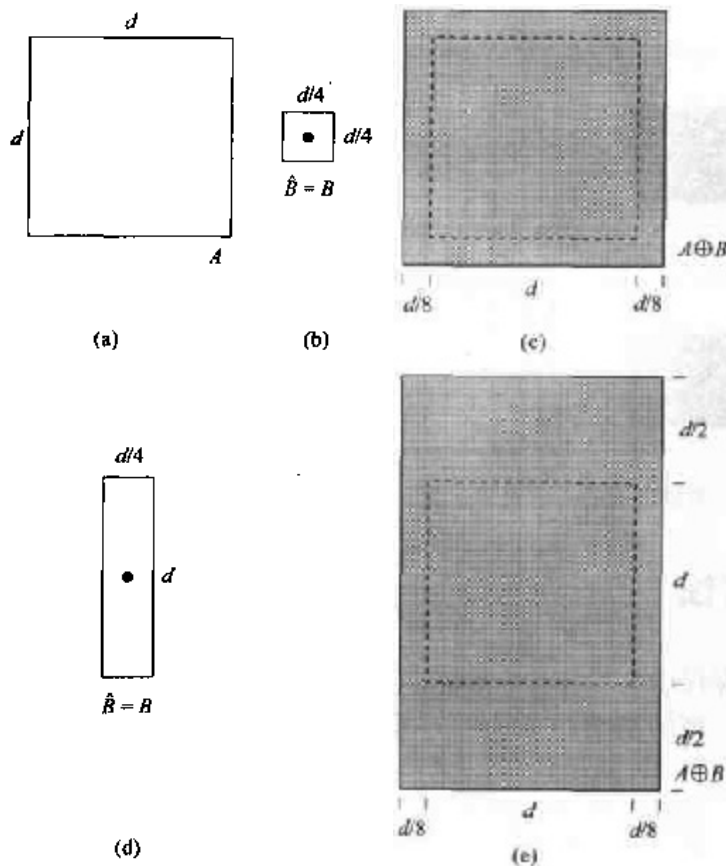


图 9.4 (a)集合 A , (b)方形结构元素(黑点为中心), (c) B 对 A 的膨胀以阴影显示, (d)拉长的结构元素, (e)使用这个元素进行的 A 的膨胀

例 9.1 将裂缝桥接起来的形态学膨胀的应用

膨胀最简单的应用之一是将裂缝桥接起来。图 9.5(a)中显示了与我们研究过的图 4.19 相同的带有间断的图像,这与低通滤波器的使用有关。已知间断的最大长度为两个像素。

图 9.5(b)显示了能够修复这些间断的简单结构元素。图 9.5(c)显示了使用这个结构元素对原图进行膨胀后的结果。现在间断已被连接起来了。形态学方法优于我们在图 4.19中用于连接间断的低通滤波方法的一个直接优点就是,这种方法在一幅二值图像中直接得到结果。而低通滤波方法从二值图像开始并生成一幅灰度级图像,这幅灰度级图像需要用门限函数进行一次处理才能将它转变回二值图像。

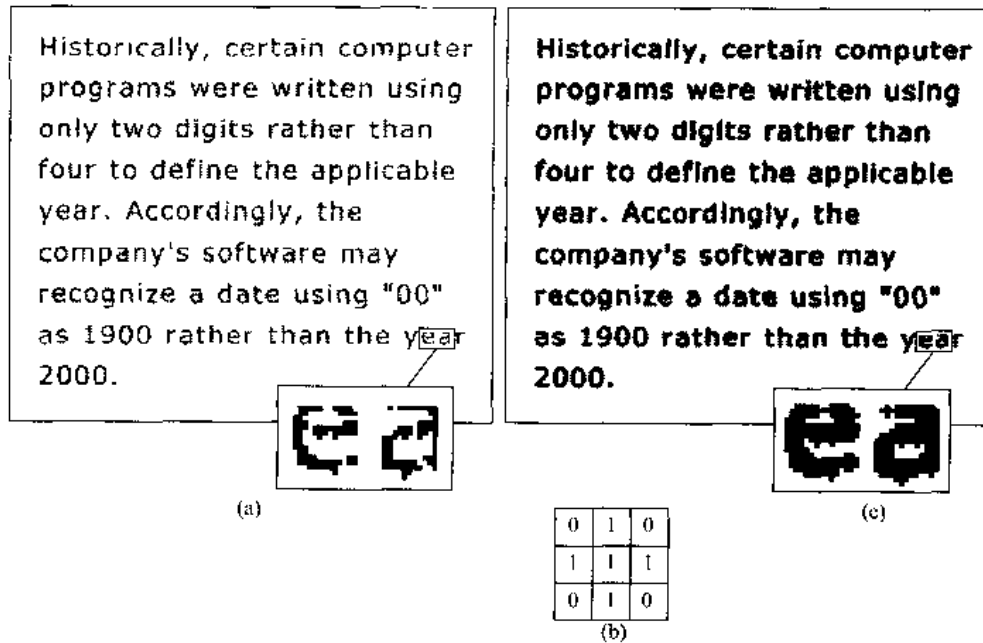


图 9.5 (a)带有间断字符的低分辨率示例文本(放大图),(b)结构元素,(c)通过(b)对(a)膨胀。断线被连接起来了

9.2.2 腐蚀

对 Z 中的集合 A 和 B ,使用 B 对 A 进行腐蚀,用 $A \ominus B$ 表示,并定义为:

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{9.2.3}$$

一句话,这个公式说明,使用 B 对 A 进行腐蚀是所有 B 中包含于 A 中的点 z 的集合用 z 平移。同膨胀的情况一样,式(9.2.3)并不是腐蚀惟一的定义形式(见习题 9.12 和习题 9.13 的两种不同但等价的定义形式)。然而,由于与式(9.2.1)相同的原因,在形态学实际应用过程中人们偏爱使用式(9.2.3)。

图 9.6 显示了与图 9.4 相似的过程。对比图 9.6(c),以前的集合 A 显示为虚线。阴影区域的边界说明 B 的原点进一步移动的界限。超出这个界限会使集合不再完全包含于集合 A 中。因此,在这个边界内(也就是阴影区域),点的位置构成了使用 B 对 A 进行的腐蚀。图 9.6(d)显示了一个拉长的结构元素,图 9.6(e)显示了用这个元素腐蚀 A 的结果。注意原来的集合被腐蚀成一条线。

膨胀和腐蚀对于集合求补运算和反射运算是彼此对偶的。即:

$$(A \ominus B)^c = A^c \oplus \hat{B} \tag{9.2.4}$$

为了说明确定形态学表达式有效性的典型方法,我们对这个结果进行正规的证明。从腐蚀的

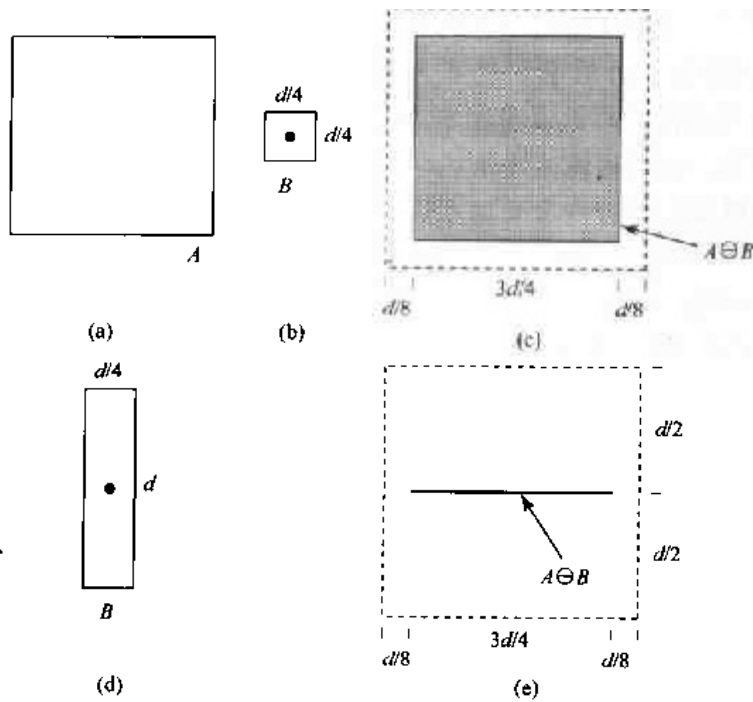


图 9.6 (a)集合 A, (b)方形结构元素, (c)由 B 对 A 腐蚀, 如阴影所示, (d)拉长的结构元素, (e)使用这个元素对 A 腐蚀

定义开始, 我们有:

$$(A \ominus B)^c = \{z \mid (B)_z \subseteq A\}^c$$

如果集合 $(B)_z$ 包含于集合 A, 则 $(B)_z \cap A^c = \emptyset$, 此时前述公式变为:

$$(A \ominus B)^c = \{z \mid (B)_z \cap A^c = \emptyset\}^c$$

但, 满足 $(B)_z \cap A^c = \emptyset$ 的 z 的集合的补集是满足 $(B)_z \cap A^c \neq \emptyset$ 的集合。因此:

$$(A \ominus B)^c = \{z \mid (B)_z \cap A^c \neq \emptyset\} = A^c \oplus \hat{B}$$

这是从式(9.2.1)得出的最后一步, 这就证明了这个结论。

例 9.2 使用形态学腐蚀除去图像的某些部分

腐蚀的一种最简单的用途是从二值图像中消除不相关的细节(根据尺寸)。图 9.7(a)显示的二值图像包含边长为 1, 3, 5, 7, 9 和 15 个像素的正方形。假设这里只留下最大的正方形而除去其他的正方形, 我们可以通过用比我们要保留的对象稍小的结构元素对图像进行腐蚀。在这个例子中我们选择 13×13 像素大小的结构元素。图 9.7(b)显示了用这个结构元素对原图像进行腐蚀后得到的结果。图中只有部分最大的正方形保留下来。如图 9.7(c)所示, 我们可以通过使用用来腐蚀的结构元素对这 3 个正方形进行膨胀恢复它们原来 15×15 像素的尺寸(通常膨胀不能完全恢复被腐蚀的对象; 见习题 9.18)。注意例子中所有 3 幅图像中的对象都是用白色像素表示, 而不是像前面的例子中用黑色表示。如我们前面提到的, 两种表达方法在实际中均有使用。除非申明, 否则我们令结构元素中起作用的元素选择与感兴趣对象一样的二进制值。这个例子中提到的概念是形态学过滤的基础, 将在接下来的小节中进行介绍。

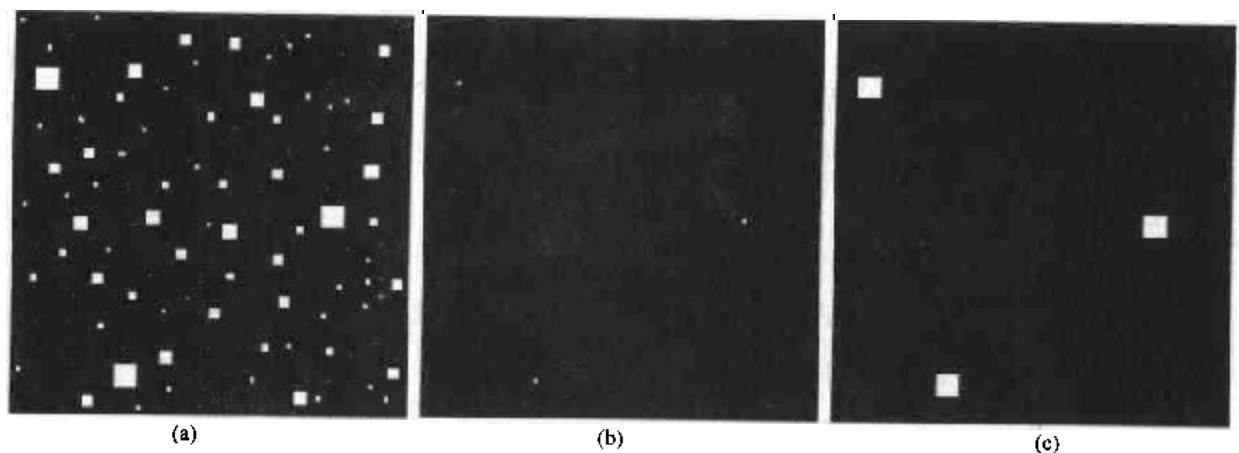


图 9.7 (a)图像内部边长为 1,3,5,7,9 和 15 个像素的正方形的图像,(b)用第一个方形结构元素(一边有 13 个像素)对(a)进行腐蚀,(c)使用相同的结构元素对(b)进行膨胀

9.3 开操作与闭操作

如我们所见到的,膨胀使图像扩大而腐蚀使图像缩小。本节中,我们讨论另外两个重要的形态学操作:开操作与闭操作。开操作一般使对象的轮廓变得光滑,断开狭窄的间断和消除细的突出物。闭操作同样使轮廓线更为光滑,但与开操作相反的是,它通常消弥狭窄的间断和长细的鸿沟,消除小的孔洞,并填补轮廓线中的断裂。

使用结构元素 B 对集合 A 进行开操作,表示为 $A \circ B$,定义为:

$$A \circ B = (A \ominus B) \oplus B \quad (9.3.1)$$

因此,用 B 对 A 进行开操作就是用 B 对 A 腐蚀,然后用 B 对结果进行膨胀。

同样,使用结构元素 B 对集合 A 的闭操作,表示为 $A \cdot B$,定义如下:

$$A \cdot B = (A \oplus B) \ominus B \quad (9.3.2)$$

这个公式说明,使用结构元素 B 对集合 A 的闭操作就是用 B 对 A 进行膨胀,而后用 B 对结果进行腐蚀。

开操作有一个简单的几何解释(图 9.8)。假设我们将结构元素 B 看做一个(扁平的)“转球”。 $A \circ B$ 的边界通过 B 中的点完成,即 B 在 A 的边界内转动时, B 中的点所能到达的 A 的边界的最远点。这个开操作的几何拟合特性使我们得出了集合论的一个公式。这个公式说明用 B 对 A 进行开操作是通过求取 B 在拟合 A 时的平移的并集得到的。就是说,开操作可以表示为一个拟合操作:

$$A \circ B = \bigcup \{ (B)_z \mid (B)_z \subseteq A \} \quad (9.3.3)$$

这里 $\bigcup \{ \cdot \}$ 表示大括号中所有集合的并集。

闭操作有相似的几何解释,只是我们现在在边界的外部转动 B (图 9.9)。简而言之,开操作和闭操作是一对对偶操作,所以闭操作在边界外部转动球是预料之中的事。从几何上讲,当且仅当对包含 w 的 $(B)_z$ 进行的所有平移都满足 $(B)_z \cap A \neq \emptyset$ 时,点 w 是 $A \cdot B$ 的一个元素。图 9.9 说明了闭操作这一基本的几何性质。

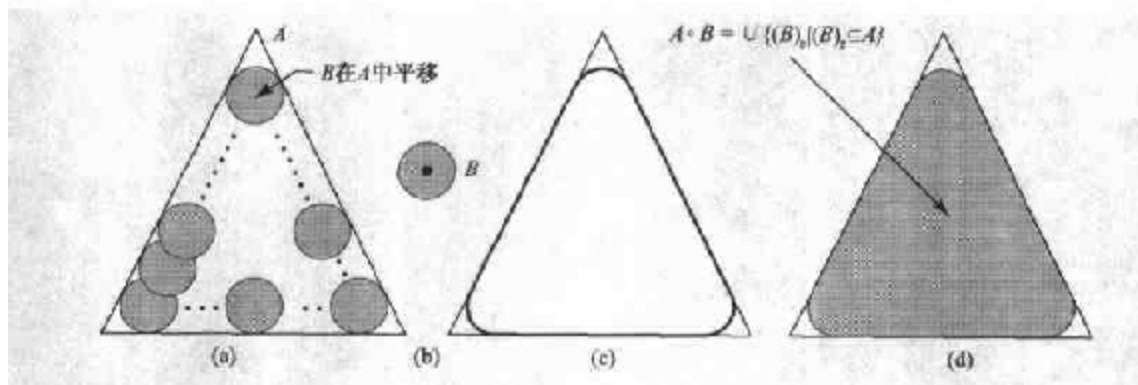


图 9.8 (a)结构元素 B 沿着 A 的内部边界转动(点表示 B 的圆心), (b)结构元素 B , (c)粗线是开操作的外部边界, (d)完全开操作(阴影部分)

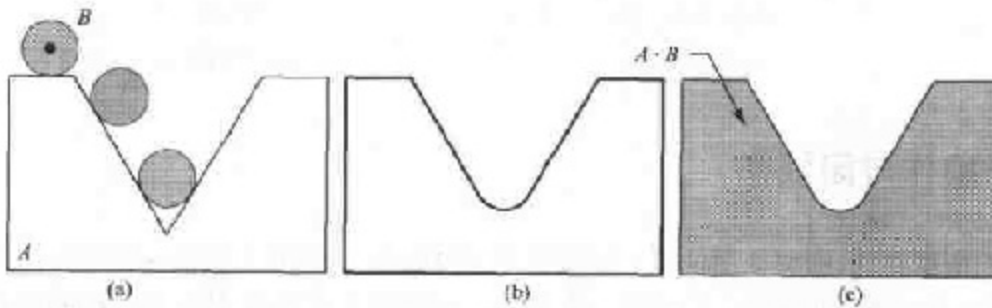


图 9.9 (a)结构元素 B 在集合 A 的外部边界上转动, (b)粗线表示闭操作的外部边界, (c)完全的闭操作(阴影部分)

例 9.3 对形态学上的开操作和闭操作的简单说明

图 9.10 进一步对开操作和闭操作进行了说明。图 9.10(a)显示了集合 A , 图 9.10(b)显示了在腐蚀过程中的一块圆盘形结构元素的各种位置。当腐蚀完成时, 得到图 9.10(c)中显示的连接图。注意, 两个主要部分之间的桥接部分消失了。这部分的宽度与结构元素的直径相比较要细; 就是说, 这部分集合不能完全包含结构元素, 因此, 无法满足式(9.2.3)的条件。对象最右边的两个部分也是如此。圆盘无法拟合的突出部分被消除了。在图 9.10(d)中显示了对腐蚀后的集合进行膨胀的操作, 图 9.10(e)显示了开操作最后的结果。注意, 方向向外的角变圆滑了, 而方向向内的角没受影响。

同样, 图 9.10(f)到(i)显示了使用同样的结构元素对 A 进行闭操作的结果。我们注意到方向向内的拐角变得圆滑了, 而方向向外的拐角没有变化。在 A 的边界上, 最左边的侵入部分在尺寸上明显地减小了。因为在这个位置上, 圆盘无法拟合。同时也要注意在使用圆形结构元素对集合 A 进行开操作和闭操作后, 所得对象的各个部分得到了平滑处理。

如此处的膨胀和腐蚀一样, 开操作和闭操作也是一对关于集合求补及映像的对偶操作。

即:

$$(A \cdot B)^c = (A^c \circ \hat{B}) \quad (9.3.4)$$

我们将这个结果的证明作为练习(习题 9.14)留给读者。

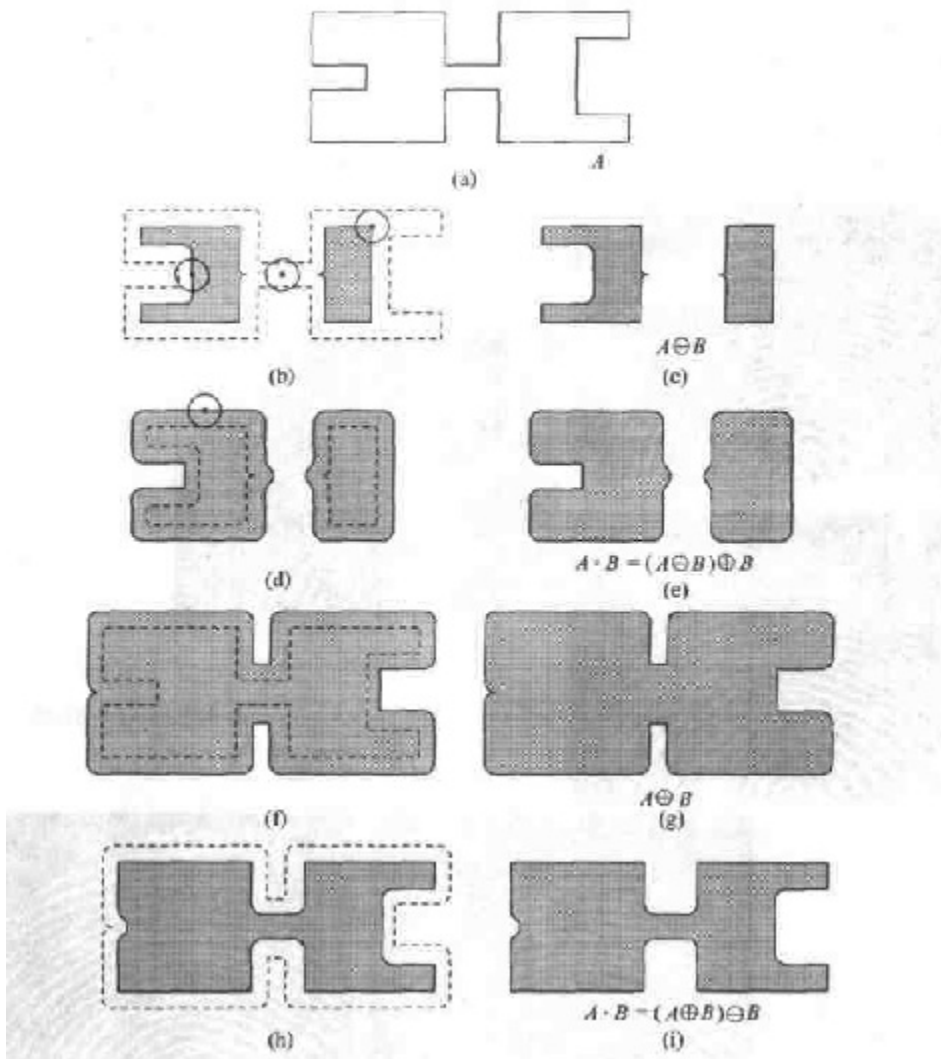


图 9.10 形态学开操作和闭操作。结构元素是(b)中各部分显示的小圆。黑点是结构元素的中心

开操作满足下列性质：

- (i) $A \circ B$ 是 A 的子集合(子图)。
- (ii) 如果 C 是 D 的子集, 则 $C \circ B$ 是 $D \circ B$ 的子集。
- (iii) $(A \circ B) \circ B = A \circ B$ 。

同样, 闭操作也满足下列性质：

- (i) A 是 $A \cdot B$ 的子集(子图)。
- (ii) 如果 C 是 D 的子集, 则 $C \cdot B$ 是 $D \cdot B$ 的子集。
- (iii) $(A \cdot B) \cdot B = A \cdot B$ 。

注意, 由两种情况下的条件(iii)可知, 算子应用一次后, 一个集合进行多少次开操作或闭操作都不会有变化。

例 9.4 形态学滤波的开操作和闭操作的使用

形态学运算可以用于构造与第 3 章中讨论的空间滤波概念相类似的滤波器。显示于

图 9.11(a)中的二值图像显示了受噪声污染的部分指纹图像。这里噪声表现为黑色背景上的亮元素和亮指纹部分的暗元素。我们的目的是消除噪声及它对印刷所造成的影响,使图像失真尽可能减小。由闭操作后紧接着进行开操作形成的形态学滤波器可用于实现这个目的。

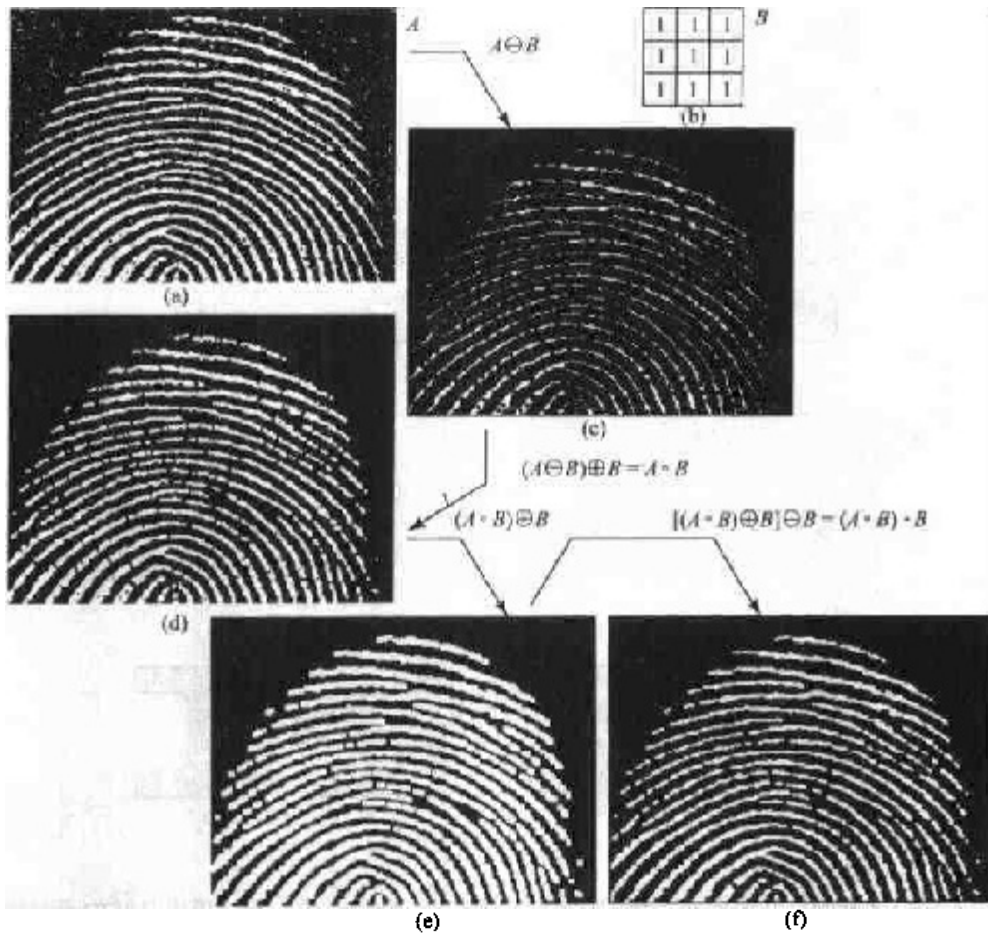


图 9.11 (a)有噪声的图像,(c)腐蚀图像,(d)A的开操作,(e)开操作的膨胀,(f)开操作的闭操作[这个例子的原图由美国国家标准技术研究所(NIST)提供]

图 9.11(b)显示了所使用的结构元素。图 9.11 余下的部分显示了滤波操作的每一步过程。图 9.11(c)显示了使用结构元素对 A 进行腐蚀的结果。背景噪声在开操作的腐蚀过程中被完全消除了,因为在这种情况下,图像中噪声部分的物理尺寸均比结构元素小。而包含于指纹中的噪声元素(黑点)的尺寸却有增加。原因是,当目标被腐蚀时,这些元素被作为应该增大尺寸的内部边界进行了处理。这种增大在图 9.11(c)中进行膨胀的过程中被抵消了。图 9.11(d)显示了该结果。包含于指纹中的噪声分量的尺寸被减小或被完全消除了。

刚才描述的两步操作构成了用 B 对 A 进行的开操作。我们注意到图 9.11(d)中开操作的实际效果是消除背景和指纹中的所有噪声。然而,在指纹纹路间产生了新的间断。为了防止这种不希望的影响,我们在开操作的基础上进行膨胀,如图 9.11(e)所示。大部分间断被恢复了,但指纹的纹路变粗了,可以通过腐蚀来弥补出现的这种情况。示于图 9.11(f)的结果构成了对图 9.11(d)中开操作的闭操作。最后结果的噪声斑点清除得相

当干净,但这种方法有缺点,即有些指纹纹路没有被完全修复,并还有间断。对这种情况的改进也并非毫无希望,只是因为这一过程中在保持连续性方面没有提出任何条件而已(我们将在例9.8中讨论这一问题并在11.1.5节中给出处理这个问题的方法)。

9.4 击中或击不中变换

形态学上的击中或击不中变换是形状检测的基本工具。我们用图9.12作为辅助介绍这个概念。图9.12显示了一个由3种形状(子集)组成的集合 A ,子集用 X , Y 和 Z 表示。图9.12(a)到(c)中的阴影部分指明了初始集合,而图9.12(d)和(e)中的阴影部分指出了进行形态学操作后的结果。目的是找到3种形状之一的位置,如 X 的位置。

令每种形状的重心为它的原点。设 X 被包围在一个小窗口 W 中。与 W 有关的 X 的局部背景定义为集合的差($W - X$),如图9.12(b)所示。图9.12(c)显示了 A 的补集,在后面将使用到它。图9.12(d)显示了由 X 对 A 腐蚀的结果(显示虚线作为参考)。使用 X 对 A 进行的腐蚀是 X 原点位置的集合。这样, X 就完全包含在 A 中了。换一个角度解释, $A \ominus X$ 从几何上可以被看做 X 的原点所有位置的集合,在这些位置 X 找到了在 A 中的匹配(击中)。请记住图9.12中 A 只包含3种彼此不相连的集合 X , Y 和 Z 。

图9.12(e)显示了由局部背景集合($W - X$)对集合 A 的补集腐蚀的结果。图9.12(e)的外圈阴影区域是腐蚀部分。我们根据图9.12(d)和(e)注意到, X 在 A 内能得到精确拟合的位置集合,是由 X 对 A 的腐蚀和由($W - X$)对 A^c 的腐蚀的交集,如图9.12(f)所示。这个交集正好是我们要找的位置。换句话说,如果 B 表示由 X 和 X 的背景构成的集合,则在 A 中对 B 进行的匹配(或匹配操作的集合)表示为 $A \circledast B$:

$$A \circledast B = (A \ominus X) \cap [A^c \ominus (W - X)] \quad (9.4.1)$$

我们可以通过令 $B = (B_1, B_2)$ 对这种表示法稍微进行推广。这里 B_1 是由与一个对象相联系的 B 元素构成的集合。 B_2 是与相应背景有关的 B 元素的集合。根据前面的讨论, $B_1 = X$, $B_2 = (W - X)$ 。用这个表示方法,式(9.4.1)变为:

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2) \quad (9.4.2)$$

因此,集合 $A \circledast B$ 同时包含了所有的原点, B_1 在 A 内找到匹配, B_2 在 A^c 中找到匹配。通过应用式(9.1.8)给出的集合之差的定义和式(9.2.4)给出的腐蚀与膨胀间的对偶关系,我们可以将式(9.4.2)写成:

$$A \circledast B = (A \ominus B_1) - (A \oplus \hat{B}_2) \quad (9.4.3)$$

然而,式(9.4.2)更为直观。我们将上述3个公式称为形态学上的击中或击不中变换。

使用与对象有关的结构元素 B_1 和与背景有关的 B_2 的原因是基于以下假设的定义,即,只有在两个或更多对象构成彼此不相交(不连通)的集合时,这些对象才是可区分的。要保证这个假设,需要在每个对象周围至少被一圈一个像素宽的背景围绕的条件。在某些应用中,我们也许对在某个集合中检测1和0组成的某种模式感兴趣,而此时是不需要背景的。在这种场合,击中或击不中变换转变成简单的腐蚀过程。正如前边所指出的那样,腐蚀是进行一系列的匹配,但对于检测单个对象来说不需要额外的背景匹配。这种简化的模式检测方案将用于下面章节讨论的某些算法中。