

Chapter 17

Shape-Based Operations

An shape-based operation identifies or acts on groups of pixels that belong to the same object or image component. We have already seen how components may be identified on the basis of pixel gray level, color (multispectral gray level), or features such as edges, corners, texture, etc.. The output of an shape-based operation may not be an image at all, but rather a description of the objects, locations, etc. in the image.

Applications:

- image segmentation
- image description
- image compression

One application of both point and local neighborhood operations is image segmentation, i.e., classification of pixels that belong in one group. In the simple cases considered thus far, pixels were segmented by identifying clusters in a feature space (e.g., the multidimensional color histogram). Clustered pixels are assumed to belong to the same object class. This approach is appropriate in multispectral classification (e.g., ground-cover classification in remote sensing) or when regions may be distinguished by convolution with a particular kernel (e.g., patterned or textured regions). However, certain important problems may be more readily attacked by other means. For example, suppose that we want to build a machine vision system to read ideal binary text (i.e., black text on white background); individual characters or words must be segmented and identified. The histogram feature-space approach is not appropriate because all text pixels have the same gray level. Pixel classification (i.e., assigning membership of pixels to specific letters or words) must be based on a different criterion such as the shape of a group of adjoining pixels with similar gray levels.

This type of problem leads naturally to the concept of shape-based processing, where groups of pixels (objects) are processed as a whole. In the text processing example, pixels that belong to an individual text character are connected, i.e., adjacent pixels of the same gray level belong to the same character, while pixels with the same

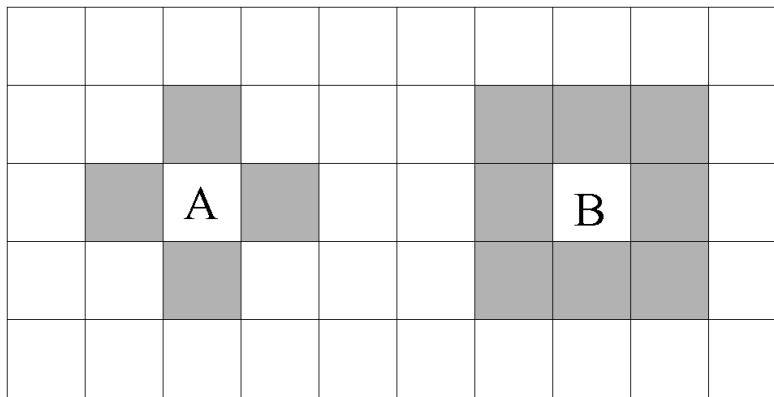
gray level that are not connected belong to different characters. In this discussion, we will ignore problems with overdots (e.g., in lower case i and j), which would be identified as separate characters.

Shape-based processing is also used for symbolic image description. Once pixels have been identified as connected, simpler shape descriptors may be derived by structural operations such as thinning to find the object skeleton, border-following, computation of moments of objects, and morphological transformations. The discussions of these operations will assume that the images are binary; the principles may be applied to gray-level imagery with some difficulty.

17.1 Pixel Connectivity

In a binary image, the component objects are clusters of foreground pixels (white) on a background (black). Two pixels with the same gray level that are adjacent or that are linked through an unbroken chain of foreground pixels are said to be connected and are assumed to belong to the same object. Unconnected foreground pixels must belong to different objects. The concepts of connectivity and adjacency must be well defined before a program may be written to identify and distinguish the foreground objects in the image.

A pixel on a rectangular grid are said to be four-connected or eight-connected when it has the same properties (gray level) as one of its nearest four or eight neighbors.

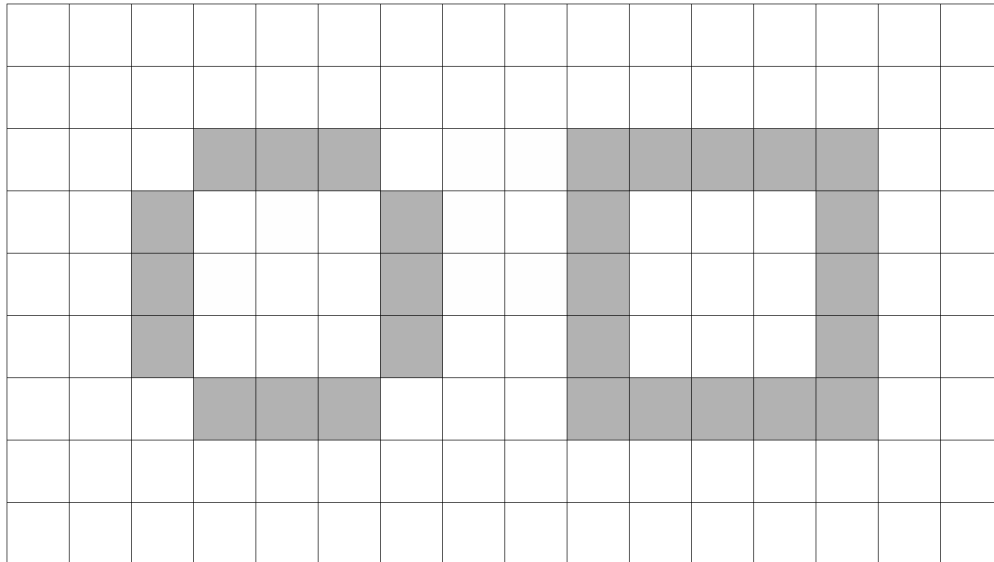


Definition of pixel connectivity: the four dark pixels are “four-connected” to pixel A; the eight dark pixels are “eight-connected” to pixel B.

Pixel A is 4-connected and Pixel B is 8-connected to dark pixels. Some questions may arise about whether some groups of connected pixels are connected to other groups. The picture below is composed of dark foreground objects on a bright background. How many dark objects are there? Regardless of the connectivity of the foreground and background, the dark pixels on the right compose a single foreground object which divides the background into two distinct regions. The number of foreground objects on the left depends on the definition of connectivity. If both foreground and background are considered to be 4-connected, the dark foreground pixels on the left actually compose four distinct objects and the background has two distinct components. Of course, if there are four distinct 4-connected foreground objects, then

all background pixels should belong to a single object. Thus if the foreground is 4-connected, the background should be 8-connected. If the foreground is 8-connected, then there is a single foreground object on the left which divides the background into two components (inside and outside), and thus the background should be 4-connected.

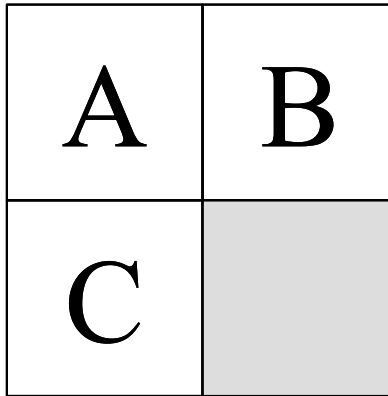
.If both foreground and background are considered to be 8-connected, then the four clusters A, B,C, and D are one foreground object, and E and F comprise a single background object. Perceptually, however, E would be considered to be a hole in the foreground object. If the foreground and background are both considered 4-connected, the clusters A, B, C,and D are separate foreground objects and E and F are separate background objects. The foreground and background are usually assumed to have complementary connectivity, e.g., 8-connected foreground and 4-connected background.



The complementarity of connectivity; if the “foreground” object is 4-connected, the “background” object is assumed to be 8-connected, and vice versa. This assumption ensures that the gray pixels on the left form one object if 8 connected, so that the white pixels “inside” and “outside” the object are not connected.

17.2 Image Labeling

The membership of a pixel is specified by assigning a specific label to all pixels belonging to that component. Algorithms have been developed to automatically assign labels and the labeled clusters may then be trivially segmented by histogram thresholding. One simple technique requires two row-by-row scans of a binary image. When a foreground pixel is encountered during the first scan, its neighbors of each pixel are examined as shown:

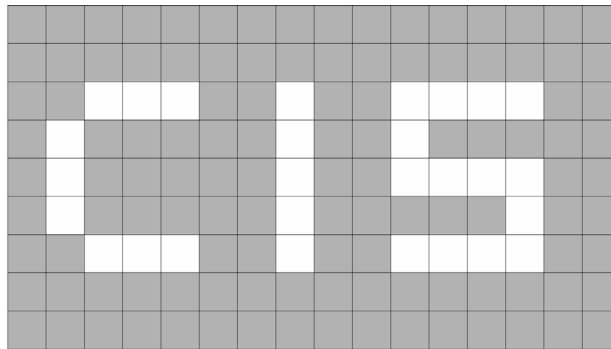


Pixel labeling; the shaded pixel is to be labeled.

The shaded pixel is to be labeled; if the object is assumed to be 4-connected, the already assigned labels of pixels B and C are checked. If either is also foreground, then the shaded pixel must belong to the same foreground object and receives the same label. If neither B nor C is foreground, then the shaded pixel is assigned a new label. If both B and C are foreground, but with different labels, then the shaded pixel connects two previously unconnected regions; it is assigned one of the labels and the equivalence of the two labels is noted in an equivalence table. During the second scan, all equivalent labels are redefined to generate the final labeled image. If the foreground is assumed to be 8-connected, the same procedure is followed, but the label of pixel A is also considered when labels are assigned.

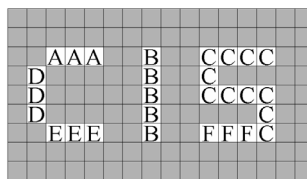
After labeling the components, image segmentation is quite trivial; different labels may be used like different gray levels to segment the image by thresholding.

17.2.1 Example:

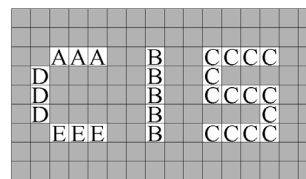


Bitonal Input Image to be Labeled

Four Connectivity

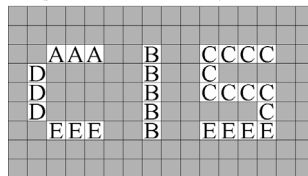


Labels After First Pass

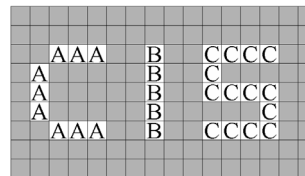


Labels After Second Pass

Eight Connectivity



Labels After First Pass



Labels After Second Pass

The pixels that divide the fore- and background define the borders of objects in the image, and are very useful for pattern recognition and image compression. The border may be defined as a pixel or as the line between the pixels in one of three ways:

It is often useful to label the border pixels of objects in the image, which may be defined in several ways. Recall that the connectivity of the foreground and background must be complementary (4-C foreground \implies 8-C background). The border may be defined as:

1. pixels belonging to the foreground object that are adjacent to background pixels, or
2. pixels belonging to the background that are adjacent to foreground pixels, or
3. the crack between foreground and background.

The borders must define a closed curve, which be specified by its beginning (e.g., upper left corner) and the pixel-by-pixel direction around the object. A common

notation for specifying the border code is to assign a direction code at each pixel using this numbering system:

3	2	1
4	*	0
5	6	7

where the asterisk indicates the edge pixel being examined. This window is placed at the location of the upper-left-most edge pixel in the image of the edge. The number specifies the direction to the next pixel around the edge in a counterclockwise direction. The next edge pixel encountered around this loop is encoded by the number; the code is easily remembered; the m^{th} pixel is in the direction $45^\circ \cdot m$. The border code must be consistent with the connectivity of the region; if the border is defined by 4-connected foreground pixels, then only codes 0, 2, 4, and 6 are allowed and only 2 bits are required to store the code; if 8-connected, all 8 directions are allowed and 3 bits are needed. The border code is often called a chain code. If using definition [3], the four directions of the resulting crack code are specified in an identical fashion and requires 2 bits per segment.

Border coding is a useful tool in image recognition and compression, i.e., it reduces the number of bits required to store/transmit the image. Binary foreground objects may be completely specified by the border codes with fewer bits than storing the locations of each foreground pixel.

17.3 Border Operations

17.3.1 Contraction/erosion and expansion/dilation

Bright objects foreground in binary images may be shrunk by deleting (turning to black) the border pixels of the foreground (as defined in [1] above) or expanded by converting the border pixels of the background to white (foreground). These two operations (erosion and dilation) are the basic components of image morphology. This operations may be applied to a number of imaging problems, including noise removal, image compression, and image specification. This field of image processing has generated recent interest because of its applications to machine vision problems.

Most morphological operations are based on the two basic operations of *erosion* and *dilation*. In the simplest cases, erosion transforms all object pixels that are adjacent to background pixels to background. Dilation is the opposite; background pixels adjacent to object pixels are transformed to object pixels. In the more general case, a shape function $p[x, y]$ is applied to objects in the image and determines which pixels would be converted. The shape function is often called the structuring element or probe, and must specify the origin of coordinates, i.e., the pixel $p[0, 0]$. The erosion of the image is the set of pixels defining those locations of $p[0, 0]$ for which $p[x, y]$ fits wholly within the image object. The dilation of the image is the set of pixels

defined by the locations of the coordinate origin of the structure element where the element touches any object pixels in the image. The generalized erosion and dilation are identical to the simple definitions if the structure element is defined as the single pixel at the origin.

In the remainder of this discussion, we will denote the erosion of an image by the symbol “ \ominus ”, so that the erosion of $f[x, y]$ by the “probe” function $p[x, y]$ is:

$$f[x, y] \ominus p[x, y],$$

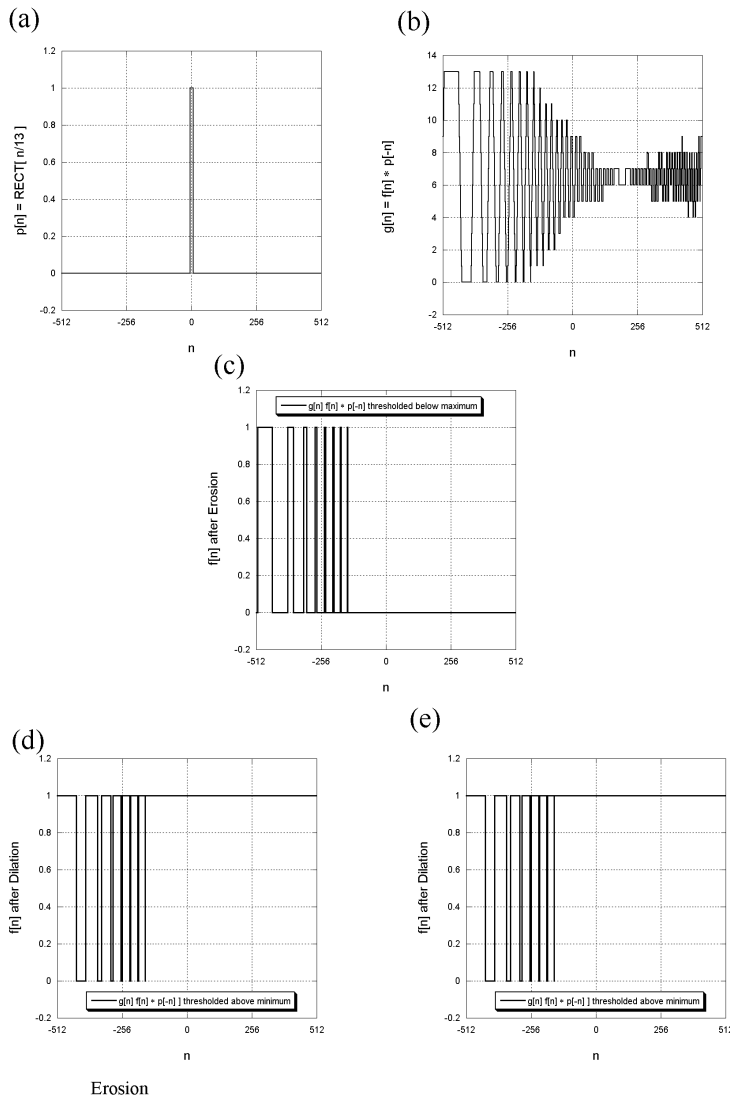
The dilation will be represented by the symbol “ \boxplus ”, so that the dilation of $f[x, y]$ by the “probe” function $p[x, y]$ is:

$$f[x, y] \boxplus p[x, y].$$

In general, the operation of dilation commutes but erosion does not, i.e.,

$$\begin{aligned} f[x, y] \boxplus p[x, y] &= p[x, y] \boxplus f[x, y] \\ f[x, y] \ominus p[x, y] &\neq p[x, y] \ominus f[x, y] \end{aligned}$$

The morphological dilation and erosion operations are shift-invariant, because a translation of the object results in an identical translation of the erosion or dilation. However, they are not linear, i.e., the erosion of the sum of two images is not the sum of the erosions, and thus may not be computed by convolutions, but rather by determining if each pixel satisfies the requisite properties for the operation. For this reason, morphological operations tend to be computationally intensive, to the point where hours of CPU time may be required on general-purpose computers. However, they may be quickly computed in the binary case by evaluating the gray-scale cross-correlation of the binary image and the binary “probe” function (sometimes called the “structuring element”). The gray-scale crosscorrelation is thresholded; the dilation is obtained by thresholding just above the minimum value and erosion by thresholding at a level just below the maximum.



1-D illustration of erosion and dilation of bitonal object via correlation: (a) $f[n]$; (b) “probe” function $p[n]$; (c) $g[n] = f[n] * p[-n] = f[n] \star p[n]$; (d) erosion obtained by thresholding $g[n]$ just below maximum; (e) dilation by thresholding $g[n]$ just above minimum.

17.4 Cascaded Morphological Operations – “Opening” and “Closing”

The general erosion and dilation operators shrink and expand bright foreground objects. They are near-inverses of each other, but if an erosion removes an object consisting of an isolated foreground pixel, a subsequent dilation will not recreate the original object. Because they are not exact inverses, the cascade of erosion followed by dilation will yield a different result than a dilation followed by erosion. The former is the morphological opening of the image because it will open up regions which are

smaller than the structuring element. The opening operation smooths the boundary of bright binary objects while breaking apart any objects connected by thin lines. The cascade of dilation-erosion is the morphological closing of the image and will fill in areas of the foreground which are smaller than the structuring element. The closing operator fills up holes within or spaces between bright objects in the image.

$$\begin{aligned} \textit{Opening} &= (f[x, y] \ominus p[x, y]) \oplus p[x, y] \\ \textit{Closing} &= (f[x, y] \oplus p[x, y]) \ominus p[x, y] \end{aligned}$$

The opening and closing operations may be further cascaded or combined in various ways to perform useful operations. For example, the difference between the set of pixels that belong to the object and the set resulting from erosion with an averaging-like element will yield the boundary pixels of the object:

$$\textit{Boundary} = (f[x, y] \ominus p[x, y]) - f[x, y] \text{ for } p[x, y] = \begin{array}{|c|c|c|} \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array}$$

The so-called “hit-or-miss” operator is obtained by computing the erosion of the image and the dilation of the complementary image, and then finding the intersection of the two:

$$\begin{aligned} \textit{Hit-or-Miss} &= (f[x, y] \ominus p[x, y]) \cap ((1 - f[x, y]) \oplus p[x, y]) \\ &= (f[x, y] \ominus p[x, y]) \cap (\hat{f}[x, y] \oplus p[x, y]) \end{aligned}$$

where \hat{f} denotes the complement image to $f[x, y]$, i.e.,

$$\hat{f}[x, y] = 1 - f[x, y]$$

The concepts of morphological operations are presently being extended to gray-level (non-binary) images.

17.5 Applications of Morphological Operations

17.5.1 Noise Removal

Additive noise in a binary image is often called salt-and-pepper; dark pixels are sprinkled in the white foreground and white pixels in the dark background. Isolated dark pixels may be removed by sequentially expanding and contracting borders of the

foreground; the complementary cycle will remove isolated white pixels.

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

Original Binary Image

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

After Erosion

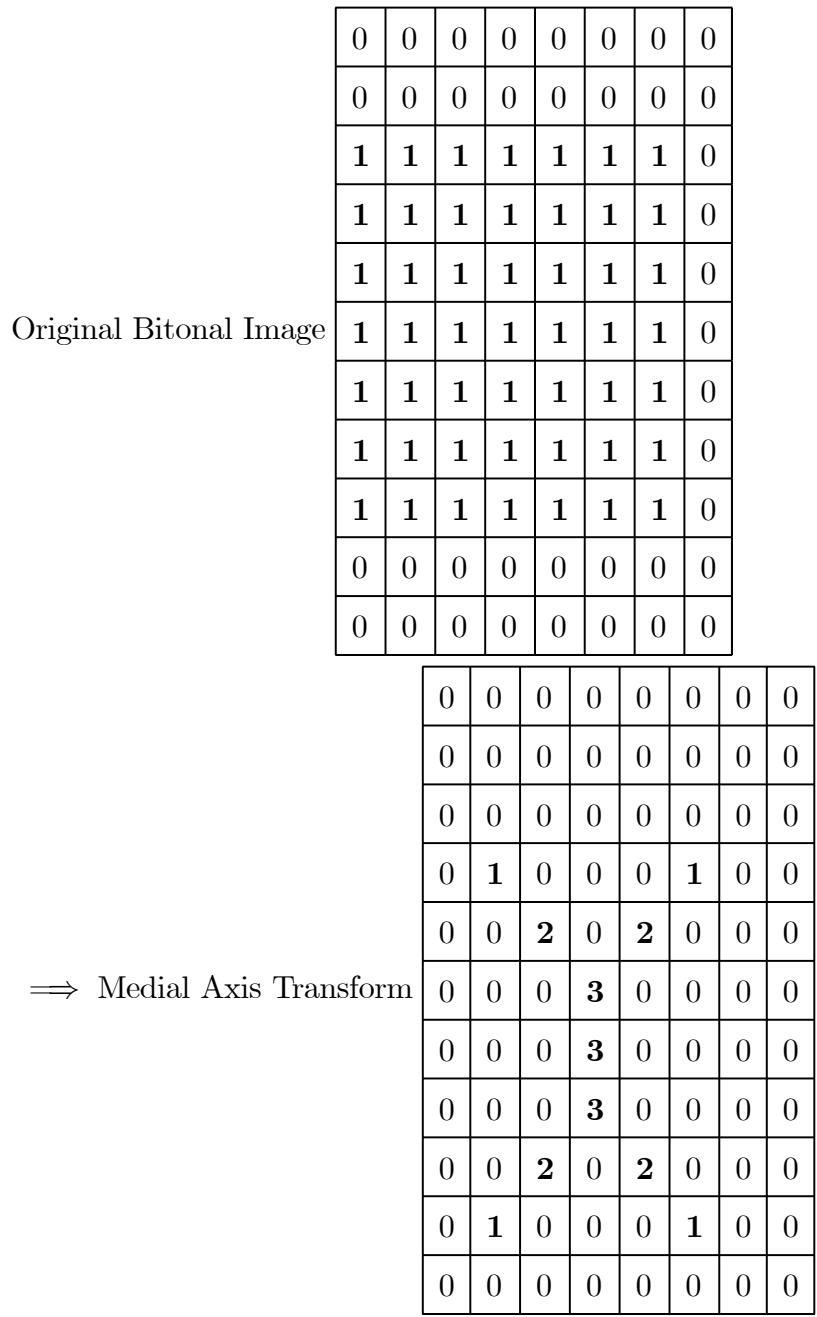
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

After Subsequent Dilatation

17.5.2 Medial Axis Transform

When subjected to repeated erosion operations, the objects in the image will gradually shrink to the point where any further erosions will cut the object into discontinuous parts. The image obtained by assigning the number of erosions required to reach that point to each remaining pixel is the medial axis of the object, and the ensemble of those pixels is the skeleton of the object. These images are very useful for identifying components of the image and/or their orientation, i.e., image representation, and to reduce the number of data bits necessary to specify the representation, i.e., image

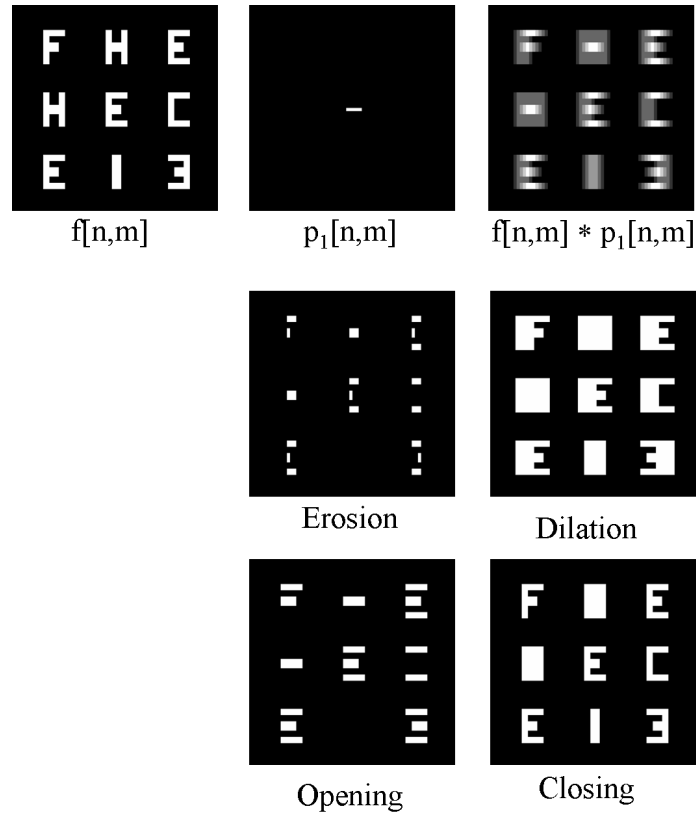
compression.



Shrinking and expansion operations are special cases of a more general field of image processing operations known as morphological operations, which specify the pixels that are inscribed by or circumscribed by other user-defined shapes (probes or structure elements).

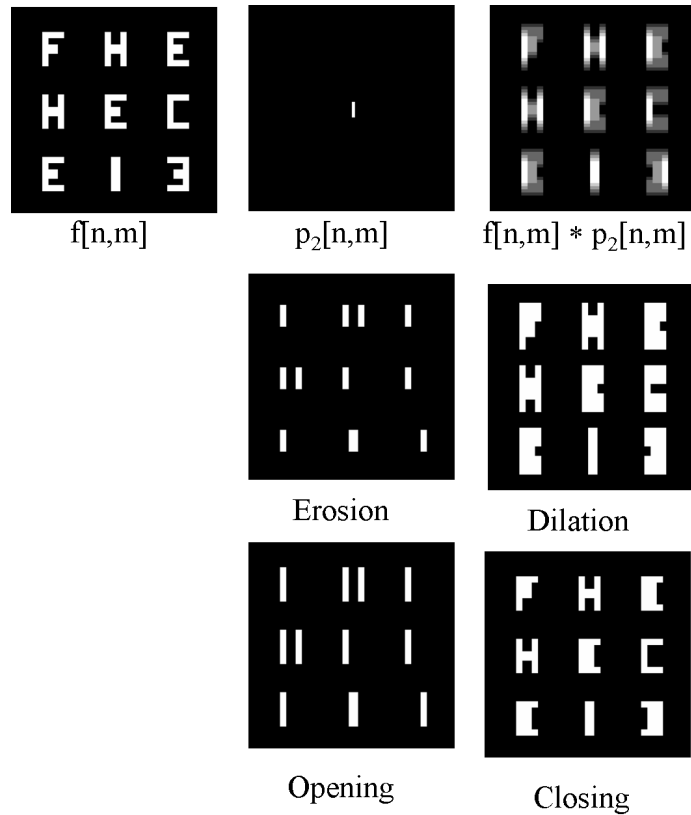
17.6 Binary Morphological Operations

17.6.1 Horizontal Structuring Element



Bitonal morphological erosion, dilation, opening (dilation of erosion), and closing (erosion of dilation) for horizontal “structuring element” (probe function). Note that the horizontal “spaces” in the characters are filled in the closing.

17.6.2 Vertical Structuring Element



Bitonal morphological erosion, dilation, opening (dilation of erosion), and closing (erosion of dilation) for vertical “structuring element” (probe function). Note that the vertical “spaces” in the characters are filled in the closing.

Thin horizontal structures are removed by opening with vertical structure element. Vertical holes in object are filled by closing with vertical structure element

