Binary images, as discussed in the preceding chapter, consist of groups of pixels selected on the basis of some property. The selection may be performed by thresholding brightness values, perhaps using several grey scale images containing different color bands, or processed to extract texture or other information. The goal of binarization is to separate features from background, so that counting, measurement, or matching operations can be performed.

As shown by the examples in Chapter 6, however, the result of the segmentation operation is rarely perfect. For images of realistic complexity, even the most elaborate segmentation routines misclassify some pixels as foreground or background. These may either be pixels along the boundaries of regions or patches of noise within regions. The major tools for working with binary images fit broadly into two groups: Boolean operations, for combining images, and morphological operations which modify individual pixels within images.

**Boolean operations**

In the section on thresholding color images, in Chapter 6, a Boolean operation was introduced to combine the data from individual color plane images. Setting thresholds on brightness values in each of the RGB (red, green, blue) planes allows pixels to be selected that fall into these ranges. This technique produces three binary images, which can then be combined with a logical “AND” operation. The procedure examines the three images pixel by pixel, keeping pixels for the selected regions if they are turned on in all three images.

The color thresholding example is an example of a situation in which pixel brightness values at the same location in several different images (the color channels) must be compared and combined. In some situations it is useful to compare the location and brightness value of pixels in two images. Figure 1 shows an example. Two x-ray maps of the same area on a mineral sample show the intensity distributions, and hence represent the concentration distributions for aluminum and silicon. A colocalization plot uses the pixel brightness values for each location in both images as coordinates and increments the plot. Regions in the resulting plot that have many counts represent combinations of elemental concentrations in the original sample. In the example plot, there are four phases present based on Si/Al combinations, and these can be observed in the original images. Colocalization is also used for biological samples prepared with multiple stains.
When a colocalization plot shows specific combinations of intensity values that are shared at the same location, each image can be thresholded and the two binary images combined with an AND to produce an image of the selected regions.

**Note:** The terminology used here will be that of “ON” (pixels that are part of the selected foreground features) and “OFF” (the remaining pixels, which are part of the background). There is no universal standard for whether the selected pixels are displayed as white, black, or some other color. In many cases, systems that portray the selected regions as white on a black background on the display screen may reverse this and print hardcopy of the same image with black features on a white background. This reversal apparently arises from the fact that in each case, the selection of foreground pixels is associated with some positive action in the display (turning on the electron beam) or printout (depositing ink on the paper). It seems to cause most users little difficulty, provided that something is known about the image. Many of the images used here are not common objects and some are made-up examples; therefore, it is important to be consistent in defining the foreground pixels (those of interest) in each case. The convention used here is that ON pixels (features) are shown as black while OFF pixels (background) are white.

Returning to our desire to combine the information from several image planes, the AND operation requires that a pixel at location \(i, j\) be ON in each individual plane to show up in the result. Pixels having the correct amount of blue but not of red will be omitted, and vice versa. As noted previously, this marks out a rectangle in two dimensions, or a rectangular prism in higher dimensions, for the pixel values to be included. More complicated combinations of color values can be described by delineating an irregular region in \(n\) dimensions for pixel selection. The advantage of simply ANDing discrete ranges is that it can be performed very efficiently and quickly using binary images.

*Figure 1. Colocalization: (a, b) x-ray maps showing the intensity distribution for Al and Si in a mineral. (c) Colocalization plot.*
Other Boolean logical rules can be employed to combine binary images. The four possibilities are AND, OR, Ex-OR (Exclusive OR) and NOT. **Figure 2** illustrates each of these basic operations. **Figure 3** shows a few of the possible combinations. All are performed pixel-by-pixel. The illustrations are based on combining two images at a time, because any logical rule involving more than two images can be broken down to a series of steps using just two at a time. The illustrations in the figures are identical to the Venn diagrams used in logic.

As described previously, AND requires that pixels be ON in both of the original images in order to be ON in the result. Pixels that are ON in only one or the other original image are OFF in the result. The OR operator turns a pixel ON in the result if it is ON in either of the original images. In the example shown in **Figure 32 of Chapter 6**, complementary directions and thus grey-scale values, result from the Sobel direction operator as it encounters opposite sides of each striation. Thresholding each direction separately would require an OR to combine them to show the correct regions.

Ex-OR turns a pixel ON in the result if it is ON in either of the original images, but not if it is ON in both. That means that combining (with an OR) the results of ANDing together two images with those from Ex-ORing them produces the same result as an OR in the first place. There are, in fact, many ways to arrange different combinations of the four Boolean operators to produce identical results.

---

**Figure 2. Simple Boolean operations:**

(a, b) two binary images;
(c) A OR B;
(d) A AND B;
(e) A exOR B;
(f) NOT A.

© 2002 by CRC Press LLC
AND, OR, and Ex-OR require two original images and produce a single image as a result. NOT requires only a single image. It simply reverses each pixel, turning pixels that were ON to OFF and vice versa. Some systems implement NOT by swapping black and white values for each pixel. As long as we are dealing with pixel-level detail, this works correctly. Later, when feature-level combinations are described, the difference between an eight-connected feature and its four-connected background (discussed in Chapter 6) will have to be taken into account.

Given two binary images \( A \) and \( B \), the combination \((\text{NOT } A) \text{ AND } B\) will produce an image containing pixels that lie within \( B \) but outside \( A \). This is quite different from \(\text{NOT} (A \text{ AND } B)\), which selects pixels that are not ON in both \( A \) and \( B \). It is also different from \( A \text{ AND } (\text{NOT } B)\), as shown in Figure 3. The order of operators is important and the liberal use of parentheses to clarify the order and scope of operations is crucial. Actually, the four operations discussed previously are redundant. Three would be enough to produce all of the same results. Consequently, some systems may omit one of them (usually Ex-OR). For simplicity, however, all four will be used in the examples that follow.

---

**Figure 3. Combined Boolean operations:**

(a) \((\text{NOT } A) \text{ AND } B\);
(b) \( A \text{ AND } (\text{NOT } B)\);
(c) \((\text{NOT } A) \text{ AND } (\text{NOT } B)\);
(d) \((\text{NOT } A \text{ AND } B)\);
(e) \((\text{NOT } A) \text{ OR } B\);
(f) \( A \text{ OR } (\text{NOT } B)\)

---

AND, OR, and Ex-OR require two original images and produce a single image as a result. NOT requires only a single image. It simply reverses each pixel, turning pixels that were ON to OFF and vice versa. Some systems implement NOT by swapping black and white values for each pixel. As long as we are dealing with pixel-level detail, this works correctly. Later, when feature-level combinations are described, the difference between an eight-connected feature and its four-connected background (discussed in Chapter 6) will have to be taken into account.

Given two binary images \( A \) and \( B \), the combination \((\text{NOT } A) \text{ AND } B\) will produce an image containing pixels that lie within \( B \) but outside \( A \). This is quite different from \(\text{NOT} (A \text{ AND } B)\), which selects pixels that are not ON in both \( A \) and \( B \). It is also different from \( A \text{ AND } (\text{NOT } B)\), as shown in Figure 3. The order of operators is important and the liberal use of parentheses to clarify the order and scope of operations is crucial. Actually, the four operations discussed previously are redundant. Three would be enough to produce all of the same results. Consequently, some systems may omit one of them (usually Ex-OR). For simplicity, however, all four will be used in the examples that follow.
Combining Boolean operations

When multiple criteria are available for selecting the pixels to be kept as foreground, they may be combined using any of these Boolean combinations. The most common situations are multiband images, such as produced by a satellite or a scanning electron microscope (SEM). In the case of the SEM, an x-ray detector is often used to create an image (called an x-ray dot map) showing the spatial distribution of a selected element. These images may be quite noisy (Chapter 3) and difficult to threshold (Chapter 6); however, by suitable long-term integration or spatial smoothing, they can lead to useful binary images that indicate locations where the concentration of the element is above some user-selected level.

This selection is usually performed by comparing the measured x-ray intensity to some arbitrary threshold, since there is a finite level of background signal resulting from the process of slowing down the electrons in the sample. The physical background of this phenomenon is not important here. The very poor statistical characteristics of the dot map (hence the name) make it difficult to directly specify a concentration level as a threshold. The x-ray intensity in one part of the image may vary from another region for the following reasons:

1. A change in that element’s concentration
2. A change in another element that selectively absorbs or fluoresces the first element’s radiation
3. A change in specimen density or surface orientation. Comparison of one specimen to another is further hampered by the difficulty in exactly reproducing instrument conditions. These effects all complicate the relationship between elemental concentration and recorded intensity.

Furthermore, the very poor statistics of the images (due to the extremely low efficiency for producing x-rays with an electron beam and the low beam intensity required for good spatial resolution in SEM images) mean that these images often require processing, either as grey-scale images (e.g., smoothing) or after binarization (using the morphological tools discussed below). For our present purpose, we will assume that binary images showing the spatial distribution of some meaningful concentration level of several elements can be obtained.

As shown in Figure 4, the SEM also produces more conventional images using secondary or backscattered electrons. These have superior spatial resolution and better feature shape definition, but with less elemental specificity. The binary images from these sources can be combined with the x-ray or elemental information.

**Figure 5** shows one example: The x-ray maps for iron (Fe) and silicon (Si) were obtained by smoothing and thresholding the grey scale image. Notice that in the grey scale images, there is a just-detectable difference in the intensity level of the Fe x-rays in two different areas. This is too small a difference for reliable thresholding. Even the larger differences in Si intensity are difficult to separate, however, Boolean logic easily combines the images to produce an image of the region containing Fe but not Si.

**Figure 6** shows another example from the same data. The regions containing silver (Ag) are generally bright in the backscattered electron image, but some other areas are also bright. On the other hand, the Ag x-ray map does not have precise region boundaries because of the poor statistics. Combining the two binary images with an AND produces the desired regions. More complicated sequences of Boolean logical operations can easily be imagined (**Figure 7**).

It is straightforward to imagine a complex specimen containing many elements. Paint pigment particles with a diverse range of compositions provide one example. In order to count or measure a particular class of particles (pigments, as opposed to brighteners or extenders), it might be necessary to specify those containing iron or chromium or aluminum, but not titanium or sulfur. This would be written as...
The resulting image might then be combined with a higher-resolution binary produced by thresholding a secondary or backscattered electron image to delineate particle boundaries. Performing these operations can be cumbersome but is not difficult.

Most of the examples shown in earlier chapters that used multiple image planes (e.g., different colors or elements) or different processing operations (e.g., combining brightness and texture) use a Boolean AND to combine the separately thresholded binary images. The AND requires that the pixels meet all of the criteria in order to be kept. There are some cases in which the Boolean OR is more appropriate. One is illustrated in Chapter 4, Figure 81. This is an image of sand grains in a sandstone, viewed through polarizers. Each rotation of the analyzer causes different grains to be-

\[(\text{Fe OR Cr OR Al}) \text{ AND NOT (Ti OR S)}\]  

© 2002 by CRC Press LLC
come bright or colored. In the earlier chapter, it was shown that keeping the brightest pixel value at each location as the analyzer is rotated gives an image that shows all of the grains.

**Figure 8** shows an alternative approach to the same problem. Each individual image is thresholded to select those grains that are bright for that particular analyzer rotation angle. Then, all of the binary images are combined using a Boolean OR. The resulting combination delineates most of the grains, although the result is not as good as the grey-level operation for the same number of analyzer rotations.

**Masks**

The previous description of using Boolean logic to combine images makes the assumption that both images are binary (that is, black and white). It is also possible to use a binary image as a mask to modify a grey-scale image. This is most often done to blank out (i.e., set to background) some portion of the grey-scale image, either to create a display in which only the regions of interest are visible or to select regions whose brightness, density, and so forth are to be measured. **Figure 9** shows an example (a protein separation gel) in which the dark spots are isolated by thresholding, and then the thresholded binary image is applied as a mask to produce separated features for measurement that retain the original density values.

This operation can be performed in several physical ways. The binary mask can be used in an overlay, or alpha channel, in the display hardware to prevent pixels from being displayed. It is also possible to use the mask to modify the stored image. This can done by multiplying the grey-scale image by the binary image, with the convention that the binary image values are 0 (OFF) or 1 (ON) at each pixel. In some systems this result is implemented by combining the grey-scale and binary images to keep whichever value is darker or brighter. For instance, if the mask is white for background and black for foreground pixels then the brighter pixel values at each location will erase all background pixels and keep the grey value for the foreground pixels.

This capability has been used in earlier chapters to display the results of various processing and thresholding operations. It is easier to judge the performance of thresholding by viewing selected
pixels with the original grey-scale information, rather than just looking at the binary image. This format can be seen in the examples of texture operators in Chapter 4, for instance, as well as in Chapter 6 on Thresholding. It is also useful to use a mask obtained by thresholding one version of an image to view another version. Figure 10 shows an example, in which values represent the orientation angle (from the Sobel derivative) of grain boundaries in the aluminium alloy are masked by thresholding the magnitude of the gradient to isolate only the boundaries.

Figure 9. Preserving feature intensity values: (a) original 2D gel; (b) thresholded spots; (c) masked image in which pixels within the spots retain their original brightness values.
Another use of masking and Boolean image combination is shown in Figure 11. An essentially cosmetic application, it is still useful and widely employed. A label superimposed on an image using either black or white may be difficult to read if the image contains a full range of brightness values. In this example, the label is used to create a mask that is one pixel larger in all directions, using dilation (discussed later in this chapter). This mask is then used to erase the pixels in the grey-scale image to white before writing in the label in black (or vice versa). The result maintains legibility for the label while obscuring a minimum amount of the image.

Finally, a binary image mask can be used to combine portions of two (or more) grey-scale images. This is shown in Figure 12. The composite image represents, in a very simple way, the kind of image overlays and combinations common in printing, advertising, and commercial graphic arts. Although it is rarely suitable for scientific applications, this example will perhaps serve to remind us that modifying images to create things that are not real has become relatively easy with modern computer technology. This justifies a certain skepticism in examining images, which were once considered iron-clad evidence of the truth. Detecting forgeries in digital images can be quite difficult if constructed with enough skill (Russ, 2001a).

From pixels to features

The Boolean operations described above deal with individual pixels in the image. For some purposes it is necessary to identify the pixels forming part of a connected whole. As discussed in Chapter 6, it is possible to adopt a convention for touching that is either eight-connected or...
four-connected for the pixels in a single feature (sometimes referred to as a blob to indicate that no interpretation of the connected group of pixels has been inferred as representing anything specific in the image). Whichever convention is adopted, grouping pixels into features is an important step (Levialdi, 1992; Ritter, 1996).

It is possible to imagine starting with one pixel (any ON pixel, selected at random) and checking its four- or eight-neighbor positions, labeling each pixel that is ON as part of the same feature, and then iteratively repeating the operation until no neighbors remain. Then a new unlabeled pixel would be chosen and the operation repeated, continuing until every ON pixel in the image was labeled as part of some feature. The usual way of proceeding with this deeply recursive operation is to create a stack to place pixel locations as they are found to be neighbors of already labeled pixels. Pixels are removed from the stack as their neighbors are examined. The process ends when the stack is empty.

It is more efficient to deal with pixels in groups. If the image has already been run-length or chord encoded, as discussed in Chapter 6, then all of the pixels within the chord are known to touch, touching any of them is equivalent to touching all, and the only candidates for touching are those on adjacent lines. This fact makes possible a very straightforward labeling algorithm that passes one time through the image. Each chord’s end points are compared to those of chords in the preceding line; if they touch or overlap (based on a simple comparison of values), the label from the preceding line is attached to this chord. If not, then a new label is used.

If a chord touches two chords in the previous line that had different labels, then the two labels are identified with each other (this handles the bottom of a letter “U” for example). All of the occurrences of one label can be changed to the other, either immediately or later. When the pass through the image or the list of chords is complete, all of the chords, and therefore all of the pixels, are identified and the total number of labels (and therefore features) is known. Figure 13 shows this logic in the form of a flow chart.

For boundary representation (including the special case of chain code), the analysis is partially complete, since the boundary already represents a closed path around a feature. If features contained no holes and no feature could ever be surrounded by another, this would provide complete information. Unfortunately, this is not always the case. It is usually necessary to reconstruct the pixel array to identify pixels with feature labels (Kim et al., 1988).

In any case, once the individual features have been labeled, several additional Boolean operations are possible. One is to find and fill holes within features. Any pixel that is part of a hole is defined as OFF (i.e., part of the background) and is surrounded by ON pixels. For boundary representation, that means the pixel is within a boundary. For pixel representation, it means it is not connected to other pixels that eventually form a path to the edge of the field of view.

Recalling that the convention for touching (eight- or four-connectedness) must be different for the background than for the foreground, we can identify holes most easily by inverting the image.
(replacing white with black and vice versa) and labeling the resulting pixels as though they were features, as shown step-by-step in Figure 14. Features in this inverted image that do not touch any side of the field of view are the original holes. If the pixels are added back to the original image (using a Boolean OR), the result is to fill any internal holes in the original features.

One very simple example of the application of this technique is shown in Figure 15. In this image of spherical particles, the center of each feature has a brightness very close to that of the substrate due to the lighting. Thresholding the brightness values gives a good delineation of the outer boundary of the particles, but the centers have holes. Filling them as described produces a corrected representation of the particles, which can be measured.

This problem is not restricted to convex surfaces nor to the SEM. Figure 16 shows a light microscope image of spherical pores in an enamel coating. The light spots in the center of many of the pores vary in brightness, depending on the depth of the pore. They must be corrected by filling the features in a thresholded binary image.

Figure 17 shows a more complicated situation requiring several operations. The SEM image shows the boundaries of the spores clearly to a human viewer, but they cannot be directly revealed by thresholding because the shades of grey are also present in the substrate. Applying an edge-finding algorithm (in this example, a Frei and Chen operator) delineates the boundaries, and it is then possible to threshold them to obtain feature outlines, as shown. These must be filled using the method described above. Further operations are then needed before measurement: erosion, to remove the other thresholded pixels in the image, and watershed segmentation, to separate the touching objects. Both are described later in this chapter.

The use of edge-enhancement routines, discussed in Chapter 4, is often followed by thresholding the outlines of features and then filling in the interior holes. In some situations, several different methods must be used and the information combined. Figure 18 shows a very difficult example,
Figure 14. Light microscope image of red blood cells:
(a) original;
(b) thresholded, which shows the thicker outer edges of the blood cells but not the thinner central regions;
(c) image b inverted;
(d) removing the edge-touching background from image c;
(e) combining the features in image d with those in image b using a Boolean OR;
(f) removing small features (dirt), edge-touching features (which cannot be measured), and separating touching features in e.

Figure 15. Image of buckshot with near-vertical incident illumination:
(a) original grey-scale image;
(b) brightness thresholded after leveling illumination;
(c) internal holes filled and small regions (noise) in background removed by erosion.
Chapter 7: Processing Binary Images

Figure 17. Segmentation of an image using multiple steps:
(a) original SEM image of spores on a glass slide;
(b) application of a Frei and Chen edge operator to image a;
(c) thresholding of image b;
(d) filling of holes in the binary image of the edges;
(e) erosion to remove the extraneous pixels, in image d;
(f) watershed segmentation to separate touching features in image e.

Figure 16. Light microscope image of a polished section through an enamel coating on steel (courtesy V. Benes, Research Institute for Metals, Panenske Brezany, Czechoslovakia) shows bright spots of reflected light within many pores (depending on their depth).

Figure 18. Section through an epoxy resin containing bubbles. To delineate the bubbles for measurement, the bright, dark, and outlined pores must be processed in different ways and the results combined with a Boolean OR.
bubbles in epoxy resin. Some of the concave pores are dark, some light, and some bounded by a bright edge. Processing and thresholding each type of pore and then combining the results with a Boolean OR produces an image delineating all of the pores.

The Boolean AND operation is also widely used to apply measurement templates to images. For instance, consider the measurement of coating thickness on a wire or plate viewed in cross section. In the examples of Figures 19 and 20, the layer can be readily thresholded, but it is not uniform in thickness. In order to obtain a series of discrete thickness values for statistical interpretation, it is easy to AND the binary image of the coating with a template or grid consisting of lines normal to the coating. These lines can be easily measured. In Figure 19, for the case of a coating on a flat surface, the lines are vertical. For a cylindrical structure such as a similar coating on a wire, or the wall thickness of a tube, a set of radial lines can be used.

In the example of Figure 20, the vein is approximately circular in cross section and the lines do not perpendicularly intersect the wall, introducing a cosine error in the measurement which may or may not be acceptable. That the cross section is not round may indicate that the section plane is not perpendicular to the vein axis, which would introduce another error in the measurement.

The measurement of three-dimensional structures from two-dimensional section images is dealt with by stereological techniques discussed in more detail in Chapter 8.

Figure 21 illustrates a situation in which the length of the lines give the layer thickness indirectly, requiring stereological interpretation. The image shows a section plane through coated particles.

Figure 19. Measurement of layer thickness:
(a) paint layer viewed in cross section;
(b) thresholded layer, with superimposed grid of vertical lines; (c) AND of lines with layer producing line segments for measurement.

Figure 20. Measurement of layer thickness:
(a) cross section of vein in tissue;
(b) thresholded wall with superimposed grid of radial lines;
(c) AND of lines with layer producing line segments for measurement (note the cosine errors introduced by non-perpendicular alignment of grid lines to wall).
embedded in a metallographic mount and polished. The section plane does not go through the center of the particles, so the coating appears thicker than the actual three-dimensional thickness. This is handled by placing a grid of random lines in the template. The distribution of line intercept lengths is related to that of the coating thickness in the normal direction. The average of the inverse intercept lengths is two-thirds the inverse of the true coating thickness, so this value can be obtained even if the image does not include a perpendicular cross section through the coating.

Selection of an appropriate grid is crucial to the success of measurements. Chapter 8 discusses the principal stereological measurements made on microstructures, to determine the volumes, surface areas, lengths, and topological properties of the components present. Many of these procedures are performed by counting the intersections made by various grids with the structures of interest. The grids typically consist of arrays of points or lines, and the lines used include regular and random grids of straight lines, circular arcs and cycloids, depending on the type of measurement desired, and the procedure used to select and prepare the specimens being imaged. In all cases, if the image can be thresholded successfully to delineate the structure, then a Boolean AND with the appropriate grid produces a result that can be measured. In some situations this requires measuring the lengths of lines, and in others simply counting the number of intersections produced.

Even for very complex or subtle images for which automatic processing and thresholding cannot delineate the structures of interest, the superimposition of grids as a mask may be important. Many stereological procedures that require only counting of intersections of various types of grids with features of interest are extremely efficient and capable of providing unbiased estimates of valuable structural parameters. Combining image capture and processing to enhance the visibility of structures with overlays of the appropriate grids — arrays of points or lines, the latter including straight lines, circles and cycloids — allows the human user to recognize the important features and intersections (Russ, 1995a). The counting may be performed manually or the computer may also assist by tallying mouse-clicks or counting marks that the user places on the image. The combination of human recognition with computer assistance provides efficient solutions to many image analysis problems.

**Boolean logic with features**

Having identified or labeled the pixel groupings as features, it is possible to carry out Boolean logic at the feature level, rather than at the pixel level. Figure 22 shows the principle of a feature-based AND. Instead of simply keeping the pixels that are common to the two images, entire features are kept if any part of them touches. This preserves the entire feature, so that it can be correctly counted or measured if it is selected by the second image.
Feature-AND requires a feature labeling operation to be performed on at least one of the images to be combined. Touching pixels in one image are identified as features as described previously. Then each pixel that is ON in one of those features is checked against the second image. If any of the pixels in the feature match an ON pixel in the second image, the entire feature in the first image is copied to the result. This is not the only possible implementation. It would be equally possible to check each pixel in the second image against the first, but that is less efficient. The method outlined limits the comparison to those pixels which are on, and halts the test for each feature whenever any pixel within it is matched.

Notice that unlike the more common pixel based AND, this statement does not commute; this means that (A Feature-AND B) does not produce the same result as (B Feature-AND A), as illustrated in Figure 23. The use of NOT with Feature-AND is straightforwardly implemented, for instance by carrying out the same procedure and erasing each feature in the first image that is matched by any pixel in the second. However, there is no need for a Feature-OR statement, since this would produce the identical result as the conventional pixel-based OR.

One use for the Feature-AND capability is to use markers within features to select them. For example, these might be cells containing a stained organelle or fibers in a composite containing a characteristic core. In any case, two binary images are produced by thresholding. In one image, the entire features are delineated, and in the second the markers are present. Applying the Feature-AND logic then selects all of the features which contain a marker.

This use of markers to select features is a particularly valuable capability in an image analysis system. Figure 24 illustrates one way that it can be used. The original image has several red features, only some of which contain darker regions within. If one copy of the image is thresholded...
Figure 23. Feature-based Boolean logic used to combine two test images (a, b) (c) a Feature-AND b (d) b Feature-AND a

Figure 24. Example of feature selection using markers. The red features and the dark spots in the original image a are thresholded to produce separate binary images b and c. The dark spots are used as markers to select those red features which contain dark markers d.
The Image Processing Handbook

for dark spots and a second copy is thresholded for red features, then the first can be used as a set of markers to select the features of interest. A Feature-AND can be used to perform that operation.

In real applications the marker image that selects the features of interest may be obtained by separate thresholding, by processing, or by using another plane in a multiplane image. Figure 25 shows an example. Only those cells containing green-stained nuclei are selected, but they are selected in their entirety so that they can be measured. A related procedure that uses the Feature-AND capability is the use of the nucleator (Gundersen et al., 1988), a stereological tool that counts cells in thin sections of tissue according to the presence of a unique marker within the cell such as the nucleus.

At a very different scale, the method might be used with aerial photographs to select and measure all building lots that contain any buildings, or fields that contain animals. The technique can also be used with x-ray images to select particles in SEM images, for instance, if the x-ray signal comes only from the portion of the particle which is visible to the x-ray detector. The entire particle image can be preserved if any part of it generates an identifying x-ray signal.

Feature-AND is also useful for isolating features that are partially within some region, or adjacent to it. For example, in Figure 26 the colonies contain bacterial cells that are to be counted and measured, but some of them extend beyond the boundaries of the colony. The logic of Feature-AND allows them to be assigned to the appropriate colony and counted, and not to be counted more than once if they exit and re-enter the region. And in Figure 27 the outline of a region has been generated (using dilation as discussed below) and used as a marker to select features that are adjacent to the substrate, so that they can be measured.

Figure 25. Application of Feature-AND:
(a) original image of cells with stained nuclei;
(b) nuclei thresholded based on green intensity;
(c) cells thresholded based on red intensity;
(d) Feature-AND result showing only those cells containing green-stained nuclei;
(e) outlines of features from image d superimposed on original.
Chapter 7: Processing Binary Images

Figure 26. Colony counting:
(a) image representing colonies of bacterial cells, some of which extend beyond the stained area;
(b) counted results showing the number of cells in each colony.

Figure 27. Identifying adjacent features:
(a) image showing cross-section of a blue substrate with some orange features touching it;
(b) thresholded substrate;
(c) pixels immediately adjacent to the substrate, produced by dilating and Ex-ORing;
(d) orange features;
(e) Feature-AND of image c with image d;
(f) features identified in image e superimposed on the original.
Selecting features by location
In a generalization of the method for identification of touching features shown in Figure 27, Feature-AND is also useful when applied in conjunction with images that map regions according to distance. We will see below that dilating a line, such as a grain boundary or cell wall, can produce a broad line of selected thickness. Using this line to select features that touch it selects those features which, regardless of size or shape, come within that distance of the original boundary. Counting these for different thickness lines provides a way to classify or count features as a function of distance from irregular boundaries. Figure 28 shows an example and Figure 29 shows an actual image of grain-boundary depletion.

Figure 30 shows a similar situation in which a pixel-based AND is appropriate. The image shows a metallurgical cross-section of a plasma-sprayed coating applied to a turbine blade. There is always a certain amount of oxide present in such coatings, which in general causes no difficulties; but if the oxide, which is a readily identifiable shade of grey, is preferentially situated at the coating-substrate interface, it can produce a region of weakness that may fracture and cause spalling of the coating. Thresholding the image to select the oxide, then ANDing this with the line representing the interface (itself obtained by thresholding the metal substrate phase, dilating, and EX-ORing to get the crust; discussed more extensively later in this chapter) gives a direct measurement of the contaminated fraction of the interface.

An aperture or mask image can be used to restrict the analysis of a second image to only those areas within the aperture. Consider counting spots on a leaf: either the spots are due to an aerial spraying operation to assess uniformity of coverage, or perhaps they are spots of fungus or mold to assess the extent of disease. The acquired image is normally rectangular, but the leaf is not. There may well be regions outside the leaf that are similar in brightness to the spots. Creating a binary image of the leaf, then Feature-ANDing it with the total image selects those spots lying on the

Figure 28. Comparison of pixel- and feature-AND:
(a) diagram of an image containing features and a boundary;
(b) the boundary line, made thicker by dilation;
(c) pixel-based AND of images b and a (incomplete features and one divided into two parts);
(d) feature-AND of images b and a (all features within a specified distance of the boundary).
leaf itself. If the spots are small enough, this could be done as a pixel-based AND; however, if the spots can touch the edge of the leaf, the feature-based operation is safer because systems may not count or measure edge-touching features (as discussed in Chapter 9). Counting can then provide the desired information, normally expressed as number-per-unit-area where the area of the leaf forms the denominator. This procedure is similar to the colony-counting problem in Figure 26.

Figure 29. Light microscope image of polished section through a steel used at high temperature in boiler tubes. Notice the depletion of carbides (black dots) in the region near grain boundaries. This effect can be measured using procedures described in the text.

Figure 30. Isolating the oxide in a coating/substrate boundary:
(a) original grey-scale microscope image of a cross section of the plasma-sprayed coating on steel;
(b) thresholding of the metal in the coating and the substrate;
(c) applying erosion and dilation (discussed later in this chapter) to image b to fill holes and remove small features;
(d) boundary line produced by dilating image c and Ex-ORing with the original;
(e) thresholding the oxide in the coating, including that lying in the interface;
(f) a pixel-based AND of image d with image b, showing just the fraction of the interface which is occupied by oxide.
Figure 31 shows another situation, in which two different thresholding operations and a logical combination are used to select features of interest. The specimen is a polished cross section of an enamel coating on steel. The two distinct layers are different colored enamels containing different size distributions of spherical pores. Thresholding the darker layer includes several of the pores in the lighter layer, which have the same range of brightness values, but the layer can be selected by discarding features that are small or do not touch both edges of the field. This image then forms a mask that can be used to select only the pores in the layer of interest. Similar logic can be employed to select the pores in the light layer. Pores along the interface will generally be included in both sets, unless additional feature-based logic is employed.

A similar application allows identifying grains in ores that are contained within other minerals, for instance, to determine the fraction that are “locked” within a harder matrix that cannot easily be recovered by mechanical or chemical treatment, as opposed to those that are not so enclosed and are easily liberated from the matrix.

Figure 31. Selecting pores in one layer of enamel on steel:
(a) original light microscope image (courtesy V. Benes, Research Inst. for Metals, Panenské Brezany, Czechoslovakia);
(b) image a thresholded to select dark pixels;
(c) discarding all features from image b that do not extend from one side to the other leaves just the layer of interest;
(d) thresholding the original image to select only dark pores produces a binary image containing more pores than those in the layer;
(e) combining images b and d with a Boolean Feature-AND leaves only the pores within the dark layer.
A rather different use of feature-based Boolean logic implements the disector, a stereological tool discussed in Chapter 8 that gives an unbiased and direct measure of the number of features per unit volume (Sterio, 1984). It requires matching features in two images that represent parallel planes separated by a distance T. The features represent the intersection of three-dimensional objects with those planes. Those objects which intersect both planes are ignored, but those which intersect only one plane or the other are counted. The total number of objects per unit volume is then

\[ N_v = \frac{\text{Count}}{\frac{\text{Area}}{T}} \]  

where Area is the area of each of the images. This method has the advantage of being insensitive to the shape and size of the objects, but it requires that the planes be close enough together that no information is lost between the planes. In effect, this means that the distance T must be small compared to any important dimension of the objects.

When T is small, most objects intersect both planes. The features in those planes will not correspond exactly, but are expected to overlap at least partially. In the case of a branching three-dimensional object, both of the intersections in one plane are expected to overlap with the intersection in the second plane. Of course, since most of the objects do pass through both planes when T is small, and only the few that do not are counted, it is necessary to examine a large image area to obtain a statistically useful number of counts. That requirement makes the use of an automated method based on the Feature-AND logic attractive.

The features which overlap in the two images are those which are not counted; therefore, a candidate procedure for determining the value of N to be used in the calculation of number of objects per unit volume might be to first count the number of features in each of the two plane images (N1 and N2). Then, the Feature-AND can be used to determine the features which are present in both images, and a count of those features (N(common)) obtained, giving

\[ N = N_1 + N_2 - 2 \cdot N_{\text{common}} \]  

However, this is correct only for the case in which each object intersects each plane exactly once. For branching objects, it will result in an error.

A preferred procedure is to directly count the features in the two planes that are not selected by the Feature-AND. The logical operation does not commute, so it is necessary to perform both operations: (NOT1 F-AND #2) and (NOT2 F-AND #1), and count the features remaining. This is illustrated schematically in Figure 32.

Figure 33 shows a typical application. The two images are separate slices reconstructed from X-ray tomography of a sintered ceramic sample. Each image is thresholded to generate a binary image of particle intersections. Each of the Feature-AND operations is performed, and the final image is the OR combination showing those features that appear in one (and only one) of the two slices. It would be appropriate to describe this image as a feature-based version of the exclusive-OR operation between the two images.

**Double thresholding**

Another application for Feature-AND logic arises in the thresholding of difficult images such as grain boundaries in materials or cell boundaries in tissue. It is not unusual to have nonuniform etching or staining of the cell or grain boundaries in specimen preparation. In the example of Figure 34, this is due to thermal etching of the interiors of the grains. The result is that direct thresholding of the image cannot produce a complete representation of the etched boundaries that does not also include “noise” within the grains.
Figure 32. Implementation of the Disector:
(a) two section images, overlaid in different colors to show matching features.
(b) 1 F-AND 2 showing features in plane 2 matched with plane 1;
(c) 2 F-AND 1 showing the features matched in the other plane;
(d) ORing together the 1 NOT F-AND 2 with 2 NOT F-AND 1 leaves just the unmatched features in both planes that area to be counted.

Figure 33. Application of the disector to x-ray tomography slices through a ceramic:
(a) slice 1;
(b) slice 2;
(c) binary image from slice 1;
(d) binary image from slice 2;
(e) [#1 NOT Feature-AND #2] OR [#2 NOT Feature-AND #1].
A technique for dealing with such situations has been described as “double thresholding” by Olson (1993), but can be implemented by using Feature-AND. As illustrated in Figure 34, the procedure is first to threshold the image to select only the darkest pixels that are definitely within the etched boundaries, even if they do not form a complete representation of the boundaries. Then, a second binary image is produced to obtain a complete delineation of all the boundaries, accepting some noise within the grains. In the example, a variance operator was applied to a copy of the original image to increase the contrast at edges. This process allows thresholding more of the boundaries, but also some of the intra-grain structures. Then a morphological closing (discussed later in this chapter) was applied to fill in noise within the boundaries. The increase in apparent width of the boundaries is not important, because skeletonization (also discussed in the following section) is used to reduce the boundary lines to minimum width (the actual grain boundaries are only a few atoms thick).

**Figure 34. Double thresholding of grain boundaries in alumina:**
(a) original image.
(b) first thresholding of dark grain boundary markers.
(c) variance operator applied to original.
(d) second thresholding of image c for all boundaries plus other marks.
(e) Feature-AND of image b with image d.
(f) closing applied to e.
(g) skeletonized and pruned boundary overlaid on original.
The two binary images are combined with a Feature-AND to keep any feature in the second image that touches one in the first. This uses the few dark pixels that definitely lie within the boundaries as markers to select the broader boundaries, while rejecting the noise within the grains. Finally, as shown in the figure, the resulting image is skeletonized and pruned to produce an image useful for stereological measurements of grain boundary area, grain size, and so forth.

In the preceding example, the grain boundary network is a continuous tessellation of the image. Hence, it could be selected by using other criteria than the double-threshold method (for instance, touching multiple edges of the field). Figure 35 shows an example requiring the double-threshold method. The acoustic microscope image shows a cross section through a fiber-reinforced material. These images are inherently noisy, but double-thresholding (in this example selecting the bright pixels) allows the boundaries around the fibers to be selected. The fibers touch each other, so it is also necessary to separate them for measurement using a watershed segmentation as discussed in the next section.

**Figure 35. Double thresholding of fiber boundaries:**
(a) original image;
(b) first thresholding;
(c) second thresholding;
(d) Feature-AND;
(e) filled boundaries;
(f) segmented fibers.
Erosion and dilation

The most extensive class of binary image processing operations is often collectively described as morphological operations (Serra, 1982; Coster and Chermant, 1985; Dougherty and Astola, 1994, 1999; Soille, 1999). These include erosion and dilation, and modifications and combinations of these operations. All are fundamentally neighbor operations, as were discussed in Chapters 3 and 4 to process grey-scale images in the spatial domain. Because the values of pixels in the binary images are restricted to 0 or 1, the operations are simpler and usually involve counting rather than sorting or weighted multiplication and addition. However, the basic ideas are the same, and it is possible to perform these procedures using the same specialized array-processor hardware sometimes employed for grey-scale kernel operations.

Rich literature, much of it French, is available in the field of mathematical morphology. It has developed a specific language and notation for the operations and is generally discussed in terms of set theory. A much simpler and more empirical approach is taken here. Operations can be described simply in terms of adding or removing pixels from the binary image according to certain rules, which depend on the pattern of neighboring pixels. Each operation is performed on each pixel in the original image, using the original pattern of pixels. In practice, it may not be necessary to create an entirely new image; the existing image can be replaced in memory by copying a few lines at a time. None of the new pixel values are used in evaluating the neighbor pattern.

Erosion removes pixels from features in an image or, equivalently, turns pixels OFF that were originally ON. The purpose is to remove pixels that should not be there. The simplest example is pixels that have been selected by thresholding because they fell into the brightness range of interest, but do not lie within large regions with that brightness. Instead, they may have that brightness value either accidentally, because of finite noise in the image, or because they happen to straddle a boundary between a lighter and darker region and thus have an averaged brightness that happens to lie in the range selected by thresholding.

Such pixels cannot be distinguished by simple thresholding because their brightness value is the same as that of the desired regions. It may be possible to ignore them by using two-parameter thresholding, for instance using the grey level as one axis and the gradient as a second one, and requiring that the pixels to be kept have the desired grey level and a low gradient. For our purposes here, however, we will assume that the binary image has already been formed and that extraneous pixels are present.

The simplest kind of erosion, sometimes referred to as classical erosion, is to remove (set to OFF) any pixel touching another pixel that is part of the background (is already OFF). This removes a layer of pixels from around the periphery of all features and regions, which will cause some shrinking of dimensions and may create other problems if it causes a feature to break up into parts. We will deal with these difficulties below. Erosion can entirely remove extraneous pixels representing point noise or line defects (e.g., scratches) because these defects are normally only a single pixel wide.

Instead of removing pixels from features, a complementary operation known as dilation (or sometimes dilatation) can be used to add pixels. The classical dilation rule, analogous to that for erosion, is to add (set to ON) any background pixel which touches another pixel that is already part of a foreground region. This will add a layer of pixels around the periphery of all features and regions, which will cause some increase in dimensions and may cause features to merge. It also fills in small holes within features.

Because erosion and dilation cause a reduction or increase in the size of regions, respectively, they are sometimes known as etching and plating or shrinking and growing. A variety of rules are followed in order to decide which pixels to add or remove and for forming combinations of erosion and dilation.

© 2002 by CRC Press LLC
In the rather simple example described previously and illustrated in Figure 36, erosion to remove the extraneous lines of pixels between light and dark phases causes a shrinking of the features. Following the erosion with a dilation will more or less restore the pixels around the feature periphery, so that the dimensions are (approximately) restored. Isolated pixels that have been completely removed, however, do not cause any new pixels to be added. They have been permanently erased from the image.

Opening and closing

The combination of an erosion followed by a dilation is called an opening, referring to the ability of this combination to open up gaps between just-touching features, as shown in Figure 37. It is one of the most commonly used sequences for removing pixel noise from binary images. Performing the same operations in the opposite order (dilation followed by erosion) produces a different result. This sequence is called a closing because it can close breaks in features. Several parameters can be used to adjust erosion and dilation operations, particularly the neighbor pattern and the number of iterations, as discussed below. In most opening operations, these are kept the same for both the erosion and the dilation.
Openings can be used in some cases to separate touching features. In the example shown in Figure 38, the features are all similar in size. This fact makes it possible to continue the erosion until all features have separated but none have been completely erased. After the separation is complete, dilation grows the features back toward their original size. They would merge again unless logic is used to prevent it. A rule that prevents turning a pixel ON if its neighbors belong to different features maintains the separation shown in the figure. This requires performing feature identification for the pixels, so the logic discussed previously is required at each step of the dilation. An additional rule prevents turning on any pixel that was not on in the original image, so that the features are restricted to their original sizes. If the features had different original sizes, the separation lines would not lie correctly at the junctions. The watershed segmentation technique discussed later in this chapter performs better in such cases.

If the sequence is performed in the other order, that is, a dilation followed by an erosion, the result is not the same. Instead of removing isolated pixels that are ON, the result is to fill in places where isolated pixels are OFF, missing pixels within features or narrow gaps between portions of a feature. Figure 39 shows an example of a closing used to connect together the parts of the cracked fibers shown in cross section. The cracks are all narrow, so dilation causes the pixels from either side to spread across the gap. The increase in fiber diameter is then corrected by an erosion, but the cracks do not reappear.

The classical erosion and dilation operations illustrated above turn a pixel ON or OFF if it touches any pixel in the opposite state. Usually, touching in this context includes any of the adjacent 8 pixels, although some systems deal only with the 4 edge-sharing neighbors. These operations would also be much simpler and more isotropic on a hexagonal pixel array, because the pixel neighbor distances are all the same, but practical considerations lead to the general use of a grid of square pixels.

A wide variety of other rules are possible. One approach is to count the number of neighbor pixels in the opposite state, compare this number to some threshold value, and only change the state of the central pixel if that test coefficient is exceeded. In this method, classical erosion corresponds to a coefficient of 0. One effect of different coefficient values is to alter the rate at which features grow or shrink and to some extent to control the isotropy of the result. This will be illustrated in the next section.
It is also possible to choose a large coefficient, from 5 to 7, to select only the isolated noise pixels and leave most features alone. For example, choosing a coefficient of 7 will cause only single isolated pixels to be reversed (removed or set to OFF in an erosion, and vice versa for a dilation). Coefficient values of 5 or 6 may be able to remove lines of pixels (such as those straddling a boundary) without affecting anything else.

An example of this method is shown in Figure 40. Thresholding the original image of the pigment cell produces a binary image showing the features of interest and creates many smaller and irregular groups of pixels. Performing a conventional opening to remove them would also cause the shapes of the larger features to change and some of them to merge. Applying erosion with a neighbor coefficient of 5 removes the small and irregular pixel groups without affecting the larger and more rounded features, as shown. The erosion is repeated until no further changes take place (the number of ON pixels in the binary image does not change). This procedure works because a corner pixel in a square has exactly five touching background neighbors and is not removed, while more irregular clusters have pixels with six or more background neighbors.

The test image in Figure 41 shows a variety of fine lines and narrow gaps that can be removed or filled in using different neighbor coefficients and number of iterations (number of erosions followed by dilations, or vice versa).
Isotropy

It is not possible for a small $3 \times 3$ neighborhood to define an isotropic neighbor pattern. Classic erosion applied to a circle will not shrink the circle uniformly, but will proceed at a faster rate in the 45° diagonal directions because the pixel spacing is greater in those directions. As a result, a circle will erode toward a diamond shape, as shown in Figure 42. Once the feature reaches this shape, it will continue to erode uniformly, preserving the shape. In most cases, however, features are not really diamond-shaped, which represents a potentially serious distortion.

Likewise, classic dilation applied to a circle also proceeds faster in the 45° diagonal directions, so that the shape dilates toward a square (also shown in Figure 42). Again, square shapes are stable in dilation, but the distortion of real images toward a block appearance in dilation can present a problem for further interpretation.
Figure 4.1. Illustration of the effect of different neighbor coefficients and number of iterations:
(a) original test image;
(b) erosion, neighbor coefficient = 3, 1 iteration, removes isolated lines and points;
(c) closing, neighbor coefficient = 2, 2 iterations, fills in gaps to connect features while removing
isolated points;
(d) closing using classical operations (neighbor coefficient = 0, 1 iteration) connects most features but
leaves isolated points;
(e) opening, neighbor coefficient = 7, 1 iteration, removes point noise without affecting anything else;
(f) opening, neighbor coefficient = 1, 4 iterations, removes all small features including the frame of the
picture.
A neighbor coefficient of 1 instead of 0 produces a markedly different result. For dilation, a background pixel that touches more than one foreground pixel (i.e., two or more out of the possible eight neighbor positions) will be turned ON and vice versa for erosion. Eroding a circle with this procedure tends toward a square and dilation tends toward a diamond, just the reverse of using a coefficient of 0. This is shown in Figure 43.

No possible intermediate value exists between 0 and 1, because the pixels are counted as either ON or OFF. If the corner pixels were counted as 2 and the edge-touching pixels as 3, it would be possible to design a coefficient that better approximated an isotropic circle. This would produce a ratio of 3/2 = 1.5, which is a reasonable approximation to $\sqrt{2}$, the distance ratio to the pixels. In practice, this is rarely done because of the convenience of dealing with pixels in binary images as a simple 0 or 1 value with no need to take into account their neighborhood.

Another approach that is much more commonly used for achieving an intermediate result between the coefficients of 0 and 1 with their directional bias is to alternate the two tests. As shown in Figure 44, this alternating pattern produces a much better approximation to a circular shape in both erosion and dilation. This procedure raises the point that erosion or dilation need not be performed only once. The number of repetitions, also called the depth of the operation, corresponds roughly to the distance that boundaries will grow or shrink radially. It may be expressed in pixels or converted to the corresponding scale dimension.
Using a larger neighborhood can also moderate the anisotropy. In Figure 45, a 5-pixel-wide circular neighborhood is used with ten iterations of erosion and dilation. As for the alternating 0 and 1 coefficients, the shapes evolve toward octagons, although the larger neighborhood provides less control over the distance used for erosion and dilation.

Each neighbor pattern or coefficient has its own characteristic anisotropy. Figure 46 shows the rather interesting results using a neighborhood coefficient of 3. Similar to an alternating 0,1 pattern, this operation produces an 8-sided polygon, however, the rate of erosion is much lower. In dilation, the figure grows to the bounding octagon and then becomes stable, with no further pixels being added. This coefficient is sometimes used to construct bounding polygons around features.
Measurements using erosion and dilation

Erosion performed \( n \) times (using either a coefficient of 0 or 1, or alternating them) will cause features to shrink radially by about \( n \) pixels (with local variations depending on the shape of the original feature). This will cause features whose smallest dimension is less than \( 2n \) pixels to disappear altogether. Counting the features that have disappeared (or subtracting the number that remain from the original) gives an estimate of the number of features smaller than that size. This means that erosion and counting can be used to get an estimate of size distributions without actually performing feature measurements (Ehrlich et al., 1984).

For irregularly shaped and concave features, the erosion process may cause a feature to subdivide into parts. Simply counting the number of features as a function of the number of iterations of erosion is therefore not a good way to determine the size distribution. One approach to this problem is to follow erosion by a dilation with the same coefficient(s) and number of steps. This will merge together many (but not necessarily all) of the separated parts and give a better estimate of their number, although there is still considerable sensitivity to the shape of the original features. A dumbbell-shaped object will separate into two parts when the handle between the two main parts erodes; they will not merge. This separation may be desirable, if indeed the purpose is to count the two main parts.

A second method is to use Feature-AND, discussed earlier. After each iteration of erosion, the remaining features are used to select only those original features that touch them. The count of original features then gives the correct number. This is functionally equivalent to keeping feature labels on each pixel in the image and counting the number of different labels present in the image after each cycle of erosion. This method of estimating size distributions without actually measuring features, using either of these correction techniques, has been particularly applied to measurements in geology, such as mineral particle sizes or sediments.

The opposite operation, performing dilations and counting the number of separate features as a function of the number of steps, is less common. It provides an estimate of the distribution of the nearest distances between features in the image. When this is done by conventional feature measurement, the \( x,y \) location of each feature is determined, then sorting in the resulting data file is used to determine the nearest neighbor and its distance. When the features are significantly large compared to their spacing or when their shapes are important, it can be more interesting to characterize the distances between their boundaries. This dilation method can provide that information.

Instead of counting the number of features that disappear at each iteration of erosion, it is much easier simply to count the number of ON pixels remaining, which provides some information about the shape of the boundaries. Smooth Euclidean boundaries erode at a constant rate. Irregular and especially fractal boundaries do not, since many more pixels are exposed and touch opposite neighbors. This effect has been used to estimate fractal dimensions, although several more accurate methods are available as discussed below.

Fractal dimensions and the description of a boundary as fractal based on a self-similar roughness is a fairly new idea that is finding many applications in science and art (Mandelbrot, 1982, Feder, 1988, Russ, 1994). No description of the rather interesting background and uses of the concept is included here for want of space. The basic idea behind measuring a fractal dimension by erosion and dilation comes from the Minkowski definition of a fractal boundary dimension. By dilating a region and Ex-ORing the result with another image formed by eroding the region, the pixels along the boundary are obtained. For a minimal depth of erosion and dilation, this will be called the custer and is discussed in the section titled “The custer.”

To measure the fractal dimension, the operation is repeated with different depths of erosion and dilation (Flook, 1978), and the effective width (total number of pixels divided by length and...
The number of cycles) of the boundary is plotted vs. the depth on a log-log scale. For a Euclidean boundary, this plot shows no trend; the number of pixels along the boundary selected by the Ex-OR increases linearly with the number of erosion/dilation cycles. For a rough boundary with self-similar fine detail, however, the graph shows a linear variation on log-log axes whose slope gives the fractal dimension of the boundary directly. Figure 47 shows an example.

A variety of other methods are used to determine the boundary fractal dimension, including box-counting or mosaic amalgamation (Kaye, 1986; Russ, 1990) in which number of pixels through which the boundary passes (for boundary representation) are counted as the pixel size is increased by coarsening the image resolution, and a structured walk method (Schwarz and Exner, 1980), which requires the boundary to be represented as a polygon instead of as pixels. For a fractal boundary, these also produce straight line plots on a log-log scale, from whose slope the dimension is determined. Newer and more accurate techniques for performing the measurement are shown in Chapter 9.

Counting the number of pixels as a function of dilations also provides a rather indirect measure of feature clustering, because as nearby features merge, the amount of boundary is reduced and the region's rate of growth slows. Counting only the pixels and not the features makes it difficult to separate the effects of boundary shape and feature spacing. If all of the features are initially very small or if they are single points, this method can provide a fractal dimension (technically a Sierpinski fractal) for the clustering.

Figure 47. Measurement of Minkowski fractal dimension by erosion/dilation:
(a) test figure with upper boundary a classical Koch fractal and lower boundary a Euclidean straight line;
(b) grey pixels show difference between erosion and dilation by one iteration;
(c, d, e) differences between erosion and dilation after 2, 3, and 4 iterations;
(f) plot of log of effective width (area of grey pixels divided by length and number of iterations) vs. log of number of iterations (approximate width of grey band).
Extension to grey-scale images

In Chapter 4, one of the image processing operations described was the use of a ranking operator, which finds the brightest or darkest pixel in a neighborhood and replaces the central pixel with that value. This operation is sometimes described as a grey-scale erosion or dilation, depending on whether the use of the brightest or darkest pixel value results in a growth or shrinkage of the visible features.

Just as an estimate of the distribution of feature sizes can be obtained by eroding features in a binary image, the same technique is also possible using grey-scale erosion on a grey-scale image. Figure 48 shows an example. The lipid spheres in this SEM image are partially piled up and obscure one another, which is normally a critical problem for conventional image-measurement techniques. Applying grey-scale erosion reduces the feature sizes, and counting the bright central points that disappear at each step of repeated erosion provides a size distribution.

The assumption in this approach is that the features ultimately separate before disappearing. This works for relatively simple images with well-rounded convex features, none of which are more than about half hidden by others. No purely two-dimensional image processing method can count the number of cannon balls in a pile if the inner ones are hidden. It is possible to estimate the volume of the pile and guess at the maximum number of balls contained, but impossible to know whether they are actually there or whether something else is underneath the topmost layer.

Figure 48. Use of grey-scale erosion to estimate size distribution of overlapped spheres:
(a) original SEM image of lipid droplets;
(b–f) result of applying repetitions of grey-scale erosion by keeping the darkest pixel value in a 5-pixel-wide octagonal neighborhood.
Morphology neighborhood parameters

The important parameters for erosion and dilation are the neighborhood size and shape, the comparison test that is used and the number of times the operation is repeated. The use of a simple test coefficient based on the number of neighbors, irrespective of their location in the neighborhood, provides considerable flexibility in the functioning of the operation as shown earlier. Each coefficient produces results having a characteristic shape, which distorts the original features. Also, the greater the depth, or number of iterations in the operation, the greater this effect, in addition to the changes in the number of features present.

Specific neighbor patterns can also be used for erosion and dilation operations. The most common are ones that compare the central pixel to its 4 edge-touching neighbors (usually called a "+" pattern because of the neighborhood shape) or to the 4 corner-touching neighbors (likewise called an "x" pattern), changing the central pixel if any of the 4 neighbors is of the opposite type (ON or OFF). They are rarely used alone, but can be employed in an alternating pattern to obtain greater directional uniformity than classical erosion, similar to the effects produced by alternating coefficient tests of 0 and 1.

Any specific neighbor pattern can be used, of course. It is not even required to restrict the comparison to immediately touching neighbors. As for grey-scale operations, larger neighborhoods make it possible to respond to more subtle textures and achieve greater control over directionality. Figure 49 shows a simple example. The general case for this type of operation is called the hit-or-miss operator, which specifies any pattern of neighboring pixels divided into three classes: those that must be ON, those that must be OFF, and those that do not matter (are ignored). If the pattern is found, then the pixel is set to the specified state (Serra, 1982; Coster and Chermant, 1985).

This operation is also called template matching. The same type of operation carried out on grey-scale images is called convolution and is a way to search for specific patterns in the image. This is also true for binary images, in fact, template matching with thresholded binary images was one of the earliest methods for optical character reading and is still used for situations in which the character shape, size, and location are tightly controlled (such as the characters at the bottom of bank checks). Much more flexible methods are needed to read more general text, however. In practice, most erosion and dilation is performed using only the 8 nearest-neighbor pixels for comparison.

One method for implementing neighborhood comparison that makes it easy to use any arbitrary pattern of pixels is the fate table. The 8 neighbors each have a value of 1 or 0, depending on

![Figure 49. Example of specifying the neighborhood pixels for morphological operations:](image)

(a) a vertical neighborhood for erosion;
(b) original pattern;
(c) eroded result.
whether the pixel is ON or OFF. Assembling these 8 values into a number produces a single byte, which can have any of 256 possible values. This value is used as an address into a table, which provides the result (i.e., turning the central pixel ON or OFF). Figure 50 illustrates the relationship between the neighbor pattern and the generated address.

Efficient ways to construct the address by bitwise shifting of values, which takes advantage of the machine-language idiosyncrasies of specific computer processors, makes this method very fast. The ability to create several tables of possible fates to deal with different erosion and dilation rules, perhaps saved on disk and loaded as needed, makes the method very flexible; however, it does not generalize well to larger neighborhoods or three-dimensional voxel array images because the tables become too large.

Some applications for highly specific erosion/dilation operations are not symmetrical or isotropic. These always require some independent knowledge of the image, the desired information, and the selection of operations that will selectively extract it. This is not as important a criticism or limitation as it may seem, however, because all image processing is to some extent knowledge-directed. The human observer tries to find operations to extract information he or she has some reason to know or expect to be present.

Figure 51 shows an example. The horizontal textile fibers vary in width as they weave above and below the vertical ones. Measuring this variation is important to modeling the mechanical properties of the weave, which will be embedded into a composite. The dark vertical fibers can be thresholded based on brightness, but delineating the horizontal fibers is very difficult. The procedure shown in the figure uses the known directionality of the structure.

After thresholding the dark fibers, an erosion is performed to remove only those pixels whose neighbor immediately below or above is part of the background. These pixels, shown in Figure 51c, can then be isolated by performing an Ex-OR with the original binary. They include the few points distinguishable between horizontal fibers and the ends of the vertical fibers where they are covered by horizontal ones.

Next, a directional dilation is performed in the horizontal direction. Any background pixel whose left or right touching neighbor is ON is itself set to ON, and this operation is repeated enough times to extend the lines across the distance between vertical fibers. Finally, the resulting horizontal lines are ORed with the original binary image to outline all of the individual fibers (Figure 51d). Inverting this image produces measurable features.

**Examples of use**

Some additional examples of erosion and dilation operations will illustrate typical applications and methods. One of the major areas of use is for x-ray maps from the SEM. These are usually so sparse that even though they are recorded as grey-scale images, they are virtually binary images even before thresholding because most pixels have zero photons and a few pixels have one.

---

*Figure 50. Constructing an address into a fate table by assigning each neighbor position to a bit value.*
Regions containing the element of interest are distinguished from those that do not by a difference in the spatial density of dots, which humans are able to interpret by a gestalt grouping operation. This very noisy and scattered image is difficult to use to locate feature boundaries. Dilation may be able to join points together to produce a more useful representation.

Figure 51. Using directional erosion and dilation to segment an image:
(a) original grey-scale image of a woven textile;
(b) brightness thresholding of image a;
(c) end pixels isolated by performing a vertical erosion and EXOring with the original;
(d) completed operation by repeated horizontal dilation of image c and then ORing with the original.

Figure 52. X-ray “dot” maps from the SEM:
(a) backscattered electron image of a gold grid above an aluminium stub;
(b) secondary electron image;
(c) gold x-ray dot image;
(d) aluminium x-ray image (notice the shadows of grid).
Figure 52 shows a representative x-ray map from an SEM. Notice that the dark bands in the aluminum dot map represent the shadows where the gold grid blocks the incident electron beam or the emitted x-rays en route to the detector. Figure 53 shows the result of thresholding the gold map and applying a closing to merge the individual dots. Figure 54 illustrates the results for the aluminum map. Because it has more dots, it produces a somewhat better definition of the region edges.

Other images from the light and electron microscope sometimes have the same essentially binary image as well. Examples include ultrathin biological tissue sections stained with heavy metals and viewed in the TEM, and chemically etched metallographic specimens. The dark regions are frequently small, corresponding to barely resolved individual particles whose distribution and clustering reveal the desired microstructure (membranes in tissue, eutectic lamellae in metals, etc.) to the eye. As for the case of x-ray dot maps, it is sometimes possible to utilize dilation operations to join such dots to form a well-defined image.

In Figure 55, iron carbide particles in a steel specimen are etched to distinguish the regions with and without such structures. The islands of lamellar structure are important, but not completely defined by the individual dark carbide particles. Dilation followed by erosion (a closing) merges together the individual lamellae, dark regions are also found within the essentially white grains because of the presence of a few dark points in the original image. Following the closing with an opening (for a total sequence of dilation, erosion, erosion, dilation) produces a useful result. In the example, the closing used a neighborhood coefficient of 1 and 6 iterations, and the opening used a neighborhood coefficient of 0 and 4 iterations. The number of iterations is based on the size of the gap to be filled or feature to be removed. The presence of 45- and 90-degree edges in the processed binary images reveals the anisotropic effects of the erosion/dilation operations.
Using different coefficients in the various operations sometimes lessens the obvious geometric bias. The choice of appropriate parameters is largely a matter of experience with a particular type of image and human judgment of the correctness of the final result.

There is a basic similarity between using these morphological operations on a thresholded binary image and some of the texture operators used in Chapter 4 on grey-scale images. In most cases, similar (but not identical) results can be obtained with either approach (provided the software offers both sets of tools). For instance, Figure 56 shows the same image of curds and whey used earlier to compare several grey-scale texture processing operations. Background leveling and thresholding the smooth, white areas (the curds) produces the result shown. Clearly, many regions in the textured whey protein portion of the image are just as bright as the curds. In grey-scale texture processing, these were eliminated based on some consideration of the local variation in pixel brightness. In this image, that variation produces narrow and irregular thresholded regions. An opening, consisting of an erosion to remove edge-touching pixels and a dilation to restore pixels smoothly to boundaries that still exist, effectively removes the background clutter as shown in the figure. Small features are shaded grey and would normally be removed based on size to permit analysis of the larger curds. The erosion/dilation approach to defining the structure in this image amounts to making some assumptions about the characteristic dimensions of the features, their boundary irregularities, and their spacings.
The custer

Erosion/dilation procedures are often used along with Boolean combinations. In the examples of Figures 27 through 30, the lines used to test for adjacency were obtained by dilating the binary image and then EX-ORing the result with the original. Also, the outlines shown in many of the figures to compare the results of processing to the original image can be produced by performing an erosion followed by an EX-OR with the original binary image. This leaves the outlines which were then applied as a mask to the original grey-scale image. Whether obtained by erosion or dilation, or doing both and EX-ORing the results, the outline is called the custer of a feature, apparently in reference to George Herbert Armstrong Custer, who was also surrounded.

The custer can be used to determine neighbor relationships between features or regions. As an example, Figure 57 shows a three-phase metal alloy imaged in the light microscope. Each of the individual phases can be readily delineated by thresholding (and in the case of the medium grey image, applying an opening to remove lines of pixels straddling the white-black boundary). Then the custer of each phase can be formed as described previously.

Combining the custer of each phase with the other phases using an AND keeps only the portion of the custer that is common to the two phases. The result is to mark the boundaries as white-grey, grey-black, or black-white, so that the extent of each type can be determined by simple counting.

In other cases, Feature-AND can be used to select the entire features that are adjacent to one region (and thus touch its custer), as illustrated previously.

Euclidean distance map

The directional bias present in morphological operations because of their restriction to pixels on a square grid can be largely overcome by performing equivalent operations using a different
Figure 57. Use of Boolean logic to measure neighbor relationships: (a) original light microscope image of a three-phase metal. (b–d) Threshold white, grey, and black phases. (e–g) Surrounding outlines of each phase produced by dilation and EXOR with original. (h–j) AND of outlines of pairs of phases. (k) OR of all ANDed outlines using different colors to identify each phase/phase interface. (l) Outlines filled to show idealized phase regions.
technique. It makes use of a grey-scale image, produced from the original binary, in which every pixel within a feature is assigned a value that is its distance from the nearest background pixel. This is called the Euclidean distance map, or EDM.

Most of the image processing functions discussed in this and preceding chapters operate either on grey-scale images (to produce other grey-scale images) or on binary images (to produce other binary images). The Euclidean distance map is a tool that works on a binary image to produce a grey-scale image. The definition is simple enough: each pixel in the foreground is assigned a brightness value equal to its straight line (thus, “Euclidean”) distance from the nearest point in the background. In a continuous image, as opposed to a digitized one containing finite pixels, this is unambiguous. In most pixel images, the distance is taken from each pixel in the feature to the nearest pixel in the background.

Searching through all of the background pixels to find the nearest one to each pixel in a feature and calculating the distance in a Pythagorean sense would be an extremely inefficient and time-consuming process for constructing the EDM. Some researchers have implemented a different type of distance map in which distance measured in only a few directions. For a lattice of square pixels, this may either be restricted to the 90° directions, or it may also include the 45° directions (Rosenfeld and Kak, 1982). This measuring convention is equivalent to deciding to use a 4-neighbor or 8-neighbor convention for considering whether pixels are touching. In either case, the distance from each pixel to one of its 4 or 8 neighbors is taken as 1, regardless of the direction. Consequently, as shown in Figure 58, the distance map from a point gives rise to either square or diamond-shaped artefacts and is quite distorted, as compared to the Pythagorean distance. These measuring conventions are sometimes described as city-block models (connections in 4 directions) or chessboard models (8 directions), because of the limited moves available in those situations.

A conceptually straightforward, iterative technique for constructing such a distance map can be programmed as follows:

1. Assign a brightness value of 0 to each pixel in the background.
2. Set a variable N equal to 0.
3. For each pixel that touches (in either the 4- or 8-neighbor sense, as described previously) a pixel whose brightness value is N, assign a brightness value of N + 1.
4. Increment N and repeat step 3, until all pixels in the image have been assigned.

The time required for this iteration depends on the size of the features (the maximum distance from the background). A more efficient method is available that gives the same result with two passes through the image (Danielsson, 1980). This technique uses the same comparisons, but propagates the values through the image more rapidly. It can be programmed as follows:

```
1. Assign a brightness value of 0 to each pixel in the background.
2. Set a variable N equal to 0.
3. For each pixel that touches (in either the 4- or 8-neighbor sense, as described previously) a pixel whose brightness value is N, assign a brightness value of N + 1.
4. Increment N and repeat step 3, until all pixels in the image have been assigned.

The time required for this iteration depends on the size of the features (the maximum distance from the background). A more efficient method is available that gives the same result with two passes through the image (Danielsson, 1980). This technique uses the same comparisons, but propagates the values through the image more rapidly. It can be programmed as follows:
```

```
1. Assign a brightness value of 0 to each pixel in the background.
2. Set a variable N equal to 0.
3. For each pixel that touches (in either the 4- or 8-neighbor sense, as described previously) a pixel whose brightness value is N, assign a brightness value of N + 1.
4. Increment N and repeat step 3, until all pixels in the image have been assigned.
```

Figure 58. Arrays of pixels with their distances from the center pixel shown (from left to right) for the cases of 4- and 8-neighbor paths, and in Pythagorean units.

© 2002 by CRC Press LLC
1. Assign the brightness value of 0 to each pixel in the background and a large positive value (greater than the maximum feature width) to each pixel in a feature.

2. Proceeding from left to right and top to bottom, assign each pixel within a feature a brightness value one greater than the smallest value of any of its neighbors.

3. Repeat step 2, proceeding from right to left and bottom to top.

A further modification provides a better approximation to the Pythagorean distances between pixels (Russ and Russ, 1988b). The diagonally adjacent pixels are neither a distance 1 (8-neighbor rules) or $\sqrt{2} \approx 1.414$ (4-neighbor rules) away. The latter value is an irrational number, but closer approximations than 1.00 or 2.00 are available. For instance, modifying the above rules so that a pixel brightness value must be larger than its 90° neighbors by 2 and greater than its 45° neighbors by 5 is equivalent to using an approximation of 1.5 for the square root of 2.

The disadvantage of this method is that all of the pixel distances are now multiplied by two, increasing the maximum brightness of the EDM image by this factor. For images capable of storing a maximum grey level of 255, this represents a limitation on the largest features that can be processed in this way. If the EDM image is 16 bits deep (and can hold values up to 65,535), however, this is not a practical limitation. It also opens the way to selecting larger ratios of numbers to approximate $\sqrt{2}$, getting a correspondingly improved set of values for the distance map. For instance, $\frac{7}{5} = 1.400$ and $\frac{58}{41} = 1.415$.

It takes no longer to compare or add these values than it does any others, and the ratio 58/41 allows dimensions larger than 1024 pixels. Because this dimension is the half-width, features or background up to 2048 pixels wide can be processed ($1024 \times 41 = 41,984$, which is less than $2^{16} - 1 = 65,535$). Of course, the final image can be divided down by the scaling factor (41 in this example) to obtain a result in which pixel brightness values are the actual distance to the boundary (rounded or truncated to integers) and the total brightness range is within the 0 to 255 range that most displays are capable of showing.

The accuracy of an EDM constructed with these rules can be judged by counting the pixels whose brightness values place them within a distance s. This is just the same as constructing a cumulative histogram of pixel brightness in the image. Figure 59 plots the error in the number of pixels vs. integer brightness for a distance map of a circle 99 pixels in diameter; the overall errors are not large. Even better accuracy for the EDM can be obtained by performing additional comparisons to pixels beyond the first 8 nearest neighbors. Adding a comparison to the 8 neighbors in the 5 × 5 neighborhood whose Pythagorean distance is $\sqrt{5}$ produces values having even less directional sensitivity and more accuracy for large distances. If the integer values 58 and 41 mentioned above are used to approximate $\sqrt{2}$, then the path to these pixels consisting of a ‘knight’s move’ of one 90°- and one 45°-pixel step would produce a value of $58 + 41 = 99$. Substituting a value of 92 gives a closer approximation to the Pythagorean distance ($92/41 = 2.244$, $\sqrt{2} = 2.236$) and produces more isotropic results.

Figure 59. Difference between theoretical area value ($\pi r^2$) and the actual area covered by the EDM as a function of brightness (distance from boundary) shows increasing but still small errors for very large distances.
There is another algorithm that produces a Euclidean distance map with real number values. During the passes through the image, the X and Y distances from the nearest background point are accumulated separately for each pixel within the features, and then the actual Pythagorean distance is calculated as the square root of the sum of squares. Of course, it is still necessary to convert to an integer representation for display purposes. In general, the better the quality of the EDM values the better the results obtained using the EDM for erosion, dilation, and watershed segmentation as described in the next section.

Comparison of the pixel-by-pixel erosion and dilation described earlier with the circular pattern provided thresholding by the EDM of either the foreground (erosion) or background (dilation) to select pixels that are farther from the edge than any desired extent of erosion shows that the EDM method is much more isotropic (Figure 60). Furthermore, the distance map is constructed quickly and the thresholding requires no iteration, so the execution time of the method does not increase with feature size (as do classical erosion methods) and is preferred for large features or depths. When more irregular shapes are subjected to erosion and dilation, the difference between the iterative methods and thresholding the EDM is also apparent, with EDM methods avoiding the 90- and 45-degree boundaries present with the traditional morphological tools. Figure 61 shows the same example of closing and opening applied to the image in Figure 55. The distance used for both closing and opening was 5.5 pixels (note that with the EDM it is possible to specify distances are real numbers rather than being restricted to integers), and the final outlines trace the edges of the structures with much greater fidelity.

Watershed segmentation

A common difficulty in measuring images occurs when features touch, and therefore cannot be separately identified, counted, or measured. This situation may arise when examining an image of
The Image Processing Handbook

a thick section in transmission, where actual feature overlap may occur, or when particles resting on a surface tend to agglomerate and touch each other. One method for separating touching, but mostly convex, features in an image is known as watershed segmentation (Beucher and Lantnejoul, 1979; Lantnejoul and Beucher, 1981). It relies on the fact that eroding the binary image will usually cause touching features to separate before they disappear.

The classical method for accomplishing this separation (Jernot, 1982) is an iterative one. The image is repetitively eroded, and at each step those separate features that disappeared from the previous step are designated ultimate eroded points (UEPs) and saved as an image, along with the iteration number. Saving these is necessary because the features will in general be of different sizes and would not all disappear in the same number of iterations, as mentioned earlier in connection with Figure 38. The process continues until the image is erased.

Then, beginning with the final image of UEPs, the image is dilated using classical dilation, but with the added logical constraint that no new pixel may be turned ON if it causes a connection to form between previously separate features or if it was not ON in the original image. At each stage of the dilation, the image of UEPs that corresponds to the equivalent level of erosion is added to the image using a logical OR. This process causes the features to grow back to their original boundaries, except that lines of separation appear between the touching features.

The method just described has two practical drawbacks: the iterative process is slow, requiring each pixel in the image to be processed many times, and the amount of storage required for all of the intermediate images is quite large. The same result can be obtained more efficiently using an EDM. Indeed, the name “watershed” comes directly from the EDM. Imagine that the brightness values of each pixel within features in an EDM correspond to a physical elevation. The features then appear as a mountain peak. Figure 62 illustrates this for a circular feature.

If two features touch or overlap slightly, the EDM shows two peaks, as shown in Figure 63. The slope of the mountainside is constant, so the larger the feature the higher the peak. The ultimate eroded points are the peaks of the mountains, and where features touch, the flanks of the mountains intersect. The saddles between these mountains are the lines selected as boundaries by the
segmentation method. They are locations where water running down from the mountains arrives from two different peaks, and thus are generally called watershed lines. The placement of these lines according to the relative height of the mountains (size of the features) gives the best estimate of the separation lines between features, which are divided according to the regions that belong to each mountain top.

Implementing the segmentation process using an EDM approach (Russ and Russ, 1988b) is very efficient, both in terms of speed and storage. The distance map image required is constructed without iteration. The ultimate eroded points are located as a special case of local maxima (A further discussion of UEPs is included in the next section), and the brightness value of each directly corresponds to the iteration number at which it would disappear in the iterative method. Dilating these features is fast, because the distance map supplies a constraint. Starting at the brightest value and “walking down the mountain” covers all of the brightness levels. At each one, only those pixels at the current brightness level in the distance map need to be considered. Those that do not produce a join between feature pixels are added to the image. The process continues until all of the pixels in the features, except for those along the separation lines, have been restored.

© 2002 by CRC Press LLC
Figure 64 shows an example of this method, applied to an image consisting of touching circles. Since these are of different sizes, the method described earlier in Figure 38 does not work, but watershed segmentation separates the features. For an image of real particles, as shown in Figure 65, the method works subject to the assumption that the features are sufficiently convex so that the EDM does not produce multiple peaks within each feature.

Of course, this method is not perfect. Watershed segmentation cannot handle concave and irregular particles, nor does it separate particles whose overlap is so great that there is no minimum in the EDM between them. Depending on the quality of the original distance map, watershed segmentation may subdivide lines of constant width into many fragments because of the apparent minima produced by aliasing along the line edges. In most cases, the effort needed to correct such defects is much less than would have been required to perform manual separation of the original features.
The presence of holes within features confuses the watershed algorithm and breaks the features up into many fragments. It is therefore necessary to fill holes before applying the watershed, although there may also be holes in the image between features as well as those within them. Normal hole filling would fill them in since any region of background not connected to the edge of the image is considered a hole. This difficulty can be overcome if some difference in hole size or shape can be identified to permit filling only the holes within features and not those between them (Russ, 1995f). In the example shown in Figure 66, the holes within features (organelles within the cells) are much rounder than spaces between the touching cells. Isolating these holes by measurement, and ORing them with the original image, allows watershed segmentation to separate the cells.

Figure 66. Separation of touching cells:
(a) original grey-scale image;
(b) thresholded;
(c) erroneous watershed segmentation produced by holes within cells;
(d) inverting image b to show holes within and between cells;
(e) holes within cells selected by their rounder shapes;
(f) combining image b with e using a Boolean OR;
(g) watershed segmentation of image f;
(h) outlines superimposed on original image.
The ultimate eroded points described previously in the watershed segmentation technique can be used as a measurement tool in their own right. The number of points gives the number of separable features in the image, while the brightness of each point gives a measure of their sizes (the inscribed radius, shown in Figure 65e). In addition, the location of each point can be used as a location for the feature if clustering or gradients are to be investigated.

The formal definition of a UEP in a continuous, rather than pixel-based, image is simply a local maximum of brightness in the EDM image. Since the image is subdivided into finite pixels, the definition must take into account the possibility that more than one pixel may have equal brightness, forming a plateau. The operating definition for finding these pixels is recursive:

\[
\{U: \forall U_j \text{ neighbors of } U_i, |U_j| \leq |U_i| \text{ AND } \forall U_j \text{ neighbors of } U_i \text{ such that } |U_j| = |U_i|, U_i \in U \}
\]

In other words, the set of pixels which are UEPs must be as bright or brighter than all neighbors; if the neighbors are equal in brightness, then they must also be part of the set.

The brightness of each pixel in the distance map is the distance to the nearest boundary. For a UEP, this must be a point that is equidistant from at least three boundary locations. Consequently, the brightness is the radius of the feature’s inscribed circle. Figure 67 shows the UEPs for the touching circles from Figure 64. A histogram of the brightness values of the UEP pixels gives an immediate measure of the size distribution of the features. This is much faster than convex segmentation, because the iterative dilation is bypassed, and much faster than measurement, since no feature identification or pixel counting is required. Even for separate features, the maximum point of the EDM provides a measurement of the radius of an inscribed circle, a useful size parameter.

Other EDM-based measurements

The Euclidean distance map provides values that can be effectively used for many types of measurements. For example, the method described above for determining a fractal dimension from successive erosion and dilation operations has two shortcomings: it is slow and has an orientational bias because of the anisotropy of the operations. The EDM offers a simple way to obtain the same information (Russ, 1988) as will be discussed in detail in Chapter 9.

Because the distance map encodes each pixel with the straight line distance to the nearest background point, it can also be used to measure the distance of many points or features from irregular boundaries. In the example shown in Figure 68 the image is thresholded to define the boundary lines (which might represent grain boundaries, cell membranes, etc.) and points (particles, organelles, etc.). The image of the thresholded features is applied as a mask to the Euclidean distance map of the interior so that all pixels in the features have the distance values. Measuring the brightness of the features gives the distance of each feature from the boundary.

Figure 67. The ultimate points for the touching features from Figure 64.
EDM values can also be combined with the skeleton of features or of the background, as discussed in the next section.

**Skeletonization**

Erosion can be performed with special rules that remove pixels, except when doing so would cause a separation of one region into two. The rule for this is to examine the touching neighbors; if they do not form a continuous group, then the central pixel cannot be removed (Pavlidis, 1980; Nevatia and Babu, 1980; Davidson, 1991; Lan et al., 1992; Ritter and Wilson, 1996). The definition of this condition is dependent on whether four- or eight-connectedness is used. In either case, the selected patterns can be used in a fate table to conduct the erosion (Russ, 1984). The more common convention is that features, and thus skeletons, are eight-connected while background is four-connected, and that is the convention used in the following examples.

Skeletonization by erosion is an iterative procedure, and the number of iterations required is proportional to the largest dimension of any feature in the image. An alternative method for constructing the skeleton uses the Euclidean distance map. The ridge of locally brightest values in the

Figure 68. Measurement of distance from a boundary:

(a) example image;  
(b) thresholded interior region;  
(c) thresholded features;  
(d) Euclidean distance map of the interior (color coded);  
(e) distance value assigned to features;  
(f) histogram of distances for features.
EDM contains those points that are equidistant from at least two points on the boundaries of the feature. This ridge constitutes the medial axis transform (MAT). As for the UEPs, the MAT is precisely defined for a continuous image but only approximately defined for an image composed of finite pixels (Mott-Smith, 1970).

In most cases, the MAT corresponds rather closely to the skeleton obtained by sequential erosion. Since it is less directionally sensitive than any erosion pattern and because of the pixel limitations in representing a line, it may differ slightly in some cases. The uses of the MAT are the same as the skeleton, and in many cases, the MAT procedure is used but the result is still described as a skeleton.

Figure 69 shows several features with their (eight-connected) skeletons. The skeleton is a powerful shape factor for feature recognition, containing both topological and metric information. The topological values include the number of end points, the number of nodes where branches meet, and the number of internal holes in the feature. The metric values are the mean length of branches (both those internal to the feature and those having a free end) and the angles of the branches. These parameters seem to correspond closely to what human observers see as the significant characteristics of features. Figure 70 shows the nomenclature used.

The numbers of each type of feature are related by Euler’s equation:

\[ \text{# Loops} = \text{# Branches} - \text{# Ends} - \text{# Nodes} + 1 \]  

A few specific cases appear to violate this basic rule of topology, requiring careful interpretation of the digitized skeleton. Figure 71 shows two of them. The ring skeletonizes to a single circular branch that has one loop, a single branch, and no apparent node, however, the rules of topology
require that there be a “virtual” node someplace on the ring where the two ends of the linear branch are joined. Likewise, the symmetrical circle figure skeletonizes to a single point, which, having fewer than two neighbors, would be classified as an end. In reality, this point represents a short branch with two ends. Special rules can correctly handle these special cases.

Locating the nodes and end points in a skeleton is simply a matter of counting neighbors. Points along the skeleton branches have exactly two neighbors. End points have a single neighbor, while nodes have more than two. The topology of features is an instantly recognizable shape descriptor that can be determined quickly from the feature skeleton. For example, in Figure 72 the number of points in each star is something that humans identify easily as a defining shape parameter. Counting the number of skeleton pixels that have just one neighbor allows labeling them with this topological property. Similarly, an easy distinction between the letters A, B, and C is the number of loops (1, 2, and 0, respectively). As topological properties, these do not depend on size, position, or any distortion of the letters (for example by the use of different fonts).

Segment lengths are important measures of feature size, as will be discussed in Chapter 9. These can also be determined by counting, keeping track of the number of pixel pairs that are diagonally or orthogonally connected, or by fitting smoothed curves through the points and to measure the length, which gives more accurate results. Counting the number of nodes, ends, loops, and branches defines the topology of features. These topological events simplify the original image and assist in characterizing structure, as illustrated in Figure 73.

Skeletons are very useful for dealing with images of crossed fibers. Figure 74 shows a diagrammatic example in which several fibers cross each other. In a few situations, it is necessary to actually follow individual fibers in such a tangle. This can be done (generally using rather specialized software and some prior knowledge about the nature of the fibers) by skeletonizing the image. The regions around each nodes where the skeletons cross are then examined, and the branches identified that represent the continuation of a single fiber (Talbot et al., 2000). The criteria are typically that the local direction change be small, and perhaps that the width, color or density of the fiber be consistent. When fibers cross at a shallow angle, the skeleton often shows two nodes with a segment that belongs to both fibers. Images of straight fibers are much easier to disentangle than curved ones.
A much simpler result is possible if the required information is just the total number and average length of the fibers. Regardless of the number of nodes or fiber crossings, the number of fibers is just half the number of end points, which can be counted directly (with a small error introduced by the probability that an end of one fiber will lie on a second fiber). Measuring the total length of the skeleton (with a correction for the end points as discussed below) and dividing by the number gives the average value. In Figure 74, there are 12 ends, thus 6 fibers, and a total of 40.33 inches of skeleton length, for an average length of 6.72 inches.

In other situations, such as the example shown in Figure 75, it may be useful to separate the branches of the skeleton for individual measurements of parameters such as length or orientation angle. Removing the exact node pixels is not sufficient to accomplish this, because the remaining branches may still be connected. This arises from the nature of eight-connected logic. Figure 76 shows an enlargement of a portion of the skeleton network from Figure 75 in which the node points for topological counting and near-node points that must also be removed to separate the branches are color coded for illustration. This technique is particularly appropriate for branched structures such as the roots of plants, provided that they can be spread out to produce a two-dimensional image. A stereological method is also used for measuring the total length of three-dimensional structures from projections discussed in Chapter 8.

Just as the skeleton of features may be determined in an image, it is also possible to skeletonize the background. This is often called the "skiz" of the features. Figure 77 shows an example.
Consisting of points equidistant from feature boundaries, it effectively divides the image into regions of influence around each feature (Serra, 1982). It may be desirable to eliminate from the skiz those lines that are equidistant from two portions of the boundary of the same feature. This elimination is easily accomplished, since branches have an end; other lines in the skiz are continuous and have no ends except at the image boundaries. Pruning branches from a skeleton (or skiz) simply requires starting at each end point (points with a single neighbor) and eliminating touching pixels until a node (a point with more than two neighbors) is reached.

Boundary lines and thickening

Another use for skeletonization is to thin down boundaries that may appear broad or of variable thickness in images. This phenomenon is particularly common in light microscope images of metals whose grain boundaries are revealed by chemical etching. Such etching preferentially attacks the boundaries, but in order to produce continuous dark lines, it also broadens them. In order to measure the actual size of grains, the adjacency of different phases, or the length of boundary lines, it is preferable to thin the lines by skeletonization.
Figure 78. Skeletonization of grain boundaries:
(a) metallographic image of etched 1040 steel;
(b) thresholded image showing boundaries and dark patches of iron carbide (and pearlite);
(c) skeletonized from image b;
(d) pruned from image c;
(e) enlarged to show eight-connected line;
(f) converted to four-connected line;
(g) grains separated by thickened lines;
(h) identification of individual grains.
Figure 78 shows an example. The original polished and etched metal sample has dark and wide grain boundaries, as well as dark patches corresponding to carbides and pearlite. Thresholding the image produces broad lines, which can be skeletonized to reduce them to single-pixel width. Since this is properly a continuous tesselation, it can be cleaned up by pruning all branches with end points.

The resulting lines delineate the grain boundaries, but because they are eight-connected, they do not separate the grains for individual measurement. Converting the lines to four-connected, called thickening, can be accomplished with a dilation that adds pixels only for a few neighbor patterns corresponding to eight-connected corners (or the skeleton could have been produced using four-connected rules to begin with). The resulting lines separate the grains, which can be identified and measured as shown.

Figure 79 shows how this approach can be used to simplify an image and isolate the basic structure for measurement. The original image is a light micrograph of cells. It might be used to measure the variation in cell size with the distance from the two stomata (openings). This process is greatly simplified by reducing the cell walls to single lines. Leveling the background brightness of the original image and then thresholding leaves boundary lines of variable width. Skeletonizing them produces a network of single-pixel-wide lines that delineate the basic cell arrangement.

Unfortunately, the grain boundary tesselation produced by simple thresholding and skeletonization is incomplete in many cases. Some of the boundaries may fail to etch because the crystallographic mismatch across the boundary is small or the concentration of defects or impurities is low. The result is a tesselation with some missing lines, which would bias subsequent analysis. Figure 80 shows one of the simplest approaches to dealing with this situation. Skeletonizing the incomplete network is used to identify the end points (points with a single neighbor). It is reasoned that these points should occur in pairs, so each is dilated by some arbitrarily selected distance which, it is hoped, will span half of the gap in the network.

The resulting dilated circles are OrRed with the original network and the result is again skeletonized. Wherever the dilation has caused the circles to touch, the result is a line segment that joins the corresponding end points. This method is imperfect, however. Some of the points may be too far apart for the circles to touch, while in other places, the circles may obscure details by touching several existing lines, oversimplifying the resulting network. It is not easy to select an appropriate dilation radius, because the gaps are not all the same size (and not all of the grains are either). In addition, unmatched ends, or points due to dust or particulates within the grains, can cause difficulties.

Figure 79. Light microscope image of cells in plant tissue:
(a) original;
(b) thresholded;
(c) skeleton superimposed on original (image courtesy Data Translations, Inc.)
Other methods are also available. A computationally intensive approach locates all of the end points and uses a relaxation method to pair them up, so that line direction is maintained, lines are not allowed to cross, and closer points are matched first. This method suffers some of the same problems as dilation if unmatched end points or noise are present, but at least it deals well with gaps of different sizes. A third approach, the use of watershed segmentation based on the EDM, is perhaps the most efficient and reasonably accurate method. As shown in Figure 81, it correctly draws in most of the missing lines, but erroneously segments grains with concave shapes (which are fortunately rare in real microstructures).

Combining skeleton and EDM

The skeleton and the EDM are both important measurement tools for images, and by combining them in various ways it is possible to efficiently extract quite a variety of numeric values to quantify image data. A few examples will illustrate the variety of techniques available.

Figure 80. Dilation method for completing grain boundary tessellation:
(a) incomplete network;
(b) dilation of end point by arbitrary radius, shown as circles overlaid on the original;
(c) re-skeletonization of network, showing typical errors such as removal of small grains (1), large gaps still not joined (2), and dangling single ends (3).

Figure 81. Watershed segmentation applied to the same image as Figure 80:
(a) the image is inverted to deal with the grains rather than the boundaries;
(b) watershed lines are drawn in, connecting most of the broken boundaries;
(c) in the re-inverted result typical errors appear such as large gaps not joined (1) and false segmentation of irregular shaped grains (2).
The Euclidean distance map discussed previously provides values that measure the distance of every pixel from the background. For features of irregular shape or width, the pixels along the center line correspond to the centers of inscribed circles, and their EDM values can be used to measure the width and its variation. The skeleton provides a way to sample these pixels, for example by using the skeleton as a mask and then examining the histogram as shown in Figure 82.

The skeleton provides a basic tool for measuring the length of such irregular features, but in general is too short. The EDM values for the pixels at the end points of the skeleton give the radii of the inscribed circles at the ends. Adding these values to the skeleton length corrects for the shortness of the skeleton and provides a more accurate measure of the length of the irregular feature.

The skeleton of the background (the skiz) can be combined with the EDM of the background to determine the minimum separation distance between features. Minimum EDM values along the pruned skiz correspond to the centers of circles that touch two features, and twice those values correspond to the separation distances.

The example in Figure 83 shows a diagram of a neuron with neurites that branch. Thresholding the central cell body, inverting the image, and creating the EDM produces a measurement of the

Figure 82. An irregular feature shown with its skeleton superimposed on the EDM (using pseudo-color), and the histogram of the EDM values selected by the skeleton.

Figure 83. Relating distance to length: (a) diagram of a neural cell; (b) skeleton segments of the neurites, color coded according to distance from the cell body (values obtained from the EDM as described in the text); (c) plot of distance vs. length.
distance of points from the cell body. The skeleton of the neurites can be separated into its com-
ponent branches by removing the nodes, and the resulting segments applied as a mask to the 
EDM. This assigns numeric values to the pixels in the branches that determine the distance from the
 cell body. It may be desirable to use either the minimum or the mean value as an effective dis-
tance measurement. Plotting the skeleton length for each branch against the EDM values shows that
the lengths are correlated with distance.

Measuring the distance of each feature in an image from the nearest point on the skiz (using the
method shown in Figure 68) provides a measure of clustering in images. Many other combinations
can be devised to solve measurement problems in images that combine distance and topological
information.