18 Morphology

18.1 Introduction

In Chapters 16 and 17 we discussed the segmentation process that extracts objects from images, i. e., identifies which pixels belong to which objects. Now we can perform the next step and analyze the *shape* of the objects. In this chapter, we discuss a class of neighborhood operations on binary images, the morphological operators that modify and analyze the form of objects.

18.2 Neighborhood Operations on Binary Images

18.2.1 Binary Convolution

In our survey of digital image processing, operators relating pixels in a small neighborhood emerged as a versatile and powerful tool for scalar and vector images (Chapter 4). The result of such an operation in binary images can only be a zero or a one. Consequently, neighborhood operators for binary images will work on the shape of object, adding pixels to an object or deleting pixels from an object. In Sections 4.2 and 4.3 we discussed the two basic operations for combining neighboring pixels of gray value images: convolution ("weighting and summing up") and rank value filtering ("sorting and selecting"). With binary images, we do not have much choice as to which kind of operations to perform. We can combine pixels only with the logical operations of Boolean algebra. We might introduce a *binary convolution* by replacing the multiplication of the image and mask pixels with an *and operation* and the summation by an *or operation*:

$$\boxed{g'_{mn} = \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} m_{m',n'} \wedge g_{m+m',n+n'}.} \tag{18.1}$$

The $\wedge$ and $\vee$ denote the logical *and* and *or* operations, respectively. The binary image $\boldsymbol{G}$ is convolved with a symmetric $2R + 1 \times 2R + 1$ mask $\boldsymbol{M}$. Note that in contrast to convolution operations, the mask is not mirrored at the origin (see Section 4.2.5).

B. Jähne, Digital Image Processing
ISBN 3-540-24035-7

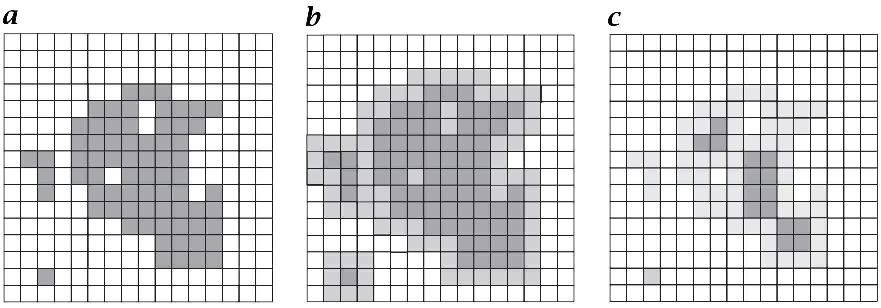

Figure 18.1: ***b*** *Dilation and* **c** *erosion of a binary object in* ***a*** *with a* 3×3 *mask. The removed (erosion) and added (dilation) pixels are shown in a lighter color.*

What does this operation achieve? Let us assume that all the coefficients of the mask are set to 'one'. If one or more object pixels, i. e., 'ones', are within the mask, the result of the operation will be one, otherwise it is zero (Fig. 18.1a, b). Hence, the object will be dilated. Small holes or cracks will be filled and the contour line will become smoother, as shown in Fig. 18.2b. The operator defined by Eq. (18.1) is known as the *dilation operator*.

Interestingly, we can end up with the same effect if we apply *rank-value filter* (see Section 4.3) to binary images. Let us take the *maximum operator*. The maximum will then be one if one or more 'ones' are within the mask, just as with the binary convolution operation in Eq. (18.1).

The *minimum operator* has the opposite effect. Now the result is only one if the mask is completely within the object (Fig. 18.1c). In this way the object is eroded. Objects smaller than the mask disappear completely and objects connected only by a small bridge will become disconnected. The *erosion* of an object can also be performed using binary convolution with logical *and* operations:

$$\boxed{g'_{mn} = \bigwedge_{m'=-R}^{R} \bigwedge_{n'=-R}^{R} m_{m',n'} \wedge g_{m+m',n+n'}} \tag{18.2}$$

For higher-dimensional images, Eqs. (18.1) and (18.2) just need to be appended by another loop for each coordinate. In 3-D space, the dilation operator is, for instance,

$$g'_{lmn} = \bigvee_{l'=-R}^{R} \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} m_{l'm'n'} \wedge g_{l+l',m+m',n+n'}. \tag{18.3}$$

By transferring the concepts of neighborhood operations for gray value images to binary images we have gained an important tool to operate on the form of objects. We have already seen in Fig. 18.1 that these operations can be used to fill small holes and cracks or to eliminate small objects.

The size of the mask governs the effect of the operators, therefore the mask is often called the *structure element*. For example, an erosion operation works like a net that has holes in the shape of the mask. All objects that fit through the hole will slip through and disappear from the image. An object remains only if at least at one point the mask is completely covered by object pixels. Otherwise it disappears.

An operator that works on the form of objects is called a *morphological operator*. The name originates from the research area of morphology which describes the form of objects in biology and geosciences.

18.2.2 Operations on Sets

We used a rather unconventional way to introduce morphological operations. Normally, these operations are defined as operations on sets of pixels. We regard $\boldsymbol{G}$ as the set of all the pixels of the matrix that are not zero. $\boldsymbol{M}$ is the set of the non-zero mask pixels. With $\boldsymbol{M}_p$ we denote the mask shifted with its reference point (generally but not necessarily its center) to the pixel p. Erosion is then defined as

$$\boxed{\boldsymbol{G} \ominus \boldsymbol{M} = \{p : \boldsymbol{M}_p \subseteq \boldsymbol{G}\}} \tag{18.4}$$

and dilation as

$$\boxed{\boldsymbol{G} \oplus \boldsymbol{M} = \{p : \boldsymbol{M}_p \cap \boldsymbol{G} \neq \emptyset\}.} \tag{18.5}$$

These definitions are equivalent to Eqs. (18.1) and (18.2), respectively. We can now express the erosion of the set of pixels $\boldsymbol{G}$ by the set of pixels $\boldsymbol{M}$ as the set of all the pixels p for which $\boldsymbol{M}_p$ is completely contained in $\boldsymbol{G}$. In contrast, the dilation of $\boldsymbol{G}$ by $\boldsymbol{M}$ is the set of all the pixels for which the intersection between $\boldsymbol{G}$ and $\boldsymbol{M}_p$ is not an empty set. As the set-theoretical approach leads to more compact and illustrative formulas, we will use it from now on.

Equations Eqs. (18.1) and (18.2) still remain important for the implementation of morphological operators with logical operations. The erosion and dilation operators can be regarded as elementary morphological operators from which other more complex operators can be built. Their properties are studied in detail in the next section.

18.3 General Properties

Morphological operators share most but not all of the properties we have discussed for linear convolution operators in Section 4.2. The properties discussed below are not restricted to 2-D images but are generally valid for N-dimensional image data.

18.3.1 Shift Invariance

Shift invariance results directly from the definition of the erosion and dilation operator as convolutions with binary data in Eqs. (18.1) and (18.2). Using the *shift operator* S as defined in Eq. (4.17) and the operator notation, we can write the shift invariance of any morphological operator $\mathcal{M}$ as

$$\mathcal{M}\left({}^{mn}S\boldsymbol{G}\right) = {}^{mn}S\left(\mathcal{M}\boldsymbol{G}\right). \qquad (18.6)$$

18.3.2 Principle of Superposition

What does the *superposition principle* for binary data mean? For gray value images it is defined as

$$\mathcal{H}(a\boldsymbol{G} + b\boldsymbol{G}') = a\mathcal{H}\boldsymbol{G} + b\mathcal{H}\boldsymbol{G}'. \qquad (18.7)$$

The factors a and b make no sense for binary images; the addition of images corresponds to the union or *logical or* of images. If the superposition principle is valid for morphological operations $\mathcal{M}$ on binary images, it has the form

$$\mathcal{M}(\boldsymbol{G} \cup \boldsymbol{G}') = (\mathcal{M}\boldsymbol{G}) \cup (\mathcal{M}\boldsymbol{G}') \quad \text{or} \quad \mathcal{M}(\boldsymbol{G} \vee \boldsymbol{G}') = (\mathcal{M}\boldsymbol{G}) \vee (\mathcal{M}\boldsymbol{G}'). \qquad (18.8)$$

The operation $\boldsymbol{G} \vee \boldsymbol{G}'$ means a pointwise *logical or* of the elements of the matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$. Generally, morphological operators are not additive in the sense of Eq. (18.8). While the dilation operation conforms to the superposition principle, erosion does not. The erosion of the union of two objects is generally a superset of the union of two eroded objects:

$$\begin{array}{lcl} (\boldsymbol{G} \cup \boldsymbol{G}') \ominus \boldsymbol{M} & \supseteq & (\boldsymbol{G} \ominus \boldsymbol{M}) \cup (\boldsymbol{G}' \ominus \boldsymbol{M}) \\ (\boldsymbol{G} \cup \boldsymbol{G}') \oplus \boldsymbol{M} & = & (\boldsymbol{G} \oplus \boldsymbol{M}) \cup (\boldsymbol{G}' \oplus \boldsymbol{M}). \end{array} \qquad (18.9)$$

18.3.3 Commutativity and Associativity

Morphological operators are generally not *commutative*:

$$\boldsymbol{M}_1 \oplus \boldsymbol{M}_2 = \boldsymbol{M}_2 \oplus \boldsymbol{M}_1, \text{ but } \boldsymbol{M}_1 \ominus \boldsymbol{M}_2 \neq \boldsymbol{M}_2 \ominus \boldsymbol{M}_1. \qquad (18.10)$$

We can see that erosion is not commutative if we take the special case that $\boldsymbol{M}_1 \supset \boldsymbol{M}_2$. Then the erosion of $\boldsymbol{M}_2$ by $\boldsymbol{M}_1$ yields the empty set. However, both erosion and dilation masks consecutively applied in a cascade to the same image $\boldsymbol{G}$ are commutative:

$$\boxed{\begin{array}{lll} (\boldsymbol{G} \ominus \boldsymbol{M}_1) \ominus \boldsymbol{M}_2 & = \boldsymbol{G} \ominus (\boldsymbol{M}_1 \oplus \boldsymbol{M}_2) & = (\boldsymbol{G} \ominus \boldsymbol{M}_2) \ominus \boldsymbol{M}_1 \\ (\boldsymbol{G} \oplus \boldsymbol{M}_1) \oplus \boldsymbol{M}_2 & = \boldsymbol{G} \oplus (\boldsymbol{M}_1 \oplus \boldsymbol{M}_2) & = (\boldsymbol{G} \oplus \boldsymbol{M}_2) \oplus \boldsymbol{M}_1. \end{array}} \qquad (18.11)$$

These equations are important for the implementation of morphological operations. Generally, the cascade operation with k structure elements

$\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_k$ is equivalent to the operation with the structure element $\boldsymbol{M} = \boldsymbol{M}_1 \oplus \boldsymbol{M}_2 \oplus \ldots \oplus \boldsymbol{M}_k$. In conclusion, we can decompose large structure elements in the very same way as we decomposed linear shift-invariant operators. An important example is the composition of separable structure elements by the horizontal and vertical elements $\boldsymbol{M} = \boldsymbol{M}_x \oplus \boldsymbol{M}_y$. Another less trivial example is the construction of large one-dimensional structure elements from structure elements including many zeros:

$$\begin{array}{c} [1\ 1\ 1] \oplus [1\ 0\ 1] = [1\ 1\ 1\ 1\ 1] \\ [1\ 1\ 1\ 1\ 1] \oplus [1\ 0\ 0\ 0\ 1] = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \oplus [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \\ = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \end{array}. \tag{18.12}$$

In this way, we can build up large exponentially growing structure elements with a minimum number of logical operations just as we built up large convolution masks by cascading in Section 11.5. It is more difficult to obtain isotropic, i. e., circular-shaped, structure elements. The problem is that the dilation of horizontal and vertical structure elements always results in a rectangular-shaped structure element, but not in a circular mask. A circular mask can be approximated, however, with one-dimensional structure elements running in more directions than only along the axes. As with smoothing convolution masks, large structure elements can be built efficiently by cascading multistep masks.

18.3.4 Monotony

Erosion and dilation are monotonic operations

$$\begin{array}{lcl} \boldsymbol{G}_1 \subseteq \boldsymbol{G}_2 & \rightsquigarrow & \boldsymbol{G}_1 \oplus \boldsymbol{M} \subseteq \boldsymbol{G}_2 \oplus \boldsymbol{M} \\ \boldsymbol{G}_1 \subseteq \boldsymbol{G}_2 & \rightsquigarrow & \boldsymbol{G}_1 \ominus \boldsymbol{M} \subseteq \boldsymbol{G}_2 \ominus \boldsymbol{M}. \end{array} \tag{18.13}$$

Monotony means that the subset relations are invariant with respect to erosion and dilation.

18.3.5 Distributivity

Linear shift-invariant operators are *distributive* with regard to addition. The corresponding distributivities for erosion and dilation with respect to the union and intersection of two images $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are more complex:

$$\begin{array}{lcl} (\boldsymbol{G}_1 \cap \boldsymbol{G}_2) \oplus \boldsymbol{M} & \subseteq & (\boldsymbol{G}_1 \oplus \boldsymbol{M}) \cap (\boldsymbol{G}_2 \oplus \boldsymbol{M}) \\ (\boldsymbol{G}_1 \cap \boldsymbol{G}_2) \ominus \boldsymbol{M} & = & (\boldsymbol{G}_1 \ominus \boldsymbol{M}) \cap (\boldsymbol{G}_2 \ominus \boldsymbol{M}) \end{array} \tag{18.14}$$

and

$$\begin{array}{lcl} (\boldsymbol{G}_1 \cup \boldsymbol{G}_2) \oplus \boldsymbol{M} & = & (\boldsymbol{G}_1 \oplus \boldsymbol{M}) \cup (\boldsymbol{G}_2 \oplus \boldsymbol{M}) \\ (\boldsymbol{G}_1 \cup \boldsymbol{G}_2) \ominus \boldsymbol{M} & \supseteq & (\boldsymbol{G}_1 \ominus \boldsymbol{M}) \cup (\boldsymbol{G}_2 \ominus \boldsymbol{M}). \end{array} \tag{18.15}$$

Erosion is distributive over the intersection operation, while dilation is distributive over the union operation.

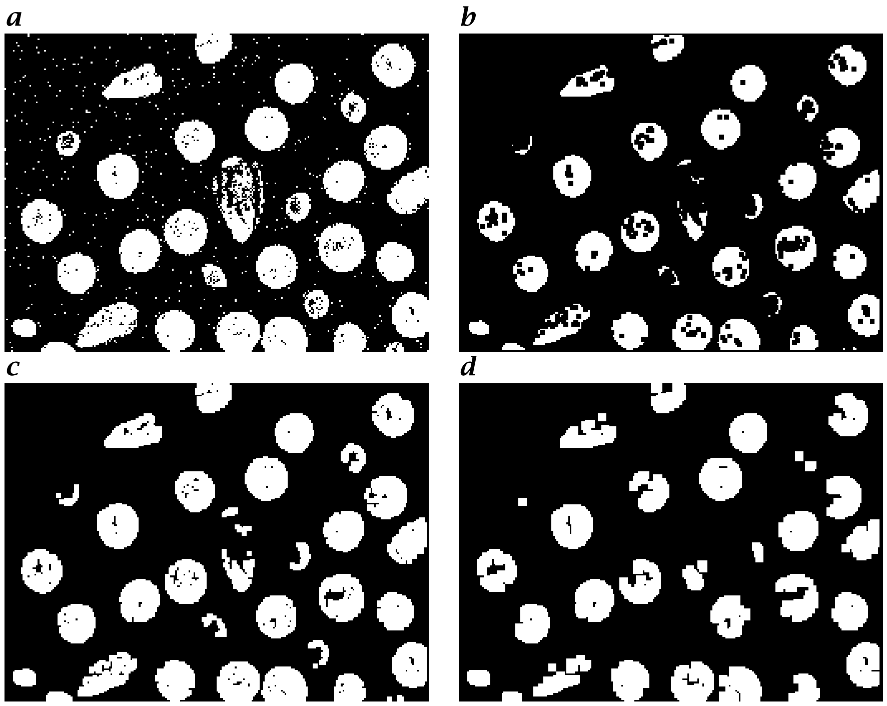

Figure 18.2: *Erosion and opening:* ***a*** *original binary image;* ***b*** *erosion with a* 3×3 *mask;* ***c*** *opening with a* 3×3 *mask;* ***d*** *opening with a* 5×5 *mask.*

18.3.6 Duality

Erosion and dilation are *dual operators*. By negating the binary image, erosion is converted to dilation and vice versa:

$$\begin{aligned} \overline{G} \ominus M &= \overline{G \oplus M} \\ \overline{G} \oplus M &= \overline{G \ominus M}. \end{aligned} \qquad (18.16)$$

18.4 Composite Morphological Operators

18.4.1 Opening and Closing

Using the elementary erosion and dilation operations we now develop further useful operations to work on the form of objects. While in the previous section we focused on the general and theoretical aspects of morphological operations, we now concentrate on application.

The erosion operation is useful for removing small objects. However, it has the disadvantage that all the remaining objects shrink in size. We can avoid this effect by dilating the image after erosion with the same

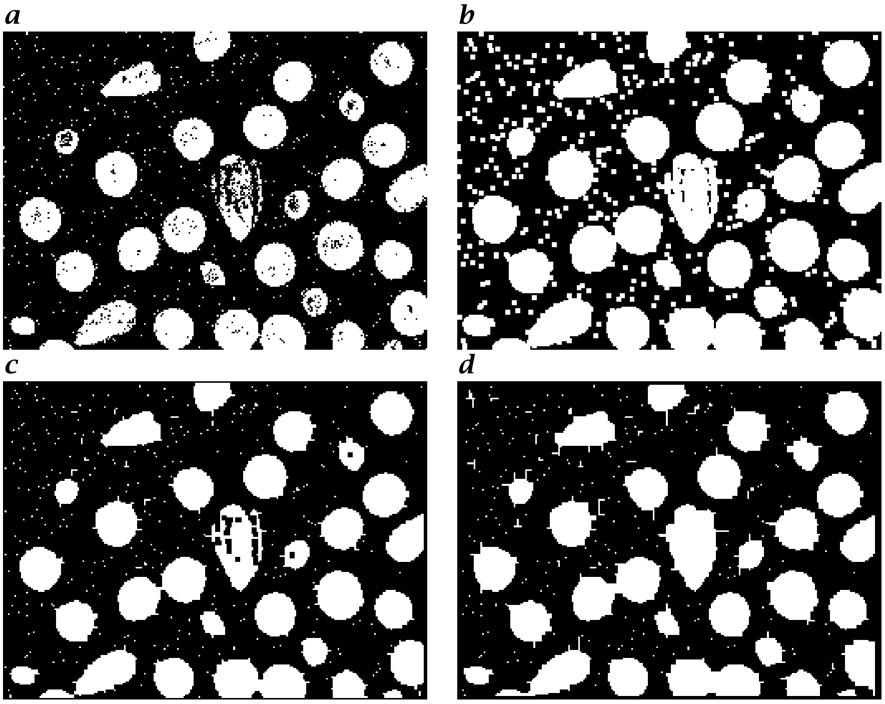

Figure 18.3: *Dilation and closing:* ***a*** *original binary image;* ***b*** *dilation with a* 3×3 *mask;* ***c*** *closing with a* 3×3 *mask;* ***d*** *closing with a* 5×5 *mask.*

structure element. This combination of operations is called an *opening operation*

$$\boldsymbol{G} \circ \boldsymbol{M} = (\boldsymbol{G} \ominus \boldsymbol{M}) \oplus \boldsymbol{M}. \tag{18.17}$$

The opening sieves out all objects which at no point completely contain the structure element, but avoids a general shrinking of object size (Fig. 18.2c, d). It is also an ideal operation for removing lines with a thickness smaller than the diameter of the structure element. Note also that the object boundaries become smoother.

In contrast, the dilation operator enlarges objects and closes small holes and cracks. General enlargement of objects by the size of the structure element can be reversed by a following erosion (Fig. 18.3d and e). This combination of operations is called a *closing operation*

$$\boldsymbol{G} \bullet \boldsymbol{M} = (\boldsymbol{G} \oplus \boldsymbol{M}) \ominus \boldsymbol{M}. \tag{18.18}$$

The area change of objects with different operations may be summarized by the following relations:

$$\boldsymbol{G} \ominus \boldsymbol{M} \subseteq \boldsymbol{G} \circ \boldsymbol{M} \subseteq \boldsymbol{G} \subseteq \boldsymbol{G} \bullet \boldsymbol{M} \subseteq \boldsymbol{G} \oplus \boldsymbol{M}. \tag{18.19}$$

Opening and closing are *idempotent operations*:

$$\begin{aligned} \boldsymbol{G} \bullet \boldsymbol{M} &= (\boldsymbol{G} \bullet \boldsymbol{M}) \bullet \boldsymbol{M} \\ \boldsymbol{G} \circ \boldsymbol{M} &= (\boldsymbol{G} \circ \boldsymbol{M}) \circ \boldsymbol{M}, \end{aligned} \tag{18.20}$$

i. e., a second application of a closing and opening with the same structure element does not show any further effects.

18.4.2 Hit-Miss Operator

The *hit-miss operator* originates from the question of whether it is possible to detect objects of a specific shape. The erosion operator only removes objects that at no point completely contain the structure element and thus removes objects of very different shapes. Detection of a specific shape requires a combination of two morphological operators. As an example, we discuss the detection of objects containing horizontal rows of three consecutive pixels.

If we erode the image with a 1×3 mask that corresponds to the shape of the object

$$\boldsymbol{M}_1 = [1\ 1\ 1], \tag{18.21}$$

we will remove all objects that are smaller than the target object but retain also all objects that are larger than the mask, i. e., where the shifted mask is a subset of the object $\boldsymbol{G}$ ($\boldsymbol{M}_p \subseteq \boldsymbol{G}$, Fig. 18.4d). Thus, we now need a second operation to remove all objects larger than the target object.

This can be done by analyzing the background of the original binary image. Thus, we can use as a second step an erosion of the background with a 3×5 mask $\boldsymbol{M}_2$ in which all coefficients are zero except for the pixels in the background, surrounding the object. This is a negative mask for the object:

$$\boldsymbol{M}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{18.22}$$

The eroded background then contains all pixels in which the background has the shape of $\boldsymbol{M}_2$ or larger ($\boldsymbol{M}_2 \subseteq \overline{\boldsymbol{G}}$, Fig. 18.4b). This corresponds now to objects having the sought shape or a smaller one. As the first erosion obtains all objects equal to or larger than the target, the intersection of the image eroded with $\boldsymbol{M}_1$ and the background eroded with $\boldsymbol{M}_2$ gives all center pixels of the objects with horizontal rows of three consecutive pixels (Fig. 18.4e). In general, the *hit-miss operator* is defined as

$$\boxed{\begin{aligned} \boldsymbol{G} \otimes (\boldsymbol{M}_1, \boldsymbol{M}_2) &= (\boldsymbol{G} \ominus \boldsymbol{M}_1) \cap (\overline{\boldsymbol{G}} \ominus \boldsymbol{M}_2) \\ &= (\boldsymbol{G} \ominus \boldsymbol{M}_1) \cap \overline{(\boldsymbol{G} \oplus \boldsymbol{M}_2)} \\ \text{with} \quad & \boldsymbol{M}_1 \cap \boldsymbol{M}_2 = \emptyset. \end{aligned}} \tag{18.23}$$

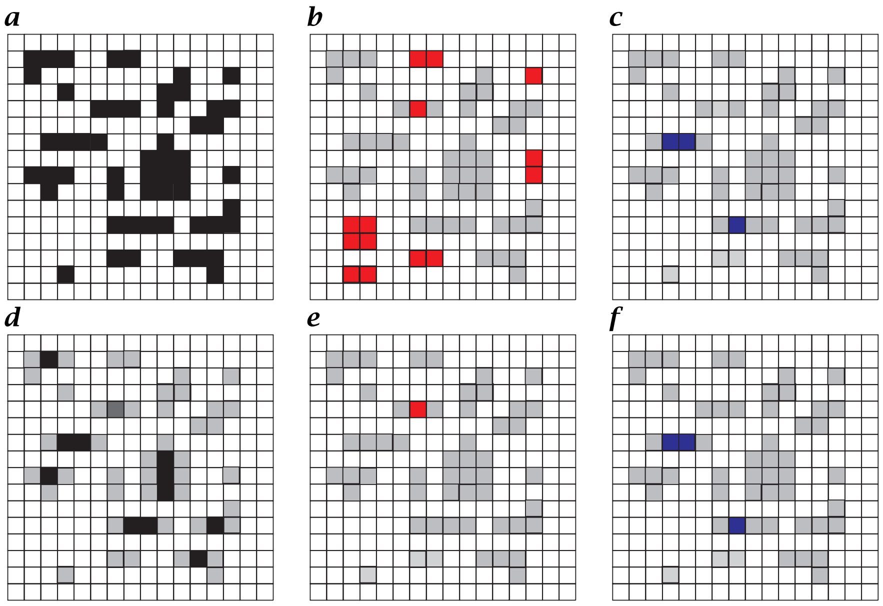

Figure 18.4: *Illustration of the hit-miss operator for extracting all objects containing horizontal rows of three consecutive pixels:*
a *original image; in all following images, the black pixels of the original image are displayed as light gray pixels and black pixels are pixels with the value 1 generated by the corresponding operator;*
b *background eroded by a* 3×5 *mask (Eq. (18.22));* ***c*** *background eroded by the* 3×7 *mask (Eq. (18.24));*
d *object eroded by the* 1×3 *mask (Eq. (18.21));*
e *intersection of* ***b*** *and* ***d*** *extracting the objects with horizontal rows of 3 consecutive pixels;*
f *intersection of* ***c*** *and* ***d*** *extracting objects with 3 to 5 horizontal rows of consecutive pixels in a* 3×7 *free background.*

The condition $\boldsymbol{M}_1 \cap \boldsymbol{M}_2 = \emptyset$ is necessary, because otherwise the hit-miss operator would result in the empty set.

With the hit-miss operator, we have a flexible tool with which we can detect objects of a given specific shape. The versatility of the hit-miss operator can easily be demonstrated by using another miss mask

$$\boldsymbol{M}_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \qquad (18.24)$$

Erosion of the background with this mask leaves all pixels in the binary image where the union of the mask $\boldsymbol{M}_3$ with the object is zero (Fig. 18.4c). This can only be the case for objects with horizontal rows of one to five consecutive pixels in a 3×7 large background. Thus, the hit-miss operator with $\boldsymbol{M}_1$ and $\boldsymbol{M}_3$ gives all center pixels of objects with horizontal rows of 3 to 5 consecutive pixels in a 3×7 large background (Fig. 18.4f).

As the hit and miss masks of the hit-miss operator are disjunct, they can be combined into one mask using a hit (1), miss (-1), and don't care (0) notation. The combined mask is marked by 1 where the hit mask is one, by 0 where the miss mask is one, and by x where both masks are zero. Thus, the hit-miss mask for detecting objects with horizontal rows of 3 to 5 consecutive pixels is

$$\boldsymbol{M} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 1 & 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \qquad (18.25)$$

If a hit-miss mask has no don't-care pixels, it extracts objects of an exact shape given by the 1-pixels of the mask. If don't-care pixels are present in the hit-miss mask, the 1-pixels give the minimum and the union of the 1-pixels and don't-care pixels the maximum of the detected objects.

As another example, the hit-mass mask

$$\boldsymbol{M}_I = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad (18.26)$$

detects isolated pixels. Thus, the operation $\boldsymbol{G}/\boldsymbol{G} \otimes \boldsymbol{M}_I$ removes isolated pixels from a binary image. The / symbol represents the set difference operator.

The hit-miss operator detects certain shapes only if the miss mask surrounds the hit mask. If the hit mask touches the edge of the hit-miss mask, only certain shapes at the border of an object are detected. The hit-miss mask

$$\boldsymbol{M}_C = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \qquad (18.27)$$

for instance, detects lower right corners of objects.

18.4.3 Boundary Extraction

Morphological operators can also be used to extract the boundary of a binary object. This operation is significant as the boundary is a complete yet compact representation of the geometry of an object from which further shape parameters can be extracted, as we discuss later in this chapter.

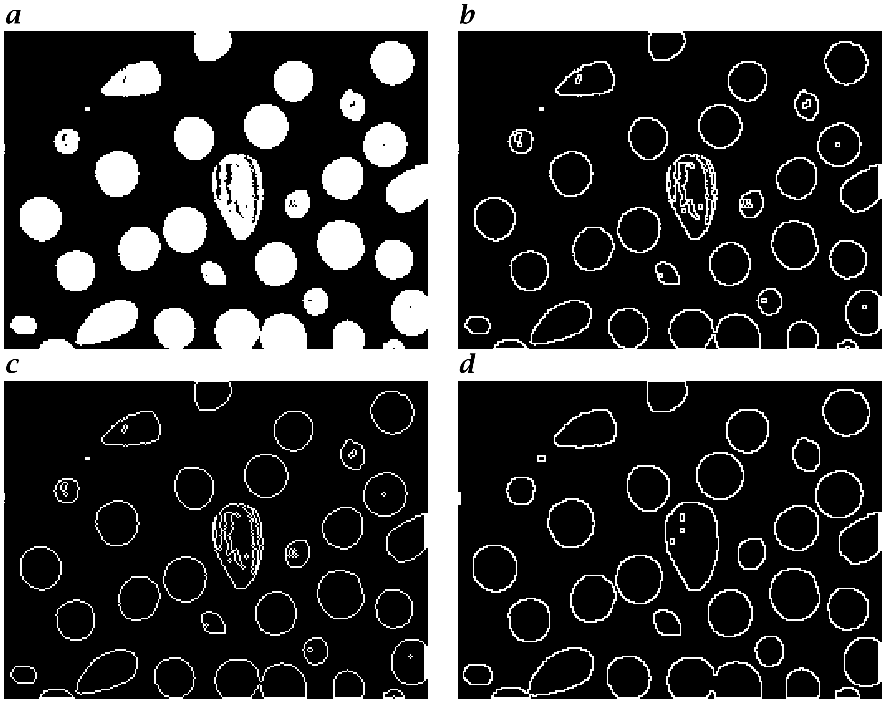

***Figure 18.5:** Boundary extraction with morphological operators: **a** original binary image; **b** 8-connected and **c** 4-connected boundaries extracted with $\boldsymbol{M}_{b4}$ and $\boldsymbol{M}_{b8}$, respectively, Eq. (18.28); **d** 8-connected boundary of the background extracted by using Eq. (18.30).*

Boundary points miss at least one of their neighbors. Thus, an erosion operator with a mask containing all possible neighbors removes all boundary points. These masks for the 4- and 8-neighborhood are:

$$\boldsymbol{M}_{b4} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{M}_{b8} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{18.28}$$

The boundary is then obtained by the set difference (/ operator) between the object and the eroded object:

$$\begin{aligned} \partial \boldsymbol{G} &= \boldsymbol{G} / (\boldsymbol{G} \ominus \boldsymbol{M}_b) \\ &= \boldsymbol{G} \cap \overline{(\boldsymbol{G} \ominus \boldsymbol{M}_b)} \\ &= \boldsymbol{G} \cap (\bar{\boldsymbol{G}} \oplus \boldsymbol{M}_b). \end{aligned} \tag{18.29}$$

As Eq. (18.29) shows, the boundary is also given as the intersection of the object with the dilated background. Figure 18.5 shows 4- and 8-connected boundaries extracted from binary objects using Eq. (18.28).

The boundary of the background is similarly given by dilating the object and subtracting it:

$$\partial \boldsymbol{G}_B = (\boldsymbol{G} \oplus \boldsymbol{M}_b)/\boldsymbol{G}. \tag{18.30}$$

18.4.4 Distance transforms

The boundary consists of all points with a distance zero to the edge of the object. If we apply boundary extraction again to the object eroded with the mask Eq. (18.28), we obtain all points with a distance of one to the boundary of the object. A recursive application of the boundary extraction procedure thus gives the distance of all points of the object to the boundary. Such a transform is called a *distance transform* and can be written as

$$\boldsymbol{D} = \bigcup_{n=1}^{\infty} \left[(\boldsymbol{G} \ominus \boldsymbol{M}_b^{n-1})/(\boldsymbol{G} \ominus \boldsymbol{M}_b^{n}) \cdot n \right], \tag{18.31}$$

where the operation $\cdot$ denotes pointwise multiplication of the binary image of the nth distance contour with the number n.

This straightforward distance transform has two serious flaws. First, it is a slow iterative procedure. Second, it does not give the preferred Euclidian distance but — depending on the chosen neighborhood connectivity — the city block or chess board distance (Section 2.2.3).

Fortunately, fast algorithms are available for computing the Euclidian distance. The Euclidian distance transform is an important transform because it introduces isotropy for morphological operations. All morphological operations suffer from the fact that the Euclidian distance is not a natural measure on a rectangular grid. Square-shaped structure elements, for instance, all inherit the chess board distance. Successive dilation with such structure elements makes the objects look more and more like squares, for instance.

The Euclidian distance transform can be used to perform isotropic erosion and dilation operations. For an erosion operation with a radius r, we keep only pixels with a distance greater than r in the object. In a similar way, an isotropic dilation can be performed by computing a Euclidian distance transform of the background and then an isotropic erosion of the background.

18.5 *Exercises*

18.1: Elementary morphological operators

Interactive demonstration of elementary morphological operators, such as erosion, dilation, opening, and closing (dip6ex18.01)

Problem 18.2: *Commutativity of morphological operators

Check if morphological erosion and dilation operators are commutative and prove your conclusion! (Hint: If one of the operators is not commutative, present a counter example.)

Problem 18.3: Hit-miss operator

Interactive demonstration of the hit-miss operator (dip6ex18.02)

Problem 18.4: Morphological boundary detection

Interactive demonstration of morphological boundary detection (dip6ex18.03)

Problem 18.5: Morphological operations with gray-scale images

Interactive demonstration of morphological operators with gray-scale images (dip6ex18.04)

Problem 18.6: *Opening and closing

Opening and closing are two of the most important morphological operators.

1. What happens if you apply an opening or a closing with the same structure element several times?
2. What is the structure element for an opening operator that should remove all horizontal lines with a width of only one pixel?

Problem 18.7: *Combination of morphological operators

What kind of operation is performed if you

1. subtract an eroded binary image from the original binary image,
2. subtract the original binary image from a dilated image, and
3. subtract an eroded image from a dilated image?

What is different with these three combined morphological operators?

Problem 18.8: **Decomposition of morphological operators

Large convolution masks can often be decomposed into a number of smaller masks and thus be performed much more efficiently. Is the same also possible with morphological masks (structure elements)? Examine this question with the following examples

1.
$$[1\ 1\ 1] \quad \text{and} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

2.
$$[1\ 1\ 1] \quad \text{and} \quad [1\ 0\ 0\ 1\ 0\ 0\ 1]$$

Problem 18.9: *Object detection with the hit-miss operator

The hit-miss operator can be used to detect objects with a specific form.

1. Show with some examples that the hit-miss mask

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

   detects isolated pixels.
2. Which objects are extracted with the following two hit-miss masks?

$$[0\;1\;\;-1] \quad \text{and} \quad [-1\;1\;0]\,?$$

18.6 *Further Readings*

The authoritative source for the theory of morphological image processing is a monograph written by the founders of image morphology, see Serra [184]. The more practical aspects are covered by Jähne and Haußecker [93, Chapter 14] and Soille [192]. Meanwhile morphological image processing is a mature research area with a solid theoretical foundation and a wide range of applications as can be seen from recent conference proceeding, e. g., Serra and Soille [185].