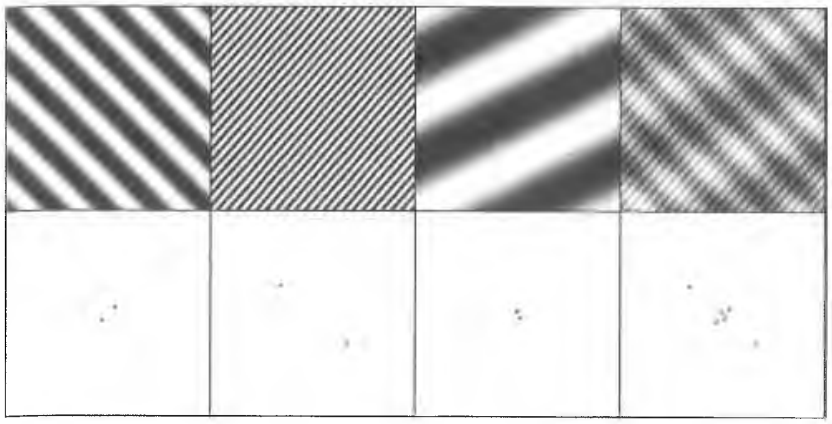
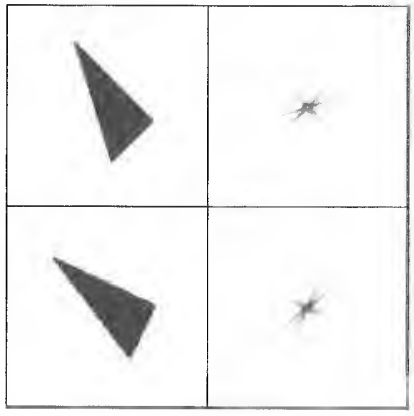


Figure 5. *Three sinusoidal patterns, their frequency transforms, and their sum.*

The power spectrum of the low frequency (greatest spacing) sinusoid has a point close to the origin, and the higher frequency sinusoids have points farther away, and in directions that identify the orientation of the lines. The superposition of the three sinusoids produces an image whose frequency transform is simply the sum of the three individual transforms. This principle of additivity will be important for much of the discussion in the following paragraphs. Subtracting the information from a location in the frequency transform is equivalent to removing the corresponding information from every part of the spatial-domain image. **Figure 6** shows two images with the same shape in different orientations. The frequency transforms rotate with the feature.

Figure 7 shows a two-dimensional step consisting of a rectangle. The two-dimensional frequency transform of this image produces the same series of diminishing peaks in the x and y axis directions as the one-dimensional step function. The darkness of each point in the power spectrum display represents the log of the square of the amplitude of the corresponding frequency. Limiting the reconstruction to only the central (low-frequency) terms produces the reconstructions shown. Just as for the one-dimensional case, this limits the sharpness of the edge of the step and produces some ringing (oscillations near the edge) in the overall shape. The line profiles through the image show the same shape as previously discussed for the one-dimensional case.

Figure 6. *Rotation of a spatial domain image (left), and the corresponding rotation of the frequency transform (right).*



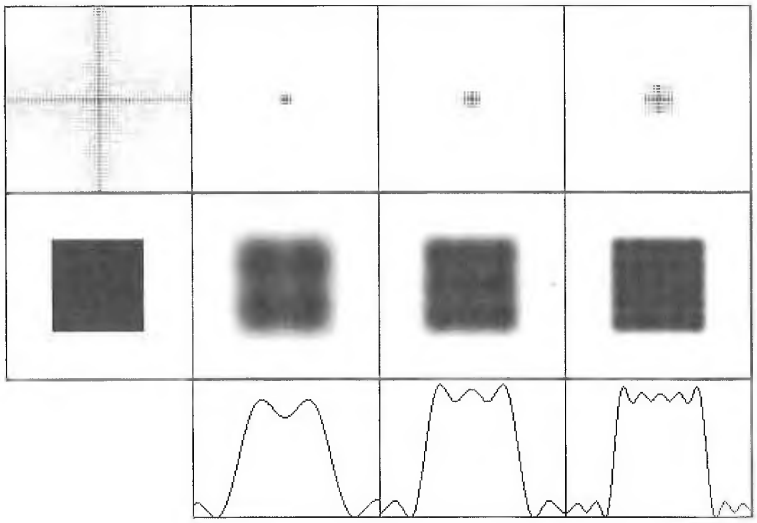


Figure 7. A two-dimensional step function and its frequency transform (left), and reconstructions with different numbers of terms (shown as a portion of the frequency transform). Bottom row shows horizontal line profiles through the center of the reconstructed spatial image.

Frequencies and orientations

It is helpful to develop a little familiarity with the power spectrum display of the frequency-space transform of the image using simple images. In **Figure 8a**, the lines can be represented by a single peak because their brightness profile is perfectly sinusoidal; thus only a single frequency is present. If the line profile is different, more terms are needed to represent the shape and consequently more peaks appear in the power spectrum. **Figure 8b** shows an example in which the frequency transform consists of a series of peaks at multiples of the lowest frequency, in the same orientation (perpendicular to the line angle).

In **Figure 8c**, the lines have the aliasing common in computer displays (and in halftone printing technology), in which the lines at a shallow angle on the display are constructed from a series of steps corresponding to the rows of display pixels. This further complicates the frequency transform, which now has additional peaks representing the horizontal and vertical steps in the image that correspond to the aliasing, in addition to the main line of peaks seen in the figure.

It is possible to select only the peaks along the main row and eliminate the others with a mask or filter, as will be discussed in the next section. After all, the frequency-domain image can be modified just like any other image. If this is done and only the peaks in the main row are used for the inverse transformation (back to the spatial domain), the aliasing of the lines is removed. In fact, that is how the images in **Figures 8a** and **8b** were produced. This will lead naturally to the subject of filtering (discussed in a later section): removing unwanted information from spatial-domain images by operating on the frequency transform. For example, it offers one practical technique to remove aliasing from lines and make them visually smooth.

Measuring images in the frequency domain

Orientation and spacing

The idealized examples shown in the preceding tutorial show that any periodic structure in the original spatial-domain image will be represented by a peak in the power spectrum image at a radius corresponding to the spacing and a direction corresponding to the orientation. In a real image, which typically consists of mostly non-periodic information, any such peaks will be superimposed on a broad and sometimes noisy background, although, finding the peaks is generally much

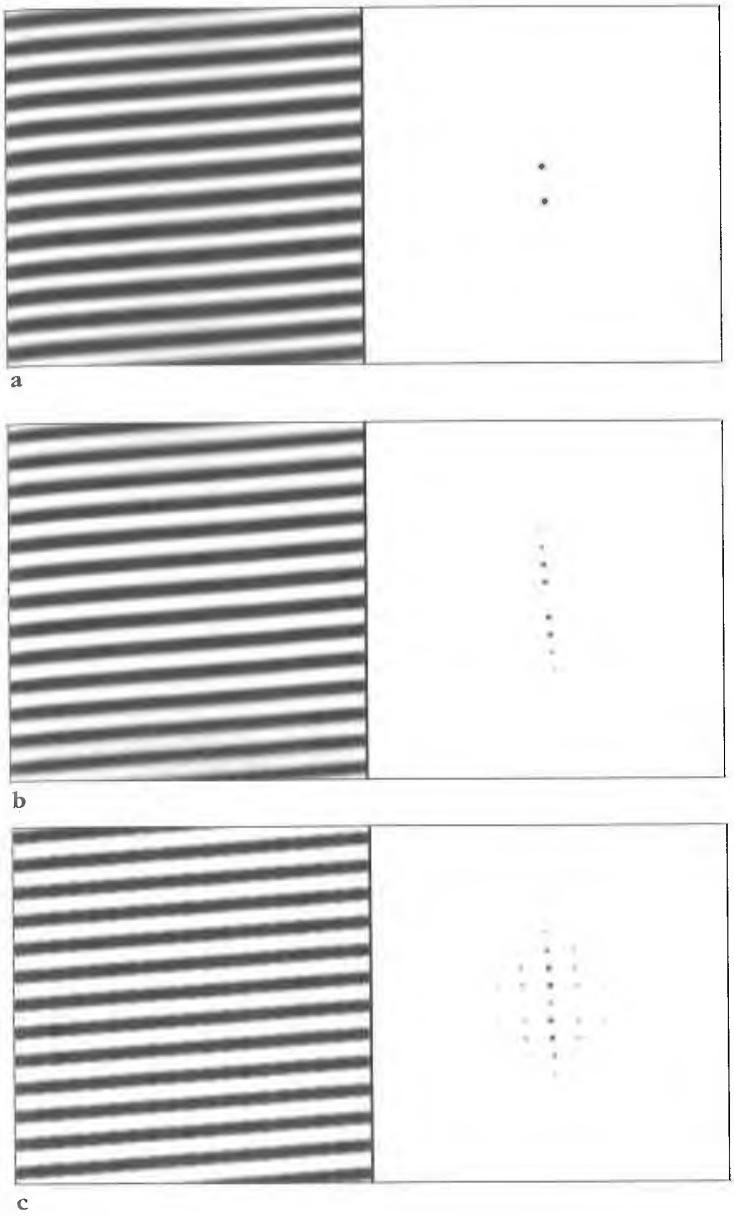


Figure 8. A set of lines (left) and their frequency transform (right): *(a) sinusoidal lines; (b) with a non-sinusoidal brightness profile; (c) the lines from* **b** *with aliasing.*

easier than finding the original periodic structure. Also, measuring the peak locations accurately is much easier and more accurate than trying to extract the same information from the original image, because all of the occurrences are effectively averaged together in the frequency domain.

Figure 9 shows an example of this kind of peak location measurement. The spatial-domain image is a high-resolution transmission electron microscope (TEM) image of the lattice structure in pure silicon. The regular spacing of the bright spots represents the atomic structure of the lattice. Measuring all of the individual spacings of the spots would be very time-consuming and not particularly accurate. The frequency-domain representation of this image shows the periodicity clearly. The series of peaks indicates that the variation of brightness is not a simple sine wave, but contains many higher harmonics. The first-order peak gives the basic atomic spacing (and orientation),

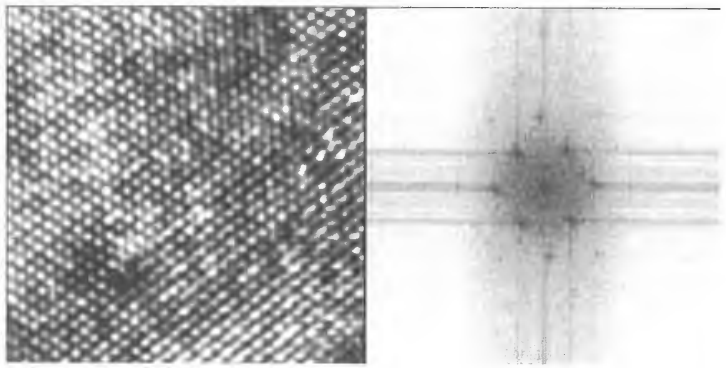


Figure 9. *High resolution TEM image of atomic lattice in silicon (left), with the frequency transform (right) (image courtesy Sopa Chevacharoenkul, Microelectronics Center of North Carolina).*

which can be measured by interpolating the peak position to a fraction of a pixel width, corresponding to an accuracy for the atom spacing of a few parts in 10,000. The spacing of the features that produce the point in the power spectrum is simply the width of the image (e.g., 256 pixels in the example, times whatever calibration applies) divided by the distance from the origin to the center of the peak in the power spectrum.

To the electron microscopist, the power spectrum image of the frequency-domain transform looks just like an electron diffraction pattern, which in fact it is. The use of microscope optics to form the diffraction pattern is an analog method of computing the frequency-domain representation. This can be done with any image by setting up suitable optics. While it is a fast way to obtain the frequency-domain representation, this method has two serious drawbacks for use in image processing.

First, the phase information is lost when the diffraction pattern is recorded, so it is not possible to reconstruct the spatial-domain image from a photograph of the diffraction pattern. (Recording the entire diffraction pattern including the phase information results in a hologram, from which the original image *can* be reconstructed.) It is possible to perform the reconstruction from the diffraction pattern in the microscope by using suitable lenses (indeed, that is how the microscope functions), so in principle it is possible to insert the various masks and filters discussed below. Making these masks and filters is difficult and exacting work, however, which must usually be performed individually for each image to be enhanced. Consequently, it is much easier (and more controllable) to use a computer to perform the transform and to apply any desired masks.

It is also easier to perform measurements on the frequency-domain representation using the computer. Locating the centers of peaks by curve fitting would require recording the diffraction pattern (typically with film, which may introduce nonlinearities or saturation over the extremely wide dynamic range of many patterns), followed by digitization to obtain numerical values. Considering the speed with which a spatial-domain image can be recorded, the frequency transform calculated, and interactive or automatic measurement performed, the computer is generally the tool of choice. This analysis is made easier by the ability to manipulate the display contrast so that both brighter and dimmer spots can be seen, and to use image processing tools such as background leveling and a top hat filter to isolate the peaks of interest.

When spots from a periodic structure are superimposed on a general background, the total power in the spots expressed as a fraction of the total power in the entire frequency transform gives a useful quantitative measure of the degree of periodicity in the structure. This may also be used to compare different periodicities (different spacings or orientations) by comparing summations of values in the power spectrum. For electron diffraction patterns, this is a function of the atomic density of various planes and the atomic scattering cross sections.

Although the display of the power spectrum corresponds to a diffraction pattern and is the most familiar presentation of frequency-space information, it must not be forgotten that the phase information is also needed to reconstruct the original image. **Figure 10** shows a test image, consisting of a regular pattern of spots, and its corresponding power spectrum and phase values. If the phase information is erased (all phases set to zero), the reconstruction (**Figure 11**) shows some of the same periodicity, but the objects are no longer recognizable. The various sine waves have been shifted in phase, so that the feature boundaries are not reconstructed.

The assumption that the image is one repetition of an endless sequence is also important. Most real images do not have perfectly matching left and right or top and bottom edges. This produces a large step function at the edge, which is more apparent if the image is shifted by an arbitrary offset (**Figure 12**). As noted previously, this does not alter the power spectrum image, although the phase image is shifted. The discontinuity requires high-frequency terms to fit, and since the edges are precisely horizontal and vertical, the power spectrum display shows a central cross superimposed on the rest of the data, which is visible in **Figure 10**. For the test pattern, the result of eliminating these lines from the original frequency transform and then retransforming is shown in **Figure 13**. The central portion of the image is unaffected, but at the edges the discontinuity is no longer sharp because the many frequencies needed to create the step functions at the edges are missing. The pattern from each side has been reproduced on the other side of the boundary, superimposed on the correct data.

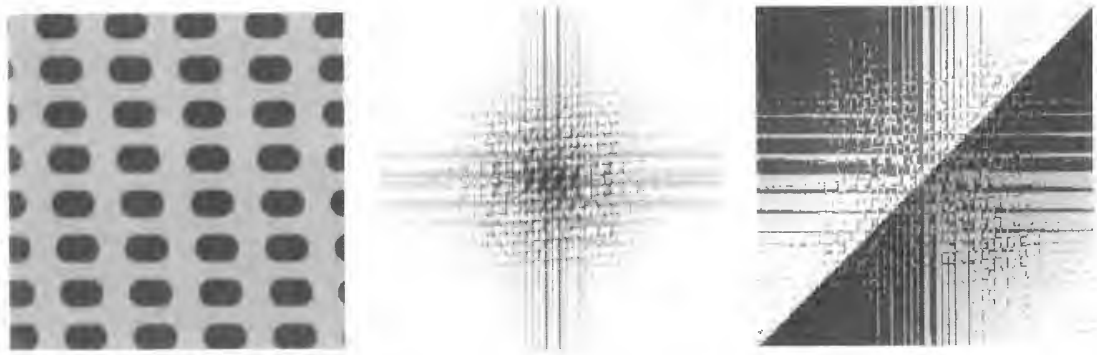


Figure 10. *Test image consisting of a regular pattern* ***(a)****, with its frequency transform power spectrum* ***(b)*** *and phase values* ***(c)****.*

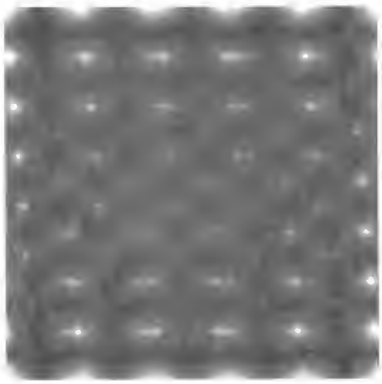


Figure 11. *Retransformation of* ***Figure 10*** *with all phase information set to zero.*

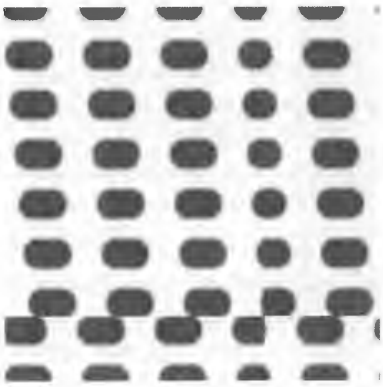


Figure 12. *The test image of* ***Figure 10*** *with an arbitrary spatial shift, showing the discontinuities at the image boundaries*

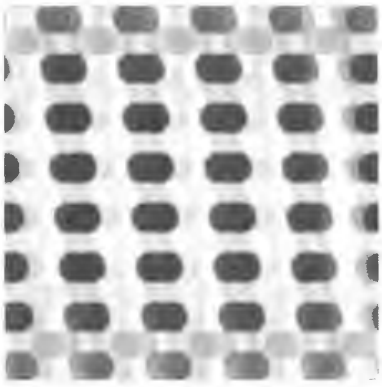


Figure 13. *Retransformation of* ***Figure 10*** *with the central cross (horizontal and vertical lines) reduced to zero magnitude, so that the left and right edges of the image, and the top and bottom edges, are forced to match.*

Preferred orientation

Figure 14 shows another example of a periodic structure, only much less perfect and larger in scale than the lattice images above. The specimen is a thin film of magnetic material viewed in polarized light. The stripes are oppositely oriented magnetic domains in the material that are used to store information in the film. The frequency transform of this image clearly shows the width and spacing of the domains. Instead of a single peak, some arcs show the variation in orientation of the stripes, which is evident in the original image but difficult to quantify.

The length of the arcs and the variation of brightness (power) with angle along them can be easily measured to characterize the preferred orientation in the structure. Even for structures that are not perfectly periodic, the integrated power as a function of angle can be used to measure the preferred orientation. This is identical to the results of autocorrelation operations carried out in the spatial domain, in which a binary image is shifted and combined with itself in all possible displacements to obtain a matrix of fractional values, but it is much faster to perform with the frequency-domain representation. Also, this makes it easier to deal with grey scale values.

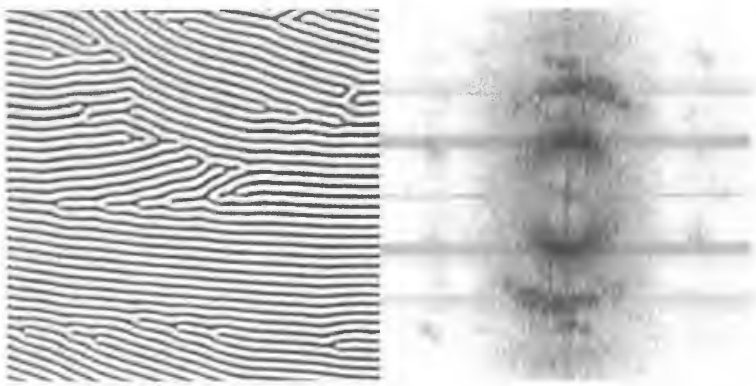


Figure 14. *Polarized light image of magnetic domains in thin film material (left), with the frequency transform (right).*

Reconstructing periodic structures that are not perfectly aligned can be performed by selecting the entire arc in the frequency transform. **Figure 15** illustrates this with a virus particle. The TEM image hints at the internal helical structure, but does not show it clearly. In the frequency transform, the periodic spacing and the variation in direction is evident. The spacing can be measured (2.41 nm) and the helix angle determined from the length of the arc. Retransforming only these arcs shows the periodicity, but this is not limited spatially to the virus particle. Using the spatial-domain image as a mask (as discussed in Chapter 7) makes the internal helical pattern evident.

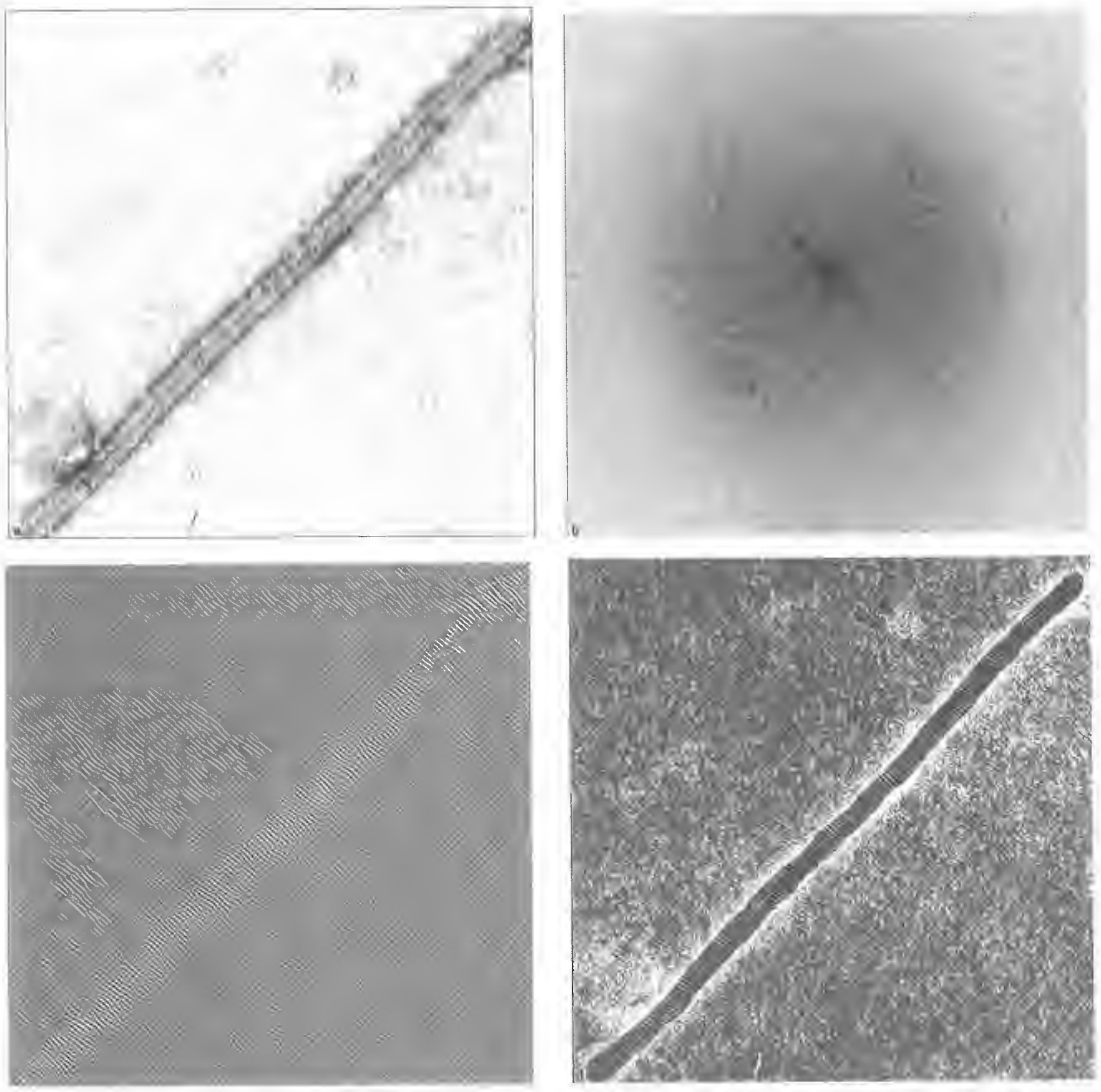


Figure 15. TEM image of a virus *(courtesy Dr. R. L. Grayson, Virginia Polytechnic Institute, Blacksburg, VA):*
(a) *original image, in which the internal helical structure is difficult to discern;*
(b) *frequency transform of image* ***a****, in which the regular repeating structure of the virus and its angular variation in orientation is evident;*
(c) *retransformation of just the peaks in the frequency transform, in which the periodic lines are not limited to the virus;*
(d) *using the virus particle as a mask, the helical pattern becomes evident.*

One particular type of preferred orientation in images, which arises not from the specimen but rather from the imaging system itself, is astigmatism. This is a particular problem with electron microscopes because of the operating principles of electromagnetic lenses. Even skilled operators devote considerable time to making adjustments to minimize astigmatism, and it is often very difficult to recognize it in images in order to correct it. Astigmatism results in the defocusing of the image and a consequent loss of sharpness in one direction, sometimes with an improvement in the perpendicular direction. This becomes immediately evident in the frequency transform, since the decrease in brightness or power falls off radially (at higher frequencies) and the asymmetry can be noted.

Figure 16 shows an example. The specimen is a cross section with three layers. The bottom is crystalline silicon, above which is a layer of amorphous (non-crystalline) silicon, followed by a layer of glue used to mount the sample for thinning and microscopy. The glue is difficult to distinguish by eye from the amorphous silicon. Frequency transforms for the three regions are shown. The regular structure in the pattern from the crystalline silicon gives the expected diffraction pattern. While the two regions above do not show individual peaks from periodic structures, they are not the same. The amorphous silicon has short-range order in the atomic spacings based on strong covalent bonding that is not visible to the human observer because of its chaotic overall pattern. This shows up in the frequency transform as a white cross in the dark ring, indicating that in the 45-degree directions there is a characteristic distance and direction to the next atom. This pattern is absent in the glue region, where there is no such structure.

In both regions, the dark circular pattern from the amorphous structure is not a perfect circle, but an ellipse. This indicates astigmatism. Adjusting the microscope optics to produce a uniform circle will correct the astigmatism and provide uniform resolution in all directions in the original image. It is much easier to observe the effects of small changes in the frequency-space display than in the spatial-domain image.

The frequency transform of an image can be used to optimize focus and astigmatism. When an image is in focus, the high-frequency information is maximized in order to sharply define the edges.

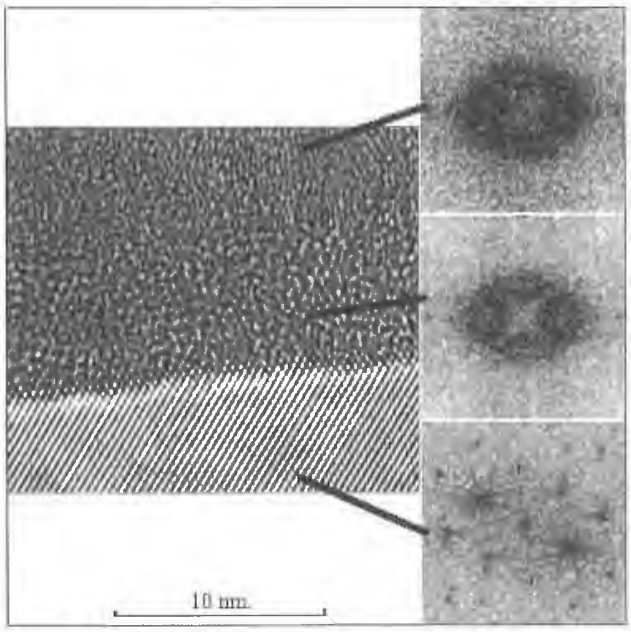


Figure 16. *TEM image of cross section of crystalline and amorphous silicon, and glue, with frequency transforms of each region shown at right (image courtesy Sopa Chevacharoenkul, Microelectronics Center of North Carolina).*

This provides a convenient test for sharpest focus. **Figure 17** shows a light microscope image that is in focus. The line profiles of the power spectrum in both the vertical and horizontal directions show a more gradual drop-off at high frequencies than **Figure 18**, which is the same image out of focus. When astigmatism is present (**Figure 19**), the power spectrum is asymmetric, as shown by the profiles.



Figure 17. *Out-of-focus image with its power spectrum and horizontal and vertical line profiles, showing presence of high-frequency information as compared with* ***Figure 18****. Red arrows indicate actual resolution limit as discussed in text.*

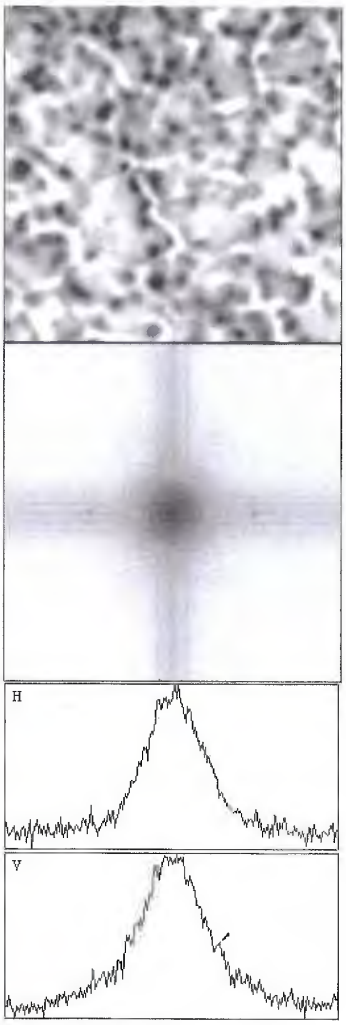


Figure 18. *Out-of-focus image with its power spectrum and horizontal and vertical line profiles, showing loss of high frequency information as compared to* ***Figure 17****.*

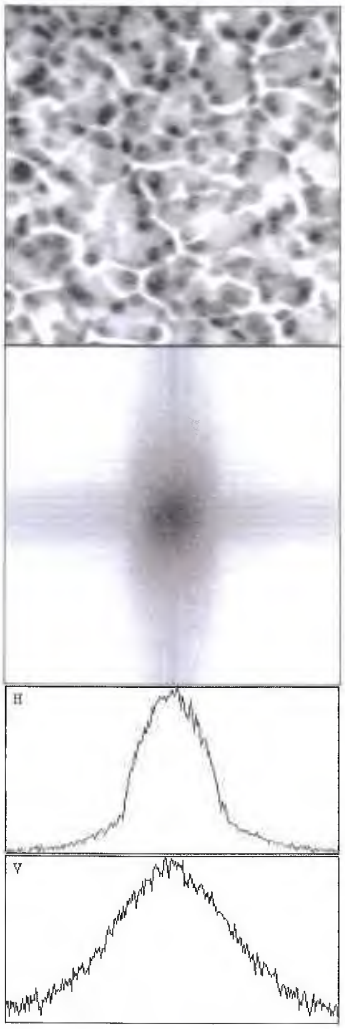


Figure 19. *Astigmatic image produced by misaligning lens, with its power spectrum and horizontal and vertical line profiles showing different high-frequency components.*

Profiles of the power spectrum also provide a convenient tool for measuring the actual resolution of images, to characterize various types of cameras and acquisition devices as discussed in Chapter 1. Most image acquisition procedures store an image that has somewhat more pixels than the actual resolution. As shown by the red arrows in **Figure 17**, the power spectrum shows a definite break in slope. This indicates the frequency at which the real information in the image ends; only noise is present at higher frequencies (smaller spacings).

Figure 20 shows a test pattern of radial lines. Due to the finite spacing of detectors in the video camera used, as well as limitations in electronics bandwidth that eliminate the very high frequencies required to resolve small details, these lines are incompletely resolved where they are close together. **Figure 21** shows the two-dimensional Fourier transform power spectrum of this image. The power spectrum is plotted isometrically in **Figure 22** to emphasize the drop-off in magnitude, which is different in the horizontal and vertical directions. As is common in video, the resolution along each scan line is poorer than the vertical resolution. **Figure 23** shows the complete set of data using color. One image shows the real and imaginary components of the transform

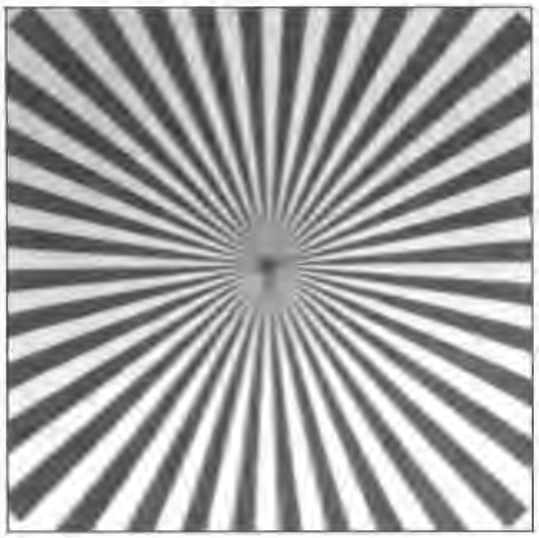
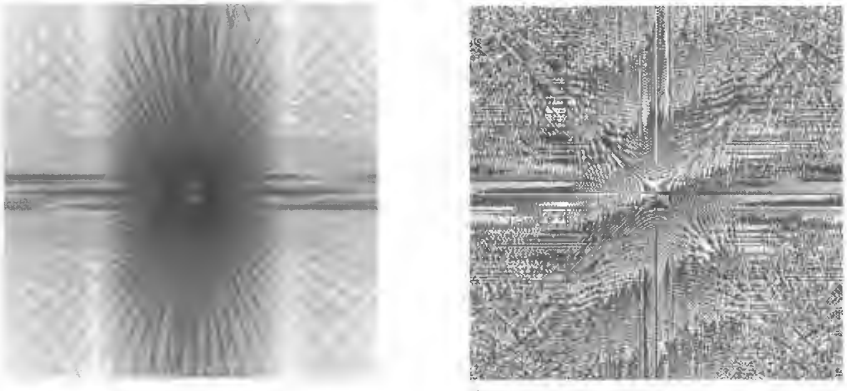


Figure 20. *Test pattern image.*



a b

Figure 21. Fourier transform of the test image in Figure 20:
(a) *power spectrum magnitude;*
(b) *phase (angles from 0 to 180° displayed as grey-scale values from black to white).*

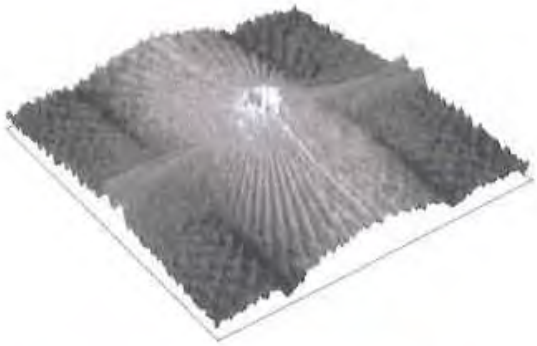
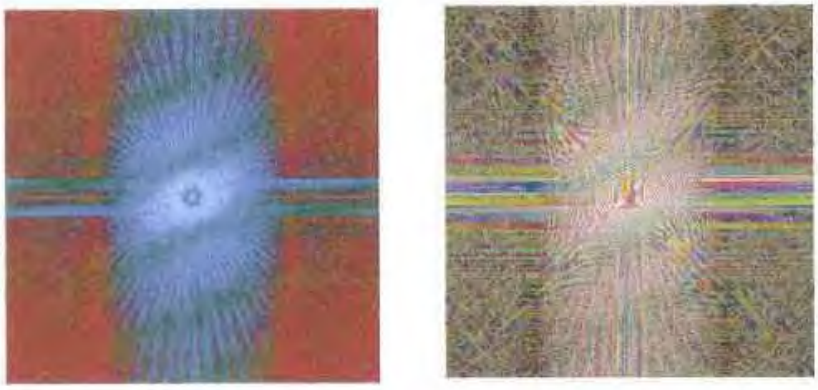


Figure 22. The power spectrum from ***Figure 21*** *presented as an isometric view.*



a b

Figure 23. Color representations of the Fourier transform from Figure 21:
(a) *YUV image in which U = Real, V = Imaginary;*
(b) *RGB image in which Intensity = Magnitude, Hue = Phase.*

and the other the magnitude and phase, encoded in different color channels. Although technically complete, these displays are rarely used because the phase information is confusing to the viewer.

In the power spectrum, it is evident that there is a well-defined boundary, different in the x and y directions, beyond which the magnitude drops abruptly. This corresponds to the image resolution, which is different in the horizontal and vertical directions. In many cases, it is not so obvious where the physical source of resolution limitation lies. It can arise from the finite spacing of detectors in the camera, or various electronic effects in the amplification and digitization process; however, the Fourier transform power spectrum will still show the limit, permitting the resolution of any imaging system to be ascertained.

Texture and fractals

Besides the peaks in the power spectrum resulting from periodic structures that may be present and the ultimate limitation at high frequencies due to finite image resolution, it may seem as though there is only a noisy background containing little useful information. This is far from true. Many images represent the brightness of light scattered from surfaces, or other data such as surface elevation. In these images, the roughness or texture of the surface is revealed and may be measured from the power spectrum.

The concept of a fractal surface dimension will not be explained here in detail, but is discussed in Chapter 13. Surfaces that are fractal have an area that is mathematically undefined. It is greater than the projected area covered by the irregular surface and increases as the measurement scale becomes finer. The fractal dimension may be determined the slope of a line on a log-log plot of measured area vs. the size of the measuring tool. Many naturally occurring surfaces resulting from wear, erosion, agglomeration of particles, or fracture are observed to have this character. It has also been shown that images of these surfaces, whether produced by the scattering of diffuse light or the production of secondary electrons in an scanning electron microscope (SEM), are also fractal. That is, the variation of brightness with position obeys the same mathematical relationship. The fractal dimension is an extremely powerful and compact representation of the surface roughness, which can often be related to the history of the surface and the properties that result.

Measuring surface fractals directly is rarely practical, although it can be done physically by determining the number of molecules of various gases that can adhere to the surface as a function of their size. One common imaging approach is to reduce the dimensionality and measure the fractal dimension of a boundary line produced by intersecting the surface with a sampling plane. This may either be produced by cross sectioning or by polishing down into the surface to produce islands. In either case, the rougher the surface, the more irregular the line. In the case of polishing parallel to the surface to produce islands, the perimeter line also has a fractal dimension (the slope of a log-log plot relating the measured line length and the length of the measurement tool), which is just 1.0 less than that of the surface.

For a fractal surface, the power spectrum shows the superposition of sinusoids of all frequencies and orientations to have a specific shape: the magnitudes of the coefficients in a Fourier transform of a fractal curve decrease exponentially with the log of frequency, while the phases of the terms are randomized. This can be understood qualitatively, since by definition a fractal curve is self-similar and has detail extending to ever-finer scales (or higher frequencies). This also implies that the proportion of amplitudes of higher frequency terms must be self-similar. An exponential curve satisfies this criterion.

Plotting the log of the amplitude (or the power spectrum, which is the square of the amplitude) versus the logarithm of frequency produces a straight line plot, which is easily analyzed. There is a simple relationship between the fractal dimension of a profile and the exponential decrease in magnitude of the terms in a Fourier expansion, as had been predicted by Feder (1989). This correlation makes it practical to use the radial decrease of magnitude in a two-dimensional Fourier-transform image as a measure of roughness and the directional variation of that decrease as a measure of orientation (Mitchell and Bonnell, 1990; Russ, 1990; Russ, 1994).

Figure 24 shows a range image (brightness represents elevation) for a fractal surface, with its Fourier transform power spectrum. Plotting log (amplitude) vs. log (frequency) as shown in **Figure 25** produces a straight line, and a plot of the histogram of the phase values shows a uniform random distribution, which confirms the fractal geometry of the surface. The slope of the plot gives the dimension of the surface (which must lie between 2.0 for a Euclidean surface and 2.999, for one so irregular that it effectively fills three-dimensional space) as Fractal Dimension = (6 + Slope)/2, or about 2.2 for the example shown. This is an isotropic surface produced by shotblasting a metal, so the slope is the same in all directions.

Figure 26 shows another surface, this produced by machining, with tool marks oriented in a specific direction. Similar analysis of this surface shows (**Figure 27**) that while it is still fractal (with average dimension about 2.25), it is not isotropic. Actually, two methods are used to depart from isotropy: one is changing the slope of the Log (amplitude) vs. Log (frequency) plot, and the other its intercept, as a function of direction. Chapter 13 discusses fractal surface geometry in more detail (see also Russ, 1994 and Russ, 2001b).

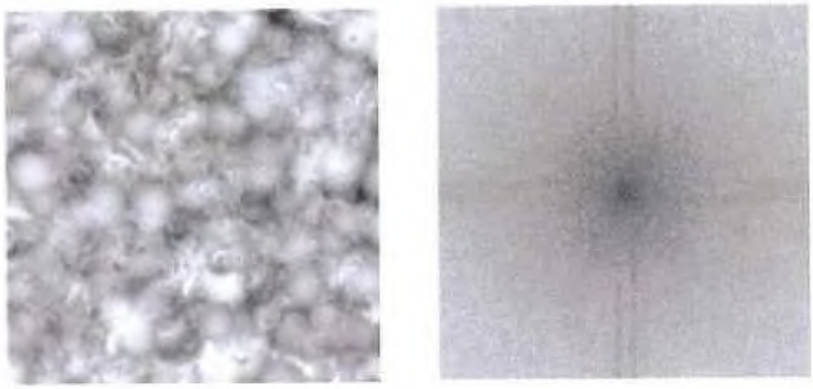


Figure 24. Range image of a shotblasted metal surface, with its Fourier transform power spectrum.

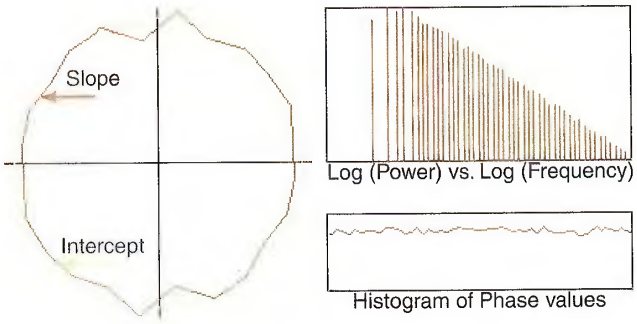


Figure 25. Analysis of the Fourier transform data showing the isotropic fractal nature of the surface in ***Figure 24.***

Measuring the two-dimensional Fourier transform is more efficient than measuring many individual brightness profiles in the original image. It also allows any periodic structures that may be present to be ignored, since these show up as discrete points in the frequency-transform image and can be skipped in determining the overall exponential decrease in the magnitude values. In other words, it is possible to look beyond the periodic structures or noise (e.g., arising from electronic components) in the images and still characterize the underlying chaotic but self-similar nature of the surface.

Filtering images

Isolating periodic noise

It was noted earlier (and shown in **Figure 5** for a simple case) that the frequency transform has a property of separability and additivity. Adding together the transforms of two original images or functions produces the same result as the transform of the sum of the originals. This idea opens the way to using subtraction to remove unwanted parts of images. It is most commonly used to remove periodic noise, which can be introduced by the devices used to record or transmit images. We will see some examples below. If the frequencies associated with the noise can be determined (often possible directly from the Fourier transform), then setting the amplitude of those terms to zero will leave only the desired part of the information. This can then be inverse transformed back to the spatial domain to produce a noise-free image.



a b

Figure 26. *Range image of a machined metal surface, with its Fourier transform power spectrum.*

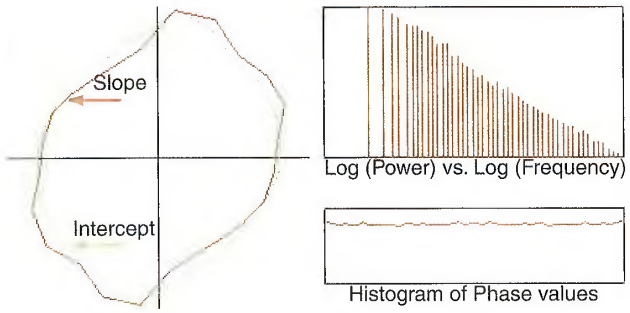


Figure 27. *Analysis of the Fourier transform data showing the anisotropic fractal nature of the surface in* ***Figure 26.***

The same method of removing selected frequencies can be used for a more basic type of filtering as well. **Figure 28** shows an image with its Fourier transform power spectrum. There are no evident "spikes" or noise peaks, just the usual gradual reduction in the amplitude of higher frequencies. Keeping the low frequencies and removing the high frequencies can be accomplished by zeroing the amplitude of all sinusoids above a selected frequency. The red circle in **Figure 28b** was used to perform this "low-pass" filtering operation (i.e., passing or keeping the low frequencies), to produce the result in **Figure 28c**. Conversely, keeping the high frequencies and removing the low frequencies (a "high-pass" filter) produces the result in **Figure 28d**.

Except for the ringing around boundaries (discussed below), the results are exactly the same as those for Gaussian smoothing and Laplacian sharpening, shown in the two preceding chapters. In fact, it can be shown mathematically that the operations are exactly the same whether performed in the frequency domain or in the pixel domain.

In this illustration of filtering, portions of the Fourier-transform image were selected based on frequency, which is why these filters are generally called low-pass and high-pass filters. Usually, selecting arbitrary regions of the frequency domain for reconstruction produces artefacts, unless some care is taken to shape the edges of the filter region to attenuate the data smoothly. This can be seen in the one-dimensional example of the step function in **Figure 1** and the corresponding two-dimensional example of **Figure 7**. If only the first few terms are used then, in addition to not

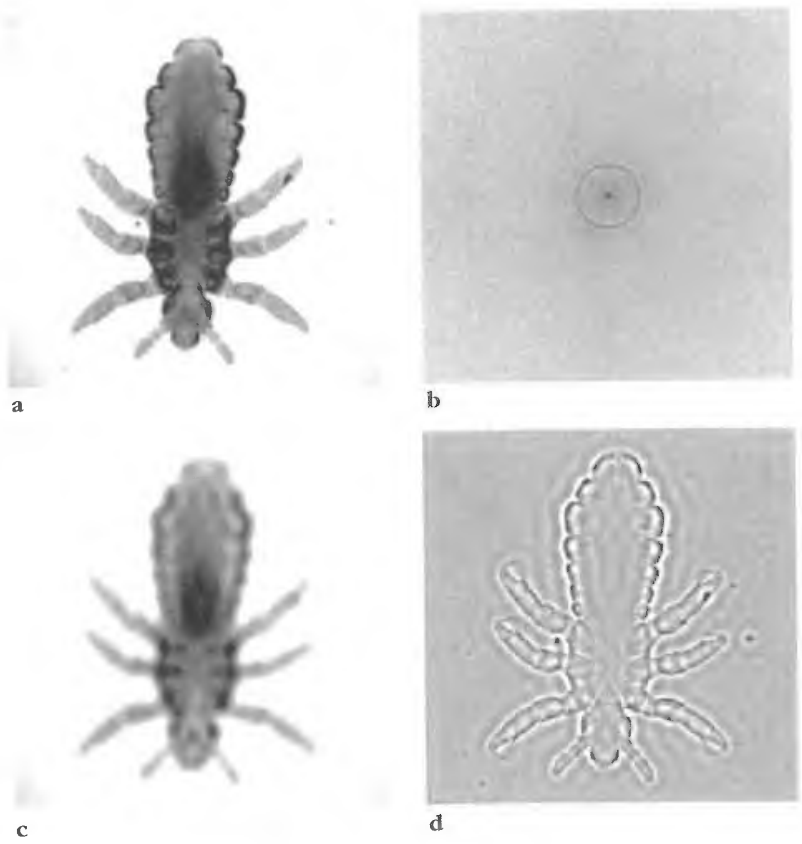


Figure 28. Frequency filtering using the Fourier transform: *(a) original image; (b) power spectrum with a circular frequency cutoff; (c) retransforming just the low frequencies inside the circle (a low-pass filter); (d) retransforming just the high frequencies outside the circle (a high-pass filter).*

modeling the steepness of the step, the reconstruction has oscillations near the edge, which are generally described as ringing.

It is necessary to shape the edge of the filter to prevent ringing at sharp discontinuities. This behavior is well-known in one-dimensional filtering (used in digital signal processing, for example). Several different shapes are commonly used. Over a specified width (usually given in pixels, but of course ultimately specified in terms of frequency or direction), the filter magnitude can be reduced from maximum to minimum using a weighting function. The simplest function is linear interpolation (also called a Parzen window function). Better results can be obtained using a parabolic or cosine function (also called Welch and Hanning window functions, respectively, in this context). The most elaborate filter shapes do not drop to the zero or minimum value, but extend a very long tail beyond the cutoff point. One such shape is a Gaussian. **Figure 29** shows several of these shapes.

Another filter shape often used in these applications is a Butterworth filter, with a magnitude that can be written as

$$H = 1 / \left[1 + C\left(\frac{R}{R_0}\right)^{2n} \right] \tag{8}$$

where R is the distance from the center of the filter (usually the center of the power spectrum image, or zero-frequency point), and R_0 is the nominal filter cutoff value. The constant C is often set equal to 1.0 or to 0.414; the value defines the magnitude of the filter at the point where $R = R_0$ as either 50% or $1/\sqrt{2}$. The integer n is the order of the filter; its most common values are 1 or 2.

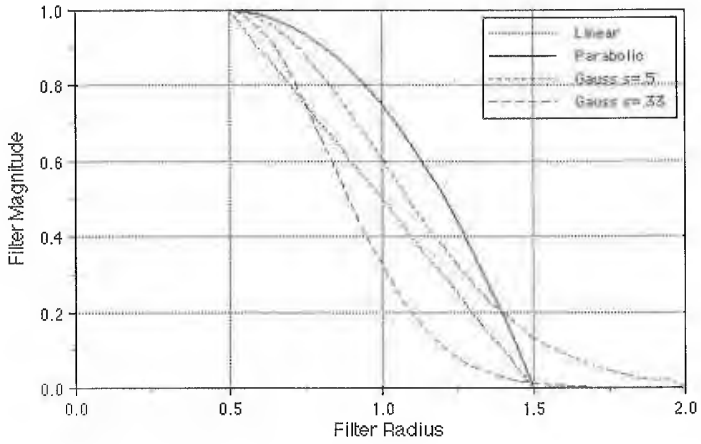


Figure 29. *Some common filter edge profiles.*

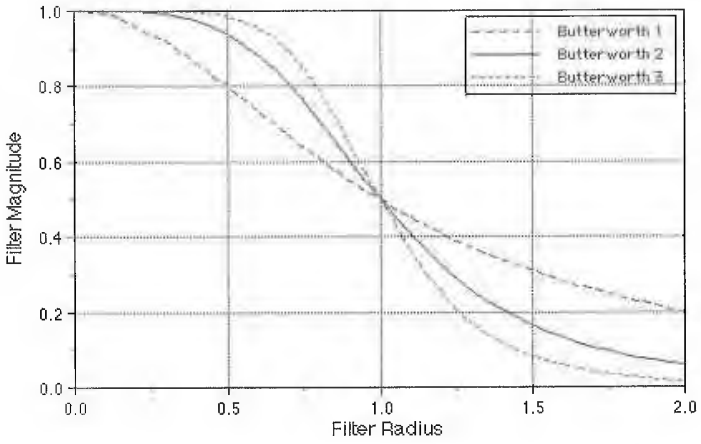
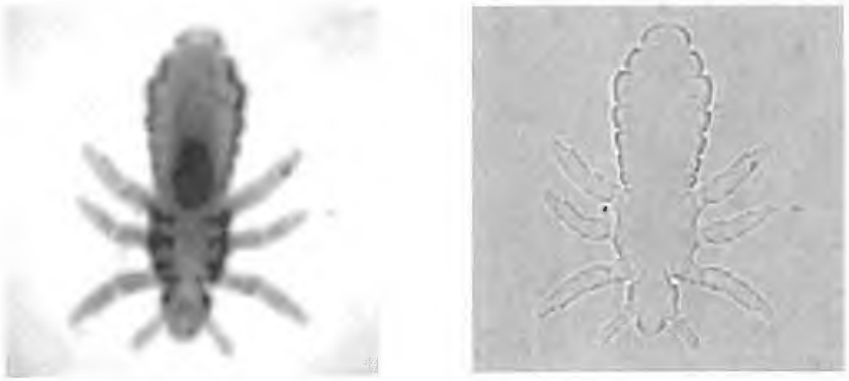


Figure 30. *Shapes for Butterworth filter profiles of order 1, 2, and 3.*

Figure 30 shows comparison profiles of several Butterworth low-pass filters (ones that attenuate high frequencies). The converse shape having negative values of n passes high frequencies and attenuates low ones. **Figure 31** shows the effect that shaping the frequency cutoff has on the quality of the retransformed images from **Figure 28**.

To illustrate the effects of these filters on ringing at edges, **Figure 32** shows a simple test shape and its two-dimensional FFT power spectrum image. The orientation of principal terms perpendicular to the major edges in the spatial-domain image is evident. Performing a reconstruction using a simple aperture with a radius equal to 25 pixels (called an ideal filter) produces the result shown in **Figure 33a**. The oscillations in brightness near the edges are quite visible.

Ringing can be reduced by shaping the edge of the filter, as discussed above. The magnitudes of frequency terms near the cutoff value are multiplied by factors less than one, whose values are based on a simple function. If a cosine function is used, which varies from 1 to 0 over a total width of 6 pixels, the result is improved (**Figure 33b**). In this example, the original 25-pixel radius used for the ideal filter (the sharp cutoff) is the point at which the magnitude of the weighting factor drops to 50%. The weights drop smoothly from 1.0 at a radius of 22 pixels to 0.0 at a radius of 28 pixels.



a b

Figure 31. Shaping the frequency cutoff with a Butterworth filter:
(a) *low-pass filter (compare with* ***Figure 28c****);*
(b) *high-pass filter (compare with* ***Figure 28d****).*

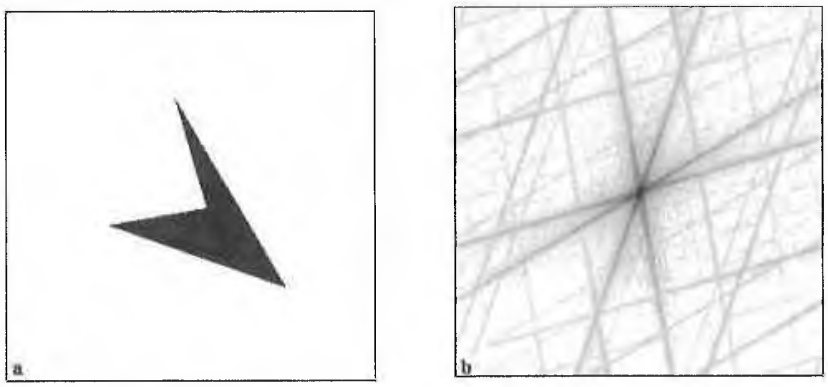


Figure 32. Test shape with its frequency transform power spectrum.

Increasing the distance over which the transition takes place further reduces the ringing, as shown in **Figure 33c**. Here, the 50% point is still at 25 pixels but the range is from 15 to 35 pixels. Note that the improvement is not achieved simply by increasing the high-frequency limit, which would improve the sharpness of the feature edges but would not by itself reduce the ringing. **Figure 33d** shows the same reconstruction using a second-degree Butterworth filter shape whose 50% point is set at 25 pixels.

Figure 34 shows an image with both fine detail and some noise along with its frequency transform. Applying Butterworth low-pass filters with radii of 10 and 25 pixels in the frequency-domain image smooths the noise with some blurring of the high-frequency detail (**Figure 35**), while the application of Butterworth high-pass filters with the same radii emphasizes the edges and reduces the contrast in the large (low-frequency) regions (**Figure 36**). All of these filters were applied to the amplitude values as multiplicative masks. Adjusting the radius of the filter cutoff controls the range of frequencies in the inverse transformed image, just as changing the size of the convolution kernel alters the degree of smoothing or sharpening when the equivalent procedure is carried out in the spatial or pixel domain.

Of course, it is not necessary for the variation in magnitude to be from 1 to 0. Sometimes the lower limit is set to a fraction, so that the high (or low) frequencies are not completely attenuated.

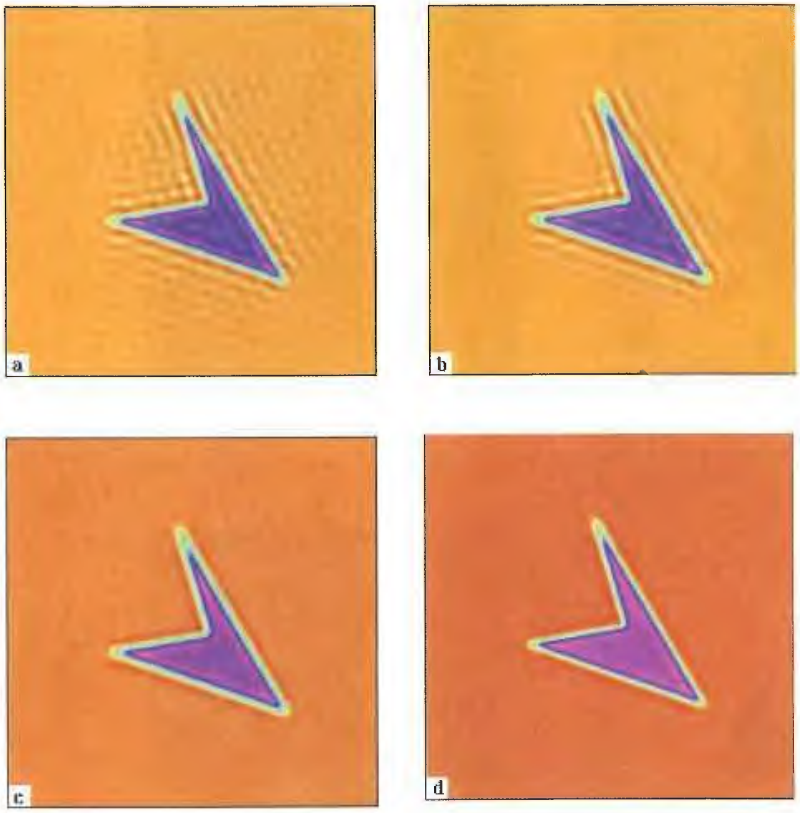


Figure 33. ***Reconstruction of shape from its frequency transform in Figure 46, using a 25-pixel aperture (mask or filter) diameter:*** ***(a)*** *ideal filter in which the cutoff is exact and abrupt;* ***(b)*** *cosine-weighted edge shape with a half-width of 3 pixels;* ***(c)*** *cosine-weighted edge shape with a half-width of 10 pixels;* ***(d)*** *Butterworth second-degree shape. False color has been added to increase the visibility of small variations.*

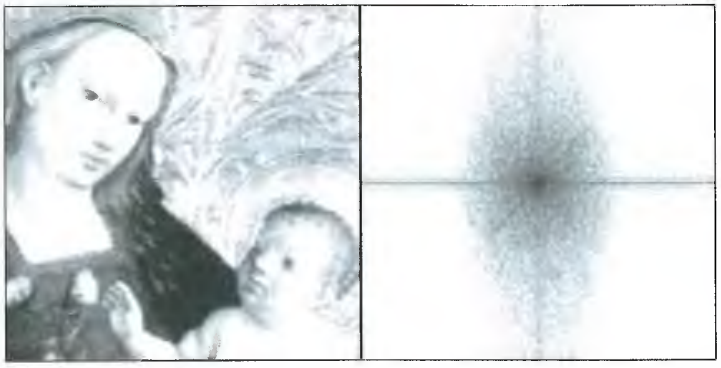


Figure 34. *Image and its transform used for filtering in* ***Figures 35*** *and* ***36.***

It is also possible to use values greater than 1. A high-frequency emphasis filter with the low-frequency value set to a reduced value, such as 0.5, and a high-frequency value greater than 1, such as 2, is called a homomorphic filter. It is usually applied to an image whose brightness values have previously been converted to their logarithms (using a LUT). This filtering operation will simultaneously increase the high-frequency information (sharpening edges) while reducing the overall brightness range to allow edge brightness values to show. The physical reasoning behind the homomorphic filter is a separation of illumination and reflectance components in the image. As with most of these filters, though, the real justification is that it improves the appearance of many images of practical interest.



Figure 35. *Filtering of* ***Figure 34*** *with low-pass Butterworth Filters having 50% cutoff diameters of 10 (left) and 25 pixels (right).*



Figure 36. *Filtering of* ***Figure 34*** *with high-pass Butterworth Filters having 50% cutoff diameters of 10 (left) and 25 pixels (right).*

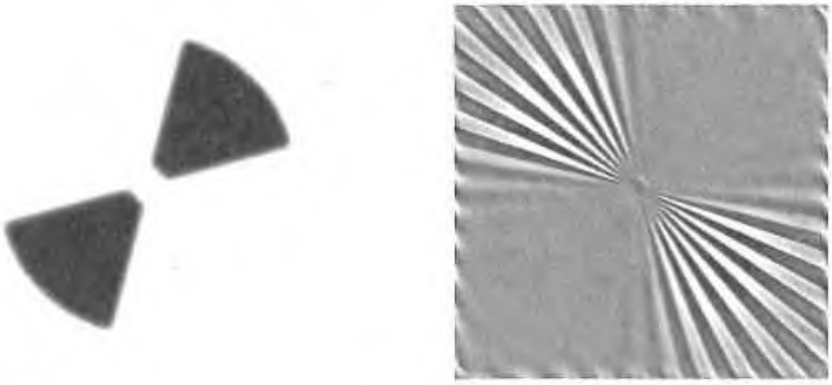
It is also possible to select a region of the Fourier transform image that is not centered on the origin. **Figure 37** shows a selection of intermediate frequency values lying in a particular direction on the transform in **Figure 20**, along with the resulting reconstruction. This kind of filtering may be useful to select directional information from images. It also demonstrates the basic characteristic of Fourier transform images: locations in the Fourier transform image identify periodicity and orientation information from any or all parts of the spatial-domain image. Note that the correct shape for off-centered regions is a combination of annuli and wedges that have cutoffs corresponding to frequencies and angles, rather than the more easily constructed circles or rectangles, and that the edges require the same type of smoothing shown previously.

Masks and filters

Once the location of periodic noise in an original image is isolated into a single point or a few points in the Fourier-transform image, it becomes possible to remove the noise by removing those sinusoidal terms. A filter removes selected frequencies and orientations by reducing the magnitude values for those terms, either partially or to zero, while leaving the phase information alone (which is important in determining where in the image that information appears).

Many different ways are used to specify and to apply this reduction. Sometimes it is practical to specify a range of frequencies and orientations numerically, but most often it will be convenient to do so using the magnitude or power spectrum display of the Fourier transform image. Manually or automatically selecting regions on this display allows specific peaks in the power spectrum, corresponding to the periodic information, to be selected for elimination.

Rather than the use of combinations of arcs and radial lines, for isolated noise peaks (often referred to as noise "spikes") it is convenient and usually acceptable to use small circles to define the



a b

Figure 37. *Inverse transform of the Fourier transform of* ***Figure 20 (b)*** *through a filter selecting a range of frequencies and angles* ***(a)****.*

regions. It is important to use a smoothing function to modify the edges of the regions. When the transition takes place over a distance of only a few pixels, the differences between the various transition curves described previously are of little importance, and a simple Gaussian smooth is most commonly applied.

Usually, it is only by first examining the transform-image power spectrum that the presence and exact location of these points can be determined. **Figure 38** shows an image of a halftone print from a magazine. The pattern results from the halftone screen used in the printing process. In the frequency transform of the image, this regular pattern shows up as well-defined narrow peaks or spikes. Filtering removes the peaks by setting the magnitude at those locations to zero (but not altering any of the phase information). This allows the image to be retransformed without the noise. The filtered Fourier transform image power spectrum shown in **Figure 39** shows the circular white spots where filtering was performed.

It is interesting to note that image compression does not necessarily remove these noise spikes in the Fourier transform. Compression is generally based on discarding the terms in the transform (Fourier, Cosine, etc.) whose magnitudes are small. The halftone pattern or other periodic noise is interpreted as an important feature of the image by these compression methods. The need to preserve the periodic noise spikes actually reduces the amount of other useful detail that can be retained.

The effects of compression are best examined in the frequency transform. **Figure 40** shows a portion of the image of Saturn used in Chapter 4, before and after compression by the JPEG algorithm. Some differences, particularly in the rings, can be seen. The frequency transforms of the two images show that this is due to the loss of lines of high frequency information that are needed to define the profiles of the rings and the detail within them.

The example in **Figure 39** relied on human observation of the peaks in the Fourier transform image, recognition that the peaks were responsible for the periodic noise, and selecting them to produce the filter. In some cases, it is possible to construct an appropriate image filter automatically from the Fourier transform power spectrum. The guiding principle is that peaks in the power spectrum that are narrow and rise significantly above the local background should be removed. If the Fourier transform magnitude image is treated like an ordinary spatial-domain grey scale image, this peak removal can often be accomplished automatically using a rank-based filter like the top hat.



Figure 38. *A halftoned printed image (left) and its frequency transform (right).*

Figure 39. *Removal of periodic information from* ***Figure 38*** *by reducing the magnitude (left) and retransforming (right).*

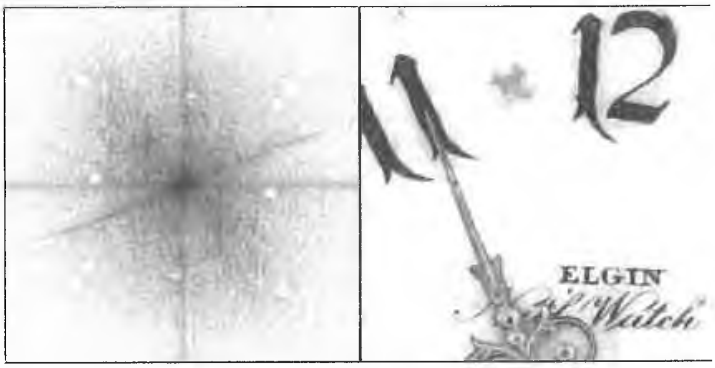
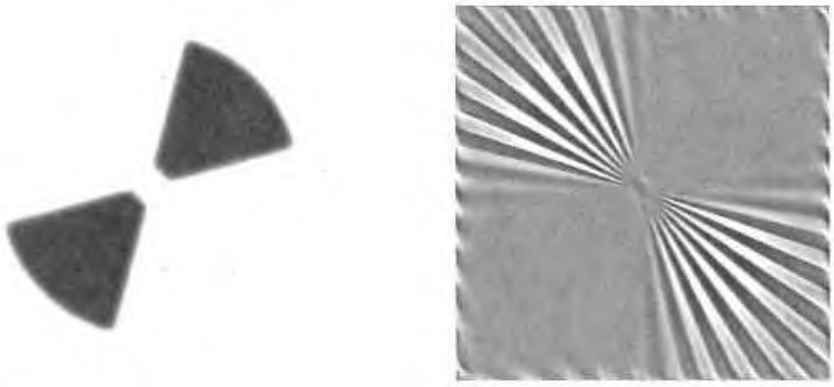


Figure 41 shows the power spectrum from the image in **Figure 38** and the result of applying a top hat filter with an inner radius of 3 pixels, an outer radius of 5 pixels, and a height of 8 grey-scale values. The brim of the hat rests on the background that drops off gradually and smoothly as frequency (radius) increases. The located features correspond to the noise spikes. Enlarging the spots by a few pixels and applying a smoothing to the edges, and inverting this result creates a mask that will remove the spikes. The figure shows the resulting modified power spectrum and the retransformed result.

Sometimes, the process of determining where the noise peaks are located can be simplified by selecting a region of the image that exhibits the noise pattern in an otherwise uniform area. This sequence is demonstrated in **Figure 42** for the same image as in **Figures 38** and **41**. A region of halftone periodic noise is selected and the rest of the image cleared. The transform of this image is then processed by smoothing and leveling, and then thresholded to locate the peaks. The inverse of this mask is then smoothed and multiplied by the frequency transform of the original image to produce the filtered result.

Removal of noise spikes is not always enough to restore an image completely. The example in **Figure 43** shows an example of an image scanned in from a newspaper; the halftone dots are very evident. The dark spots ("spikes") in the power spectrum (**Figure 43b**) correspond to the periodic structure in the image. They align with the repetitive pattern of dots, and their darkness indicates how much of various frequencies are present. Removing them is equivalent to removing the periodic component. In this case a top hat filter was used to locate the spikes and create a mask (**Figure 43c**). The result of applying this mask as a filter to remove the spikes shows that the periodic noise has been removed without affecting any of the other information present (**Figure 43d**). There is still some pixel-to-pixel noise because the image has been scanned at a higher magnification than it was printed, and the halftone cells in the original image are separated.



a b

Figure 37. Inverse transform of the Fourier transform of ***Figure 20 (b)*** *through a filter selecting a range of frequencies and angles* ***(a)****.*

regions. It is important to use a smoothing function to modify the edges of the regions. When the transition takes place over a distance of only a few pixels, the differences between the various transition curves described previously are of little importance, and a simple Gaussian smooth is most commonly applied.

Usually, it is only by first examining the transform-image power spectrum that the presence and exact location of these points can be determined. **Figure 38** shows an image of a halftone print from a magazine. The pattern results from the halftone screen used in the printing process. In the frequency transform of the image, this regular pattern shows up as well-defined narrow peaks or spikes. Filtering removes the peaks by setting the magnitude at those locations to zero (but not altering any of the phase information). This allows the image to be retransformed without the noise. The filtered Fourier transform image power spectrum shown in **Figure 39** shows the circular white spots where filtering was performed.

It is interesting to note that image compression does not necessarily remove these noise spikes in the Fourier transform. Compression is generally based on discarding the terms in the transform (Fourier, Cosine, etc.) whose magnitudes are small. The halftone pattern or other periodic noise is interpreted as an important feature of the image by these compression methods. The need to preserve the periodic noise spikes actually reduces the amount of other useful detail that can be retained.

The effects of compression are best examined in the frequency transform. **Figure 40** shows a portion of the image of Saturn used in Chapter 4, before and after compression by the JPEG algorithm. Some differences, particularly in the rings, can be seen. The frequency transforms of the two images show that this is due to the loss of lines of high frequency information that are needed to define the profiles of the rings and the detail within them.

The example in **Figure 39** relied on human observation of the peaks in the Fourier transform image, recognition that the peaks were responsible for the periodic noise, and selecting them to produce the filter. In some cases, it is possible to construct an appropriate image filter automatically from the Fourier transform power spectrum. The guiding principle is that peaks in the power spectrum that are narrow and rise significantly above the local background should be removed. If the Fourier transform magnitude image is treated like an ordinary spatial-domain grey scale image, this peak removal can often be accomplished automatically using a rank-based filter like the top hat.



Figure 38. A halftoned printed image (left) and its frequency transform (right).

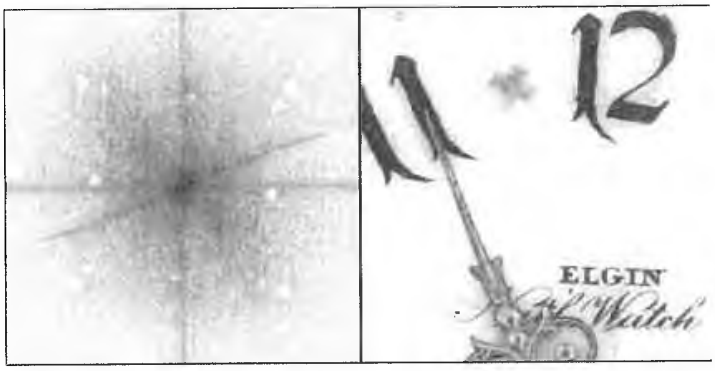


Figure 39. Removal of periodic information from ***Figure 38*** *by reducing the magnitude (left) and retransforming (right).*

Figure 41 shows the power spectrum from the image in **Figure 38** and the result of applying a top hat filter with an inner radius of 3 pixels, an outer radius of 5 pixels, and a height of 8 grey-scale values. The brim of the hat rests on the background that drops off gradually and smoothly as frequency (radius) increases. The located features correspond to the noise spikes. Enlarging the spots by a few pixels and applying a smoothing to the edges, and inverting this result creates a mask that will remove the spikes. The figure shows the resulting modified power spectrum and the retransformed result.

Sometimes, the process of determining where the noise peaks are located can be simplified by selecting a region of the image that exhibits the noise pattern in an otherwise uniform area. This sequence is demonstrated in **Figure 42** for the same image as in **Figures 38** and **41**. A region of halftone periodic noise is selected and the rest of the image cleared. The transform of this image is then processed by smoothing and leveling, and then thresholded to locate the peaks. The inverse of this mask is then smoothed and multiplied by the frequency transform of the original image to produce the filtered result.

Removal of noise spikes is not always enough to restore an image completely. The example in **Figure 43** shows an example of an image scanned in from a newspaper; the halftone dots are very evident. The dark spots ("spikes") in the power spectrum (**Figure 43b**) correspond to the periodic structure in the image. They align with the repetitive pattern of dots, and their darkness indicates how much of various frequencies are present. Removing them is equivalent to removing the periodic component. In this case a top hat filter was used to locate the spikes and create a mask (**Figure 43c**). The result of applying this mask as a filter to remove the spikes shows that the periodic noise has been removed without affecting any of the other information present (**Figure 43d**). There is still some pixel-to-pixel noise because the image has been scanned at a higher magnification than it was printed, and the halftone cells in the original image are separated.

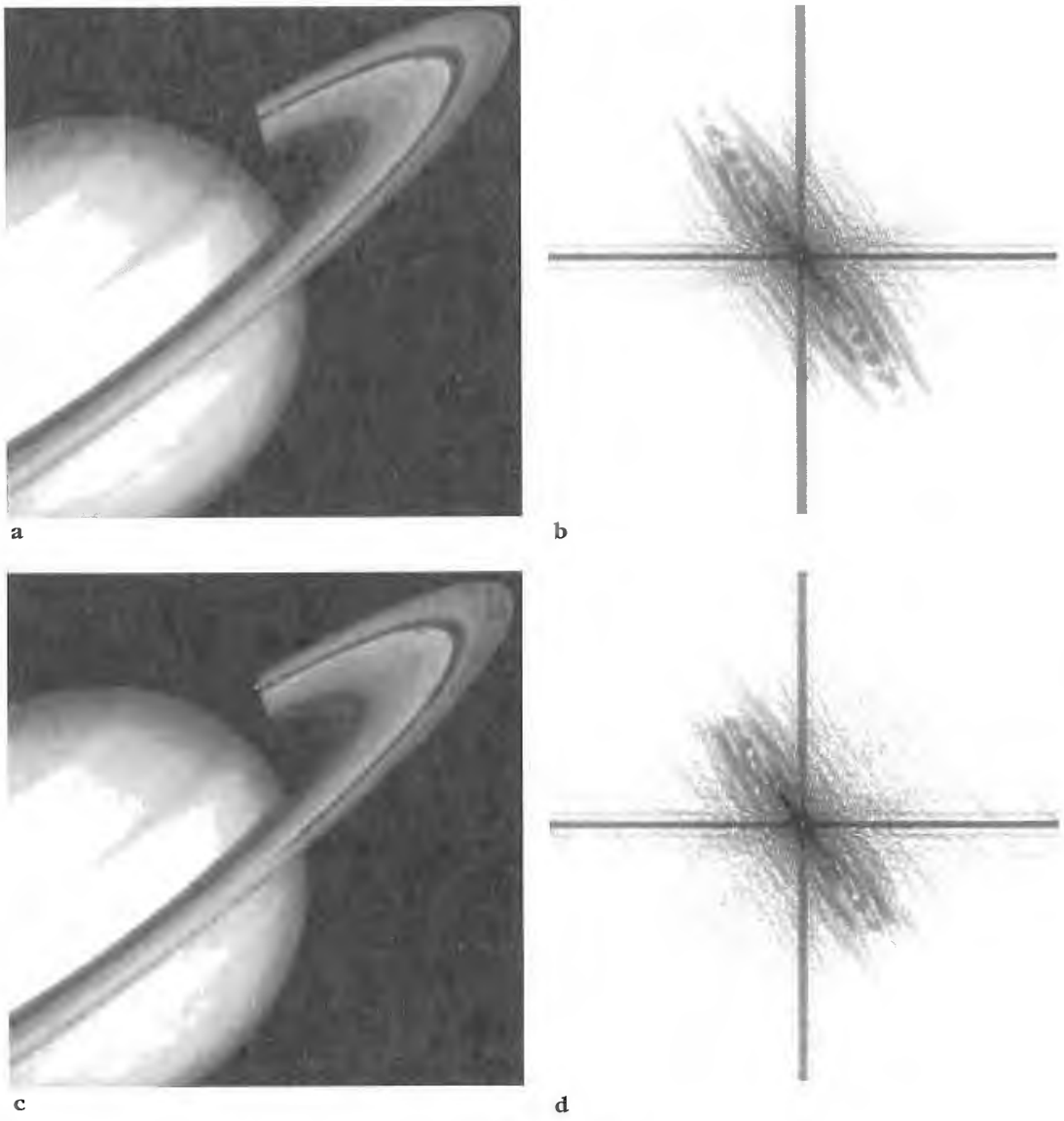


Figure 40. Portion of the image of Saturn from Chapter 4:
(a) *original;*
(b) *FT power spectrum showing high frequencies needed for detail in rings;*
(c) *image after JPEG compression (factor of 12);*
(d) *FT power spectrum of* ***c*** *showing loss of high-frequency information.*

An additional filter can be employed to fill in the space between cells. A Butterworth second-order, high-frequency cutoff filter keeps low frequencies (gradual variations in grey scale) while progressively cutting off higher ones. In this case the midpoint of the cutoff was set to the spacing of the halftone dots in the original image. The final version of the power spectrum (**Figure 43e**) shows the periodic spots removed and the high frequencies attenuated. An inverse transform produces the final image (**Figure 43f**). Note that even the cat's whiskers which are barely discernible in the original image can be clearly seen.

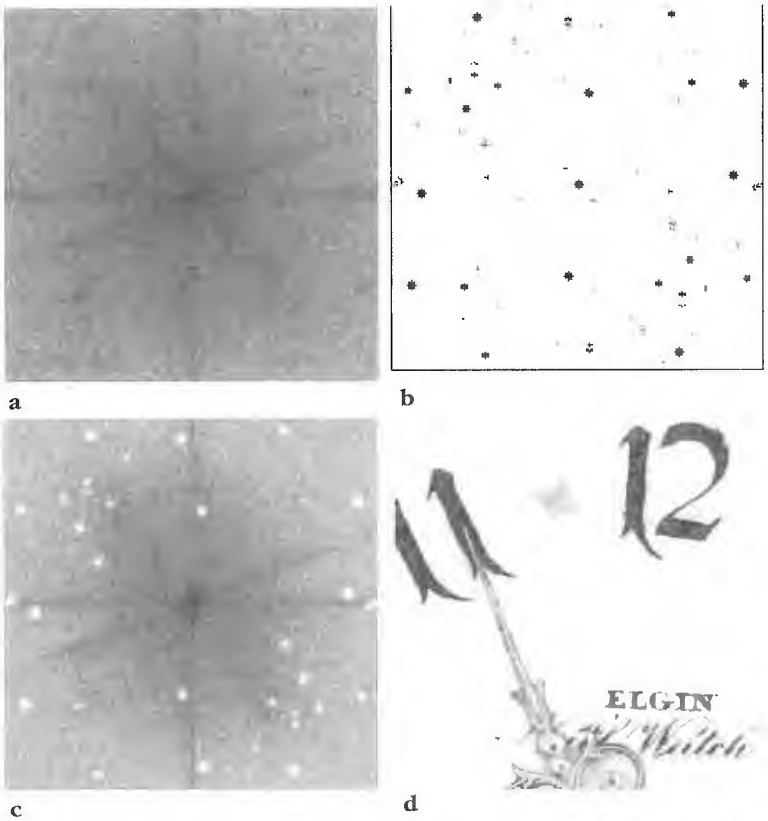


Figure 41. Automatic location and removal of noise spikes: *(**a**) original power spectrum from **Figure 38**; (**b**) top hat filter applied to **a**; (**c**) modified power spectrum after removal of spikes found in **b**; (**d**) retransformed image.*



Figure 42. Identification of noise spikes by selecting image regions that exhibit the noise pattern (a): *(**b**) Fourier transform power spectrum from the selected areas; (**c**) mask created by processing **b**; (**d**) result of applying the filter from **c** to the Fourier transform of the entire original image.*

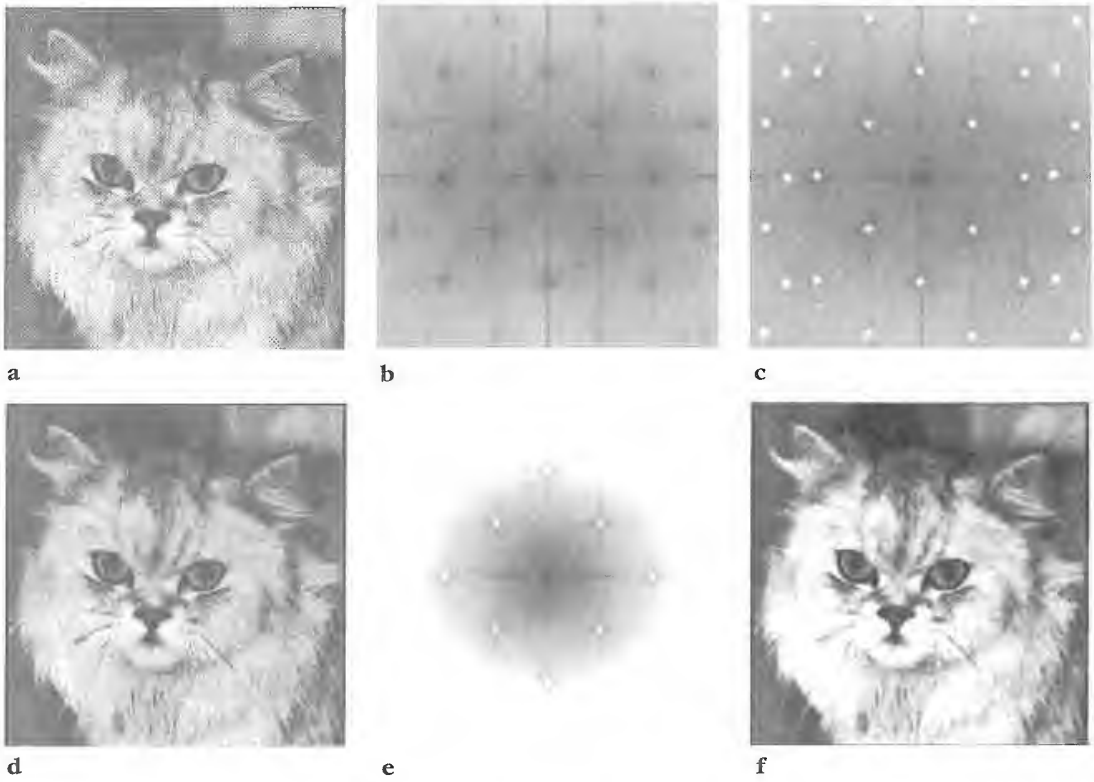


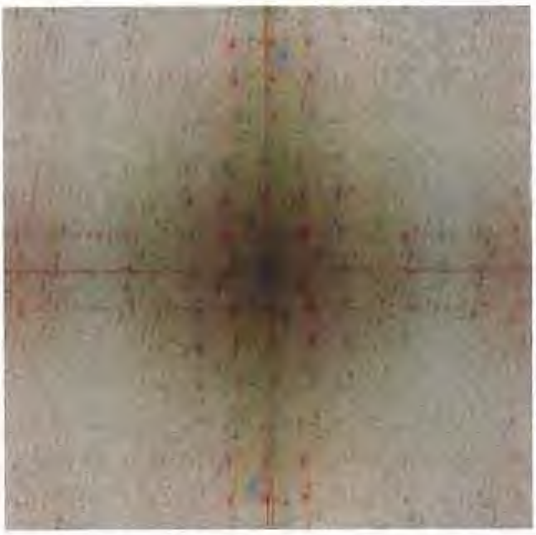
Figure 43. Removal of halftone pattern:
(a) original scanned image;
(b) power spectrum;
(c) mask applied to remove spikes;
(d) inverse transform with spikes removed;
(e) Butterworth low-pass filter applied to power spectrum;
(f) final result.

A common recommendation (found in internet newsgroups, for instance) for dealing with scanned halftone images or ones with moiré patterns from the scanner is to use smoothing to eliminate the pattern (sometimes by scanning the image at an angle and rotating it in software, using the interpolation as a low-pass filter, sometimes by scanning at a larger size and again depending on the filtering effects of interpolation as the image size is reduced, and sometimes by the overt application of a smoothing algorithm). This is a flawed strategy. The use of a smoothing or blur function in this image would have erased these fine lines long before the halftone dots were smoothed out.

When the original image has been scanned in from a color print, the situation is slightly more complicated. As mentioned in Chapter 2 and shown in **Figure 44**, colored halftone prints are typically made using cyan, magenta, yellow, and black inks, with each color having a halftone screen oriented at a different angle. The procedure required is to separate the color channels, process each one, and then recombine them for the final result. In the figure, the power spectra for each of the channels is shown superimposed, in the corresponding color, to show the different screen orientations. A top hat filter was applied to the Fourier transform power spectrum from each channel to select the noise peaks, which were then removed. Recombining the resultant color channel images produces the result shown.



a



b



c

Figure 44. Halftone removal in a color image: *(a) original image (a portion of a postage stamp, printed in CMYK); (b) Fourier transform power spectra for each color channel, superimposed in color; (c) removal of noise from each channel and recombining the color channels.*

Selection of periodic information

In some types of images, it is the periodic information that is useful and the non-periodic noise that must be suppressed. The methods for locating the periodic peaks, constructing filters, smoothing the filter edges, and so forth are unchanged. The only difference is that the filter sense is changed and in the case of a multiplicative mask, the values are inverted.

Figure 45 shows a one-dimensional example. The image is a light micrograph of stained skeletal muscle, in which there is a just-visible band spacing that is difficult to measure because of the spotty staining contrast (even with contrast expansion). The FT image shows the spots that identify the band structure. Reconstruction with only the first harmonic shows the basic band structure, and adding the second and third harmonics defines the band shape fairly well.

Figure 46 shows a high-resolution TEM lattice image from a crystalline ceramic (mullite). The two-dimensional periodicity of the lattice can be seen, but it is superimposed on a variable and

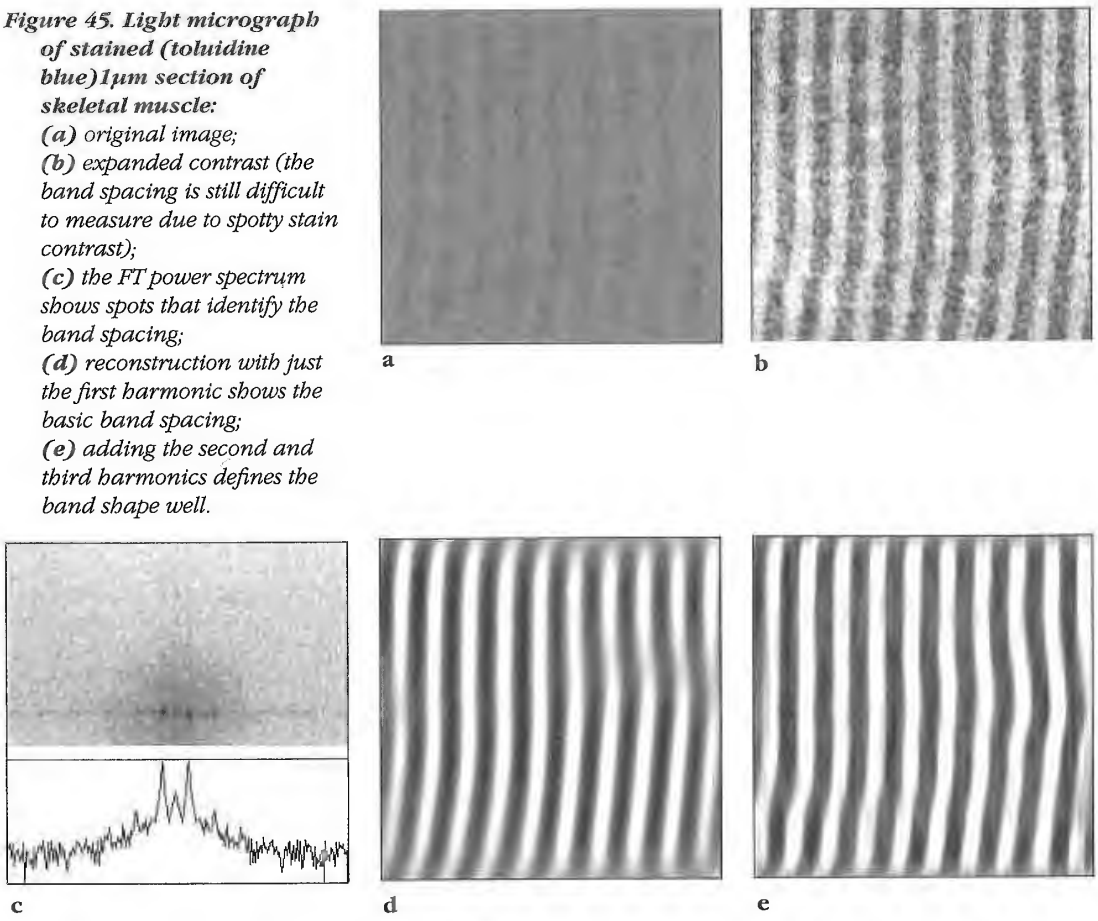


Figure 45. Light micrograph of stained (toluidine blue) 1μm section of skeletal muscle: *(a) original image; (b) expanded contrast (the band spacing is still difficult to measure due to spotty stain contrast); (c) the FT power spectrum shows spots that identify the band spacing; (d) reconstruction with just the first harmonic shows the basic band spacing; (e) adding the second and third harmonics defines the band shape well.*

noisy background, which alters the local contrast making it more difficult to observe the details in the rather complex unit cell of this material. The Fourier transform image has peaks that correspond to the periodic structure. As noted before, this image is essentially the same as would be recorded photographically using the TEM to project the diffraction pattern of the specimen to the camera plane. Of course, retransforming the spatial-domain image from the photographed diffraction pattern is not possible because the phase information has been lost. In addition, more control over the Fourier transform display is possible because a log scale or other rule for converting magnitude to screen brightness can be selected.

A filter mask is constructed to select a small circular region around each of the periodic spots. This was done, as before, by using a top hat filter with size just large enough to cover the peaks. In this case the filter is used to keep the amplitude of terms in the peaks but reduce all of the other terms to zero, which removes the random or non-periodic noise, both the short-range (high-frequency) graininess and the gradual (low-frequency) variation in overall brightness. Retransforming this image produces a spatial-domain image which shows the lattice structure clearly.

Figure 47 shows an even more dramatic example. In the original image (a cross section of muscle myofibrils) it is practically impossible to discern the periodic structure due to the presence of noise. In isolated locations, a few of the fibrils can be seen to have a regular spacing and arrangement, but human observers do not easily see through the noise and variability to find this regularity. The Fourier transform image shows the peaks from the underlying regularity,

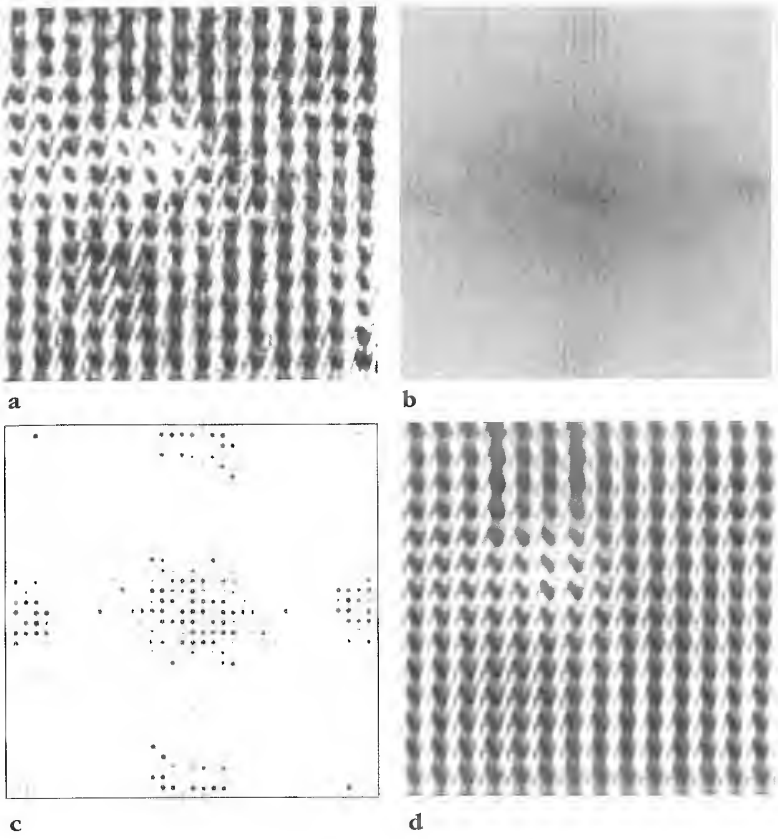


Figure 46. Transmission electron microscope lattice image of the mullite (ceramic): ***(a)*** *original;* ***(b)*** *Fourier transform power spectrum;* ***(c)*** *result of applying top hat filter to* ***b*** *to construct a filter;* ***(d)*** *retransformation of just the selected periodic information.*

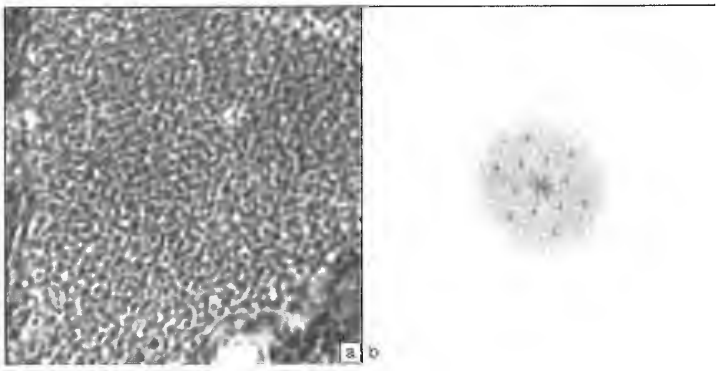


Figure 47. *Transmission electron microscope image of cross section of muscle myofibrils (left) and the frequency transform (right) (image courtesy Arlo Reeves, Dartmouth College).*

however. Selecting only these peak points in the magnitude image (with their original phase information) and reducing all other magnitude values to zero produces the result shown in **Figure 48**. The retransformed image clearly shows the sixfold symmetry expected for the myofibril structure. The inset shows an enlargement of this structure in even finer detail, with both the thick and thin filaments shown. The thin filaments, especially, cannot be seen clearly in the original image.

A caution is needed in using this type of filtering to extract periodic structures. It is possible to construct a mask that will eliminate real information from the image while keeping artefacts and noise.

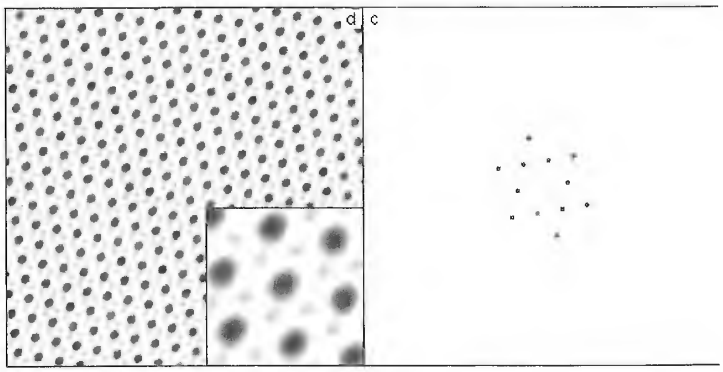


Figure 48. *Retransformation of* ***Figure 46*** *(left) with only the principal periodic peaks in the frequency transform (right). The points in the filter mask have been enlarged for visibility.*

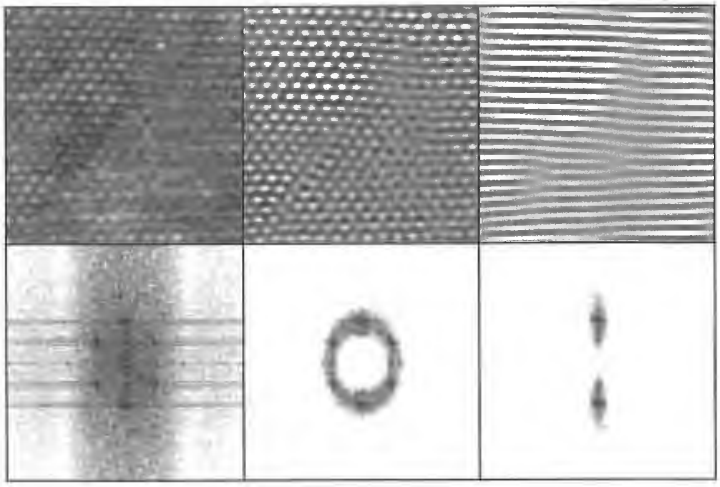


Figure 49. *Noisy high-resolution TEM image (left), and the results of applying an annular filter to select atomic spacings (center), and a slit filter to select vertical spacings only (right). Top images show the spatial domain, and bottom row shows frequency domain.*

Selecting points in the power spectrum with sixfold symmetry ensured that the filtered and retransformed spatial image would show that type of structure. This means that the critical step is the recognition and selection of the peaks in the Fourier transform image. Fortunately, many suitable tools are available for finding and isolating such points, because they are narrow peaks that rise above a gradually varying local background. The top hat filter illustrated previously is an example of this approach.

It is also possible to construct a filter to select a narrow range of spacings, such as the interatomic spacing in a high-resolution image. This annular filter makes it possible to enhance a selected periodic structure. **Figure 49** shows a high-resolution TEM image of an atomic lattice. Applying an annular filter that blocks both the low and high frequencies produces the result shown, in which the atom positions are more clearly defined. If the filter also selects a particular orientation (a slit or wedge filter), however, then the dislocation that is difficult to discern in the original image becomes clearly evident.

As for the case of removing periodic noise, a filter that selects periodic information and reveals periodic structure can often be designed by examining the Fourier transform power spectrum image itself to locate peaks. The mask or filter can be constructed either manually or automatically. In some cases, there is *a priori* information available (such as lattice spacings of crystalline specimens).

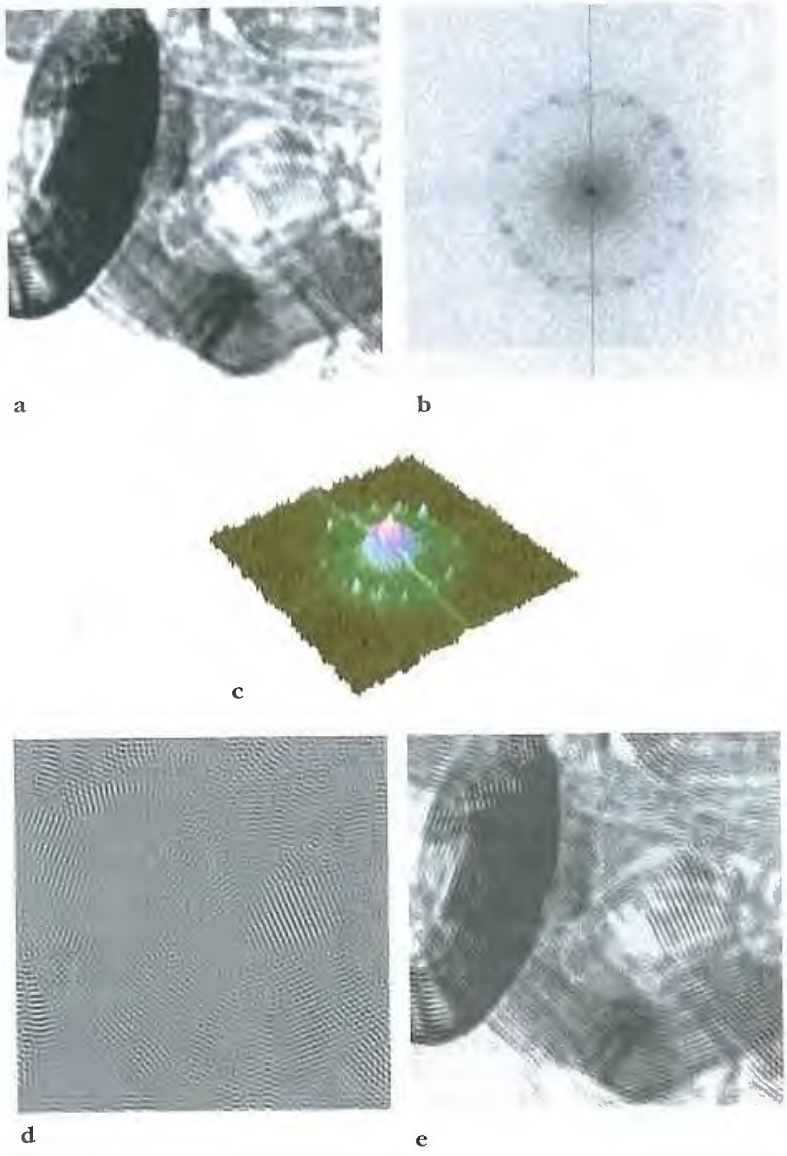


Figure 50. Transmission electron microscope (TEM) image of graphitized carbon ***(a): (b)*** *the FT power spectrum from image a showing the broken ring of spots corresponding to the 3.5 Angstrom plane spacing in the graphite;* ***(c)*** *the power spectrum plotted as a perspective drawing to emphasize the ring of peaks;* ***(d)*** *inverse transform of just the ring of spots;* ***(e)*** *image* ***d*** *added to the original image* ***a*** *to enhance the visibility of the lattice.*

In **Figure 50**, the structure (of graphitized carbon in particles used for tire manufacture) shows many atomic lattices in different orientations. In this case, the diffraction pattern shows the predominant atom plane spacing of 3.5 Ångstroms, in the form of a ring of spots. Applying an annular filter to select just that spacing, retransforming and adding the atom positions to the original image, enhances the visibility of the lattice structure.

The use of filtering in FT space is also useful for isolating structural information when the original image is not perfectly regular. **Figure 51** shows the same skeletal muscle tissue shown in **Figure 45**, but cross-sectioned and stained with uranyl acetate/lead citrate. The black dots are immunogold labeling for fast myosin, involved in muscle contraction. The spots are more or less regularly spaced because of the uniform diameters of the muscle fibers, but have no regular order to their arrangement. Sharpening the image using a top hat filter, as discussed above and in Chapter 4,

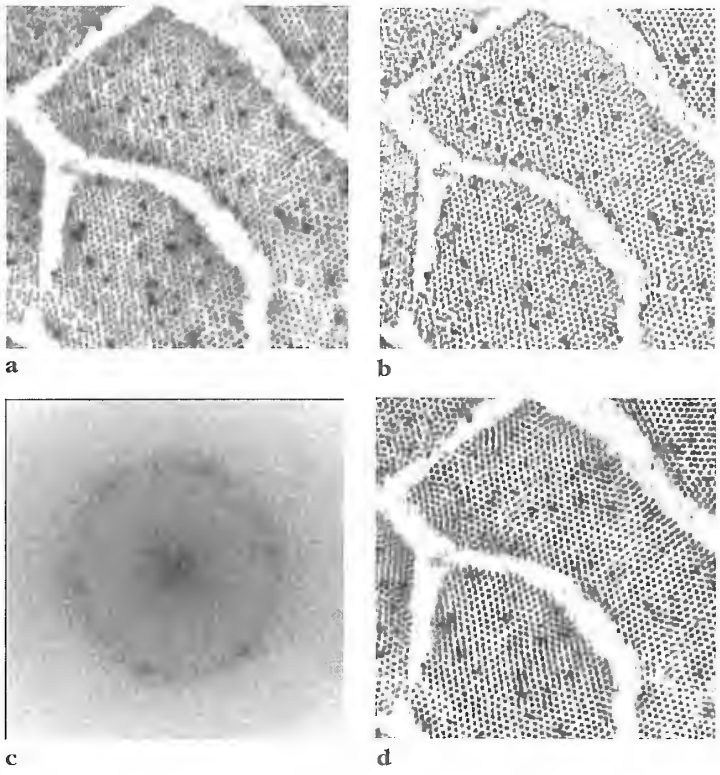


Figure 51. Light micrograph of cross section of the same skeletal muscle as in Figure 62, cross-sectioned and uranyl acetate/lead citrate stained. Black dots are immunogold labeling for fast myosin:
(a) *original;*
(b) *top hat filter applied to the image, as shown in Chapter 4;*
(c) *the power spectrum from the Fourier transform, showing a broken ring of spots corresponding to the average diameter of the fibers;*
(d) *retransforming the ring of spots with an annular filter that selects just the spacings of the gold particles and adding this back to the original image increases the contrast of the fibers.*

improves the visibility of the spots as compared to the original image. So does the use of an annular Fourier filter that selects just the ring of spots corresponding to the fiber diameter.

Similarly, measurement of "quasi-periodic" structures can also be performed in Fourier space. In **Figure 52**, the packed arrays of latex spheres are not perfectly regular, but the Fourier transform power spectrum shows the rings of nearest neighbors, second-nearest, third-nearest, and so forth. Measuring the spacing is simplified by using a plot of the circularly averaged power spectrum values, as shown.

Convolution and correlation

Fundamentals of convolution

One of the very common operations on images performed in the spatial domain is convolution, in which a kernel of numbers is multiplied by each pixel and its neighbors in a small region, the results summed, and the result placed in the original pixel location. This is applied to all of the pixels in the image. In all cases, the original pixel values are used in the multiplication and addition, and the new derived values are used to produce a new image, although as a practical matter of implementation the operation may be performed a few lines at a time, so that the new image ultimately replaces the old one.

This type of convolution is particularly common for the smoothing and derivative operations illustrated in Chapter 3 and 4. For instance, a simple smoothing kernel might contain the following values:

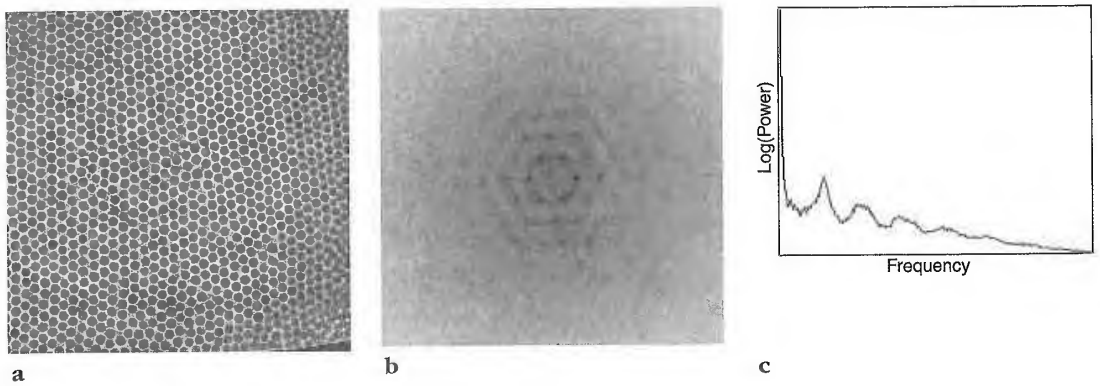


Figure 52. Quasi-periodic arrays of latex spheres:
(a) *original image;*
(b) *Fourier transform power spectrum;*
(c) *circularly averaged plot of the power spectrum, in which peak locations identify the distances of first, second-, etc., neighbors.*

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

In practice, the fastest implementation would be to multiply the pixel and its 8 immediate neighbors by the integers 1, 2, or 4, sum the products, then divide the total by 16. In this case, using integers that are powers of 2 allows the math to be extremely fast (involving only bit shifting), and the small size of the kernel (3 × 3) makes the application of the smoothing algorithm on the spatial-domain image very fast, as well.

Many spatial-domain kernels are used, including kernels that apply Gaussian smoothing (to reduce noise), take first derivatives (for instance, to locate edges), and take second derivatives (for instance, the Laplacian, which is a non-directional operator that acts as a high-pass filter to sharpen points and lines). They are usually presented as a set of integers, although greater accuracy is attainable with floating point numbers, with it understood that there is a divisor (usually equal to the sum of all the positive values) that normalizes the result. Some of these operators may be significantly larger than the 3 × 3 example shown previously, involving the adding together of the weighted sum of neighbors in a much larger region that is usually, but not necessarily, square.

Applying a large kernel takes time. **Figure 53** illustrates the process graphically for a single placement of the kernel. Even with very fast computers and with careful coding of the process to carry out additions and multiplications in the most efficient order, performing the operation with a 25 × 25 kernel on a 1024 × 1024 image would require a significant amount of time (and even larger kernels and images are often encountered). Though it can be speeded up somewhat by the use of special hardware, such as a pipelined array processor, a special-purpose investment is required. A similar hardware investment can be used to speed up the Fourier transform. Our interest here is in the algorithms, instead of their implementation.

For any computer-based system, increasing the kernel size eventually reaches a point at which it is more efficient to perform the operation in the Fourier domain. The time needed to perform the FFT transformation from the spatial domain to the frequency domain and back is more than balanced by the speed with which the convolution can be carried out. If any other reasons are needed to perform

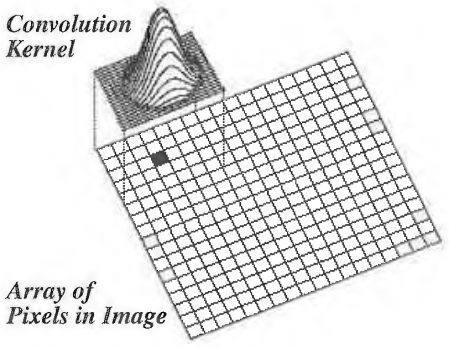


Figure 53. *Illustration of applying a convolution kernel to an image in the spatial domain (illustration courtesy Arlo Reeves, Dartmouth College).*

the transformation to the frequency-domain representation of the image, then even small kernels can be most efficiently applied there.

This is because the equivalent operation to spatial-domain convolution is a single multiplication of each pixel in the magnitude image by the corresponding pixel in a transform of the kernel. The transform of the kernel can be obtained and stored beforehand just as the kernel is stored. If the kernel is smaller than the image, it is padded with zeroes to the full image size. Convolution in the spatial domain is exactly equivalent to multiplication in the frequency domain. Using the notation presented before, in which the image is a function $f(x,y)$ and the kernel is $g(x,y)$, we describe the convolution operation in which the kernel is positioned everywhere on the image and multiplied by it as

$$g(x,y) * f(x,y) = \iint (f(\alpha,\beta) \cdot g(x-\alpha, y-\beta) d\alpha \, d\beta \tag{9}$$

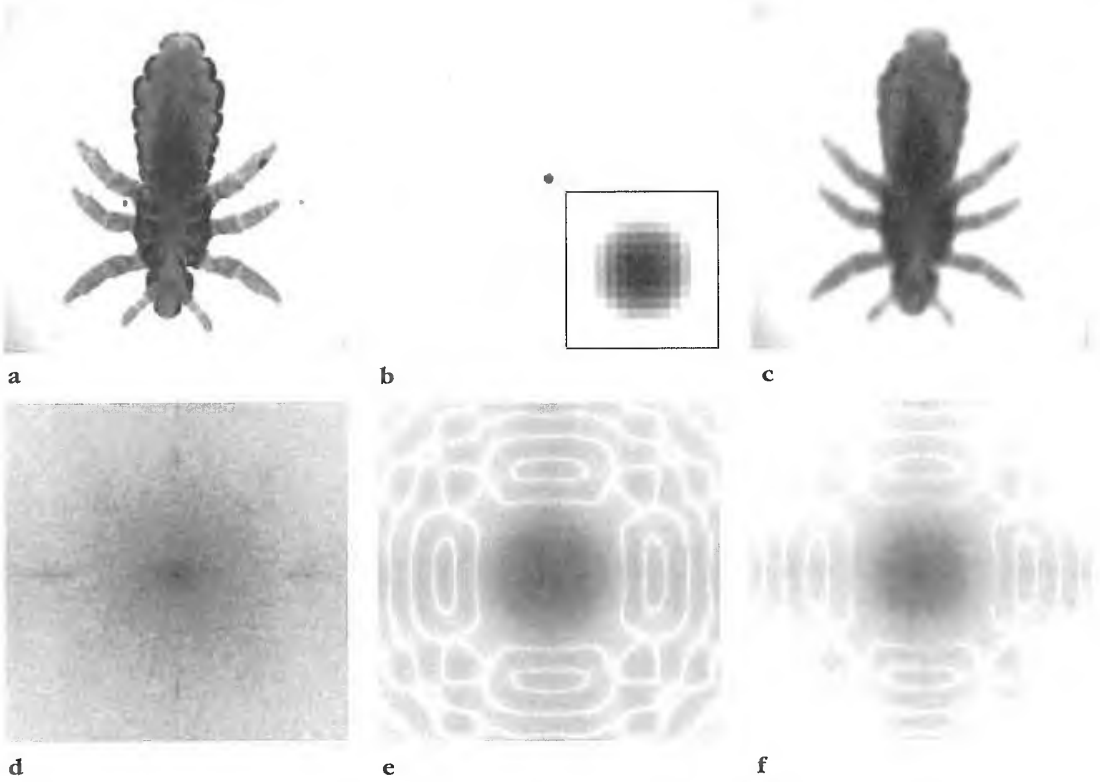
where α and β are dummy variables for the integration, the range of which is across the entire image, and the symbol * indicates convolution. If the Fourier transforms of $f(x,y)$ and $g(x,y)$ are $F(u,v)$ and $G(u,v)$, respectively, then the convolution operation in the Fourier domain is simple point-by-point multiplication, or

$$g(x,y) * f(x,y) \Leftrightarrow G(u,v)F(u,v) \tag{10}$$

A few practical differences exist between the two operations. The usual application of a kernel in the spatial domain avoids the edge pixels (those nearer to the edge than the half-width of the kernel), since their neighbors do not exist. As a practical alternative, a different kernel that is one-sided and has different weights can be applied near edges, or the edge can be considered to be a mirror. In transforming the image to the frequency domain, however, the assumption is made that the image wraps around at edges, so that the left edge is contiguous with the right and the top edge is contiguous with the bottom. Applying a convolution by multiplying in the frequency domain is equivalent to addressing pixels in this same wraparound manner when applying the kernel to the spatial image. It will usually produce some artefacts at the edges. The most common solution for this problem is to embed the image of interest in a larger one in which the borders are either filled with the mean brightness value of the image, or smoothly interpolated from the edge values.

Figure 54 shows the equivalence of convolution in the spatial domain and multiplication in the frequency domain, for the case of a smoothing kernel. The kernel, a Gaussian filter with standard deviation of 2.0 pixels, is shown as an array of grey-scale values, along with its transform. Applying the kernel to the image in the spatial domain produces the result shown in the example.

Figure 54. Smoothing by applying a large kernel in the spatial domain and convolution in the frequency domain:
(a) original image;
(b) smoothing kernel (Gaussian, standard deviation = 2.0 pixels), with enlargement to show pixel detail;
(c) smoothed image produced by spatial convolution with kernel or inverse Fourier transform of ***e****;*
(d) Fourier transform of ***a****;*
(e) Fourier transform of ***b****;*
(f) product of ***d*** *and* ***e****.*



Multiplying the kernel transform by the image transform produces the frequency-domain image with the power spectrum shown. Retransforming this image produces the identical result to the spatial-domain operation.

Notice that the equivalence of frequency-domain multiplication to spatial-domain convolution is restricted to multiplicative filters, which are also known as "linear" filters. Other neighborhood operations, such as rank filtering (saving the brightest, darkest, or median brightness value in a neighborhood) and histogram modification (e.g., local adaptive equalization), are nonlinear and have no frequency-domain equivalent.

Imaging system characteristics

Convolution can also be used as a tool to understand how imaging systems alter or degrade images. For example, the blurring introduced by imperfect lenses can be described by a function $H(u,v)$ which is multiplied by the frequency transform of the image (**Figure 55**). The operation of physical optics is readily modeled in the frequency domain. Sometimes it is possible to determine the separate characteristics of each component of the system; often it is not. In some cases,

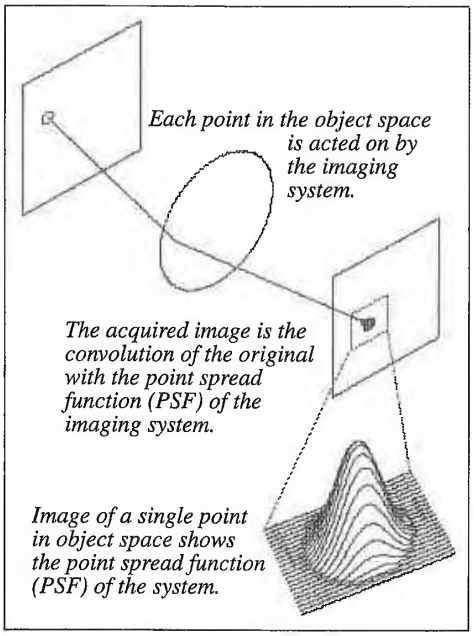


Figure 55. *System characteristics introduce a point spread function into the acquired image (illustration courtesy Arlo Reeves, Dartmouth College).*

determining the point-spread function of the system (the degree to which a perfect point in the object plane is blurred in the image plane) may make it possible to sharpen the image by removing some of the blur. This is done by dividing the transform of the point-spread image by $H(u,v)$.

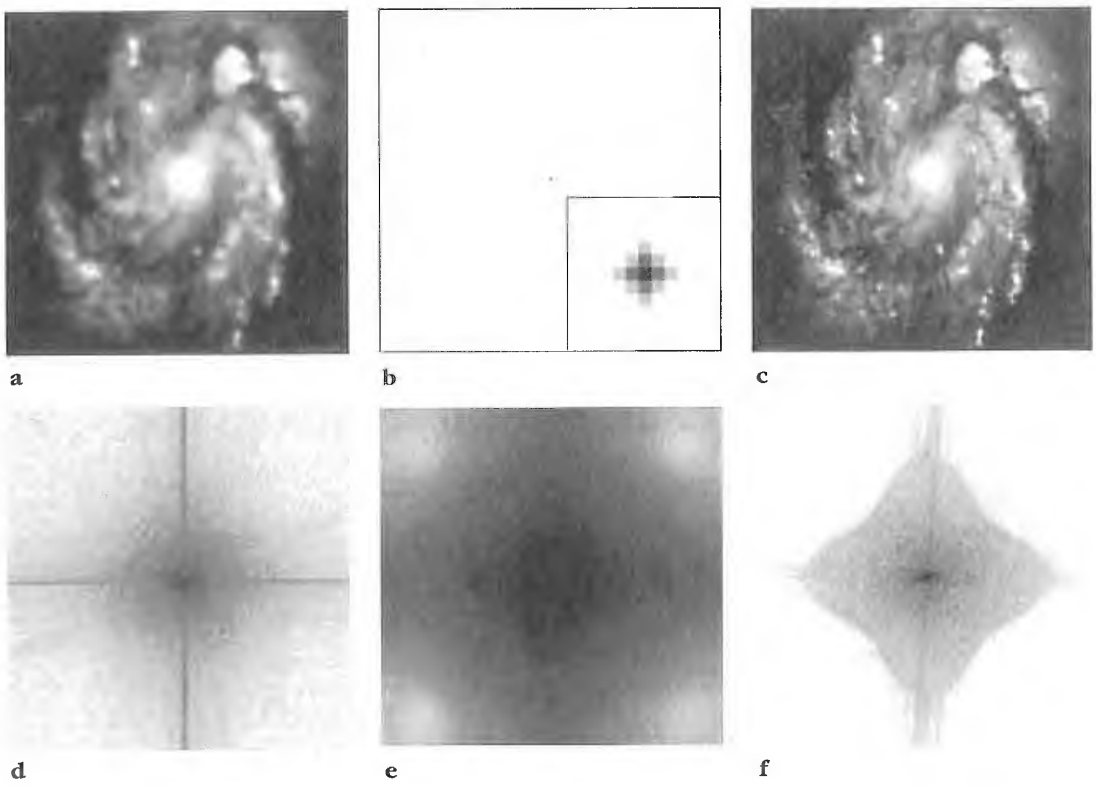
To illustrate this sharpening we can use the one of the best-known examples of correction of an out-of-focus condition due to an imperfect imaging system. As originally deployed, the Hubble telescope had an incorrectly figured main mirror that produced poorly focused images. Eventually a compensating optical device was installed in the imaging system to correct most of the defect, but even with the original telescope it was possible to obtain high resolution results by using deconvolution with the known point spread function (PSF). The corrected optical package, however, restored much of the telescope's light-gathering power, which was severely reduced due to the incorrect mirror curvature.

In this particular instance, it was possible to calculate the PSF from available measurement data on the mirror, but in many astronomical situations, the problem is simplified because the point spread function can be measured by examining the image of a single star, which is effectively a point as seen from earth. This requires a considerable grey-scale depth to the image, but in astronomy, cooled cameras often deliver 12 or 14 bits of information, which is also important for obtaining enough precision to perform the deconvolution.

Figure 56 shows the result of this method. The original, blurred image is sharpened by dividing its Fourier transform by that of the measured PSF, and the resulting inverse transform shows a considerably sharpened image.

For deconvolution, we divide the complex frequency-domain image from the out-of-focus test pattern by the image for the point spread function. This is complex division, performed by dividing the magnitude values and subtracting the phase values. One of the problems with division is that division by very small values can cause numeric overflow problems, and the Fourier transform of a symmetrical and well behaved point spread function often contains zero values. The usual

Figure 56. Hubble telescope image sharpening:
(a) *original;*
(b) *measured point spread function, with enlargement to show pixel detail;*
(c) *deconvolution result obtained as inverse transform of* ***f****;*
(d) *Fourier transform of* ***a****;*
(e) *Fourier transform of* ***b****;*
(f) ***d*** *divided by* ***e****.*



solution to this problem is apodization to restrict the division operation to those pixels in the complex transform images that will not cause overflow.

Deconvolution of image blur, which may arise from out-of-focus optics, motion, the size and arrangement of transistors in the camera chip, insufficient bandwidth in the electronics, or other causes, is an imperfect science. A mathematically optimum procedure neither exists, nor is there a proof that one exists in all cases. The practical techniques that are used have been developed under a variety of assumptions and apply to many real situations, producing impressive improvements in image quality. The results, although good, represent trade-offs between different limitations, one of which is the time required for the computations.

Deconvolution is discussed here in terms of grey scale images. In most cases, color images need to be separated into discrete color channels that correspond to the physics of the acquisition device, usually red, green and blue, and each one deconvolved separately. It is usual to find that the point spread function is different for each channel, especially if a single-chip camera with a Bayer pattern color filter has been used for acquisition (see Chapter 2).

It is important to always start with the best image possible. This means obtaining the best focus, least motion blur, etc., that you can achieve. Deconvolution is never as good a solution as

correcting the source of the problem beforehand. Next, capture the image with the best possible range of contrast, from nearly full white to nearly full black without clipping. The tonal or gray scale range should have good precision and a wide dynamic range. Eight-bit images from uncooled digital cameras are marginal, and poorer images from video cameras usually unacceptable unless special procedures such as averaging multiple frames are used. It is particularly important to have high precision and bit depth for the point spread function, whether it is obtained by measurement or by calculation.

Finally, random pixel noise (speckle) must be minimized. Noise in either the acquired image or (especially) in the point spread function is significantly amplified in the deconvolution process and will dominate the result if it is too great. Long exposures and image averaging may be useful in some cases. The usual description of the magnitude of image noise is a signal-to-noise ratio expressed in decibels. This is defined in terms of the standard deviation of values in the blurred image and in the noise (which of course may not be known).

$$SNR[\text{dB}] = 10 \cdot \log_{10}\left(\frac{\sigma_{image}}{\sigma_{noise}}\right) \tag{11}$$

When the signal-to-noise ratio is in the range of 40–50 dB, the noise is, practically speaking, invisible in the image and has a minimal effect on deconvolution. On the other hand, a low signal-to-noise ratio of 10–20 dB makes the noise so prominent that deconvolution becomes quite impractical.

The ideal and simplest form of deconvolution is to measure (or in a few cases calculate from known optical parameters) the point spread function of the system. Computing the Fourier transform of the blurred image and that of the PSF, dividing the second into the first, and performing an inverse transform, produces the deconvolved result as shown previously. The key requirement is that the blur due to the imaging system is assumed to be the same everywhere in the image, which is a good assumption for telescope images (**Figure 56**). When some portion of the blur is due to the passage of light through the sample itself, as occurs in thick sections examined in the light microscope, the blur can vary from point to point and this method becomes less useful.

An indirect measurement of the PSF can be accomplished by capturing an image of a precisely known object using the same optical setup as that used for the real image. Dividing the Fourier transform of the image of the object by that of the ideal shape, and inverse transforming the result, produces the system PSF, which can then be used to deconvolve images obtained with the optical system.

In some microscope situations, the insertion of fluorescing microbeads, or even an image of a small dust particle on the slide, may be useful as an estimate of the PSF. In the atomic force microscope, a direct measurement of the PSF may be accomplished by scanning an image of a known shape, usually a circular test pattern produced expressly for this purpose by the same methods used to etch integrated circuits. As shown in **Figure 57**, if the resulting image is deconvolved by dividing its Fourier transform by that for the ideal shape, the result is an image of the point spread function, which in this case corresponds to the shape of the scanning tip. This shape can then be used to deconvolve other images, at least until the tip is further damaged or replaced.

If an image contains many edges oriented in different directions, and the actual edge shape is known (ideally a perfectly sharp knife-edge transition in grey level), then the PSF can be determined at least in principle by measuring the actual transition across each edge, and combining the various profiles to form a PSF image. In practice this usually requires assuming that the function is symmetrical and of known shape (usually a Gaussian as a convenient approximation to the central portion of the Airy disk produced by real optics), so that measurements on a few edges can be generalized to an entire two-dimensional PSF image.

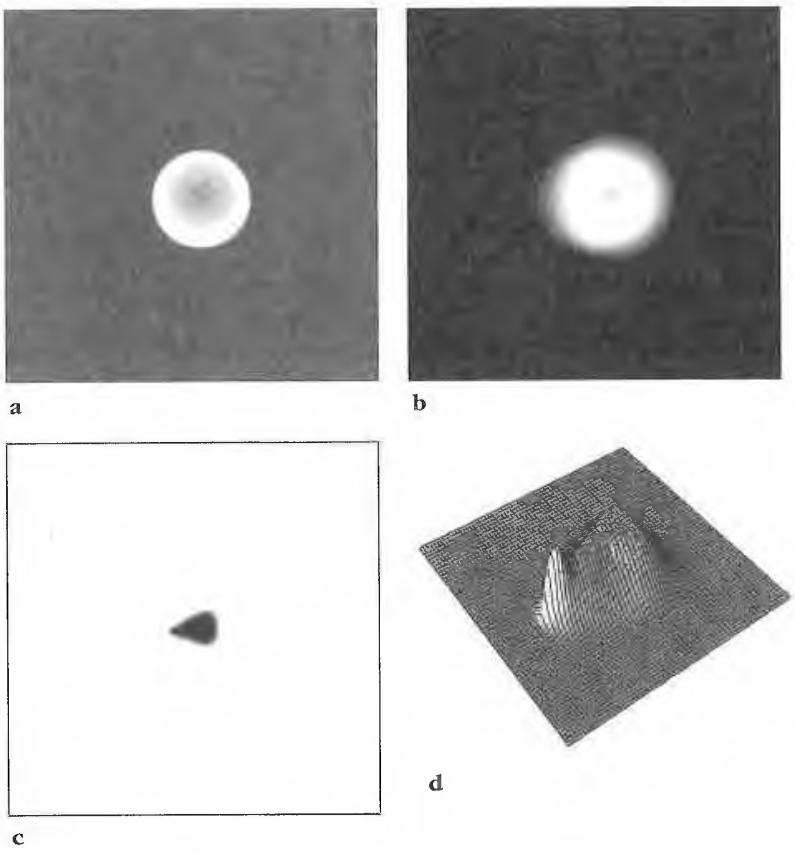


Figure 57. Measuring the PSF of an atomic force microscope: *(**a**) ideal image of a circular test pattern; (**b**) measured image using an imperfect tip; (**c**) deconvolution of **b** by **a**, producing an image of the tip; (**d**) perspective view of the shape in **c**.*

If it is not possible to obtain a PSF by measurement, it may be possible to calculate a useful approximate one based on assumptions such as a Gaussian blur function or straight line motion. Real optical defocusing does not produce an ideal Gaussian blur, but it is not necessary in many cases to have the exact PSF, just a sufficiently useful approximation to remove most of the blur by deconvolution. Sometimes important insight into the nature of the PSF can be obtained by examining the Fourier transform of the blurred image. The presence of zeroes (or near-zero values) in lines, arcs, etc., is a powerful clue to the nature of the blur. Straight lines indicate motion blur, in a direction orthogonal to the lines. Arcs indicate non-Gaussian blurring (e.g., by a uniform disk). These can be used to estimate a PSF which can be used to perform a deconvolution, perhaps iterating the estimated PSF either automatically using one of the convergence methods discussed below, or by interactive selection, to reach a satisfactory result.

The blur produced by purely optical effects is frequently uniform in all directions, although astigmatism can modify this; however, the tip shape in scanned probe microscopes can produce arbitrarily shaped point spread functions. In many cameras the shape and spacing of the transistors produces different blur magnitudes in the horizontal and vertical directions, and the effects of the color filter pattern used can also alter the shape of the PSF. In any scanned image acquisition, the electronic parameters of the amplifiers used can produce different amounts of blur in the fast scan direction (generally horizontal) as compared to the slow scan direction (vertical). Time constants in phosphors, amplifiers or other components can also produce asymmetrical blurs ("comet tails") in the output signal.

Noise and Wiener deconvolution

If there is significant noise content in the image to be sharpened or, worse yet, in the measured PSF, it can exacerbate the numerical precision and overflow problems and greatly degrades the resulting inverse transform. Removal of more than a portion of the blurring in a real image is almost never possible, but of course, some situations exist in which even a small improvement may be of considerable practical importance.

Division by the frequency transform of the blur is referred to as an inverse filter. Using the notation introduced previously, it can be written as

$$F(u,v) \approx \left[\frac{1}{H(u,v)}\right] G(u,v) \tag{12}$$

If the presence of noise in the blurred image prevents satisfactory deconvolution by simply dividing the Fourier transforms, then it may be practical to perform a Wiener deconvolution. Instead of calculating the deblurred image by dividing the Fourier transform of the original image by that of the blur function, a scalar value is used to increase the denominator. Theoretically, the additive factor K is dependent on the statistical properties of the images and their relative noise contents, but in practice these are not usually known and so the additive constant is typically treated as an adjustable parameter that controls the tradeoff between sharpening and noise.

$$F(u,v) \approx \left[\frac{1}{H(u,v)}\right] \cdot \left[\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right] \cdot G(u,v) \tag{13}$$

Figure 58 shows an example of the effect of additive random noise on a image with out-of-focus blur. In the absence of any noise, a direct deconvolution using a Gaussian model for the PSF produces useful results. The addition of random Gaussian noise (SNR = 15 dB) to the image results in a deconvolution using the same PSF that is dominated by noise and ringing that obscures the features. Wiener filtering (**Figure 59**) reduces this problem and allows adjustment which trades off the sharpness of the restoration (higher K values leave more blurring) against the amount of noise (higher K values reduce the noise). Because of the low signal-to-noise ratio, it is not possible in this case to recover all the internal details of the original image.



a b c

Figure 58. Effect of noise on deconvolution: *(a) severely blurred image of the bug in* ***Figure 54*** *with added random noise;* ***(b)*** *deconvolution that can be achieved in the absence of noise;* ***(c)*** *deconvolution with noise present.*

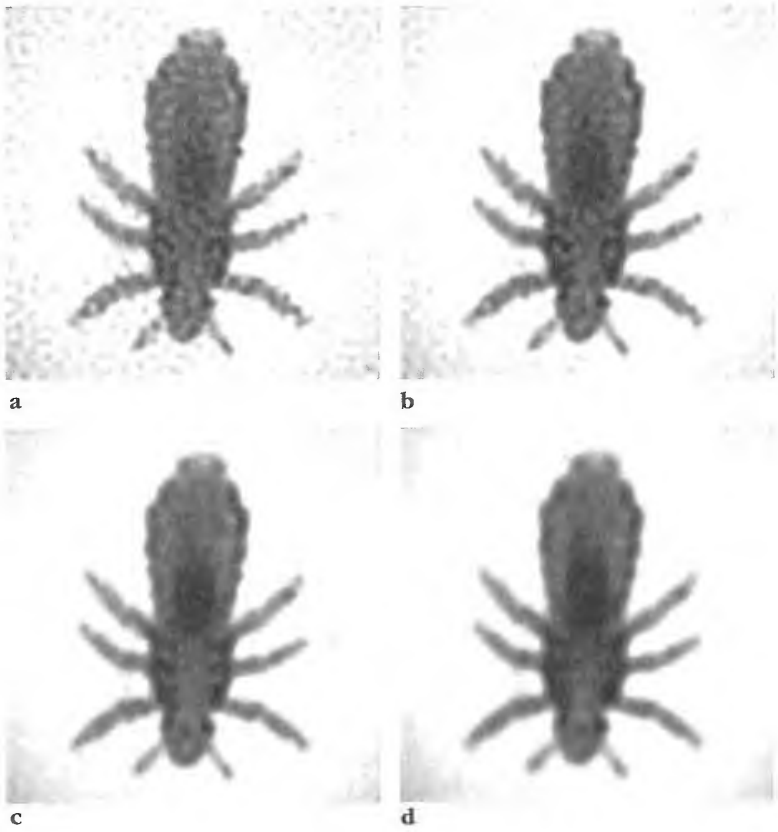


Figure 59. Wiener filtering of the image in Figure 58a: (a–d) *increasing the empirical K value reduces noise at the expense of sharpness.*

The Wiener deconvolution has a long history as a practical method for achieving useful, if not perfect blur removal. Another approach of long standing is the Van Cittert iterative technique. This method has apparently been discovered many times and is also known as a Bially or Landweber iteration. Instead of trying to remove the entire blur in one step, a series of iterative removal steps are performed. In the limit, this iteration would reach the same result as an ideal deconvolution, but instead it is terminated before convergence, resulting in a (partially) deblurred image that does not exhibit an unacceptable noise level. The quality of the results is generally similar to that of the Wiener method.

Many other methods are used, and most of them are much more complicated mathematically (and require much more computation and time). Many of them are iterative, trying to find the best form of the blur function and the deblurred image. Typically this involves trying to solve a very large number of simultaneous equations, which may be over- or under-determined. The techniques for such solutions occupy a significant mathematical literature, and are discussed further in Chapter 11. One of the issues is how to efficiently guide the iteration and how to determine when to terminate it. Another related question is the best measure of image quality, which in different implementations may be based on Bayesian statistics, maximization of entropy, etc. The calculations may be performed in either the frequency or the spatial domain. Even a small PSF in the spatial domain becomes as large as the entire image in the frequency domain, and the number of simultaneous equations is the number of pixels in the image.

Most of these methods are highly specific to particular types of applications, and depend on the *a priori* information that can be supplied, usually in the form of constraints on the solution of the equations. For example, a commonly used constraint is that negative values for pixels have no

physical meaning and so values are not permitted to become negative. If more information on the image, such as the presence of substantial background (as in astronomical or fluorescence images), or known specific pixel values that are permitted, can be incorporated it improves the quality of the result and the efficiency of reaching it. The details of these methods are far beyond the scope of this text, but a very clear and comprehensive revue and comparison can be found in Lagendijk and Biemond (1991).

Another area of current interest is the deconvolution of multiple images that are related to each other. This includes multiple channel images (e.g., different wavelengths or colors) of the same scene, or a series of images from parallel closely spaced planes in semi-transparent specimens (e.g., a series of focal planes in the light microscope). Information from one channel or plane can be used to guide the deconvolution of another.

If the blur is not known *a priori*, it can often be estimated from the power spectrum or by trial and error to find an optimum (or at least useful) result. A recent development in deconvolution that has great efficiency because it is not an iterative method has been published by Carasso (2001). Analysis of the shape of radial profile of the values in the Fourier transform of the blurred image (avoiding the regions dominated by noise) allows constructing an approximate PSF that can be used for deconvolution. This method cannot be used for all types of blur functions (particularly motion blur) but produces very good results for those situations where it is applicable.

An important area for these image restoration techniques is deblurring the images formed by optical sectioning. This is the technique in which a series of images are recorded from different depths of focus in a semitransparent specimen using a light microscope. The passage of light through the overlying layers of the specimen cause a blurring that adversely affects the sharpness and contrast of the images, preventing their being assembled into a three-dimensional stack for visualization and measurement of the three-dimensional structures present.

The confocal light microscope overcomes some of these problems by rejecting light from points away from the point of focus, which improves the contrast of the images (it also reduces the depth of field of the optics, producing higher resolution in the depth axis, but this is important only at the highest magnifications); but the scattering and diffraction of light by the upper layers of the specimen still degrade the image resolution.

In principle, the images of the upper layers contain information that can be used to deconvolute those from below. This would allow sharpening of those images. The entire process is iterative and highly computer-intensive, more so because the blurring of each point on each images may be different from other points. In practice, it is usual to make some assumptions about the blurring and noise content of the images which are used as global averages for a given specimen or for a given optical setup.

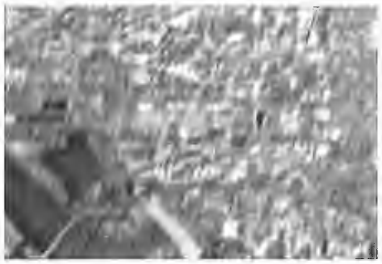
Even with these assumptions, the computations are still intensive and iterative. A considerable theoretical and limited practical literature has been published in this field (Carrington, 1990; Holmes et al., 1991; Monck et al., 1992; Joshi and Miller, 1993; Richardson, 1972; Snyder et al., 1992) A review of several of the leading methods can be found in (Van Kempen et al., 1997). The examples shown deal only with idealized structures and averaging assumptions about the noise characteristics of the images and the point spread function of the microscope, which suggests that restoration of real images will not be as good as the examples shown. Similar concerns and methods can in principle be applied to other *in situ*, three-dimensional imaging techniques such as tomography and seismic imagery.

Motion blur

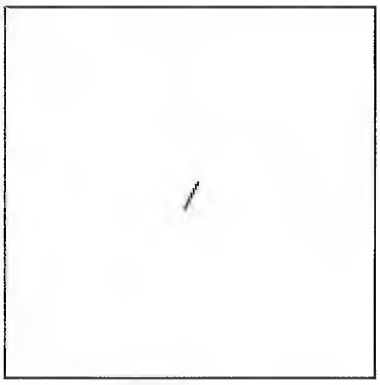
Additional defects besides out-of-focus optics can be corrected by deconvolution as well. These operations are not always performed in the frequency domain, but the basic understanding of the process of removing the blur convolution imposed by the system is most clearly illustrated there. One of the most common defects is blur caused by motion. This is rarely a problem in microscopy applications, but can be very important in remote sensing, in which light levels are low and the exposure time must be long enough for significant camera motion to occur with respect to the scene. Fortunately, in most of these circumstances the amount and direction of motion is known. That makes it possible to draw a line in the spatial domain that defines the blur. The frequency transform of this line is then divided into the transform of the blurred image. Retransforming the resulting image restores the sharp result. **Figure 60** illustrates the method.

It is important to note the similarity and the difference between this example and the removal of out-of-focus blur. Both involve dividing the transform of the blurred image by that of the defect. This follows directly from the equation presented for convolution, in which the transform of the convolved image is the product of those from the original image and the defect. The major difference is that in the motion blur case we can sometimes calculate the exact blurring vector to be removed, while in the out-of-focus blur case we must estimate the blur from an actual image. This estimation introduces unavoidable noise in the image, which is greatly magnified by the division.

As in the example shown previously for out-of-focus blur, the use of a Wiener deconvolution can partially alleviate the effects of noise. In the example of **Figure 61**, the motion blur can be estimated from the orientation and spacing of the lines of zero (or near zero) values in the power spectrum of the Fourier transform of the blurred image. Using this to determine the motion vector as the point spread function allows performing a deconvolution. Increasing the Wiener K factor controls the tradeoff between image sharpness and noise as shown in **Figure 62**. This factor is usually treated as an empirical adjustment. Human observers generally prefer an image that is noisier but less blurred than a theoretical K value based on the noise content delivers.



a



b



c



d

Figure 60. Removal of motion blur: ***(a)*** *original image (the Mall and tidal basin in Washington, D.C.);* ***(b)*** *motion vector based on airplane speed, direction, and exposure time;* ***(c)*** *deconvolution result;* ***(d)*** *deconvolution result using 8-bit images and single-precision arithmetic.*

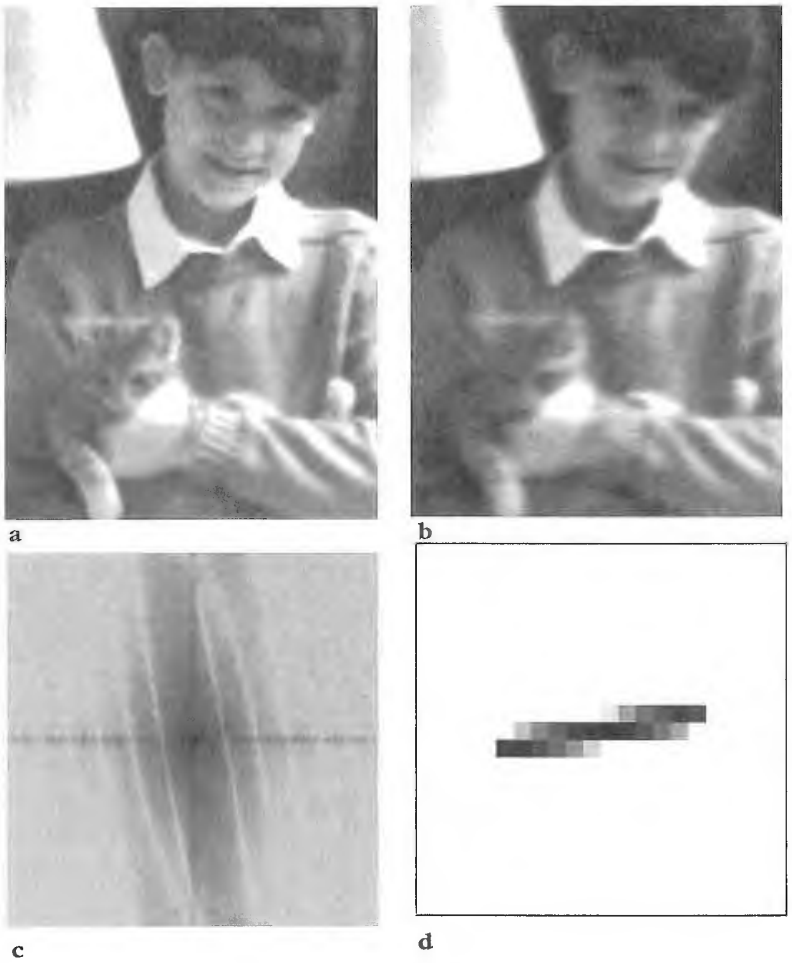


Figure 61. Motion blur: *(a) unblurred image; (b) uniform motion blur; (c) power spectrum of the Fourier transform of b; (d) motion vector shown as a point spread function (enlarged to show pixels).*

Increasing the value of the constant K in the denominator of equation 13 reduces the magnitude of all terms in the complex Fourier transform that is inverted to obtain the restored image, but it reduces terms that were originally small more than it does large ones. These terms are the ones most likely to be dominated by random noise. One effect of this reduction is the elimination of more of the small terms by apodization. **Figure 63** shows the terms eliminated in the Wiener filtered restoration of the image (compare it to the FT power spectrum of the original image in **Figure 61c**).

Additional limitations arise from the finite internal precision in computer implementation of the Fourier transform and the storage of the complex frequency-transform image. Programs may use either floating-point or fixed-point notation, the latter requiring less storage space and offering somewhat greater speed; double precision internal values give the best results because the minimum magnitude values that can be accurately recorded generally occur at higher frequencies, controlling the appearance of sharp edges in the spatial-domain image. Too small a precision in either the image or the calculation limits the ability to sharpen images by removing degrading blur due to the imaging system. To illustrate the problems introduced by images with insufficient precision, in **Figure 60d**, the original image was reduced from a bit depth of 12 (more than 4000 grey scale values obtained by scanning a 35-mm film negative) to 6 bits (the performance of a typical video camera). Note the vastly inferior (even useless) quality of the results.

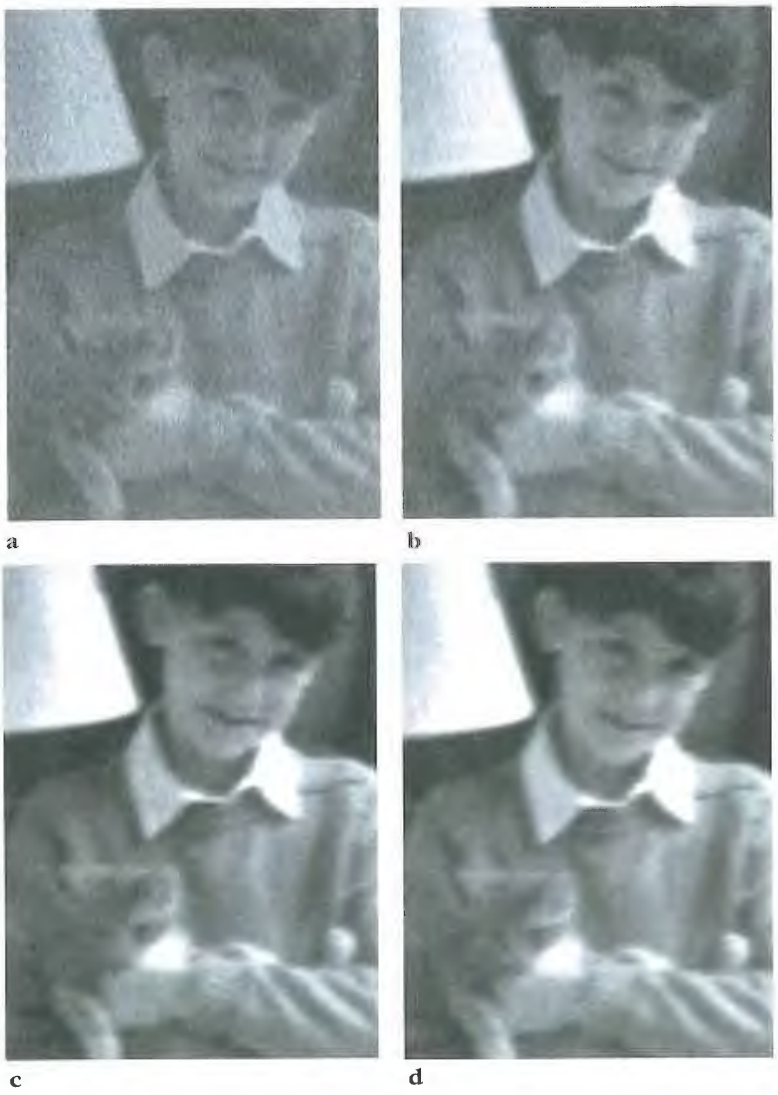


Figure 62. ***Deconvolution of the image in Figure 61b:*** ***(a)*** *Wiener constant K set to zero;* ***(b–d)*** *increasing K values covering the range from too noisy to too blurred.*

Figure 63. *Power spectrum of the image in* ***Figure 62*** *showing in red the values eliminated from the restored image by apodization (compare with* ***Figure 61c****).*



Template matching and correlation

Closely related to the spatial-domain application of a kernel for smoothing, derivatives, etc. is the idea of template matching or cross-correlation. In this case, a target pattern is shifted to every location in the image, the values are multiplied by the pixels that are overlaid, and the total is stored at that position to form an image showing where regions identical or similar to the target are located. **Figure 64** illustrates this process. The multiplication and summation process is identical to convolution, except that the target is rotated 180° first, so that the upper left corner value in the target pattern is multiplied by the lower right value in the neighborhood on the image and so forth. When the process is performed in frequency space, this amounts to a phase shift of the Fourier transform values.

This method is used in many contexts to locate features within images. One is searching reconnaissance images for particular objects such as vehicles. Tracking the motion of hurricanes in a series of weather satellite images or cells moving on a microscope slide can also use this approach. Modified to deal optimally with binary images, it can be used to find letters in text. When the target is a pattern of pixel brightness values from one image in a stereo pair and the searched image is the second image from the pair, the method can be used to perform fusion (locating matching points in the two images) to measure parallax and calculate range, as shown in Chapter 13.

For continuous two-dimensional functions, the cross-correlation image is calculated as

$$c(i,j) = \iint f(x,y)g(x-i,y-j)dx\,dy \tag{14}$$

Replacing the integrals by finite sums over the dimensions of the image gives **Equation 15**. In order to normalize the result of this template matching or correlation without the absolute brightness value of the region of the image biasing the results, the operation in the spatial domain is usually calculated as the sum of the products of the pixel brightnesses divided by their geometric mean.

$$\frac{\sum_{i,j} f_{x+i,y+j} \cdot g_{i,j}}{\sqrt{\sum_{i,j} f^2_{x+i,y+j} \cdot \sum_{i,j} g^2_{i,j}}} \tag{15}$$

When the dimensions of the summation are large, this is a slow and inefficient process compared to the equivalent operation in frequency space. The frequency-space operation is simply

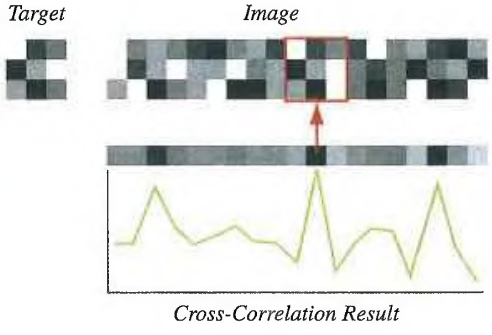


Figure 64. *Cross-correlation matches a pattern of grey scale values to many points in the target image to find the location with the best match. In the tiny example shown here, the 3 × 3 target is applied to every possible position in the image to produce the line of values shown. The marked location is most similar (but not identical) to the target.*

$$C(u,v) = F(u,v)G^*(u,v) \tag{16}$$

where * indicates the complex conjugate of the function values. The complex conjugate of the pixel values affects only the phase of the complex values. The operation is thus seen to be very similar to convolution, and indeed it is often performed using many of the same program subroutines. Operations that involve two images (division for deconvolution, multiplication for convolution, and multiplication by the conjugate for correlation) are sometimes called dyadic operations, to distinguish them from filtering and masking operations (monadic operations) in which a single frequency transform image is operated on (the filter or mask image is not a frequency-domain image nor does it contain complex values).

Usually, when correlation is performed, the wraparound assumption joining the left and right edges and the top and bottom of the image is not acceptable. In these cases, the image should be padded by surrounding it with zeroes to bring it to the next larger size for transformation (the next exact power of two, required by many FFT routines). Because the correlation operation also requires that the actual magnitude values of the transforms be used, slightly better mathematical precision can be achieved by padding with the average values of the original image brightnesses rather than with zeroes. It may also be useful to subtract the average brightness value from each pixel, which removes the zeroth (DC) term from the transformation. This value is usually the largest in the transform (it is the value at the central pixel), so its elimination allows the transform data more dynamic range.

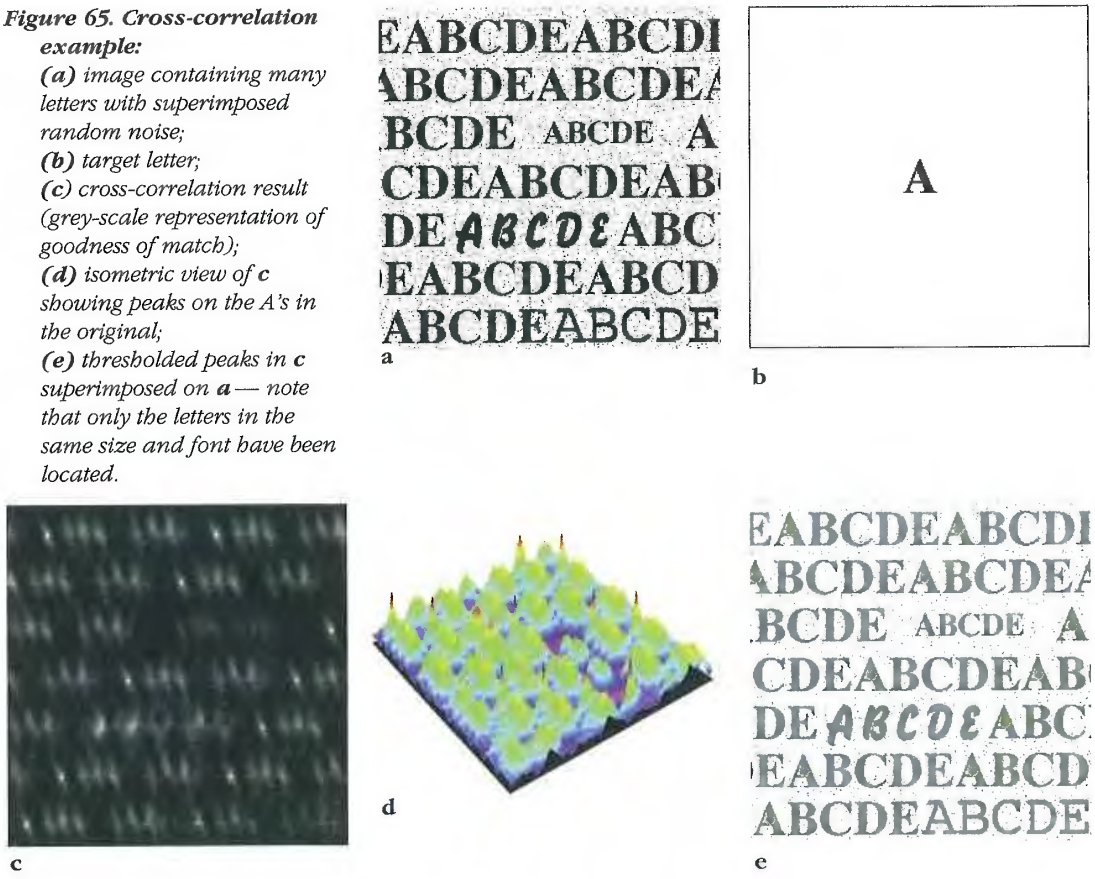


Figure 65. Cross-correlation example:
(a) *image containing many letters with superimposed random noise;*
(b) *target letter;*
(c) *cross-correlation result (grey-scale representation of goodness of match);*
(d) *isometric view of* ***c*** *showing peaks on the A's in the original;*
(e) *thresholded peaks in* ***c*** *superimposed on* ***a*** *— note that only the letters in the same size and font have been located.*

Correlation is primarily used for locating features in one image that appear in another. **Figure 65** shows an example. The image contains text, while the target contains the letter "A" by itself. The result of the cross-correlation after retransforming the image to the spatial domain shows peaks where the target letter is found, which may be more apparent when the same image presented as an isometric display. The brightest points in the correlation image correspond to the occurrences of the letter "A" in the same size and font as the target. Lower but still significant peaks exist, corresponding to two of the other letter As, in different fonts. In general, however, cross-correlation is quite size- and shape-specific as discussed in Chapter 10.

When combined with other image processing tools to analyze the cross-correlation image, this is an extremely fast and efficient tool for locating known features in images. In the example of **Figure 66**, a very large number of SEM images of Nuclepore filters with latex spheres presented a problem for automatic counting because of the variation in contrast and noise, the presence of dirt, and the texture presented by the filters. Also, the contrast of an isolated sphere is consistently different than that of one surrounded by other spheres.

A target image (**Figure 66a**) was created by averaging together ten representative latex spheres. This was then cross-correlated with each image, a top-hat filter (inner radius = 3 pixels; outer radius = 5 pixels; height = 8 grey scale values) applied to isolate each peak in the image, and the resulting spots counted automatically. The figure shows a few example images from the set, with marks showing the particles that were found and counted. After the top hat has been applied to select the peaks in the cross-correlation image, the resulting spots can be convolved with the stored target image to produce the result shown in **Figure 67**, which is just the latex spheres without the other features present in the original image.

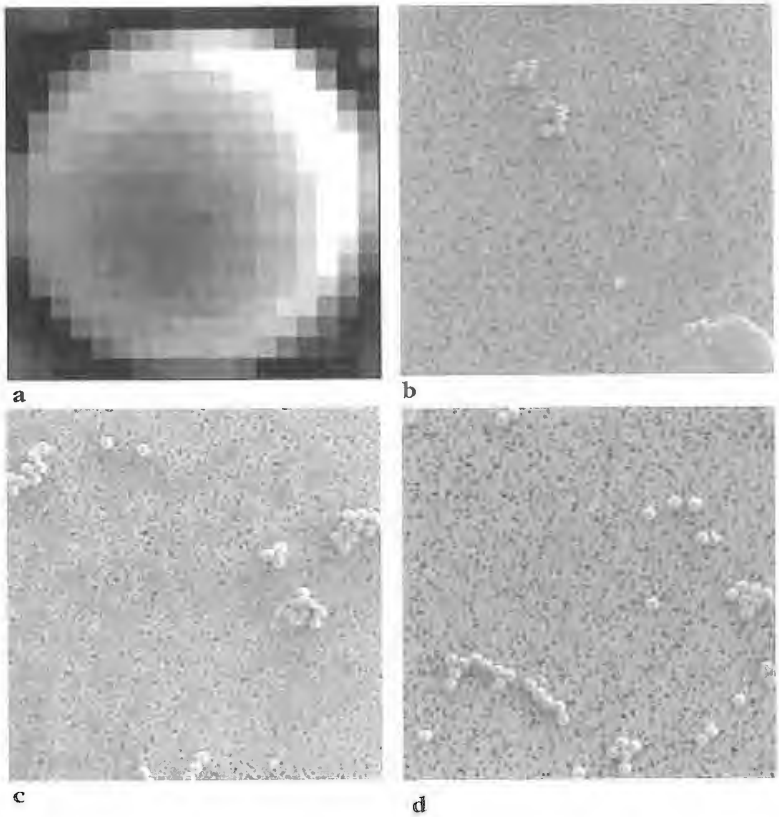


Figure 66. Cross-correlation to detect and count latex spheres on Nuclepore filters: (a) average of ten representative latex spheres, used as a target; (b–d) representative example images, with red spots marking the result of the procedure described in the text.

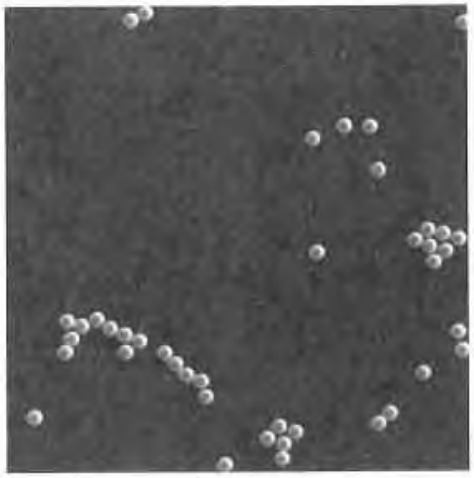


Figure 67. *Convolution of the target image from* ***Figure 66a*** *with the binary image of peaks found by the top hat filter applied to the cross-correlation image (the red marks superimposed on* ***Figure 66d****), showing just the latex spheres.*

Cross-correlation is also used as a tool for aligning serial section images. Even if the images are not identical (as in general they are not), enough common features exist so that a sharp cross-correlation peak occurs when two successive images are in best x,y alignment. The location of the peak can be determined to sub-pixel accuracy and used to shift the images into optimum alignment. The method does not deal with rotation, but it is possible to iteratively rotate the images and use the peak cross-correlation value as a measure of quality to determine the best value. Serial section alignment is discussed further in Chapter 12.

Autocorrelation

In the special case when the image functions f and g (and their transforms F and G) are the same, the correlation operation is called autocorrelation. This is used to combine together all parts of the image, in order to find repetitive structures. In Chapter 13, this method is used to measure the texture of surface roughness in terms of the characteristic distance over which amplitude of the peak drops. **Figure 68** shows another use of the autocorrelation function, to measure preferred orientation. The image is a felted textile material. Due to the fabrication process the fibers have neither a randomized nor uniform orientation; neither are they regularly arranged. The frequency transform shown indicates a slightly asymmetric shape. Performing the cross-correlation and retransforming to the spatial domain gives the image shown in **Figure 69**. The brightness profiles of the central spot along its major and minor axes show the preferred orientation quantitatively. This is the same result achieved by cross-correlation in the spatial domain (rotating the image by 180° and sliding the image across itself while recording the area of feature overlap as a function of relative offset) to determine preferred orientation.

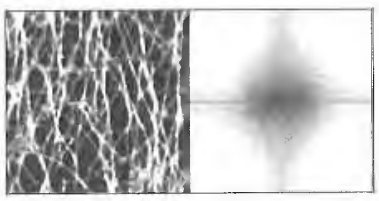


Figure 68. *Image of felted textile fibers (left) and the frequency transform (right).*

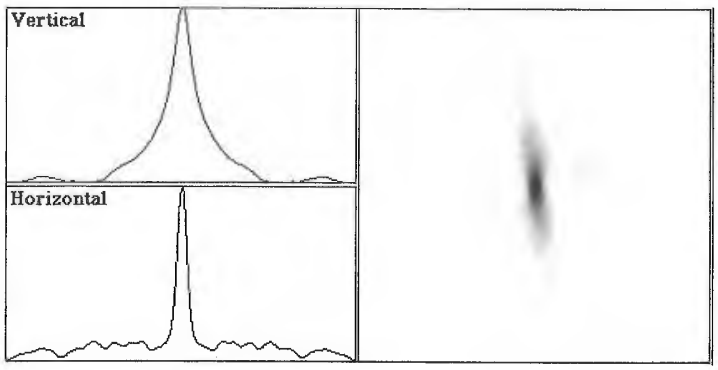


Figure 69. *Autocorrelation image from* ***Figure 68*** *(right), and its horizontal and vertical brightness profiles (left) along the major and minor axes of the peak.*

Conclusion

Processing images in the frequency domain is useful for removing certain types of noise and image blur, applying large convolution kernels, enhancing periodic structures, and locating defined structures in images. It can also be used to measure images to determine periodicity or preferred orientation. All these operations can be carried out in the spatial domain, but are often much more efficient in the frequency domain. The FFT operations can be efficiently computed in small (desktop) systems. This makes frequency-domain operations useful and important tools for image analysis.

6

Segmentation and Thresholding

One of the most critical steps in the process of reducing images to information is segmentation: dividing the image into regions that presumably correspond to structural units in the scene or distinguish objects of interest. Segmentation is often described by analogy to visual processes as a foreground/background separation, implying that the selection procedure concentrates on a single kind of feature and discards the rest.

This is not quite true for computer systems, which can generally deal much better than humans with scenes containing more than one type of feature of interest. **Figure 1** shows a common optical illusion that can be seen as a vase or as two facing profiles, depending on whether we concentrate on the white or black areas as the foreground. It appears that humans are unable to see both interpretations at once, although we can flip rapidly back and forth between them once the two have been recognized. This is true in many other illusions as well. **Figure 2** shows two others. The cube can be seen in either of two orientations, with the dark corner close to or far from the viewer; the sketch can be seen as either an old woman or a young girl. In both cases, we can switch between versions very quickly, but we cannot perceive them both at the same time.

These illusions work because human vision interprets the image in terms of the relationships between structures, which have already been unconsciously constructed at lower levels in the visual pathway. In computer-based image analysis systems, we must start at those low levels and work upwards. The initial decisions must be made at the level of individual pixels.

Thresholding

Selecting features within a scene or image is an important prerequisite for most kinds of measurement or understanding of the scene. Traditionally, one simple way this selection has been accomplished is to define a range of brightness values in the original image, select the pixels within this range as belonging to the foreground, and reject all of the other pixels to the background. Such an image is then usually displayed as a binary or two-level image, using black and white or other colors to distinguish the regions. (There is no standard convention on whether the features of interest are white or black; the choice depends on the particular display hardware in use and the designer's preference. In the examples shown here, the features are black and the background is white.)

Figure 1. *The "Vase" illusion. Viewers may see either a vase or two human profiles in this image, and can alternate between them, but cannot see both interpretations at the same time.*



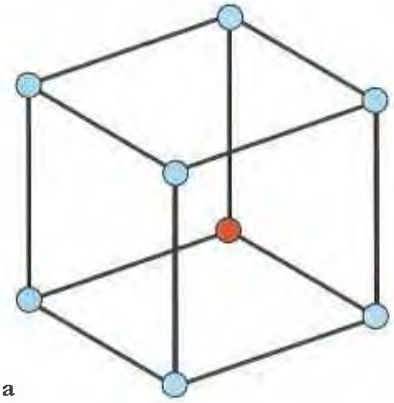




Figure 2. Examples of either/or interpretation:
(a) *the Necker cube, in which the dark corner may appear close to or far from the viewer;*
(b) *the "old woman/young girl" sketch.*

This operation is called thresholding. Thresholds may be set interactively by a user watching the image and using a colored overlay to show the result of turning a knob or otherwise adjusting the settings. As a consequence of the ubiquitous use of a mouse as the human interface to a graphical computer display, the user may adjust virtual sliders or mark a region on a histogram to select the range of brightness values. The brightness histogram of the image (or a region of it) is often useful for making adjustments. As discussed in earlier chapters, this is a plot of the number of pixels in the image having each brightness level. For a typical 8-bit monochrome image, this equals 2^8 or 256 grey scale values. The plot may be presented in a variety of formats, either vertical or horizontal, and some displays use color or grey scale coding to assist the viewer in distinguishing the white and black sides of the plot.

Systems that also handle images with a greater tonal range than 256 grey levels (8 bits per color channel), as are obtained from scanners, some digital cameras, and other instruments, may allow up to 16 bits (65,536 distinct pixel values). As a matter of convenience and consistency, both for

thresholding purposes and to preserve the meaning of the various numeric constants introduced in preceding chapters, such images may still be described as having a brightness range of 0–255 to cover the range from black to white. But instead of being limited to integer brightness values the greater precision of the data allows brightnesses to be reported as real numbers (e.g., a pixel value of 31,605 out of 65,536 would be divided by 256 and reported as 123.457). This does not solve the problem of displaying such a range of values in a histogram. A full histogram of more than 65,000 values would be too wide for any computer screen, and for a typical image size would have so few counts in each channel as to be uninterpretable. One solution is to divide the data down into a conventional 256 channel histogram for viewing, but allow selective expansion of any part of it for purposes of setting thresholds, as indicated in **Figure 3**.

Examples of histograms have been shown in earlier chapters. Note that the histogram counts pixels in the entire image (or in a defined region of interest), losing all information about the original location of the pixels or the brightness values of their neighbors. Peaks in the histogram often identify the various homogeneous regions (often referred to as phases, although they correspond to a phase in the metallurgical sense only in a few applications) and thresholds can then be set between the peaks. There are also automatic methods to adjust threshold settings (Prewitt and Mendelsohn, 1966; Weszka, 1978; Otsu, 1979; Kittler et al., 1985; Russ and Russ, 1988a; Rigaut, 1988; Lee et al., 1990; Sahoo et al., 1988; Russ, 1995c), using either the histogram or the image itself as a guide, as we will see in the following paragraphs. Methods that compare to *a priori* knowledge the measurement parameters obtained from features in the image at many threshold levels (Wolf, 1991) are too specialized for discussion here.

Many images have no clear-cut set of histogram peaks that correspond to distinct phases or structures in the image. **Figure 4** shows a real-world image in which this is the case. Under the more controlled lighting conditions of a microscope, and with sample preparation that includes selective staining or other procedures, this condition is more likely to be met. In some of the difficult cases, direct thresholding of the image is still possible but the settings are not obvious from examination of the histogram peaks. In many situations, the brightness levels of individual pixels are not uniquely related to structure. In some of these instances, prior image processing can be used to transform the original brightness values in the image to a new image, in which pixel brightness represents some derived parameter such as the local brightness gradient or direction.

Histogram analysis is sometimes done by fitting Gaussian (or other shape) functions to the histogram. Given the number of Gaussian peaks to combine for a best overall fit, the position, width and height of each can be determined by multiple regression. This permits estimating the area of each phase but of course cannot determine the spatial position of the pixels in each phase if the peaks overlap. And the result is generally poor because few real imaging situations produce ideally

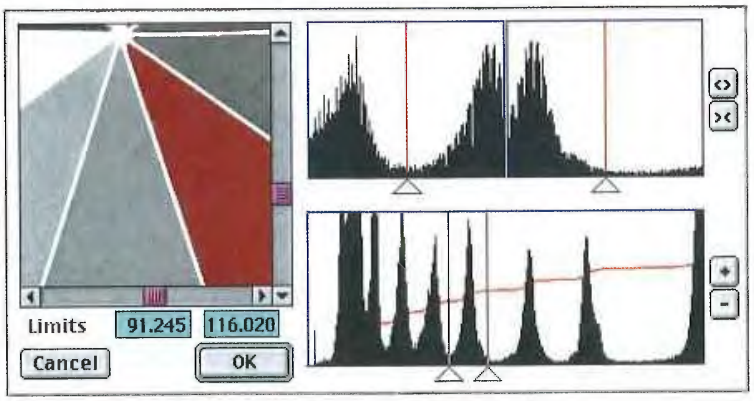


Figure 3. *Example of thresholding a histogram of a 16-bit image, with the bottom graph representing a 256-bin brightness distribution and the top graphs showing expanded details of the values near the threshold settings.*

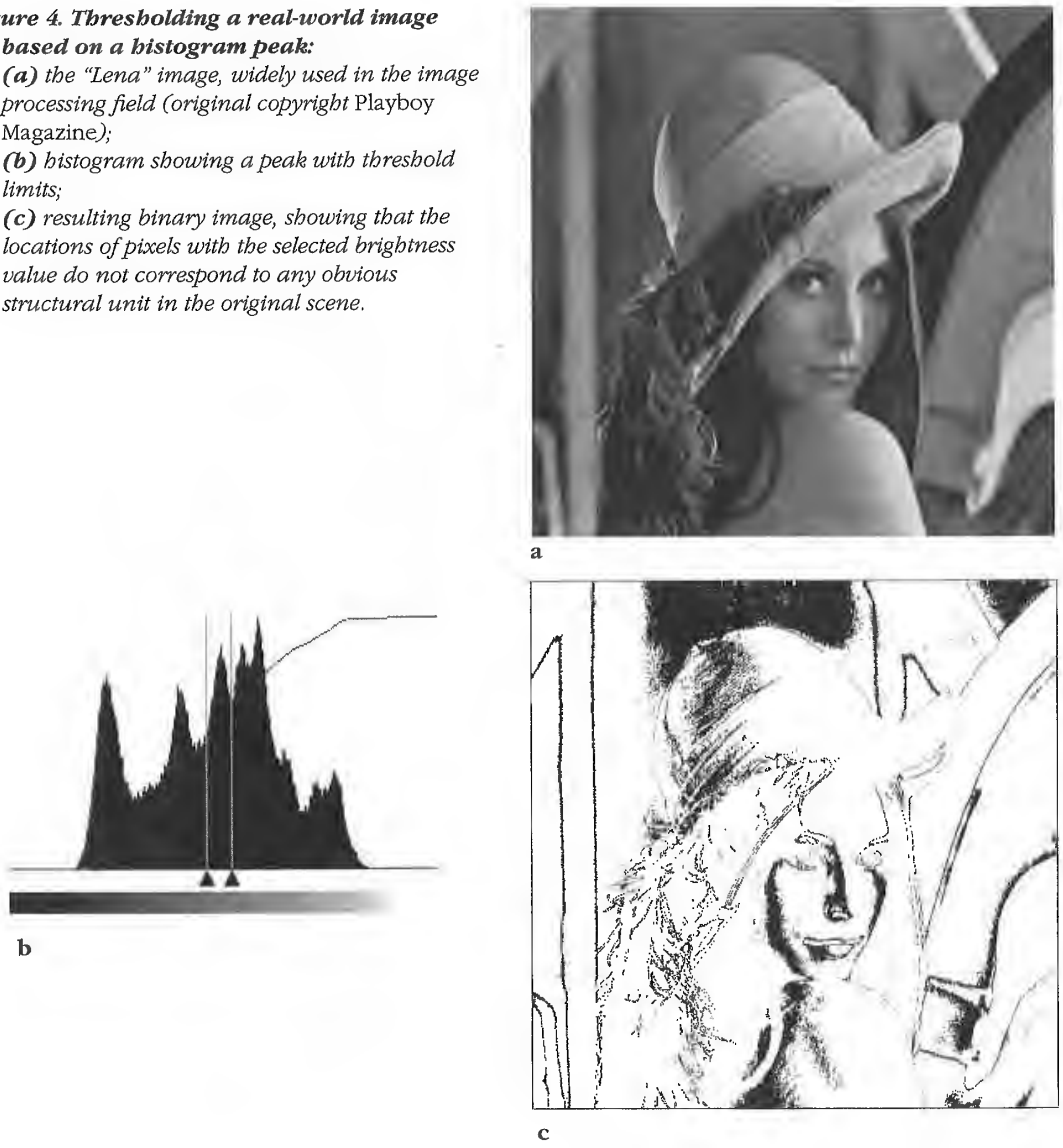


Figure 4. Thresholding a real-world image based on a histogram peak:
(a) *the "Lena" image, widely used in the image processing field (original copyright* Playboy Magazine*);*
(b) *histogram showing a peak with threshold limits;*
(c) *resulting binary image, showing that the locations of pixels with the selected brightness value do not correspond to any obvious structural unit in the original scene.*

Gaussian peaks (or any other consistent shape). One exception is magnetic resonance imaging (MRI), where this method has been applied to produce images in which pixels in the overlap areas are not converted to black or white, but shaded according to the relative contributions of the two overlapping Gaussian peaks at that brightness value (Frank et al., 1995). This does not, of course, produce a segmented image in the conventional sense, but it can produce viewable images that delineate the overlapped structures (e.g., dark matter and white matter in brain scans).

Multiband images

In some cases, segmentation can be performed using multiple original images of the same scene. The most familiar example is that of color imaging, which uses different wavelengths of light. For satellite imaging in particular, this may include several infrared bands containing important information for selecting regions according to vegetation, types of minerals, and so forth (Haralick and

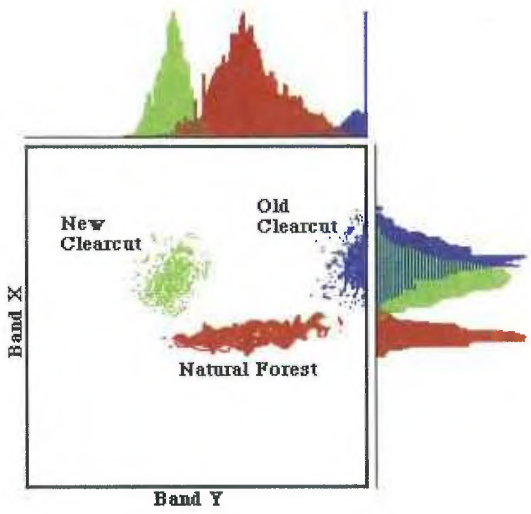


Figure 5. *Example of terrain classification from satellite imagery using multiple spectral bands. Overlaps in each band require that both be used to distinguish the types of terrain.*

Dinstein, 1975). **Figure 5** shows an example. A series of images obtained by performing different processing operations on the same original image can also be used in this way. Examples include combining one image containing brightness data, a second containing local texture information, etc., as will be described in the following paragraphs.

In general, the more independent color bands or other images that are available, the easier and better the job of segmentation that can be performed. Points that are indistinguishable in one image may be fully distinct in another; however, with multispectral or multilayer images, it can be difficult to specify the selection criteria. The logical extension of thresholding is simply to place brightness thresholds on each image, for instance to specify the range of red, green, and blue intensities. These multiple criteria are then usually combined with an AND operation (i.e., the pixel is defined as part of the foreground if its three RGB components all lie within the selected ranges). This is logically equivalent to segmenting each image plane individually, creating separate binary images, and then combining them with a Boolean AND operation afterward. Such operations to combine multiple binary images are discussed in Chapter 7.

The reason for wanting to combine the various selection criteria in a single process is to assist the user in defining the ranges for each. The optimum settings and their interaction are not particularly obvious when the individual color bands or other multiple image brightness values are set individually. Indeed, simply designing a user interface which makes it possible to select a specific range of colors for thresholding a typical visible light image (usually specified by the RGB components) is not easy. A variety of partial solutions are in use.

This problem has several aspects. First, while red, green, and blue intensities represent the way the detector works and the way the data are stored internally, they do not correspond to the way that people recognize or react to color. As discussed in Chapter 1, a system based on hue, saturation, and intensity (HSI) or lightness is more familiar. It is sometimes possible to perform satisfactory thresholding using only one of the hue, saturation or intensity planes as shown in **Figure 6**, but in the general case it may be necessary to use all of the information. A series of histograms for each of the RGB color planes may show peaks, but the user is not often able to judge which of the peaks correspond to individual features of interest.

Even if the RGB pixel values are converted to the equivalent HSI values and histograms are constructed in that space, the use of three separate histograms and sets of threshold levels still does



a b c

Figure 6. Thresholding a color image using a single channel:
(a) *original stained biological thin section;*
(b) *hue values calculated from stored red, green, and blue values;*
(c) *thresholding on the hue image delineates the stained structures.*

little to help the user see which pixels have various combinations of values. For a single monochrome image, various interactive color-coded displays allow the user either to see which pixels are selected as the threshold levels are adjusted or to select a pixel or cluster of pixels and see where they lie in the histogram. **Figure 7** shows an example, though because it consists of still images, it cannot show the live, real-time feedback possible in this situation.

For a three-dimensional color space, either RGB or HSI, interactive thresholding is more difficult. There is no easy or obvious way with present display or control facilities to interactively enclose an arbitrary region in three-dimensional space and see which pixels are selected, or to adjust that region and see the effect on the image. It is also helpful to mark a pixel or region in the image and see the color values (RGB or HSI) labeled directly in the color space. For more than three colors (e.g., the multiple bands sensed by satellite imagery), the situation is even worse.

Using three one-dimensional histograms and sets of threshold levels, for instance in the RGB case, and combining the three criteria with a logical AND selects pixels that lie within a portion of the color space that is a simple prism, as shown in **Figure 8.** If the actual distribution of color values has some other shape in the color space, for instance if it is elongated in a direction not parallel to one axis, then this simple rectangular prism is inadequate to select the desired range of colors.

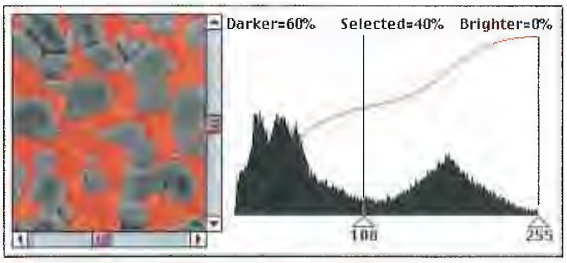
Two-dimensional thresholds

A somewhat better bound can be set by using a two-dimensional threshold. This can be done in any color coordinates (RGB, HSI, etc.), but in RGB space it is difficult to interpret the meaning of the settings. This is one of the (many) arguments against the use of RGB for color images; however, the method is well-suited for color images encoded by hue and saturation (HS). The HS plane can be represented as a circle, in which direction (angle) is proportional to hue and radius is proportional to saturation (**Figure 9**). The intensity or lightness of the image is perpendicular to this plane and requires another dimension to show or to control.

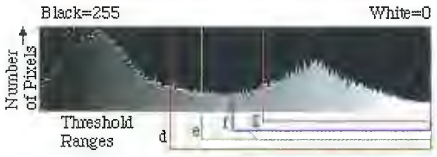
Instead of a one-dimensional histogram of brightness in a monochrome image, the figure shows a two-dimensional display in the HS plane. The number of pixels with each pair of values of hue and saturation can be plotted as a brightness value on this plane, representing the histogram with its dark peaks. Thresholds can be selected as a region that is not necessarily simple, convex, or even connected, and so can be adapted to the distribution of the actual data. **Figure 9b** illustrates this



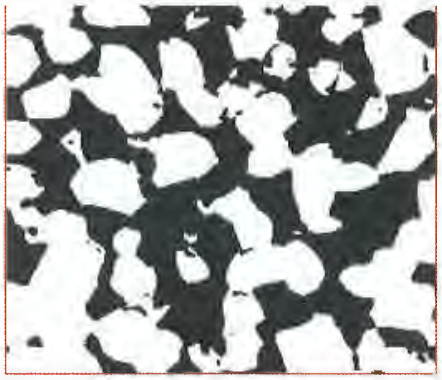
a



b



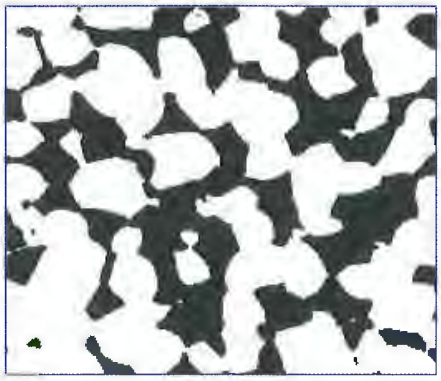
c



d



e



f



g

Figure 7. Thresholding a grey-scale image:
(a) *original SEM image of a two-phase ceramic;*
(b) *typical thresholding user dialog with upper and lower limits and a preview of the results;*
(c) *several threshold ranges marked on the image histogram;*
(d–g) *corresponding binary images produced by changing the settings of the threshold values used to select pixels, as shown on the histogram. The area fraction of the light phase varies with these setting from about 33 to 48% in these results.*

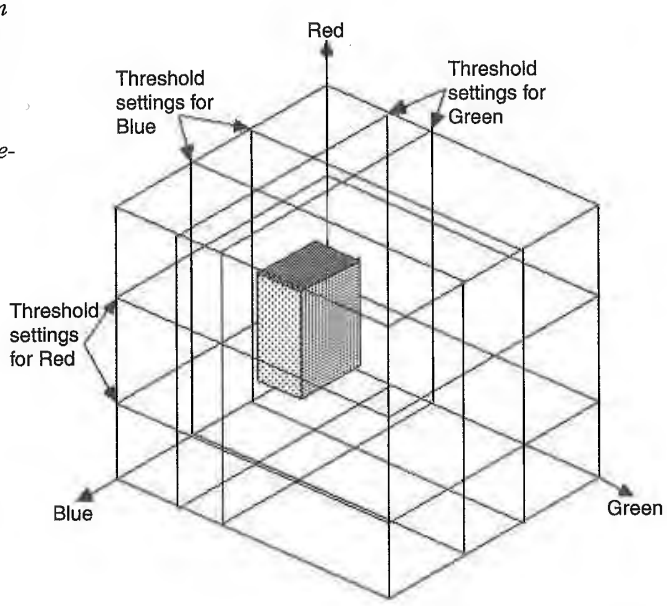


Figure 8. *Illustration of the combination of separate thresholds on individual color planes. The shaded area is the "AND" of the three threshold settings for red, green, and blue. The only shape that can be formed in the three-dimensional space is a rectangular prism.*

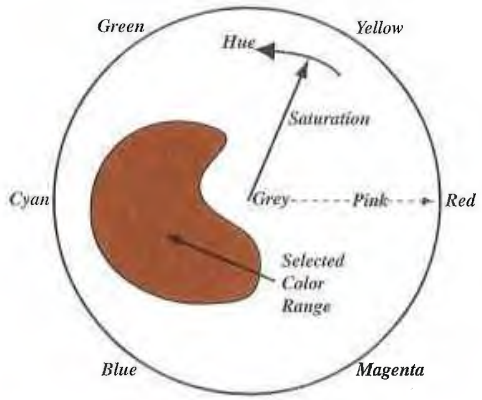
method. It is also possible to find locations in this plane as a guide to the user in the process of defining the boundary, by pointing to pixels in the image so that the program can highlight the location of the color values on the HS circle. In the figure, the third axis (intensity) is shown with a conventional histogram and adjustable limits.

Similar histogram displays and threshold settings can be accomplished using other planes and coordinates. For color images, the HS plane is sometimes shown as a hexagon (with red, yellow, green, cyan, blue, and magenta corners). The CIE color diagram shown in Chapter 1 is also a candidate for this purpose. For some satellite images, the near and far infrared intensities form a plane in which combinations of thermal and reflected IR can be displayed and selected.

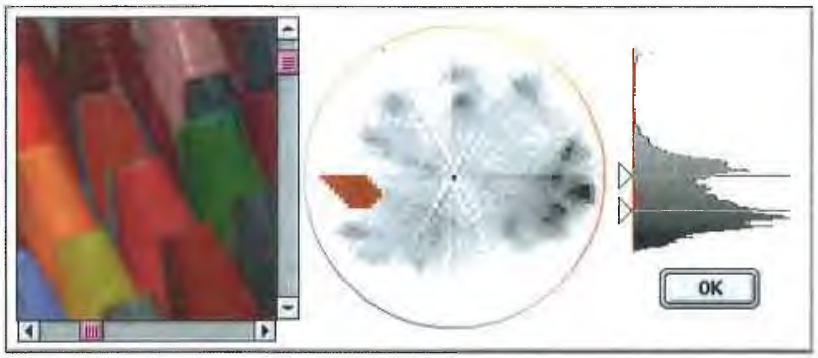
As a practical matter, the HS plane is sometimes plotted as a square face on a cube that represents the HSI space. This is simpler for the computer graphics display and is used in several of the examples that follow. The HSI cube with square faces is topologically different, however, from the cone or bi-cone used to represent HSI space in Chapter 1, and the square HS plane is topologically different from the circle in **Figure 9**. In the square, the minimum and maximum hue edges (400 nm = red and 700 nm = violet) are far apart, whereas in the circle, hue is a continuous function that wraps around. This makes using the square for thresholding somewhat less intuitive, but it is still superior in most cases to the use of RGB color space.

For the two-dimensional square plot, the axes may have unusual meanings, but the ability to display a histogram of points based on the combination of values and to select threshold boundaries based on the histogram is a significant advantage over multiple one-dimensional histograms and thresholds, even if it does not generalize easily to the n-dimensional case.

The dimensions of the histogram array are usually somewhat reduced from the actual resolution (typically one part in 256) of the various RGB or HSI values for the stored image. This is not only because the array size would become very large (256^2 = 65,536 for the square, 256^3 = 16,777,216 for the cube). Another reason is that for a typical real image, there are simply not that many distinct pairs or triples of values present, and a useful display showing the locations of peaks and clusters can be presented using fewer bins. The examples shown here use 32 × 32 bins for each of the square faces of the RGB or HSI cubes, each of which thus requires 32^2 = 1024 storage locations.



a



b



c

Figure 9. Illustration of selecting an arbitrary region in a two-dimensional parameter space (here, the hue/saturation circle) to define a combination of colors to be selected for thresholding:
(a) *schematic view;*
(b) *implementation in which the HSI values for a single crayon are selected from the histogram (which is shown on the hue-saturation circle as grey-scale values representing the number of image pixels — note the clusters for each crayon color);*
(c) *resulting image in which the selected pixels are outlined in white.*

It is possible to imagine a system in which each of the two-dimensional planes defined by pairs of signals is used to draw a contour threshold, then project all of these contours back through the multi-dimensional space to define the thresholding, as shown in **Figure 10**. As the dimensionality increases, however, so does the complexity for the user, and the AND region defined by the multiple projections still cannot fit irregular or skewed regions very satisfactorily.

Multiband thresholding

Figure 6 showed a color image from a light microscope. The microtomed thin specimen of intestine has been stained with two different colors, so that there are variations in shade, tint, and tone.

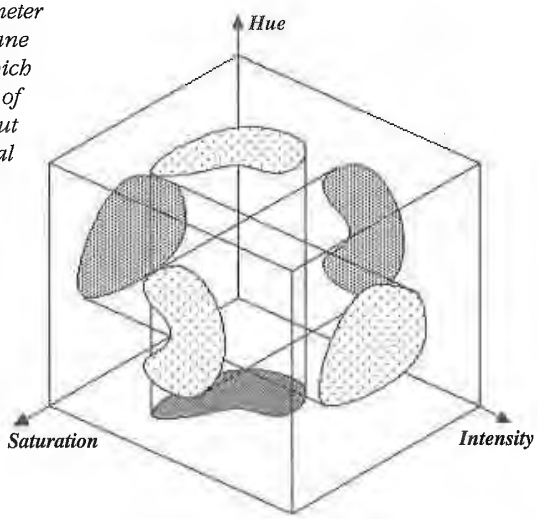


Figure 10. *Illustration of the combination of 2-parameter threshold settings. Outlining of regions in each plane defines a shape in the three-dimensional space which is more adjustable than the Boolean combination of simple one-dimensional thresholds in* ***Figure 8****, but still cannot conform to arbitrary three-dimensional cluster shapes.*

Figure 10 shows the individual red, green, and blue values. The next series of figures illustrates how this image can be segmented by thresholding to isolate a particular structure using this information.

Figure 11 shows the individual brightness histograms of the red, green, and blue color planes in the image, and **Figure 12** shows the histograms of pixel values, projected onto the red/green, green/blue, and blue/red faces of the RGB color cube. Notice that there is a trend on all faces for the majority of pixels in the image to cluster along the central diagonal in the cube. In other words,

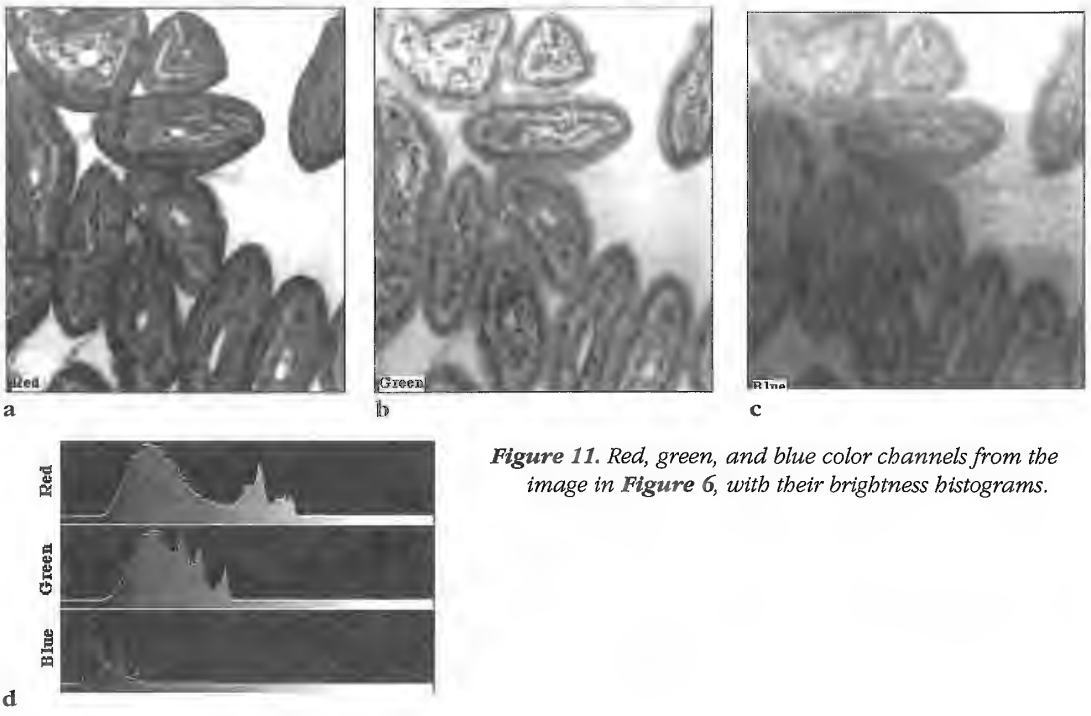


Figure 11. *Red, green, and blue color channels from the image in* ***Figure 6****, with their brightness histograms.*

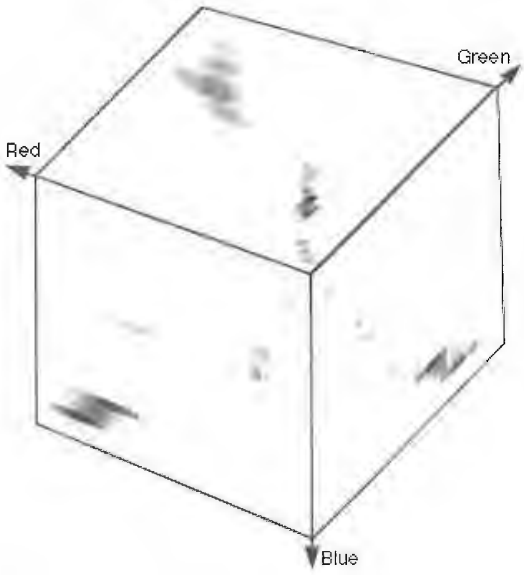
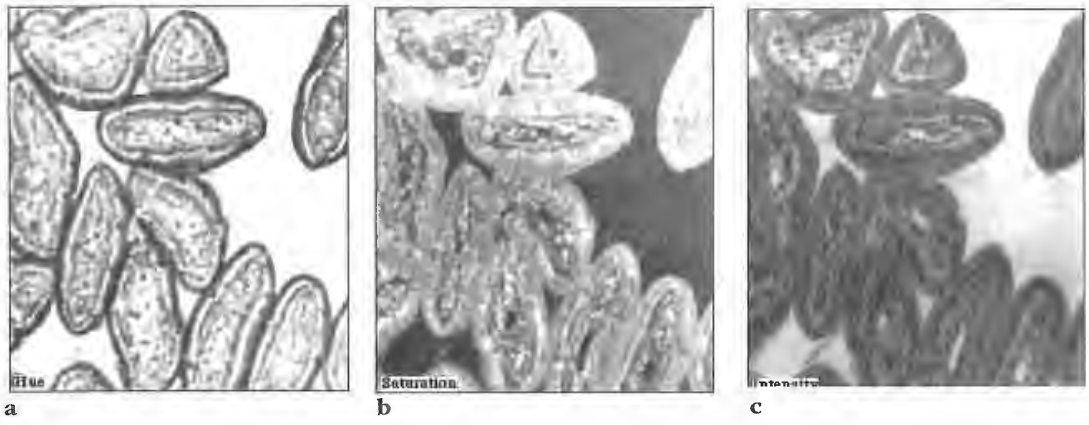


Figure 12. *Pairs of values for the pixels in the images of* ***Figure 11****, plotted on RG, BG, and RB planes and projected onto the faces of a cube.*

for most pixels, the trend toward more of any one color is part of a general increase in brightness by increasing the values of all colors. This means that RGB space poorly disperses the various color values and does not facilitate setting thresholds to discriminate the different regions present.

Figure 13 shows the conversion of the color information from **Figure 11** into hue, saturation, and intensity images, and the individual brightness histograms for these planes. **Figure 14** shows



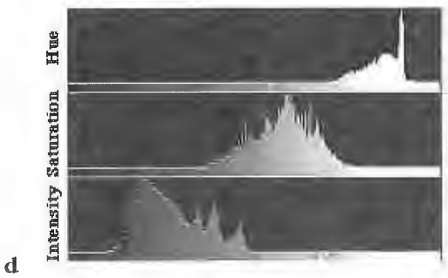


Figure 13. *Hue, saturation, and intensity channels from the image in* ***Figure 6****, with their brightness histograms.*

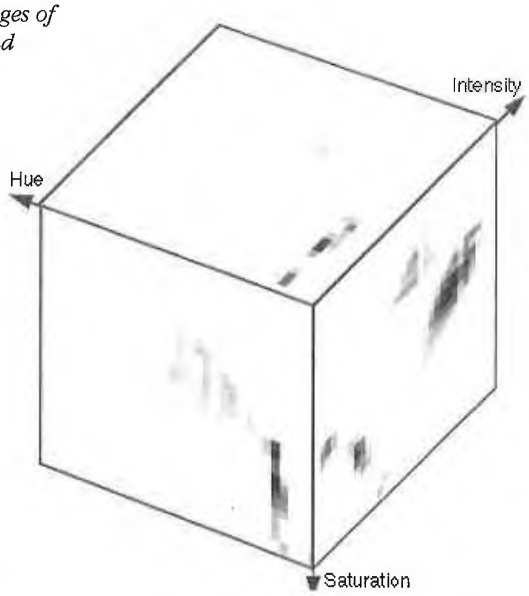


Figure 14. *Pairs of values for the pixels in the images of* ***Figure 13****, plotted on HS, SI, and HI planes and projected onto the faces of a cube.*

the values projected onto individual two-dimensional hue/saturation, saturation/intensity, and intensity/hue square plots. Notice how the much greater dispersion of peaks in the various histograms uses more of the color space and separates several different clusters of values.

In general, for stains used in biological samples, the hue image identifies where a particular stain is located while the saturation image corresponds to the amount of the stain, and the intensity image indicates the overall density of the stained specimen. Combining all three planes as shown in **Figure 15** can often select particular regions that are not well delineated otherwise.

Multiband images are not always simply different colors. A very common example is the use of multiple elemental x-ray maps from the scanning electron microscope (SEM), which can be combined to select phases of interest based on composition. In many cases, this combination can be accomplished simply by separately thresholding each individual image and then applying Boolean logic to combine the images. Of course, the rather noisy original x-ray maps may first require image processing (such as smoothing) to reduce the statistical variations from pixel to pixel (as discussed in Chapters 3 and 4), and binary image processing (as illustrated in Chapter 7).

Using x-rays or other element-specific signals, such as secondary ions or Auger electrons, essentially the entire periodic table can be detected. It becomes possible to specify very complicated combinations of elements that must be present or absent, or the approximate intensity levels needed (because intensities are generally roughly proportional to elemental concentration) to specify the region of interest. Thresholding these combinations of elemental images produces results that are sometimes described as chemical maps. Of course, the fact that several elements may be present in the same area of a specimen, such as a metal, mineral, or block of biological tissue, does not directly imply that they are chemically combined.

In principle, it is possible to store an entire analytical spectrum for each pixel in an image and then use appropriate computation to derive actual compositional information at each point, which is eventually used in a thresholding operation to select regions of interest. At present, this approach is limited in application by the large amount of storage and lengthy calculations required. As faster and larger computers and storage devices become common, however, such methods will become more widely used.

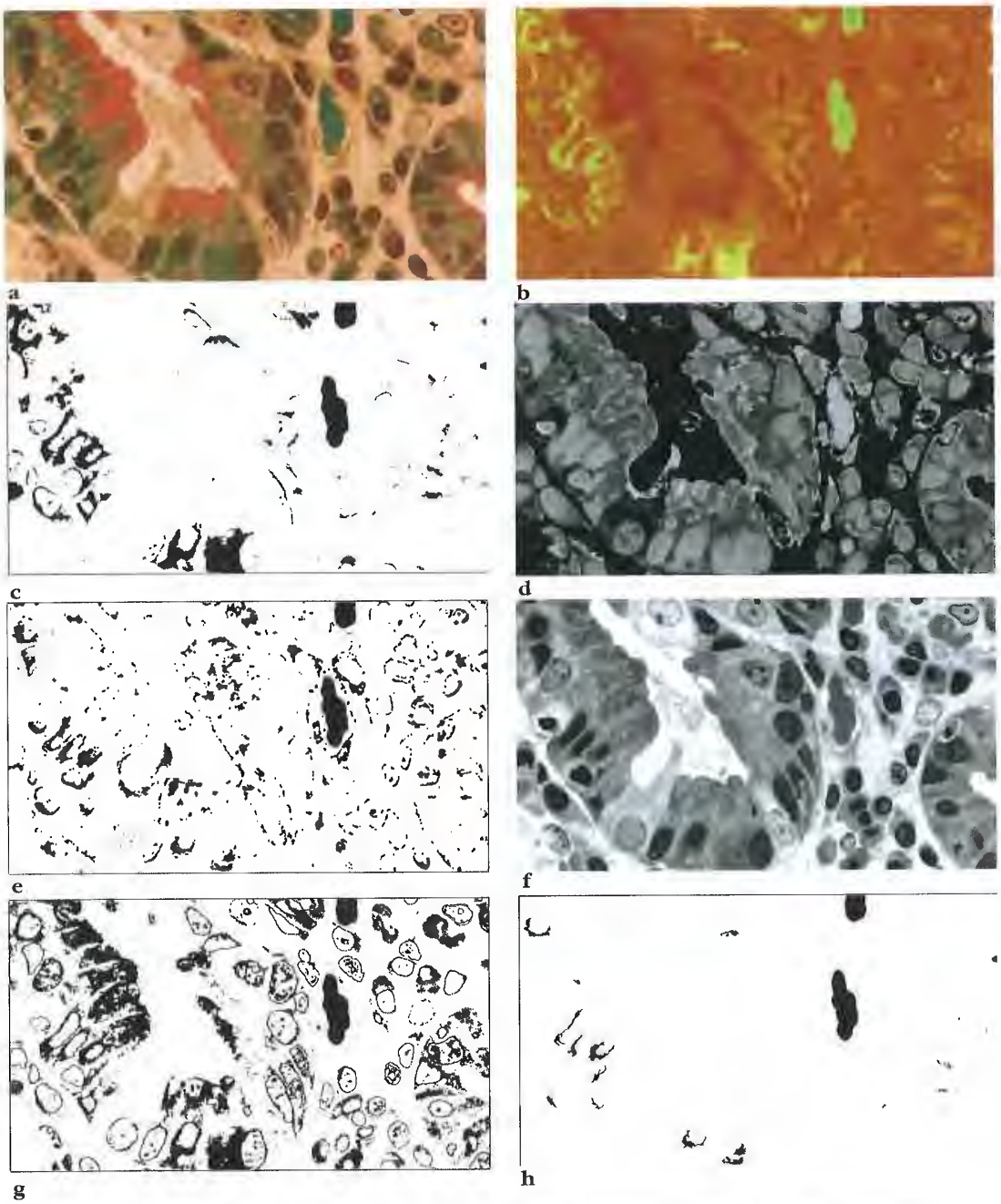


Figure 15. Light micrograph of stomach epithelium with a polychromatic stain, showing thresholding and color separations in HSI space:

(a) *original;*
(b) *hue image;*
(c) *thresholded hue image;*
(d) *saturation image;*
(e) *thresholded saturation image;*
(f) *intensity image;*
(g) *thresholded intensity image;*
(h) *Boolean AND applied to combine three binary images produced by thresholding H, S, and I planes.*

Visualization programs used to analyze complex data may also employ Boolean logic to combine multiple parameters. A simple example would be a geographical information system, in which such diverse data as population density, mean income level, and other census data were recorded for each city block (which would be treated as a single pixel). Combining these different values to select regions for test marketing commercial products is a standard technique. Another example is the rendering of calculated tensor properties in metal beams subject to loading, as modeled in a computer program. Supercomputer simulations of complex dynamical systems, such as evolving thunderstorms, produce rich data sets that can benefit from such analysis.

Other uses of image processing derive additional information from a single original grey-scale image to aid in performing selective thresholding of a region of interest. The processing produces additional images that can be treated as multiband images useful for segmentation.

Thresholding from texture

Few real images of practical interest can be satisfactorily thresholded using simply the original brightness values in a monochrome image. The texture information present in images is one of the most powerful additional tools available. Several kinds of texture may be encountered, including different ranges of brightness, different spatial frequencies, and different orientations (Haralick et al., 1973). The next few figures show images that illustrate these variables and the tools available to utilize them.

Figure 16 shows a test image containing five irregular regions that can be visually distinguished by texture. The average brightness of each of the regions is identical, as shown by the brightness histograms. Region (**e**) contains pixels with uniformly random brightness values covering the entire 0–255 range. Regions (**a**) through (**d**) have Gaussian brightness variations, which for regions (**a**) and (**d**) are also randomly assigned to pixel locations. For region (**b**) the values have been spatially averaged with a Gaussian smooth, which also reduces the amount of variation. For region (**c**) the pixels have been averaged together in one direction to create a directional texture.

One tool that is often recommended (and sometimes useful) for textural characterization is the two-dimensional frequency transform. **Figure 17** shows these power spectra for each of the

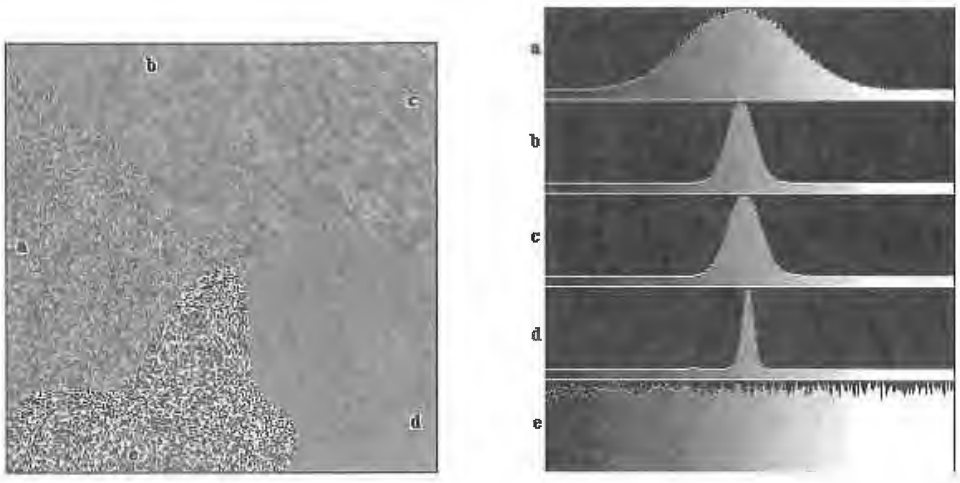


Figure 16. *Test image containing five different regions to be distinguished by differences in the textures. The brightness histograms are shown; the average brightness of each region is the same.*

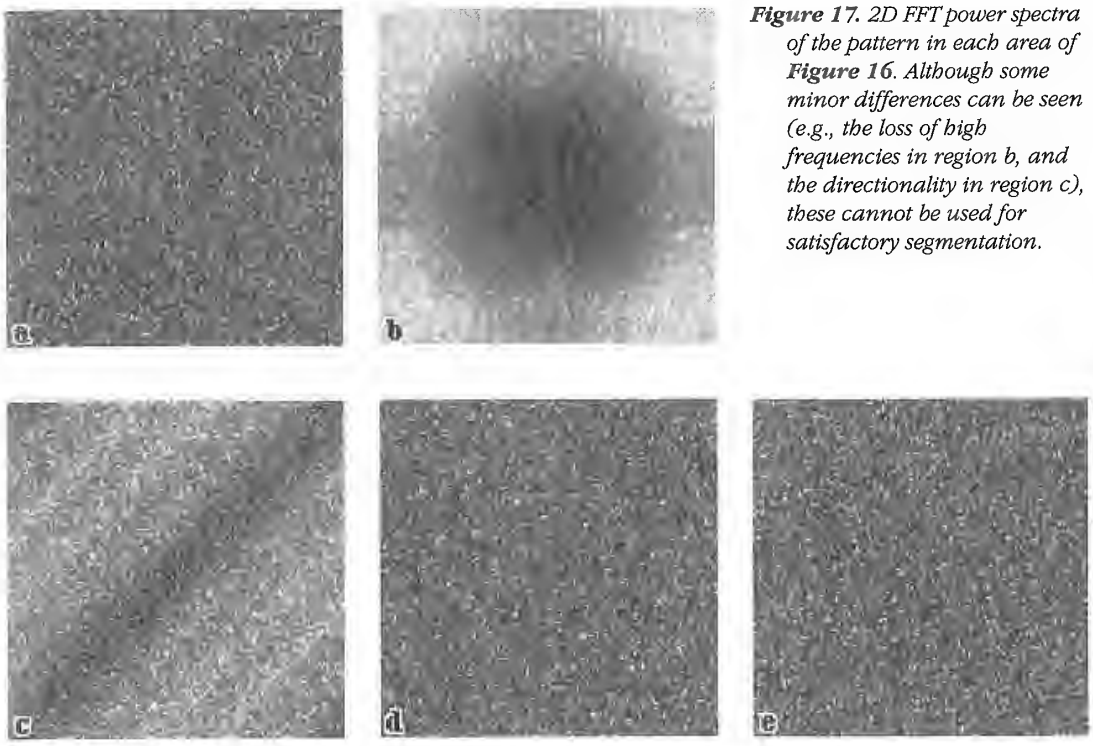


Figure 17. *2D FFT power spectra of the pattern in each area of* ***Figure 16****. Although some minor differences can be seen (e.g., the loss of high frequencies in region b, and the directionality in region c), these cannot be used for satisfactory segmentation.*

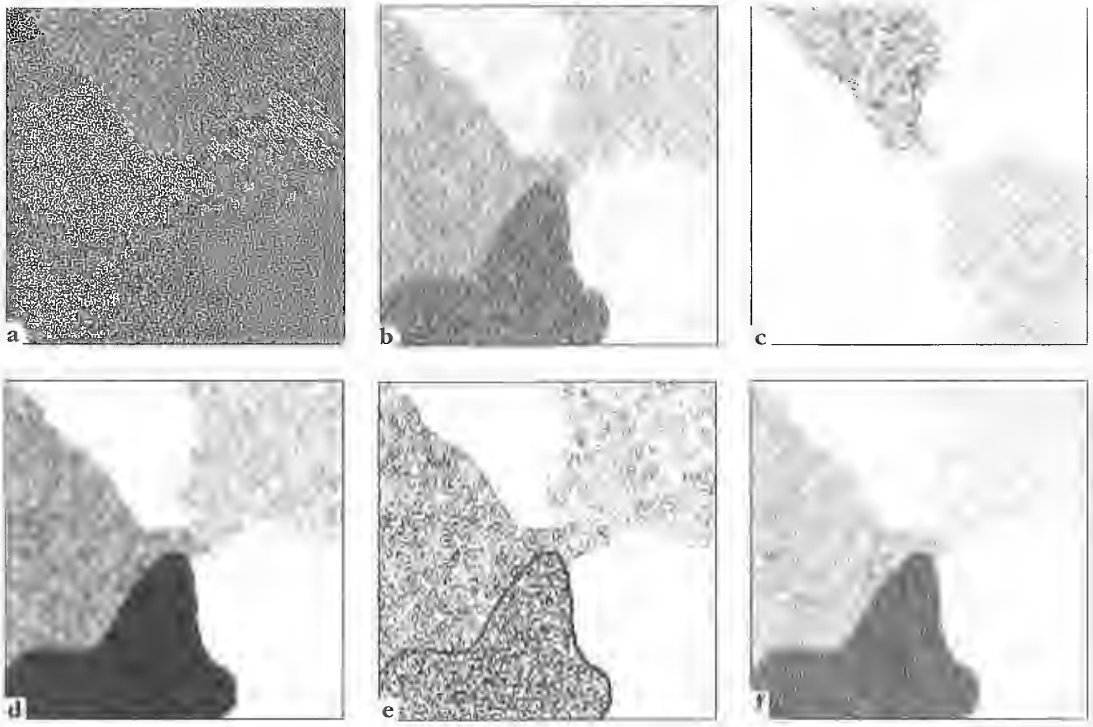
patterns in **Figure 16**. The smoothing in region (**b**) acts as a low-pass filter, so the high frequencies are attenuated. In region (**c**), the directionality is visible in the frequency transform image. For the other regions, the random pixel assignments do not create any distinctive patterns in the frequency transforms. They cannot be used to select the different regions in this case.

Several spatial-domain, texture-sensitive operators are applied to the image in **Figure 18**. The Laplacian shown in (**a**) is a 3 × 3 neighborhood operator; it responds to very local texture values and does not enhance the distinctions between the textures present here. All the other operators act on a 5 × 5 pixel octagonal neighborhood and transform the textures to grey-scale values with somewhat different levels of success. Range (**d**) and variance (**f**), both discussed in Chapter 4, give the best distinction between the different regions.

Some variation still occurs in the grey values assigned to the different regions by the texture operators, because they work in relatively small neighborhoods where only a small number of pixel values control the result. Smoothing the variance image (**Figure 19a**) produces an improved image that has unique grey-scale values for each region. **Figure 20** shows the brightness histogram of the original variance image and the result after smoothing. The spatial smoothing narrows the peak for each region by reducing the variation within it. The five peaks are separated and allow direct thresholding. **Figure 21a** shows a composite image with each region selected by thresholding the smoothed variance image.

Figure 19b shows the application of a Sobel edge (gradient) operator to the smoothed gradient image. Thresholding and skeletonizing (as discussed in Chapter 7) produces a set of boundary lines, which are shown superimposed on the original image in **Figure 21b**. Notice that because the spatial scale of the texture is several pixels wide, the location of the boundaries of regions is necessarily uncertain by several pixels. It is also difficult to estimate the proper location visually, for the same reason.

Figure 18. Application of various texture-sensitive operators to the image in Figure 16:
(a) *Laplacian;*
(b) *Frei and Chen;*
(c) *Haralick;*
(d) *range;*
(e) *Hurst;*
(f) *variance.*



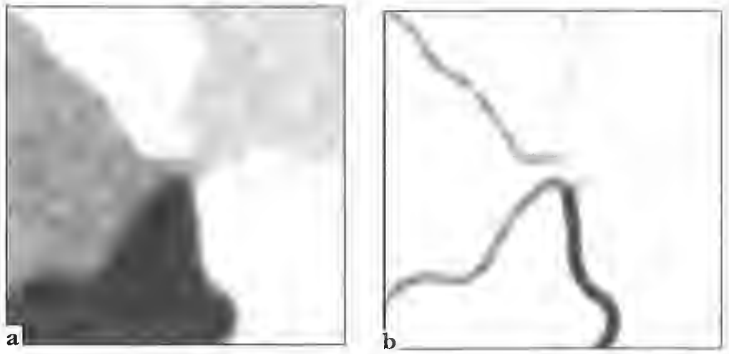
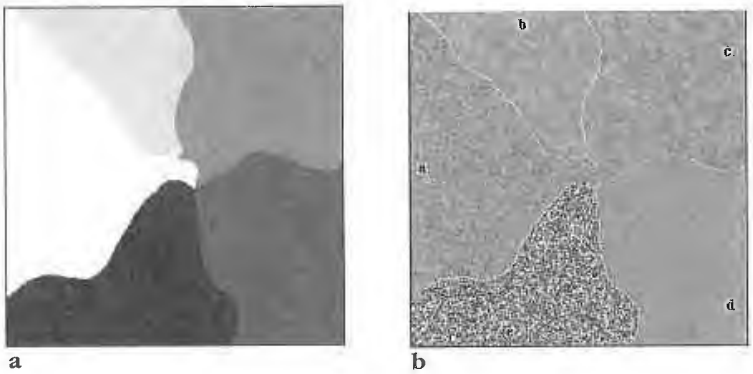


Figure 19. *Result of smoothing the variance image (**Figure 18f**) with a Gaussian kernel with standard deviation equal to 1.6 pixels **(a)**, and the Sobel edge detector applied to the smoothed image **(b)**.*

Figure 22 shows an image typical of many obtained in microscopy. The preparation technique has used a chemical etch to reveal the microstructure of a metal sample. The lamellae indicate islands of eutectic structure, which are to be separated from the uniform light regions to determine the volume fraction of each. The brightness values in regions of the original image are not distinct, but a



Figure 20. *Histograms of the variance image (**Figure 18f**) before **(a)** and after **(b)** smoothing the image with a Gaussian kernel with 1.6-pixel standard deviation. The five regions are now distinct in brightness and can be thresholded successfully.*



a b

Figure 21. Segmentation of the texture image:
(a) *thresholding the smoothed variance image (**Figure 19a**) for each of the peaks in the histogram delineates the different texture regions;*
(b) *skeletonizing the edge from the Sobel operator in **Figure 19b**.*

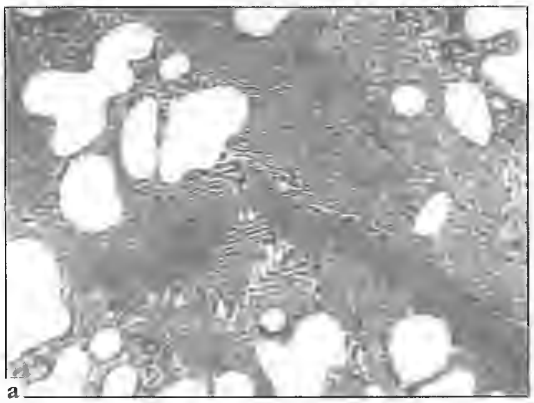
texture operator is able to convert the image to one that can be thresholded. Chapter 4 showed additional examples of converting texture to brightness differences.

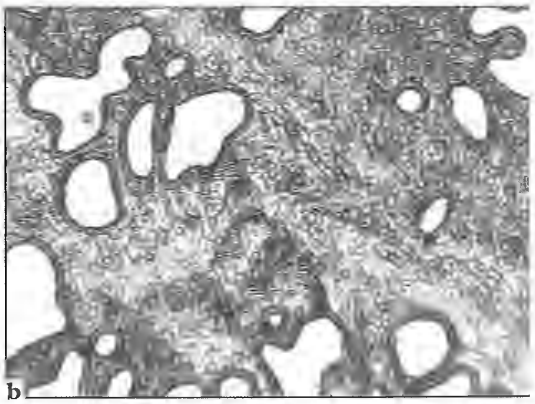
Multiple thresholding criteria

Figure 23a shows a somewhat more complex test image, in which some of the regions are distinguished by a different spatial texture and some by a different mean brightness. No single parameter can be used to discriminate all four regions. The texture values are produced by assigning Gaussian random values to the pixels. As before, a variance operator applied to a 5 × 5 octagonal neighborhood produces a useful grey-scale distinction. **Figures 23b–d** show the result of smoothing the brightness values and the variance image.

It is necessary to use both images to select individual regions. This can be done by thresholding each region separately and then using Boolean logic (discussed in Chapter 7) to combine the two binary images in various ways. Another approach is to use the same kind of two-dimensional histogram as described earlier for color images (Panda and Rosenfeld, 1978). **Figure 24** shows the individual image-brightness histograms and the two-dimensional histogram. In each of the individual histograms, only three peaks are present because the regions are not all distinct in either brightness or variance. In the two-dimensional histogram, individual peaks are visible for each of the four regions.

Figure 25a shows the result of thresholding the intermediate peak in the histogram of the brightness image, which selects two of the regions of medium brightness. **Figure 25b** shows the result of selecting a peak in the two-dimensional histogram to select only a single region. The outlines around each of the regions selected in this way are shown superimposed on the original image in **Figure 25c.**





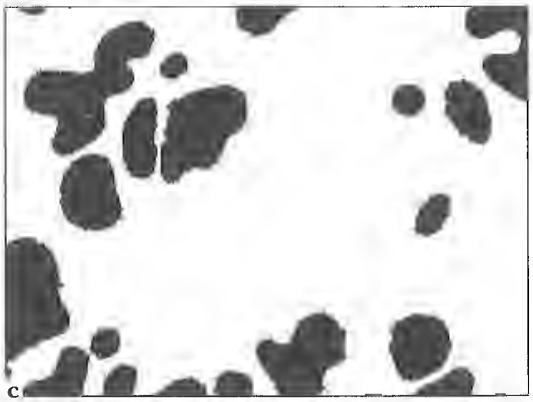
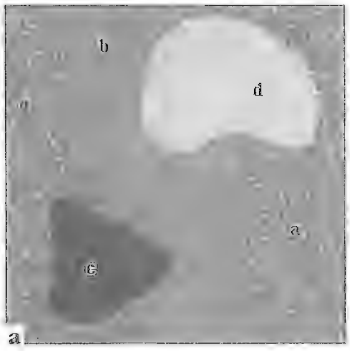


Figure 22. Application of the Hurst texture operator to a microscope image of a metal containing a eutectic:
(a) *original image, with light single-phase regions and lamellae corresponding to the eutectic;*
(b) *application of a Hurst operator (discussed in Chapter 4) to show the local texture in image* ***a****;*
(c) *binary image formed by thresholding image b to select the low-texture (single phase) regions.*







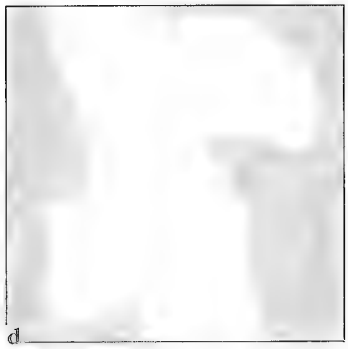


Figure 23. Another segmentation test image, in which some regions have different textures and some different mean brightness:
(a) *original;*
(b) *variance;*
(c) *smoothing* ***a*** *with Gaussian filter, standard deviation = 1.6 pixels;*
(d) *same smoothing applied to* ***b****.*

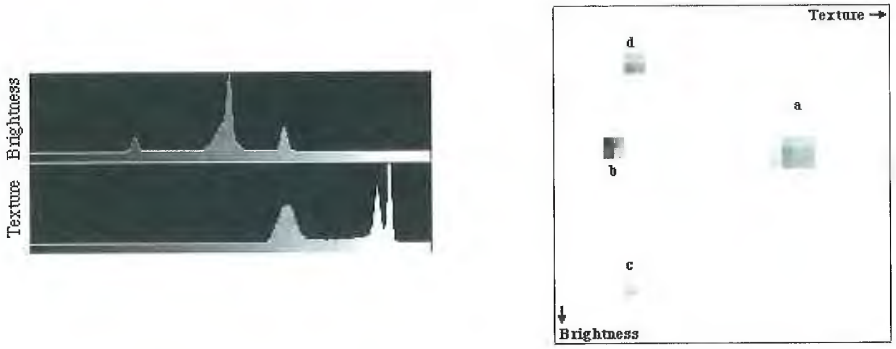


Figure 24. Histograms of the images in ***Figure 23c*** *and* ***23d****, and the two way histogram of the pixels showing the separation of the four regions.*

The different derived images used to successfully segment an image such as this one are sometimes displayed using different color planes. This is purely a visual effect, of course, because the data represented have nothing to do with color. However, it does take advantage of the fact that human vision distinguishes colors well (for most people, at least) and uses color information for segmentation, and it also reveals the similarity between this example of thresholding based on multiple textural and brightness criteria with the more commonplace example of thresholding color images based on the individual color channels. **Figure 25d** shows the information from the images in **Figures 25a** and **25b**, with the original image in the Luminance plane, the smoothed brightness values in the a plane, and the texture information from the variance operator in the b plane of an L·a·b color image.

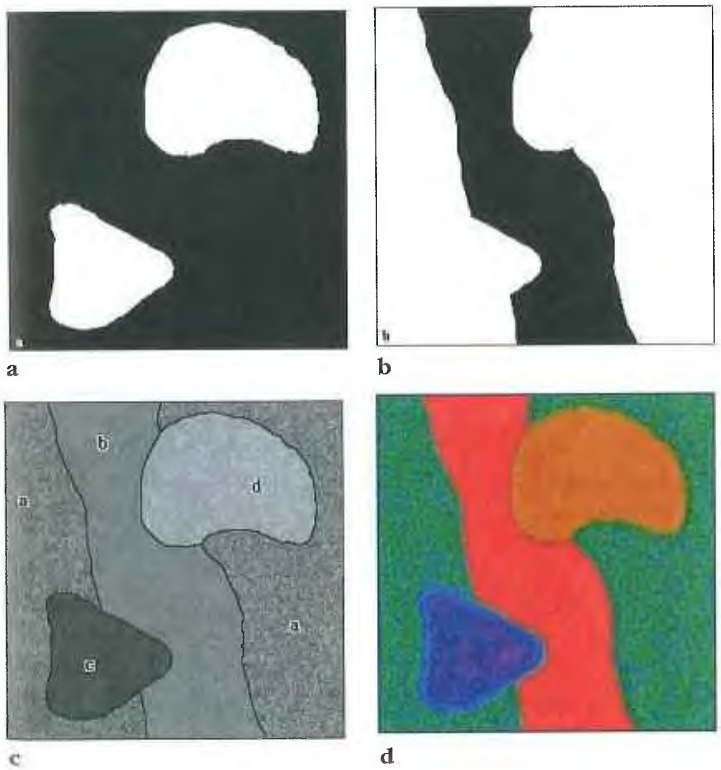


Figure 25. Thresholding of images in Figure 23:
(a) *selecting intermediate brightness values (regions* ***a*** *and* ***b****);*
(b) *selecting only region* ***b*** *by its brightness and texture;*
(c) *region definition achieved by selection ANDing binary images thresholded individually;*
(d) *color coding of the images from* ***Figure 23*** *as described in the text.*

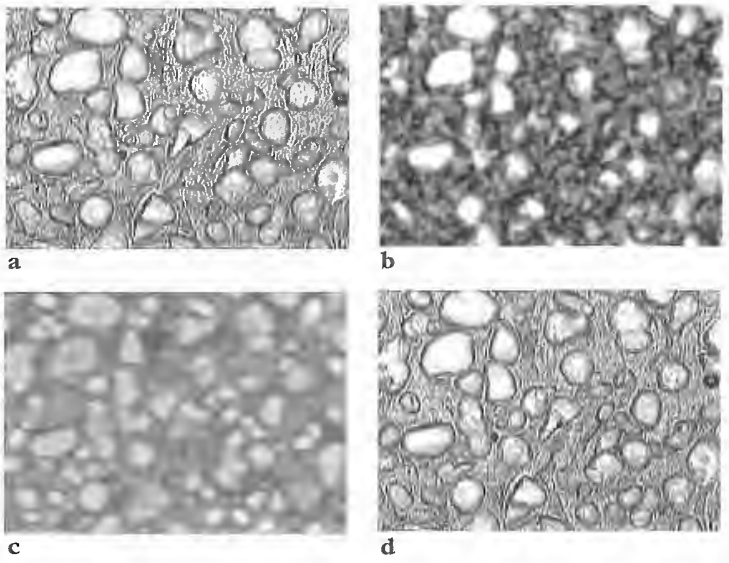


Figure 26. Segmentation of ice crystals in a food product: *(a) original image; (b) Hurst texture operator applied to the intensity channel; (c) hue channel; (d) final result shown as outlines on the original image.*

Figure 26 shows an application of thresholding using two criteria, one of them texture, to a real image. The sample is ice crystals in a food product. The Hurst texture operator partially delineates the ice crystals, as does the use of the hue channel. Thresholding each, and then combining them with Boolean logic and applying a closing (discussed in Chapter 7) produces a useful segmentation of the crystals, as shown by the outlines on the figure.

Textural orientation

Figure 27 shows another test image containing regions having different textural orientations but identical mean brightness, brightness distribution, and spatial scale of the local variation. This rather subtle texture is evident in a two-dimensional frequency transform, as shown in **Figure 28a**. The three ranges of spatial-domain orientation are revealed in the three spokes in the transform.

Using a selective wedge-shaped mask with smoothed edges to select each of the spokes and retransform the image produces the three spatial-domain images shown in **Figure 28**. Each texture orientation in the original image is isolated, having a uniform grey background in other locations. These images cannot be directly thresholded because the brightness values in the textured regions cover a range that includes the surroundings. Applying a range operator to a 5 × 5-pixel octagonal neighborhood, as shown in **Figure 29**, suppresses the uniform background regions and highlights the individual texture regions.

Thresholding these images and applying a closing operation (discussed in Chapter 7) to fill in internal gaps and smooth boundaries, produces images of each region. **Figure 29d** shows the composite result. Notice that the edges of the image are poorly delineated, a consequence of the inability of the frequency transform to preserve edge details, as discussed in Chapter 5. Also, the boundaries of the regions are rather irregular and only approximately rendered in this result.

In many cases, spatial-domain processing is preferred for texture orientation. **Figure 30** shows the result from applying a Sobel operator to the image, as discussed in Chapter 4. Two directional first derivatives in the x and y directions are obtained using a 3 × 3 neighborhood operator. These are then combined using the arc tangent function to obtain an angle that is the direction of maximum brightness gradient. The resulting angle is scaled to fit the 0 to 255 brightness range of the image, so that each step in brightness corresponds to about 1.4 degrees.

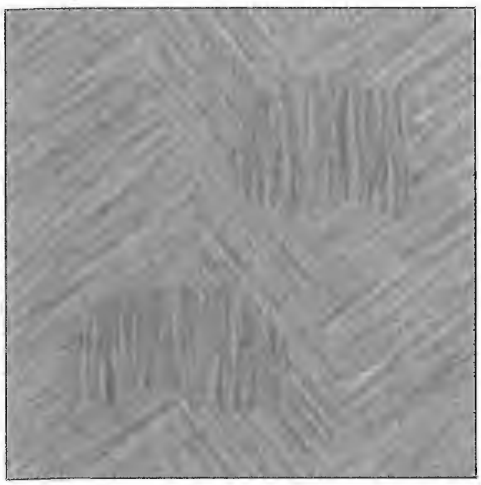


Figure 27. *An image containing regions that have different textural orientations, but the same average brightness, standard deviation, and spatial scale.*

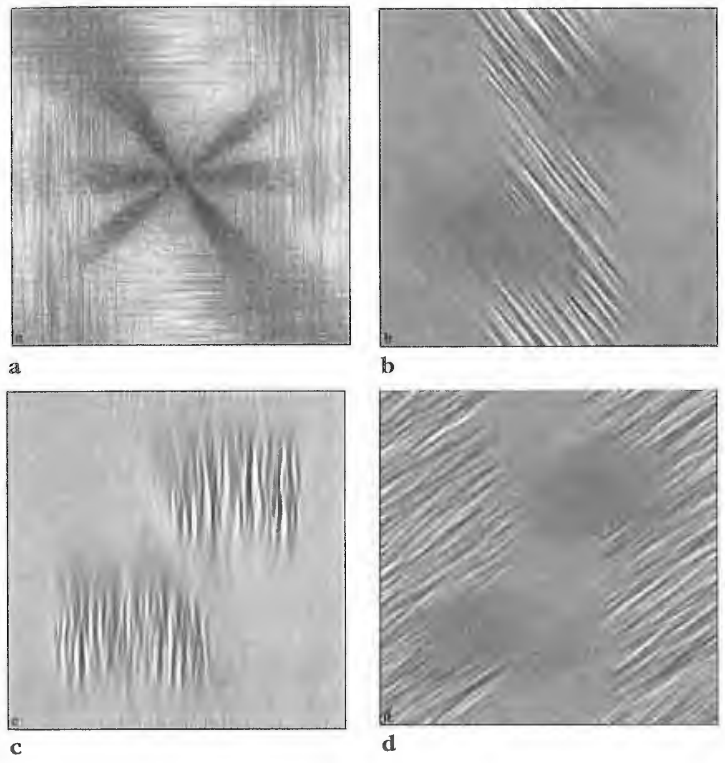


Figure 28. Isolating the directional texture in frequency space: ***(a)*** *two-dimensional frequency transform of the image in* ***Figure 27****, showing the radial spokes corresponding to each textural alignment;* ***(b–d)*** *retransformation using masks to select each of the orientations.*

The brightness histogram shown in **Figure 30b** shows six peaks. These occur in pairs 180 degrees apart, since in each texture region the direction of maximum gradient may lie in either of two opposite directions. This image can be reduced to three directions in several ways. One is to use a grey-scale LUT, as discussed in Chapter 4, which assigns the same grey-scale values to the highest and lowest halves of the original brightness (or angle) range; this converts the 0–360 degree range to 0–180 degrees and permits thresholding a single peak for each direction. A second method is to set two different threshold ranges on the paired peaks and then combine the two resulting binary images using a Boolean "OR" operation (see Chapter 7). A third approach is to set a multiple-threshold range on the two complementary peaks. All these methods are functionally equivalent.

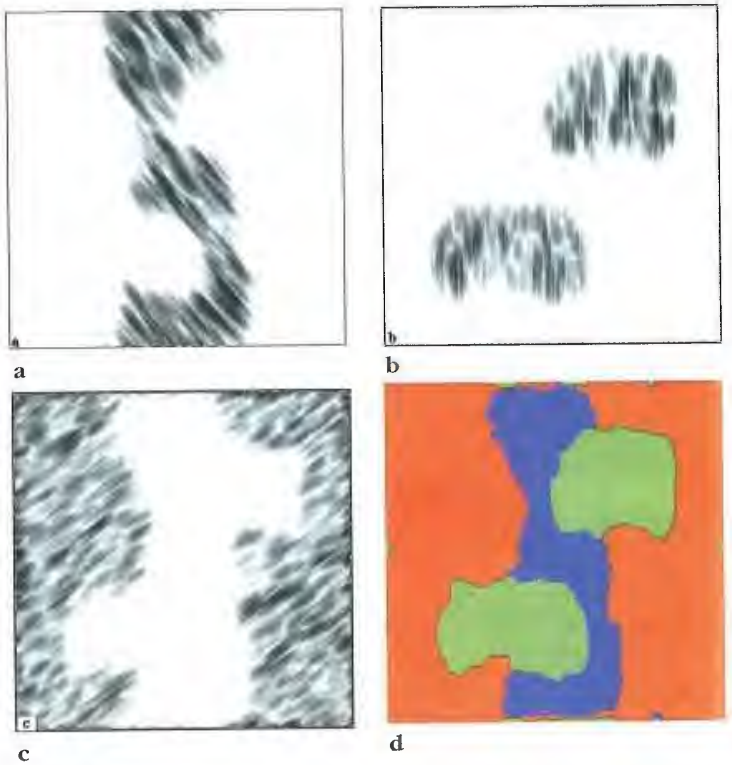


Figure 29. *Application of a range operator to the images in* ***Figure 28b, 28c,*** *and* ***28d,*** *and the combination of the regions selected by thresholding these images.*

Figure 31 shows the results of three thresholding operations to select each of the three textural orientations. Some noise occurs in these images, consisting of white pixels within the dark regions and vice versa, but these are much fewer and smaller than in the case of thresholding the results from the frequency-transform method shown previously. After applying a closing operation (a dilation followed by an erosion, as discussed in Chapter 7), the regions are well delineated, as shown by the superposition of the outlines on the original image (**Figure 31d**). This result is superior to the frequency transform and has smoother boundaries, better agreement with the visual judgment of location, and no problems at the image or region edges.

Figure 32 shows a scanned stylus image of a flycut metal surface. Two predominant directions of machining marks are present, and applying the Sobel direction operator produces an image

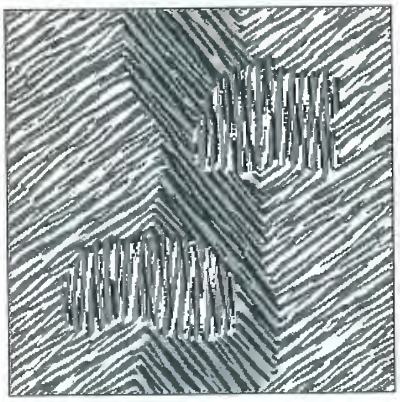
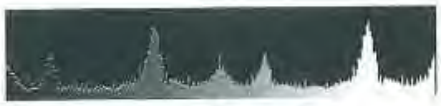


Figure 30. *Application of the Sobel direction operator to the image in* **Figure 27,** *calculating the orientation of the gradient at each pixel by assigning a grey level to the arc tangent of* $(\partial B/\partial y) / (\partial B/\partial x)$. *The brightness histogram shows six peaks, in pairs for each principal textural orientation, because the directions are complementary.*



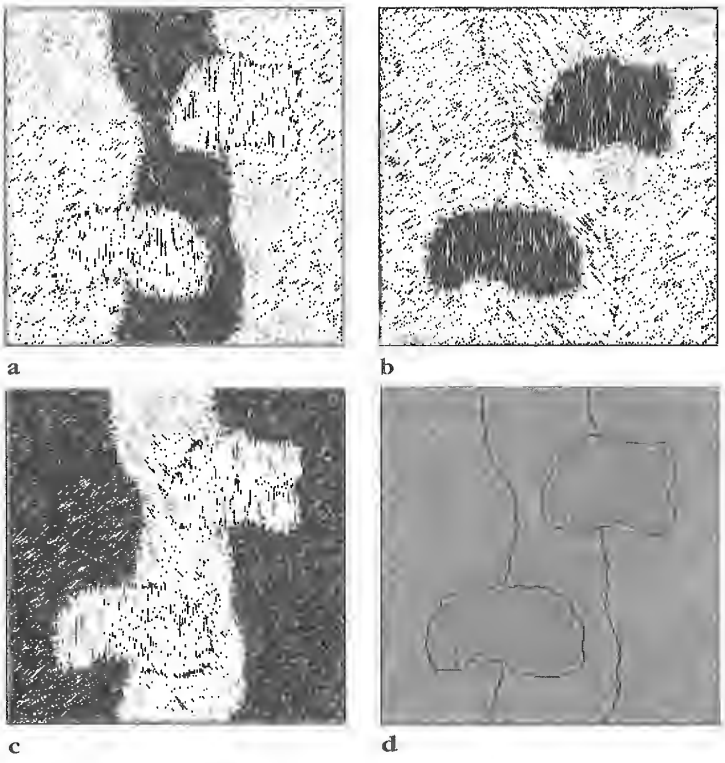


Figure 31. *Thresholded binary images from* ***Figure 30****, selecting the grey values corresponding to each pair of complementary directions, and the outlines showing the regions defined by applying a closing operation to each of the binary images.*

(**Figure 33**) with the expected four peaks, because the grey scale values represent directions covering 360 degrees. This can be reduced to 180 degrees by using a lookup table that replaces the original grey scale with the values shown in **Figure 33b**, which range from black to white over the first 128 values and then again over the second 128 values. Thresholding this produces a binary image (**Figure 34a**) that can be processed with a median filter to remove the speckle noise (**Figure 34b**), resulting in good delineation of the regions in the original image (**Figure 34c**).

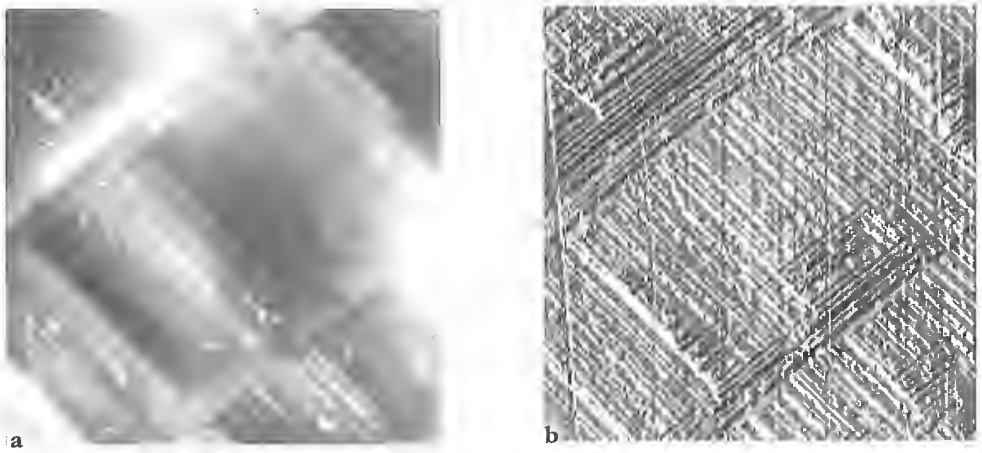
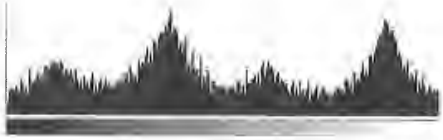
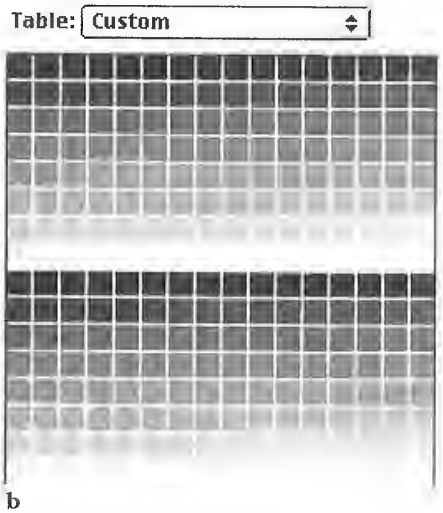


Figure 32. Surface of flycut metal (a), *and the result of applying the Sobel direction operator* ***(b)*** *to show the directionality of the machining marks.*

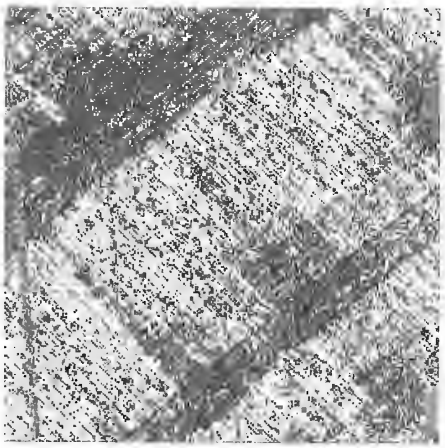
Figure 33. *Histogram of the image in* ***Figure 32b,*** *showing two pairs of peaks. The application of a look-up table* ***(b)*** *converts the image as shown in* ***Figure 34****.*



a



b



a

Figure 34. Direction information for the surface machining marks: ***(a)*** *as produced from* ***Figure 31b*** *using the lookup table in* ***Figure 33b****;* ***(b)*** *application of a median filter to remove speckle noise;* ***(c)*** *segmentation of the original image produced by thresholding* ***b****.*



b



c

Note that applying any neighborhood processing operation such as smoothing or median filtering to an image in which grey scale represents direction requires special rules to account for the modulo change in values at 0. For example, the average value in a neighborhood containing pixel values of 15 and 251 is 5, not 133. A simple but effective way to accomplish this is to process each neighborhood twice, once with the stored values and once with the values shifted to become (P + 128) mod 255, and then keep whichever result is smaller.

Figure 35 shows another example, a metallographic sample with a lamellar structure. This requires several steps to segment into the various regions. Brightness thresholding can delineate one region directly. Applying the Sobel orientation operator produces an image that can be thresholded to delineate the other two regions, but as before each region has pairs of grey-scale values that are 180 degrees or 128 grey values apart. Thresholding the various regions produces a complete map of the sample as shown.

Accuracy and reproducibility

In one or more dimensions, the selection of threshold values discussed so far has been manual. An operator interactively sets the cutoff values so that the resulting image is visually satisfying and the correspondence between what the user sees in the image and the pixels the thresholds select is as close as possible. This is not always consistent from one operator to another, or even for the same person over a period of time. The difficulty and variability of thresholding represents a serious source of error for further image analysis.

Two slightly different requirements have been established for setting threshold values. Both have to do with the typical use of binary images for feature measurement. One is to achieve reproducibility, so that variations due to the operator, lighting, etc. do not affect the results. The second

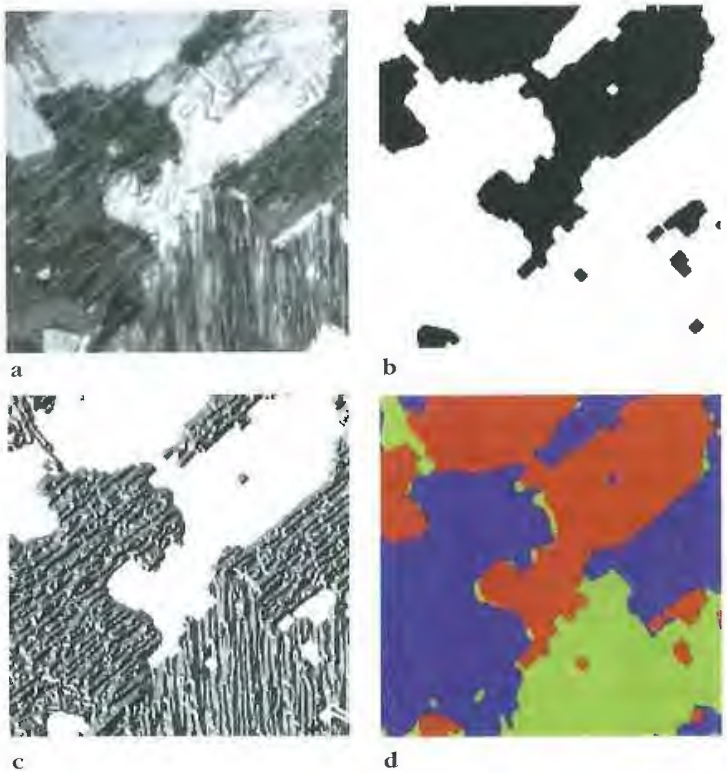


Figure 35. Thresholding on multiple criteria: ***(a)*** *original metallographic image;* ***(b)*** *one region selected by brightness thresholding;* ***(c)*** *Sobel orientation operator applied to the remaining region;* ***(d)*** *final segmentation result with regions color coded.*

goal of setting threshold values is to achieve accurate boundary delineation so that measurements are accurate.

Because pixel-based images represent, at best, an approximation to the continuous real scene being represented, and because thresholding classifies each pixel as either part of the foreground or the background, only a certain level of accuracy can be achieved. An alternate representation of features based on boundary lines can be more accurate. These may be polygons with many sides and corner points defined as x,y coordinates of arbitrary accuracy, or spline curves, etc., as compared with the comparatively coarse pixel spacing.

Such boundary-line representation is superior for accurate measurement because the line itself has no width, although determining the line is far from easy. The location of individual points can be determined by interpolation between pixels, perhaps fitting mathematical functions to pixels on either side of the boundary to improve the results. This type of approach is commonly used geographic applications in which elevation values measured at discrete points are used to construct topographic maps. It is also used in metrology applications, such as measuring dimensions of microelectronic circuit elements on silicon wafers, and is possible because the shape of the features (usually straight lines) is known *a priori*. This type of application goes beyond the typical image processing operations dealt with in this chapter.

One approach to interpolating a smoothed boundary line through the pixels is used by the super-resolution perimeter measurement routine used in Chapter 9 for feature measurement. This uses neighborhood processing (the Laplacian of a Gaussian) to fit an adaptive boundary line through each pixel, achieving improved precision and fractional-pixel accuracy.

Thresholding produces a pixel-based representation of the image that assigns each pixel to either the feature(s) or the surroundings. The finite size of the pixels allows the representation only a finite accuracy, but we would prefer to have no bias in the result. This means that performing the same operation on many repeated images of the same scene should produce an average result that approaches the true value for size or other feature measurements. This is not necessary for quality control applications in which reproducibility is of greater concern than accuracy, and some bias (as long as it is consistent) can be tolerated. Many things can contribute to bias in setting thresholds. Human operators are not very good at setting threshold levels without bias. In most cases, they are more tolerant of settings that include additional pixels from the background region along with the features than they are of settings that exclude some pixels from the features.

As indicated at the beginning of this chapter, the brightness histogram from the image can be an important tool for setting threshold levels. In many cases, it will show distinct and separated peaks from the various phases or structures present in the field of view, or it can be made to do so by prior image processing steps. In this case, it seems that setting the threshold level somewhere between the peaks should produce consistent, and perhaps even accurate, results.

Unfortunately, this idea is easier to state than to accomplish. In many real images, the peaks corresponding to particular structures are not perfectly symmetrical or ideally sharp, particularly when there may be shading either of the entire image or within the features (e.g., a brightness gradient from center to edge). Changing the field of view or even the illumination may cause the peak to shift and/or to change shape. Nonlinear camera response or automatic gain circuits can further distort the brightness histogram. If the area fraction of the image that is the bright (or dark) phase changes from one field of view to another, some method is needed to maintain a threshold setting that adapts to these changes and preserves precision and accuracy.

If the peaks are consistent and well-defined, then choosing an arbitrary location at some fixed fraction of the distance between them is a rapid method often satisfactory for quality control work. In many cases, it is necessary to consider the pixels whose brightness values lie between the peaks

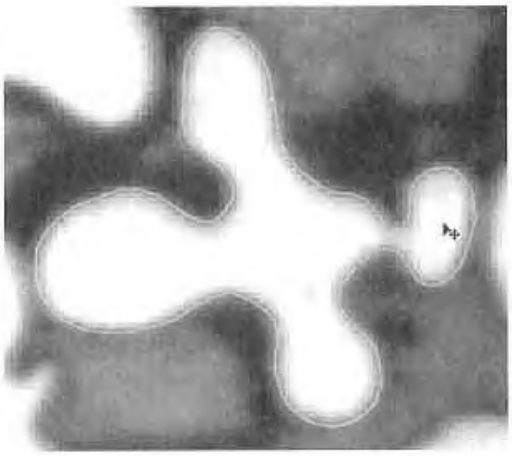


Figure 36. *Example of finite pixels straddling a boundary line, with brightness values that average those of the two sampled regions.*

in the brightness histogram. In most instances, these are pixels that straddle the boundary and have averaged together the two principal brightness levels in proportion to the area subtended within the pixel, as indicated in **Figure 36**.

Asymmetric boundaries (for example, a metallographic specimen in which etching has attacked the softer of two phases so that the boundary is skewed) can introduce bias in these brightness values, and so can prior processing steps, such as those responding to texture in the image. Many of these operations work on a finite and perhaps rather large neighborhood, so the boundary position becomes somewhat uncertain. If the processing operation responds nonlinearly to differences, as the variance operator does, the apparent boundary location will shift toward the most different pixel in the neighborhood.

Including position information

The histogram display shows only the frequency of occurrence of different values and does not preserve any information about position, the brightness of neighboring pixels, and other factors. Yet, it is this spatial information that is important for determining boundary location. It is possible, in principle, to build a co-occurrence matrix for the image in which all possible combinations of pixel brightness are counted in terms of the distance between them. This information is used to select the pixels that are part of the feature instead of simply the pixel brightness values, but this is equivalent to a processing operation that uses the same co-occurrence matrix to construct a texture image for which simple thresholding can be used.

One possible algorithm for threshold settings is to pick the minimum point in the histogram (**Figure 37**). This should correspond to the value that affects the fewest pixels and thus gives the lowest expected error in pixel classification when the image is segmented into features and background. The difficulty is that because this region of the histogram is (hopefully) very low, with few pixels having these values, the counting statistics are poor and the shape of the curve in the histogram is poorly defined. Consequently, the minimum value is hard to locate and may move about considerably with only tiny changes in overall illumination, a change in the field of view to include objects with a different shape, or more or fewer pixels along the boundary. Smoothing the histogram with a polynomial fit may provide a somewhat more robust location for a minimum point.

Figure 38 shows an image having two visibly distinguishable regions. Each contains a Gaussian noise pattern with the same standard deviation but a different mean, though the brightness values in the two regions overlap. This means that setting a threshold value at the minimum between the two peaks causes some pixels in each region to be misclassified, as shown.

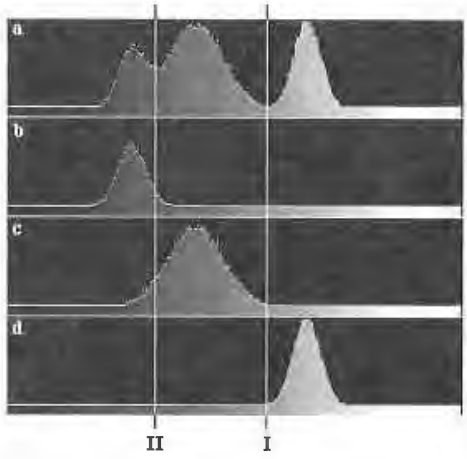


Figure 37. An example histogram from a specimen with three distinct regions. The minimum at level I cleanly separates the two peaks, while the one at level II does not because the brightness values in the two regions have variations that overlap.

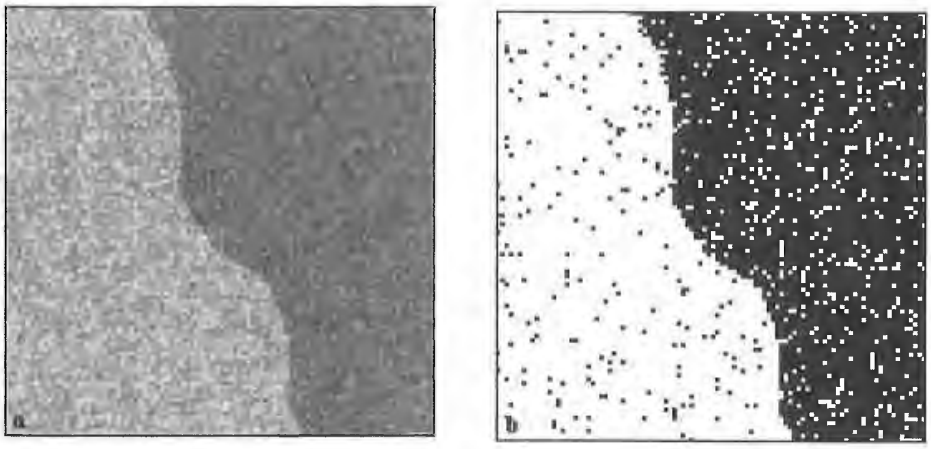


Figure 38. A test image containing two regions whose mean brightness levels are different, but which have variations in individual pixels that overlap:
(a) *original image (enlarged to show pixels);*
(b) *result of setting a simple threshold at the minimum point.*

This type of image often results from situations in which the total number of photons or other signals is low and counting statistics cause a variation in the brightness of pixels in uniform areas. Counting statistics produce a Poisson distribution, but when moderately large numbers are involved, this is very close to the more convenient Gaussian function used in these images. For extremely noisy images, such as x-ray dot maps from the SEM, some additional processing in the spatial domain may be required before attempting thresholding (O'Callaghan, 1974).

Figure 39 shows a typical sparse dot map. Most of the pixels contain 0 counts, and a few contain 1 count. The boundaries in the image are visually evident, but their exact location is at best approximate, requiring the human visual computer to group the dots together. Imaging processing can do this by counting the number of dots in a circular neighborhood around each pixel. Convolution with a kernel consisting of "ones" in a 15-pixel-diameter circle accomplishes this, producing the result shown. This grey-scale image can be thresholded to locate the boundaries shown, but there is inadequate data to decide whether the small regions, voids, and irregularities in the

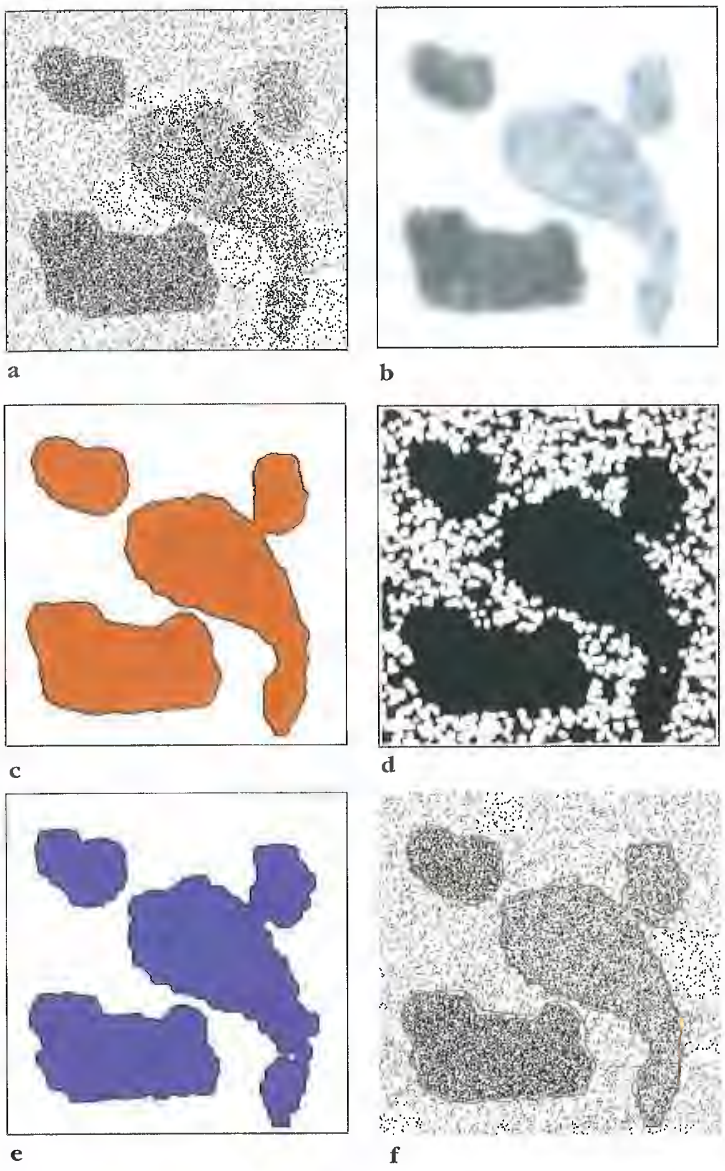


Figure 39. Thresholding a sparse dot image: ***(a)*** *x-ray dot map;* ***(b)*** *grey-scale image formed by counting the dots within a 15-pixel-diameter circle centered on each pixel;* ***(c)*** *boundary determined by thresholding image* ***b*** *at 4 standard deviations above the mean background level;* ***(d)*** *application of a closing operation (dilation followed by erosion) to fill the gaps between closely spaced pixels in image* ***a****;* ***(e)*** *application of an opening operation (erosion followed by dilation) to remove small dark regions in the background in image* ***d****;* ***(f)*** *comparison of the feature outlines determined by the smoothing and thresholding (red) vs. closing and opening (blue) methods.*

boundaries are real or simply due to the limited counting statistics. Typically, the threshold level will be set by determining the mean brightness level in the background region, and then setting the threshold several standard deviations above this to select just the significant regions.

The figure compares this approach to one based on the binary editing operations discussed in Chapter 7. Both require making some assumptions about the image. In the smoothing and thresholding case (**Figures 39b** and **c**), some knowledge about the statistical meaning of the data is required. For x-rays, the standard deviation in the count rate is known to vary in proportion to the square root of the number of counts, which is the brightness in the smoothed image. Erosion and dilation are based on assumptions about the distances between dots in the image. Closing (dilation followed by erosion) fills in the gaps between dots to create solid areas corresponding to the features, as shown in **Figure 39d**. In the background regions, this does not produce a continuous dark region and so an opening (erosion followed by dilation) can remove it (**Figure 39e**). Adding

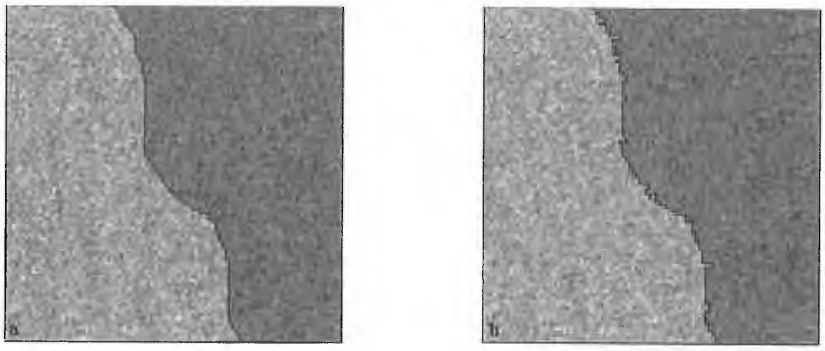


Figure 40. The boundary in the image of Figure 38:
(a) *thresholding at the minimum point in the histogram followed by closing (dilation and erosion);*
(b) *iteratively setting the minimum entropy point.*

and then removing pixels produces the final result shown. The boundaries are slightly different from those produced by smoothing and thresholding, but the original image does not contain enough data to distinguish between them.

Several possible approaches may be used to improve the segmentation of noisy regions, using **Figure 38** as a test case. Chapter 7 discusses binary image editing operations, including morphological processing. The sequence of a dilation followed by an erosion, known as a closing, fills holes, erases isolated pixels, and smooths the boundary line to produce the result shown in **Figure 40a**. By contrast, a much more complicated operation reassigns pixels from one region to the other to achieve minimum entropy in both regions. Entropy methods are generally a very computer-intensive approach to image restoration. They function to improve degraded grey-scale images, as discussed in Chapter 3.

In this case, the collection of pixels into two regions can be described as an entropy problem as follows (Kanpur et al., 1985): The total entropy in each region is calculated as $-\Sigma p_i \log_e p_i$, where p_i is the fraction of pixels having brightness i. Solving for the boundary that classifies each pixel into one of two groups to minimize this function for the two regions, subject to the constraint that the pixels in each region must touch each other, produces the boundary line shown in **Figure 40b**. Additional constraints, such as minimizing the number of touching pixels in different classes, would smooth the boundary. The problem is that such constraints are ad hoc, make the solution of the problem very difficult, and can usually be applied more efficiently in other ways (for instance by smoothing the binary image).

Setting a threshold value at the minimum in the histogram is sometimes described as selecting for minimum area sensitivity in the value (Weszka, 1978; Wall et al., 1974). This means that changing the threshold value causes the least change in the feature (or background) area, although, as noted previously, this says nothing about the spatial arrangement of the pixels that are thereby added to or removed from the features. Indeed, the definition of the histogram makes any minimum in the plot a point of minimum area sensitivity.

For the image shown in **Figure 38**, the histogram can be changed to produce a minimum that is deeper, broader, and has a more stable minimum value by processing the image. **Figure 41** shows the results of smoothing the image (using a Gaussian kernel with a standard deviation of 1 pixel) or applying a median filter (both methods are discussed in Chapters 3 and 4). The peaks are narrower and the valley is broader and deeper. The consequences for the image, and the boundary that is selected by setting the threshold level between the peaks, are shown in **Figure 42**.

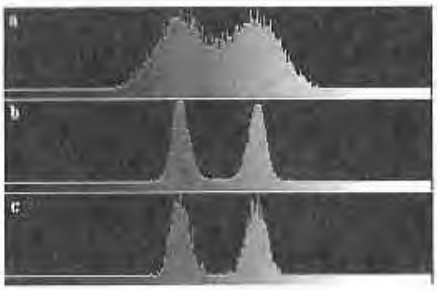


Figure 41. Histogram of the image in Figure 38: *(a) original, with overlapped peaks; (b) after smoothing; (c) after median filtering.*

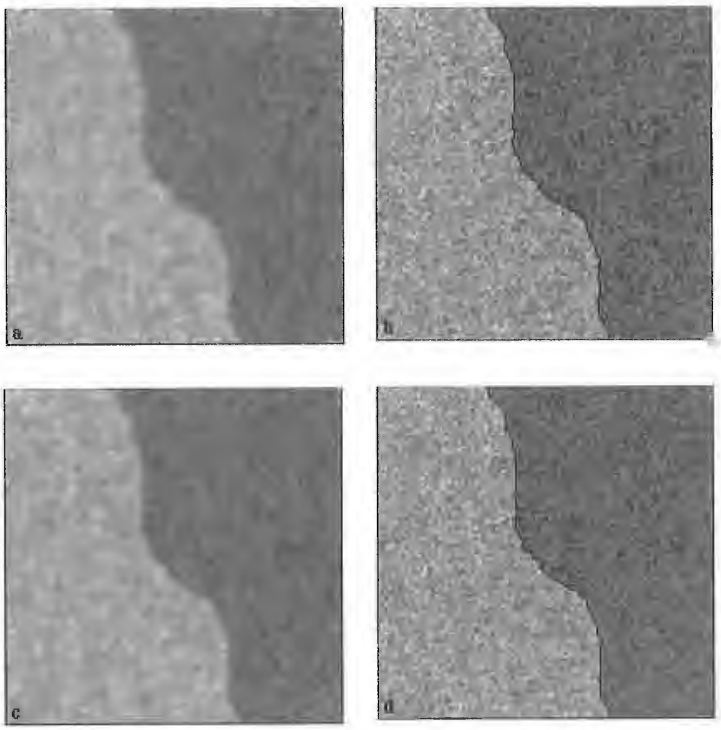


Figure 42. Processing the image in Figure 38 to modify the histogram: *(a) Smoothing with a Gaussian kernel, standard deviation = 1 pixel; (b) the boundary produced by thresholding image a, superimposed on the original; (c) median processing (iteratively applied until no further changes occurred); (d) the boundary produced by thresholding image c, superimposed on the original.*

It appears that this is not the criterion used by human operators, especially when they watch an image and interactively adjust a threshold value. Instead of the total area of features changing minutely with adjustment, which is difficult for humans to judge, another approach is to use the total change in perimeter length around the features (Russ and Russ, 1988a). This may, in fact, be the criterion actually used by skilled operators. The variation in total perimeter length with respect to threshold value provides an objective criterion that can be efficiently calculated. The minimum in this response curve provides a way to set the thresholds that is reproducible, adapts to varying illumination, etc., and mimics to some extent the way humans set the values. For the case in which both upper and lower threshold levels are to be adjusted, this produces a response surface in two dimensions (the upper and lower values), which can be solved to find the minimum point as indicated in **Figure 43**.

Figure 44 shows an image whose brightness threshold has been automatically positioned to minimize the variation in total boundary length. The brightness histogram shown in the figure has a very long valley between the two phase peaks, neither of which has a symmetric or

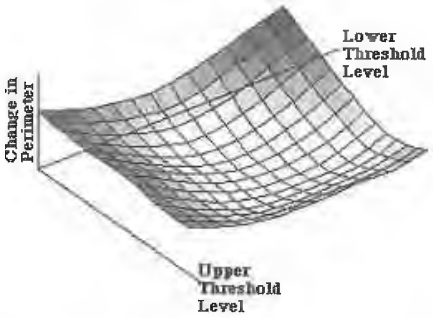


Figure 43. *A two-way plot of the change in perimeter length vs. the settings of upper and lower level brightness thresholds. The minimum indicates the optimal settings.*

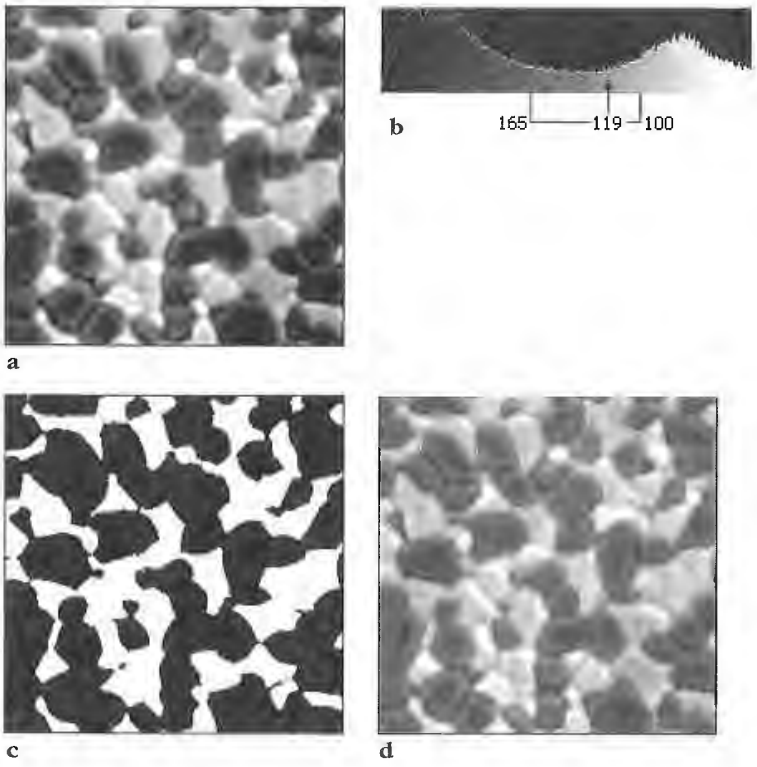
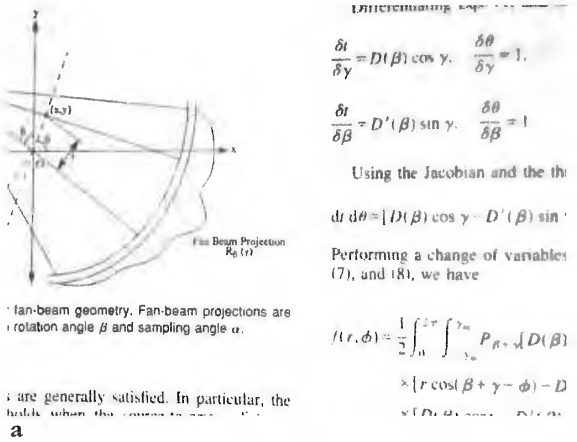


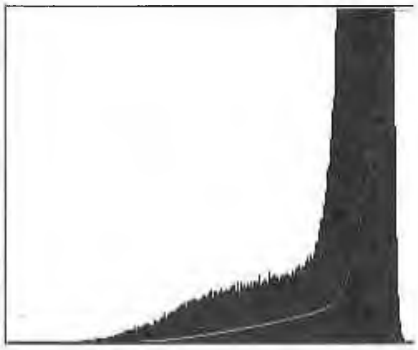
Figure 44. *A test image for automatic threshold adjustment* ***(a)****, and its brightness histogram* ***(b)****. The optimum threshold point within the search range marked, based on minimizing the change in perimeter length is shown on the histogram, with the thresholded binary result* ***(c)*** *and the boundary overlaid on the original* ***(d)****.*

Gaussian shape. The selected threshold point is not at the lowest point in the histogram. Repeated measurements using this algorithm on many images of the same objects show that the reproducibility in the presence of finite image noise and changing illumination is rather good. Length variations for irregular objects varied less than 0.5%, or 1 pixel in 200 across the major diameter of the object.

Many of the algorithms developed for automatic setting of thresholds are intended for the discrimination of printed text on paper, as a first step in programs that scan pages and convert them to text files for editing or communication. **Figure 45** shows an example of a page of scanned text with the results of several of these algorithms, as summarized in Parker (Parker, 1997; Yager, 1979; Otsu, 1979; Trussell, 1979). Note that there is no "valley between two peaks" present in this histogram. Each method makes different assumptions about the nature of the histogram and the appropriate statistical or other tests that can be used to divide it into two parts, each representing one



a



b

Using the Jacobian and the th

$dt\, d\theta = [D(\beta) \cos \gamma \quad D'(\beta) \sin$

Performing a change of variables
(7), and (8), we have

c

Using the Jacobian and the th

$dt\, d\theta = [D(\beta) \cos \gamma - D'(\beta) \sin$

Performing a change of variables
(7), and (8), we have

d

Using the Jacobian and the thi

$dt\, d\theta = [D(\beta) \cos \gamma - D'(\beta) \sin$

Performing a change of variables
(7), and (8), we have

e

Using the Jacobian and the thi

$dt\, d\theta = [D(\beta) \cos \gamma - D'(\beta) \sin$

Performing a change of variables
(7), and (8), we have

f

Figure 45. Automatic thresholding of printed text on paper using algorithms from Parker (1997):
(a) *original grey-scale scan;*
(b) *histogram;*
(c) *Yager algorithm, threshold = 134;*
(d) *Trussell method, threshold = 172;*
(e) *Shannon entropy algorithm; threshold = 184;*
(f) *Kittler algorithm, threshold = 196.*

of the two structures present (paper and ink). Many of the algorithms summarized in Parker produce closely similar results on this image, but some of the results do not separate the characters entirely and others cause them to break up. No single method will work for all types of printing, paper, and image acquisition settings. Even if a single method did exist, the problem being addressed is more specialized than the general range of images containing just two types of structures, while many images contain more than two.

The Trussell algorithm (Trussell, 1979) is probably the most widely used automatic method, because it usually produces a fairly good result (**Figure 45d**) and is easy to implement. It finds the threshold setting that produces two populations of pixels (brighter and darker) with the largest value of the student's *t* statistic, which is calculated from the difference between the means of the two groups and their standard deviations. This is a fairly standard statistical test, but it really should only be applied when the groups are known to have normal or Gaussian distributions, which is rarely the case for the distribution of pixel brightness values in typical images. Another algorithm that often produces good results (slightly different from the Trussell method) minimizes the entropy of the two sets of pixels (above and below the threshold setting), and is illustrated in **Figure 45e**.

Selective histograms

Most of the difficulties with selecting the optimum threshold brightness value between two peaks in a typical histogram arise from the intermediate brightness values of the histogram. These pixels lie along the boundaries of the two regions, so methods that eliminate them from the histogram will contain only peaks from the uniform regions and can be used to select the proper threshold value (Weszka and Rosenfeld, 1979; Milgram and Herman, 1979).

One way to perform this selection is to use another derived image, such as the Sobel gradient or any of the other edge-finding operators discussed in Chapter 4. Pixels having a high gradient value can be eliminated from the histogram of the original image to reduce the background level in the range between the two phase peaks. **Figure 46** shows an example. The original image contains three phases with visually distinct grey-levels. Several methods can be used to eliminate edge pixels. It is most straightforward to threshold a gradient image, selecting pixels with a high values. This produces a binary image that can be used as a mask, as discussed in Chapter 7. This mask restricts which pixels in the original image are to be used for the histogram to be analyzed.

As an example, using the image from **Figure 46**, the 20% of the pixels with the largest magnitude in the Sobel gradient image were selected to produce a mask used to remove those pixels from the original image and the histogram. The result, shown in **Figure 47**, is the reduction of those portions of the histogram between peaks, with the peaks themselves little affected. This makes it easier to characterize the shapes of the peaks from the phases and select a consistent point between them.

Of course, this method requires setting a threshold on the gradient image to select the pixels to be bypassed. The most often used technique is simply to choose some fixed percentage of the pixels with the highest gradient value and eliminate them from the histogram of the original image. In the example shown, however, the gradient operator responds more strongly to the larger difference between the white and grey regions than to the smaller difference between the grey and dark regions. Thus, the edge-straddling pixels (and their background in the histogram) are reduced much more between the white and grey peaks than between the grey and black peaks.

Figure 48 shows another method which alleviates the problem in this image. Beginning with a range image (the difference between the darkest and brightest pixels in a 5-pixel-wide octagonal neighborhood), non-maximum suppression (also known as grey-scale thinning, skeletonization, or ridge-finding) is used to narrow the boundaries and eliminate pixels that are not actually on the



a b c

Figure 46. Thresholding by ignoring boundary pixels:
(a) *original image containing three visually distinct phase regions with different mean grey levels;*
(b) *application of a gradient operator (Sobel) to* ***a****;*
(c) *The image without the 20% of the pixels having the largest gradient value, which eliminates the edge-straddling pixels in the original.*

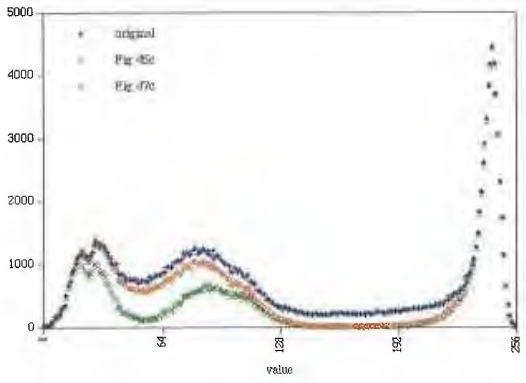
boundary. This line is uniformly dilated to 3 pixels wide and used as a mask to remove edge-straddling pixels from the original. The plot of the resulting histogram, also in **Figure 47**, shows a much greater suppression of the valley between the grey and black peaks. All of these methods are somewhat ad hoc; the particular combination of different region brightnesses present in an image will dictate what edge-finding operation will work best and what fraction of the pixels should be removed.

Boundary lines

One of the shortcomings of thresholding is that the pixels are selected primarily by brightness, and only secondarily by location. This means that there is no requirement for regions to be continuous. Instead of defining a region as a collection of pixels with brightness values that are similar in one or more images, an alternate definition can be based on a boundary.

Manually outlining regions for measurement is one way to use this approach. Various interactive pointing devices, such as graphics tablets (also called drawing pads), touch screens, mice, or light pens, may be used. The drawing can take place while the user looks at the computer screen, at a photographic print on a tablet, or through the microscope, with the pointer device optically superimposed. None of these methods is without problems. Video displays have rather limited

Figure 47. *Brightness histograms from the original image in* ***Figure 46a*** *and the masked images in* ***Figures 46c*** *and* ***48c****, showing the reduction of number of pixels with brightness in the ranges between the main peaks.*





a b c

Figure 48. Removal of edge pixels:
(a) *non-maximum suppression (grey-scale thinning) applied to* ***Figure 46b****;*
(b) *dilation of lines to 3 pixels wide;*
(c) *removal of edges leaving uniform interior regions in the original.*

resolution. Drawing on a video representation of a live image does not provide a record of where you have been. Mice are clumsy pointing devices, light pens lose precision in dark areas of the display, touch screens have poor resolution (and your finger gets in the way), and so on.

It is beyond our purpose here to describe the operation or compare the utility of these different approaches. Regardless of what physical device is used for manual outlining, the method relies on the human visual image processor to locate boundaries and produces a result that consists of a polygonal approximation to the region outline. Most people tend to draw just outside the actual boundary of whatever features they perceive to be important, making dimensions larger than they should be, and the amount of error is a function of the contrast at the edge. (Exceptions exist, of course. Some people draw inside the boundary, but bias is commonly present in all manually drawn outlines.)

Attempts to emulate the human outlining operation with a computer algorithm require a starting point, usually provided by the human. Then the program examines each adjoining pixel to find which has the characteristics of a boundary, usually defined as a step in brightness. Whichever pixel has the highest value of local gradient is selected and added to the growing polygon, and then the procedure is repeated. Sometimes, a constraint is added to minimize sharp turns, such as weighting the pixel values according to direction.

Automatic edge-following suffers from several problems. First, the edge definition is essentially local. People have a rather adaptable capability to look ahead various distances to find pieces of edge to be connected together. Gestalt psychologists describe this as grouping. Such a response is difficult for an algorithm that looks only within a small neighborhood. Even in rather simple images, there may be places along boundaries where the local gradient or other measure of edgeness drops.

In addition, edges may touch where regions abut. The algorithm is equally likely to follow either edge, which of course gives a nonsensical result. There may also be the problem of when to end the process. If the edge is a single, simple line, then it ends when it reaches the starting point. If the line reaches another feature that already has a defined boundary (from a previous application of the routine) or if it reaches the edge of the field of view, then there is no way to complete the outline.

The major problems with edge-following are:

1. It cannot by itself complete the segmentation of the image because it has to be given each new starting point and cannot determine whether there are more outlines that need to be followed.

2. The same edge-defining criteria used for following edges can be applied more easily by processing the entire image and then thresholding.

This produces a line of pixels that may be broken and incomplete (if the edge following would have been unable to continue) or may branch (if several boundaries touch). Methods are discussed in Chapter 7, however, which apply erosion/dilation logic to deal with some of these deficiencies. The global application of the processing operation finds all of the boundaries.

Figure 49 illustrates a few of these effects. The image consists of several hand-drawn dark lines, to which a small amount of random noise is added and a ridge-following algorithm applied (Van Helden, 1994). Each of the user-selected starting points is shown with the path followed by the automatic routine. The settings used for this example instruct the algorithm to consider points out to a distance of 5 pixels in deciding which direction to move in at each point. Increasing this number produces artificially smooth boundaries, and also takes more time as more neighbors must be searched. Conversely, reducing it makes it more likely to follow false turnings. Many of the paths are successful, but a significant number are not. By comparison, thresholding the image to select dark pixels, and then skeletonizing the resulting broad outline as discussed in Chapter 7, produces good boundary lines for all of the regions at once.

The same comparison can be made with a real image. **Figure 50** shows a fluorescence image from a light microscope. In this case, the inability of the fully automatic ridge-following method to track the boundaries has been supplemented by a manually assisted technique. The user draws a line near the boundary, and the algorithm moves the points onto the nearest (within some preset maximum distance) darkest point. This method, sometimes called "active contours" or "snakes," allows the user to overcome many of the difficulties in which the automatic method may wander

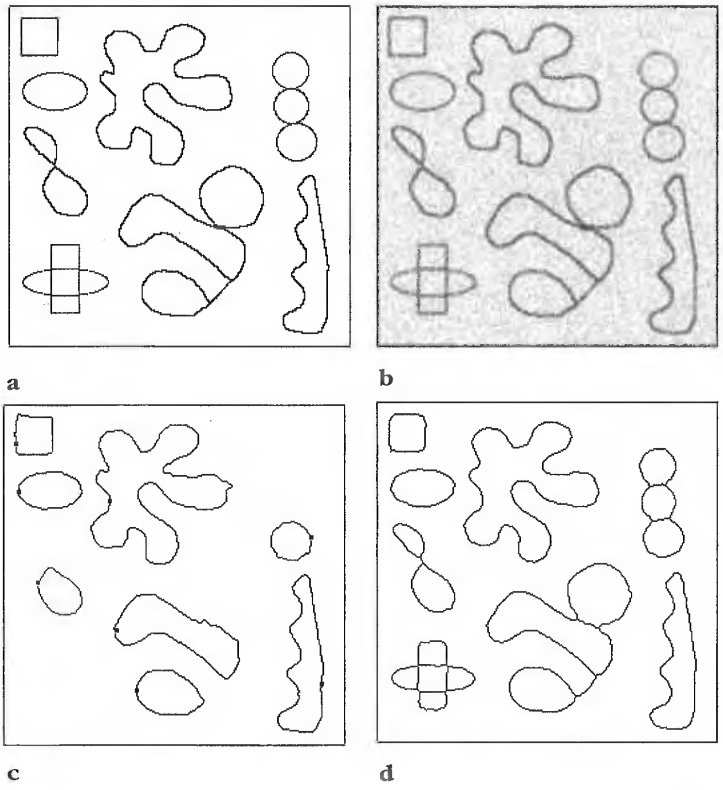
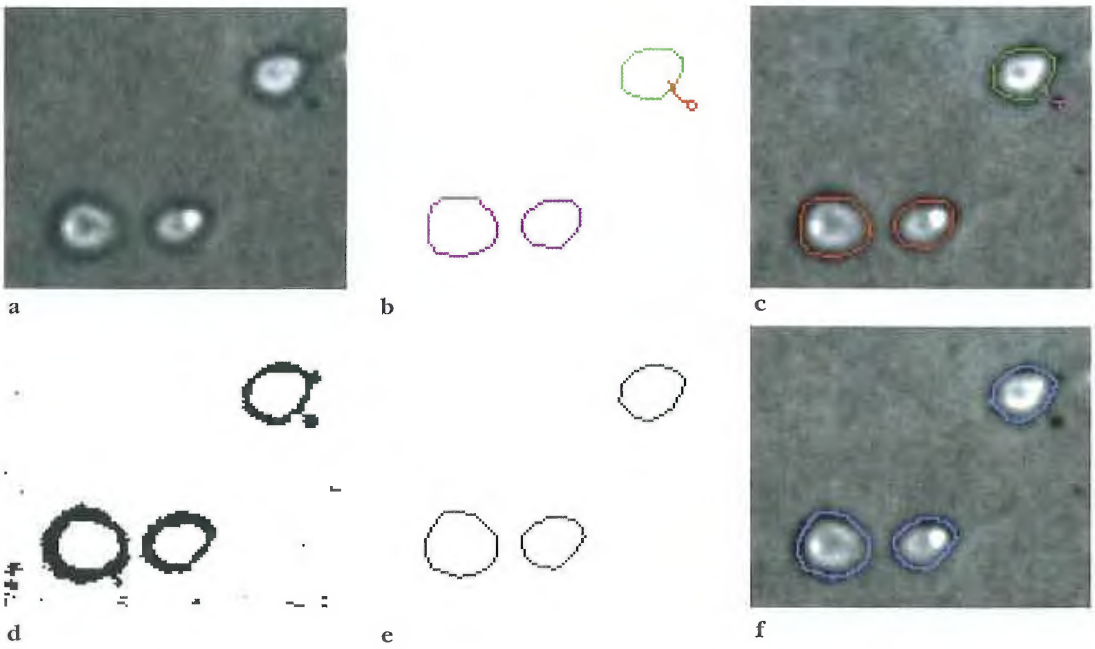


Figure 49. Test image for automatic line-following: ***(a)*** *hand-drawn lines;* ***(b)*** *addition of random noise to* ***a****;* ***(c)*** *lines found by automatic tracing, showing the starting points for each (notice that some portions of crossing or branching line patterns are not followed);* ***(d)*** *lines found by thresholding and skeletonization.*

Figure 50. Light microscope fluorescence image with three features:
(a) *original;*
(b) *edge-following algorithm (blue shows fully automatic results, purple shows a feature that required manual assistance to outline);*
(c) *outlines from* ***b*** *superimposed on the original;*
(d) *brightness thresholding the original image;*
(e) *skeletonized outlines from figure* ***d****;*
(f) *outlines from* ***e*** *superimposed on the original.*



away from the correct line, never to return. It is still faster, however, to use thresholding and skeletonizing to get the boundary lines. Although the details of the lines differ, it is not evident that either method is consistently superior for delineation.

Contours

One type of line that may provide boundary information and is guaranteed to be continuous is a contour line. This is analogous to the iso-elevation contour lines drawn on topographic maps. The line marks a constant elevation or, in our case, a constant brightness in the image. These lines cannot end, although they may branch or loop back upon themselves. In a continuous image or an actual topographic surface, there is always a point through which the line can pass. For a discrete image, the brightness value of the line may not happen to correspond to any specific pixel value. Nevertheless, if there is a pair of pixels with one value brighter than and one value darker than the contour level, then the line must pass somewhere between them.

The contour line can, in principle, be fit as a polygon through the points interpolated between pixel centers for all such pairs of pixels that bracket the contour value. This actually permits measuring the locations of these lines, and the boundaries that they may represent, to less than the dimensions of one pixel, called sub-pixel sampling or measurement. This is rarely done for an entire image because of the amount of computation involved and the difficulty in representing the boundary by such a series of points, which must be assembled into a polygon.

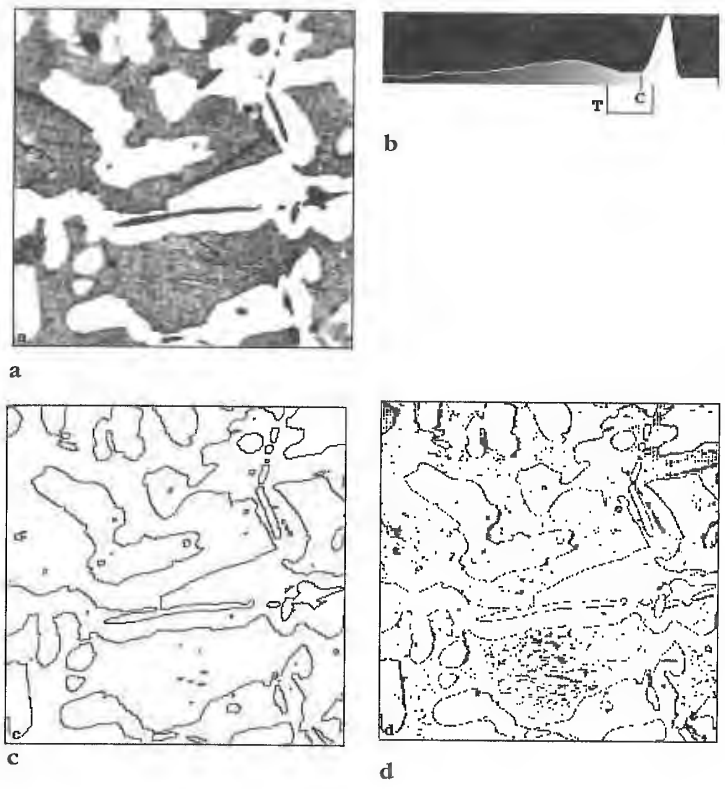


Figure 51. Cast iron (light micrograph) with ferrite (white) and graphite (dark): ***(a)*** *original image;* ***(b)*** *brightness histogram showing levels used for contour (C) and threshold (T);* ***(c)*** *contour lines drawn using the value shown in* ***b****;* ***(d)*** *pixels selected by the threshold setting shown in* ***b****.*

Instead, the most common use of contour lines is to mark the pixels that lie closest to, or closest to and above, the line. These pixels approximate the contour line to the resolution of the pixels in the original image, form a continuous band of touching pixels (touching in an eight-neighbor sense, as discussed in the following paragraph), and can be used to delineate features in many instances. Creating the line from the image is simply a matter of scanning the pixels once, comparing each pixel and its neighbors above and to the left to the contour value, and marking the pixel if the values bracket the test value.

Figure 51 shows a grey-scale image with several contour lines, drawn at arbitrarily chosen brightness values, marked on the histogram. Notice that setting a threshold range at this same brightness level, even with a fairly large range, does not produce a continuous line, because the brightness gradient in some regions is quite steep and no pixels fall within the range. The brightness gradient is very gradual in other regions, so a gradient image (**Figure 52**) obtained from the same original by applying a Sobel operator does not show all of the same boundaries, and does introduce more noise.

Drawing a series of contour lines on an image can be an effective way to show minor variations in brightness, as shown in **Figure 53**. Converting an image to a series of contour lines (**Figure 54**) is often able to delineate regions of similarity or structural meaning, even for complex three-dimensional scenes such as the figure shows. For one important class of images (range images) in which pixel brightness measures elevation, such a set of lines is the topographic map. Such images may result from radar imaging, the CSLM, interferometry, the STM or AFM, and other devices. **Figure 55** shows a scanned stylus image of a coin, with contour lines drawn to delineate the raised surfaces, and a similar image of a ball bearing. The contour lines on the ball show the roughness and out-of-roundness of the surface, and can be measured quantitatively for such a purpose.



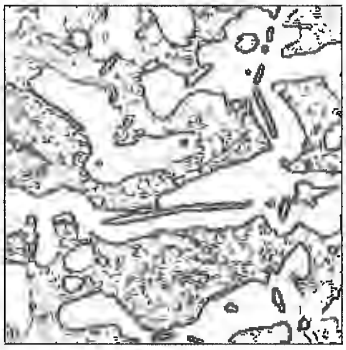


Figure 52. *Gradient image obtained by applying a Sobel operator to* ***Figure 51a****, and the pixels selected by thresholding the 20% darkest (highest gradient) values.*





Figure 53. Ion microprobe image of boron implanted in a silicon wafer: ***(a)*** *original image, in which brightness is proportional to concentration;* ***(b)*** *two iso-brightness or iso-concentration contour values, which make it easier to compare values in different parts of the image.*





Figure 54. *Real-world image* ***(a)*** *and four contour lines drawn at selected brightness values* ***(b).*** *No matter how irregular they become, the lines are always continuous and distinct.*

As discussed in Chapter 13, range images are often produced by surface elevation measurements, as shown in **Figure 56**. This image has elevation contours calculated from stereo pair views of a specimen in the TEM, in which the "mountains" are deposited contamination spots. The information from the contour lines can be used to generate a rendered surface as shown in the figure and discussed in Chapter 13, in order to illustrate the surface topography.

The contours drawn by selecting a brightness value on the histogram provide the same outline information as the edge pixels in regions determined by thresholding with the same value. The contour lines can be filled in to provide a pixel representation of the feature, using the logic discussed in Chapter 7. Conversely, the solid regions can be converted to an outline by another set of binary image processes. If the contour line is defined by pixel values, the information is identical to the thresholded regions. If sub-pixel interpolation has been used, then the resolution of the features may be better. The two formats for image representation are entirely complementary, although they have different advantages for storage, measurement, etc.

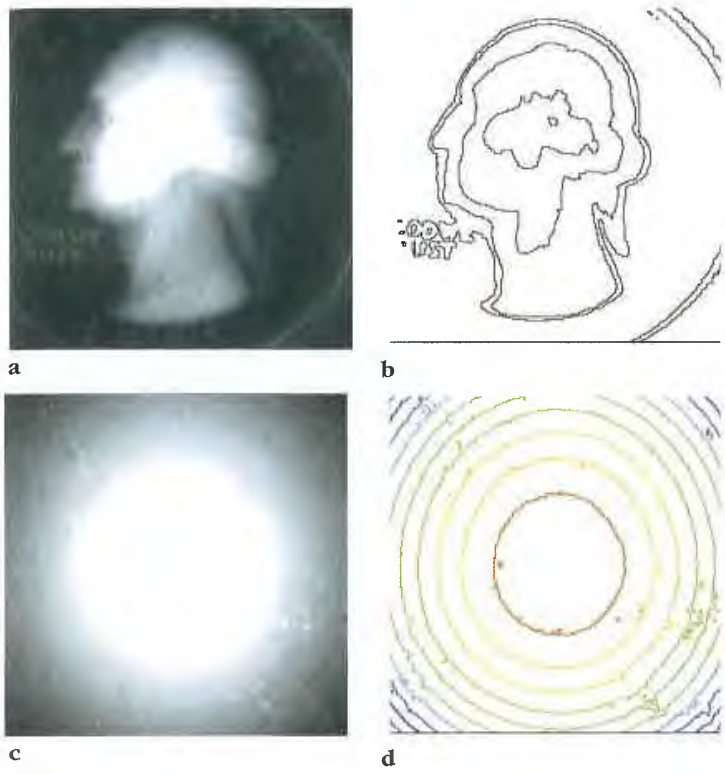


Figure 55. Range images and contour lines: *(a) range image of a coin; (b) contour lines delineating raised areas on the surface; (c) range image of a ball bearing; (d) contour lines showing roughness and out-of-roundness (color coded according to elevation).*

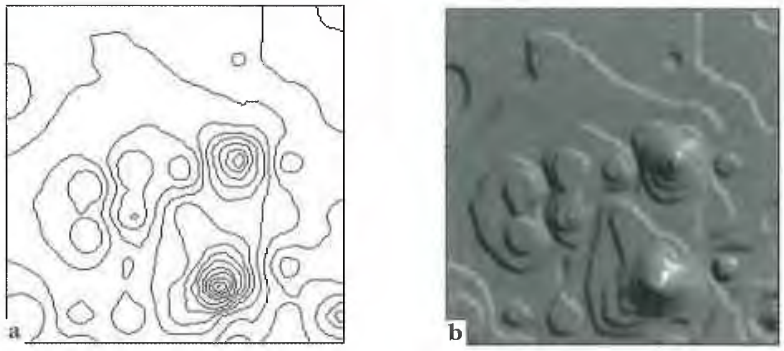


Figure 56. Elevation contour map from a range image (a) *in which pixel brightness represents surface elevation, and a reconstructed and rendered view of the surface* ***(b)***.

Image representation

Different representations of the binary image are possible; some are more useful than others for specific purposes. Most measurements, such as feature area and position, can be directly calculated from a pixel-based representation by simple counting procedures. This can be stored in less space than the original array of pixels by using run-length encoding (also called chord encoding). This treats the image as a series of scan lines. For each sequential line across each region or feature, it stores the line number, start position, and length of the line. **Figure 57** illustrates this schematically.

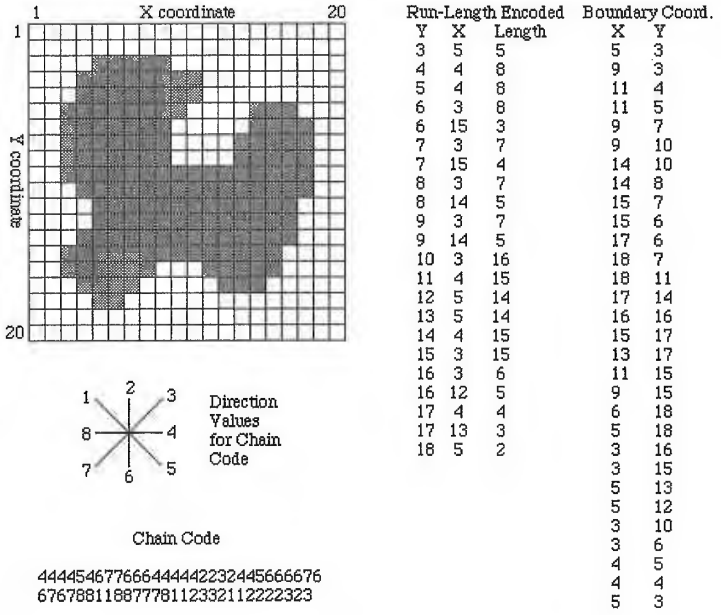


Figure 57. *Encoding the same region in a binary image by run-length encoding, boundary polygonal representation, or chain code.*

For typical images, the pixels are not randomly scattered, but collected together into regions or features so that the run-length encoded table is much smaller than the original image. This is the method used, for instance, to transmit fax messages over telephone lines. **Figure 58** shows how a black-and-white image is encoded for this purpose. In this example, the original image is 256 × 256 = 65,536 pixels, while the run-length table is only 1460 bytes long. The run-length table can be used directly for area and position measurements, with even less arithmetic than the pixel array. Because the chords are in the order in which the raster crosses the features, some logic is required to identify the chords with the features, but this is often done as the table is built.

The chord table is poorly suited for measuring feature perimeters or shape. Boundary representation, consisting of the coordinates of the polygon comprising the boundary, is superior for this task, although it is awkward for dealing with regions containing internal holes, because there is nothing to relate the interior boundary to the exterior. Again, logic must be used to identify the internal boundaries, keep track of which ones are exterior and which are interior, and construct a hierarchy of features within features, if needed.

A simple polygonal approximation to the boundary can be produced when it is needed from the run-length table by using the endpoints of the series of chords, as shown in **Figure 57**. A special form of this polygon can be formed from all of the boundary points, consisting of a series of short vectors from one boundary point to the next. On a square pixel array, each of these lines is either 1 or $\sqrt{2}$ pixels long and can only have 1 of 8 directions. Assigning a digit from 1 to 8 (or 0 to 7, or -3 to +4, depending on the particular implementation) to each direction and writing all of the numbers for the closed boundary in order produces chain code, also shown in **Figure 57**.

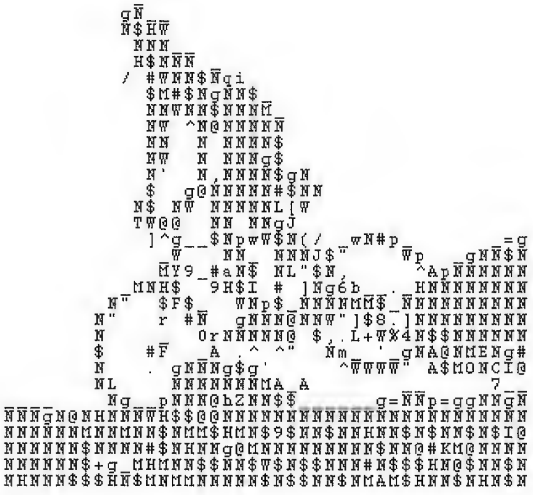
This form is particularly well suited for calculating perimeter or describing shape (Freeman, 1961, 1974; Cederberg, 1979). The perimeter is determined by counting the number of even and odd digits, multiplying the number of odd ones by $\sqrt{2}$ to correct for diagonal directions, and adding. The chain code also contains shape information, which can be used to locate corners, simplify the shape of the outline, match features independent of orientation, or calculate various shape descriptors.



a



b



c

Figure 58. Representing a black-and-white image for fax transmission:
(a) original;
(b) run-length encoded (each horizontal line is marked with a red point at its start, just the position of the red point and the length of the line are sent);
(c) representing the same image with formed characters, a trick commonly used two decades ago.

Most current-generation imaging systems use an array of square pixels, because it is well suited both to raster-scan acquisition devices and to processing images and performing measurements. If rectangular pixels are acquired by using a different pixel spacing along scan lines than between the lines, processing in either the spatial domain with neighborhood operations or in the frequency domain becomes much more difficult, because the different pixel distances as a function of orientation must be taken into account. The use of rectangular pixels also complicates measurements.

With a square pixel array, a minor problem exists, which we have already seen in the previous chapters on image processing: the four pixels diagonally adjacent to a central pixel are actually farther away than the four sharing an edge. An alternative arrangement that has been used in a few systems is to place the pixels in a hexagonal array. This has the advantage of equal spacing between all neighboring pixels, which simplifies processing and calculations. Its great disadvantage, however, is that standard cameras and other acquisition and display devices do not operate that way.

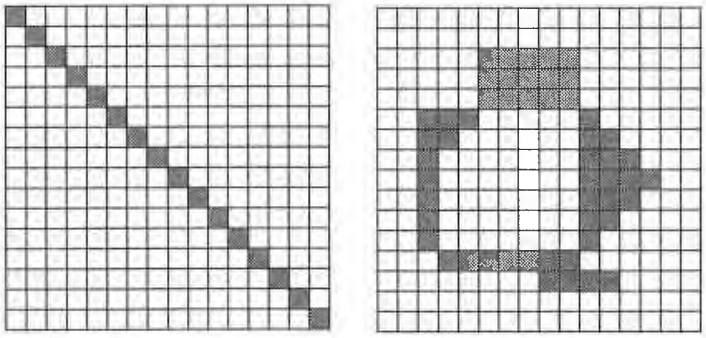


Figure 59. Ambiguous images: ***(a)*** *If the pixels are assumed to touch at their corners, then this shows a line that separates the background pixels on either side; but those pixels also touch at their corners. If 8-connectedness or 4-connectedness is selected for feature pixels, then the opposite convention applies to background pixels;* ***(b)*** *This shows either four separate features or one containing an internal hole, depending on the touching convention.*

For a traditional square pixel array, it is necessary to decide whether pixels adjacent at a corner are actually touching. This will be important for the binary processing operations in Chapter 7. It is necessary in order to link pixels into features or follow the points around a boundary, as discussed previously. Although it is not evident whether one choice is superior to the other, whichever one is made, the background (the pixels which surround the features) must have the opposite relationship.

Figure 59 shows this dual situation. If pixels within a feature are assumed to touch any of their eight adjacent neighbors (called eight-connectedness), then the line of pixels in **Figure 59a** separates the background on either side, and the background pixels that are diagonally adjacent do not touch. They are therefore four-connected. Conversely, if the background pixels touch diagonally, the pixels are isolated and only touch along their faces. For the second image fragment shown, choosing an eight-connected rule for features (dark pixels) produces a single feature with an internal hole. If a four-connected rule is used, there are four features and the background, now eight-connected, is continuous.

This means that simply inverting an image (interchanging white and black) does not reverse the meaning of the features and background. **Figure 60** shows a situation in which the holes within a feature (separated from the background) become part of a single region in the reversed image. This can cause confusion in measurements and binary image processing. When feature dimensions as small as one pixel are important, there is some basic uncertainty. This is unavoidable and argues for using large arrays of small pixels to define small dimensions and feature topology accurately.

Other segmentation methods

Other methods are used for image segmentation besides the ones based on thresholding discussed previously. These are generally associated with fairly powerful computer systems and with attempts to understand images in the sense of machine vision and robotics (Ballard and Brown, 1982; Wilson and Spann, 1988). Two of the most widely described are split-and-merge and region growing, which appear to lie at opposite extremes in method.

Split-and-merge is a top-down method that begins with the entire image. Some image property is selected as a criterion to decide whether everything is uniform. This criterion is often based on the statistics from the brightness histogram. If the histogram is multimodal, or has a high standard

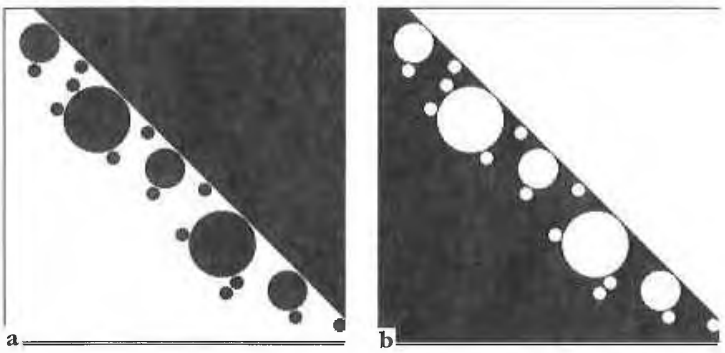


Figure 60. *Reversing an image (interchanging features and background) without changing the connectedness rules alters meaning. In image* ***a*** *the black pixels all touch at corners (8-connectedness), and so this is one feature with an irregular boundary. In image* ***b*** *the white pixels do not touch (4-connectedness), and so these are separate holes within the feature.*

deviation, etc., then the region is assumed to be nonuniform and is divided into four quadrants. Each quadrant is examined in the same way and subdivided again if necessary. The procedure continues until the individual pixel level is reached. The relationship between the parent region and the four quadrants, or children, is typically encoded in a quadtree structure, and another name is sometimes applied to this approach.

This is not the only way to subdivide the parent image and encode the resulting data structure. Thresholding can be used to divide each region into arbitrary subregions, which can be subdivided iteratively. This can produce final results having less blocky boundaries, but the data structure is much more complex, because all of the regions must be defined and the time required for the process is much greater.

Subdividing regions alone does not create a useful image segmentation. After each iteration of subdividing, each region is compared to adjacent ones that lie in different squares at a higher level in the hierarchy. If they are similar, they are merged together. The definition of "similar" may use the same tests applied to the splitting operation, or comparisons may be made only for pixels along the common edge. The latter has the advantage of tolerating gradual changes across the image.

Figure 61 shows an example in which only four iterations have been performed. A few large areas have already merged, and their edges will be refined as the iterations proceed. Other parts of the image contain individual squares that require additional subdivision before regions become visible.

An advantage of this approach is that a complete segmentation is achieved after a finite number of iterations (for instance, a 512-pixel-square image takes 9 iterations to reach individual pixels, because $2^9 = 512$). Also, the quadtree list of regions and subregions can be used for some measurements, and the segmentation identifies all of the different types of regions at one time. By comparison, thresholding methods typically isolate one type of region or feature at a time. They must be applied several times to deal with images containing more than one class of objects.

On the other hand, the split-and-merge approach depends on the quality of the test used to detect inhomogeneity in each region. Small subregions within large uniform areas can easily be missed with this method. Standard statistical tests that assume, for example, a normal distribution of pixel brightness within regions are rarely appropriate for real images, so more complicated procedures must be used (Yakimovsky, 1976). Tests used for subdividing and merging regions can also be expressed as image processing operations. A processed image can reveal the same edges and texture used for the split-and-merge tests in a way that allows direct thresholding. This is potentially less efficient, because time-consuming calculations may be applied to parts of the image that are

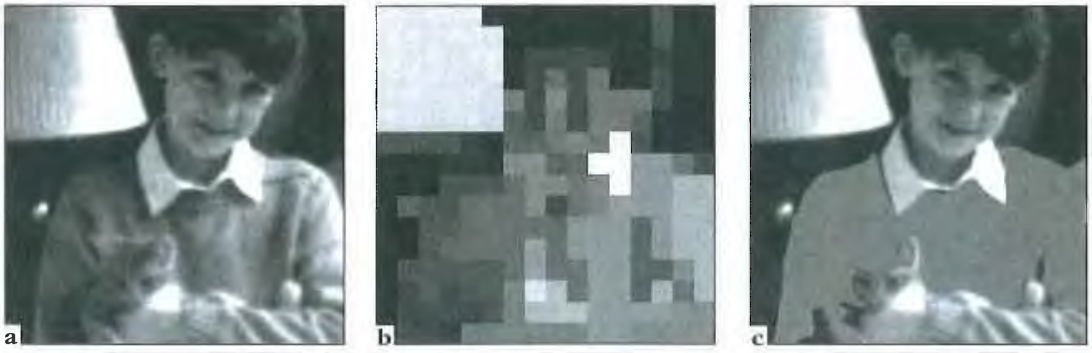


Figure 61. Other segmentation methods:
(a) *original grey-scale image;*
(b) *split and merge after four iterations;*
(c) *region growing from a point in the girl's sweater.*

uniform, but the results are the same. Thresholding also has the advantage of identifying similar objects in different parts of the field of view as the same, which may not occur with split-and-merge.

Region-growing starts from the bottom, or individual pixel level, and works upward. Starting at some seed location (usually provided by the operator but in some cases located by image processing tools such as the top hat filter), neighboring pixels are examined one at a time and added to the growing region if they are sufficiently similar. Again, the comparison may be made to the entire region or just to the local pixels, with the latter method allowing gradual variations in brightness. The procedure continues until no more pixels can be added. **Figure 61c** shows an example in which one region has been identified; notice that it includes part of the cat as well as the girl's sweater. Then a new region is begun at another location. **Figure 62** shows an example of the application of this technique to a color image. The red boundary line shows the extent of the region grown from a starting point within the intestine.

If the same comparison tests are implemented to decide whether a pixel belongs to a region, the result of this procedure is the same as top-down split-and-merge. The difficulty with this approach is that the starting point for each region must be provided. Depending on the comparison tests employed, different starting points may not grow into identical regions. Also, no ideal structure is



Figure 62. *Region-growing applied to a color image. The red lines show the boundaries of the region.*

available to encode the data from this procedure, beyond keeping the entire pixel array until classification is complete. The complete classification is slow, because each pixel must be examined individually.

Region-growing also suffers from the conflicting needs to keep the test local, to see if an individual pixel should be added to the growing region, and to make it larger in scale, if not truly global, to ensure that the region has some unifying and distinct identity. If too small a test region is used, a common result is that regions leak out into adjoining areas or merge with different regions. This leaking or merging can occur if even a single pixel on the boundary can form a bridge.

Finally, there is no easy way to decide when the procedure is complete and all of the meaningful regions in the image have been found. Region-growing may be a useful method for selecting a few regions in an image, as compared to manual tracing or edge following, for example, but it is rarely the method of choice for complex images containing many regions (Zucker, 1976).

Edge following was mentioned above in terms of an algorithm that tries to mimic a human drawing operation by tracing along a boundary, and at each point selecting the next pixel to step toward based on local neighborhood values. Human vision does not restrict itself to such local decisions, but can use remote information to bridge over troublesome points. A machine vision approach that does the same constructs outlines of objects as deformable polygons. The sides of the polygon may be splines rather than straight lines, to achieve a smoother shape. These deformable boundaries are referred to as "snakes" (Kass et al., 1987). The minimum length of any side, the maximum angular change at any vertex, and other arbitrary fitting constants must reflect some independent knowledge about the region that is to be fitted. They must generally be adjusted for each application until the results are acceptable as compared to visual judgment.

Fitting is accomplished for all of the points on the boundary at once using a minimization technique, so the snakes can accommodate some missing or confusing points. They are particularly useful for tracking moving boundaries in a sequence of images, because the deformation of the snake from one moment to the next must be small. This makes them useful for tracking moving objects in robotics vision. When applied to three-dimensional arrays of voxels, as in medical imaging, they become deformable polyhedra and are called balloons.

The general classification problem

The various methods described so far have relied on human judgment to recognize the presence of regions and to define them by delineating the boundary or selecting a range of brightness values. Methods have been discussed that can start with an incomplete definition and refine the segmentation to achieve greater accuracy or consistency. Fully automatic techniques also determine how many classes of objects are present and fully subdivide the image to isolate them. They are not used very often in small computer-based systems, however, and are often much less efficient than using some human input. The task of general image segmentation can be treated as an example of a classification problem. Similar to most techniques involving elements or artificial intelligence, it may not use the same inputs or decision methods that a human employs, but it seeks to duplicate the results (and often succeeds).

One successful approach to general classification has been used with satellite imagery, in which many wavelength bands of data are available (Reeves, 1975). If each pixel in the image is plotted in a high-dimensionality space, where each axis is the measured brightness in one of the wavelength bands, it is expected that points corresponding to different classes of land use, crop type, soil or rock type, and so forth will cluster together and the clusters will be well separated from each other, as indicated in **Figure 63**. The problem then reduces to finding the clusters and fitting boundaries between them that can be used for classification. Finding clusters and boundaries between regions is discussed in a more general context in Chapter 10.

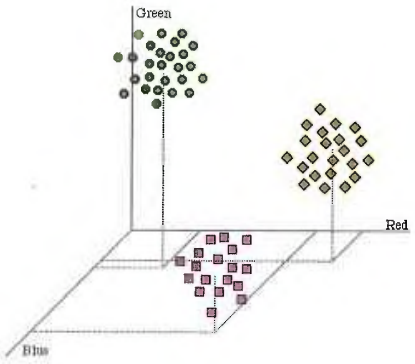


Figure 63. *Schematic illustration of pixel classification in color space. Each pixel is plotted according to its color values, and clusters identify the various regions present.*

Reduced to a single dimension (a simple grey-scale image), this classification begins with the brightness histogram. The cluster analysis looks for peaks and tries to draw thresholds between them. This is successful in a few specialized tasks, such as counting cells of one type on a microscope slide. As the number of dimensions increases, for instance using the RGB or HSI data from color imagery or adding values from a derived texture or gradient image, the separation of the clusters usually becomes more distinct. Satellite imagery with several discrete visible and infrared wavelength bands is especially well suited to this approach.

Clusters are easier to recognize when they contain many similar points, but minor regions or uncommon objects may be overlooked. Also, the number of background points surrounding the clusters (or more often lying along lines between them) confuse the automatic algorithms. These points arise from the finite size of pixels that straddle the boundaries between regions. Finding a few major clusters may be straightforward — being sure that all have been found is not.

Even after the clusters have been identified (and here, some *a priori* knowledge or input from a human can be of great assistance), different strategies can be employed to use this information to classify new points. One is to surround each cluster with a boundary, typically either a polyhedron formed by planes lying perpendicular to the lines between the cluster centers, or n-dimensional ellipsoids. Points falling inside any of these regions are immediately classified.

Particularly for the ellipsoid case, it is also possible to have a series of concentric boundaries that enclose different percentages of the points in the cluster, which can be used to give a probability of classification to new points. This is sometimes described as a "fuzzy" classification method.

A third approach is to find the nearest classified point to each new point and assign that identity to the new one. This method has several drawbacks, particularly when some densely populated clusters and others with very few members are present, or when the clusters are close or overlapping. It requires considerable time to search through a large universe of existing points to locate the closest one, as well. An extension of this technique is also used, in which a small number of nearest neighbors are identified and "vote" for the identity of the new point.

Segmentation of grey-scale images into regions for measurement or recognition is probably the most important single problem area for image analysis. Many novel techniques have been used that are rather ad hoc and narrow in their range of applicability. Review articles by Fu and Mui (1981) and Haralick and Shapiro (1988) present good guides to the literature. Most standard image analysis textbooks, such as Rosenfeld and Kak (1982), Castleman (1979), Gonzalez and Wintz (1987), Russ (1990b), and Pratt (1991) also contain sections on segmentation.

All these various methods and modifications are used extensively in other artificial intelligence situations (see, for example, Fukunaga, 1990). They may be implemented in hardware, software, or

some combination of the two. Only limited application of any of these techniques has been made to the segmentation problem, although it is likely that the use of such methods will increase in the future as more color or multiband imaging is done and readily accessible computer power continues to increase.



Processing Binary Images

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 those 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.

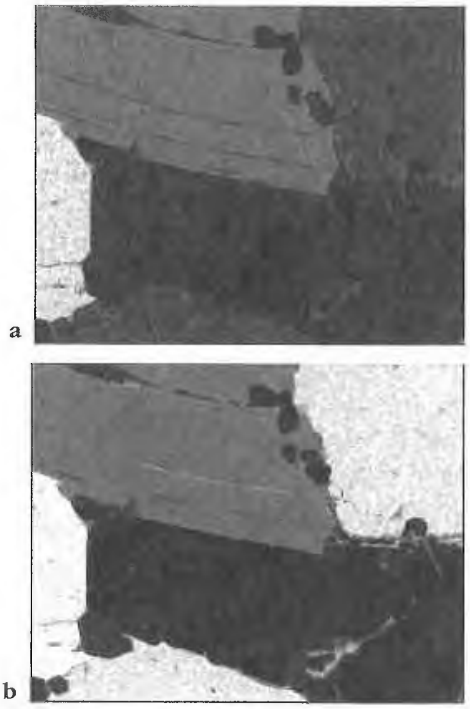


Figure 1. Colocalization: *(a, b) x-ray maps showing the intensity distribution for Al and Si in a mineral; (c) colocalization plot.*

When a colocalization plot shows specific combinations of intensity values that are share 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.

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.*

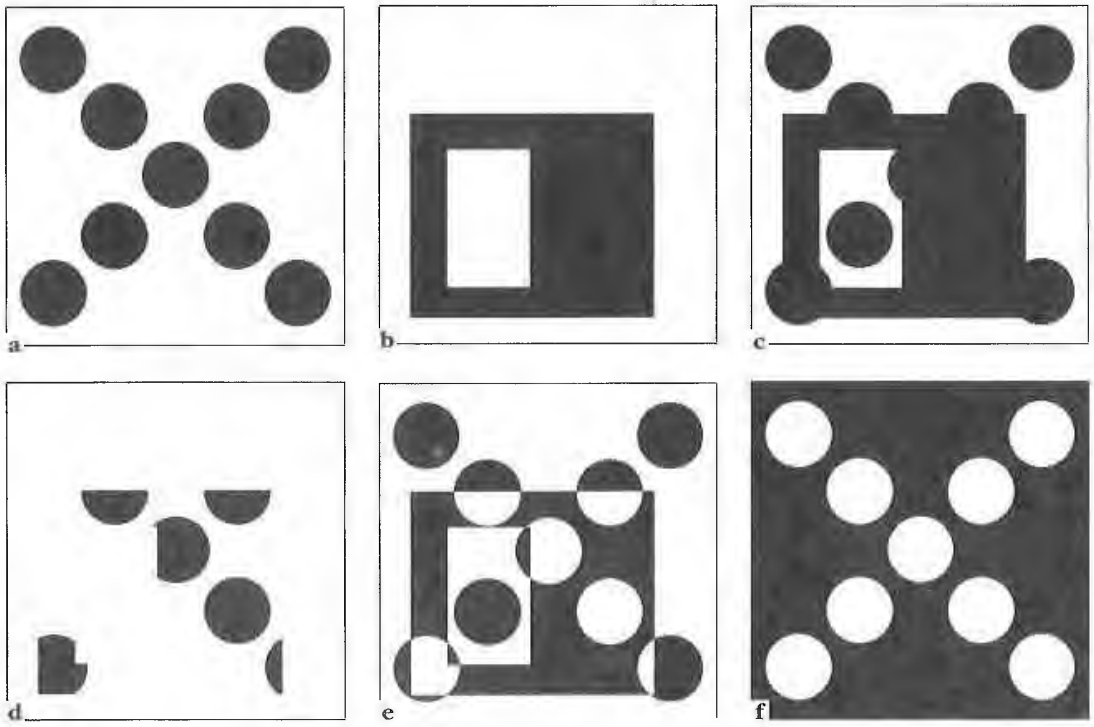
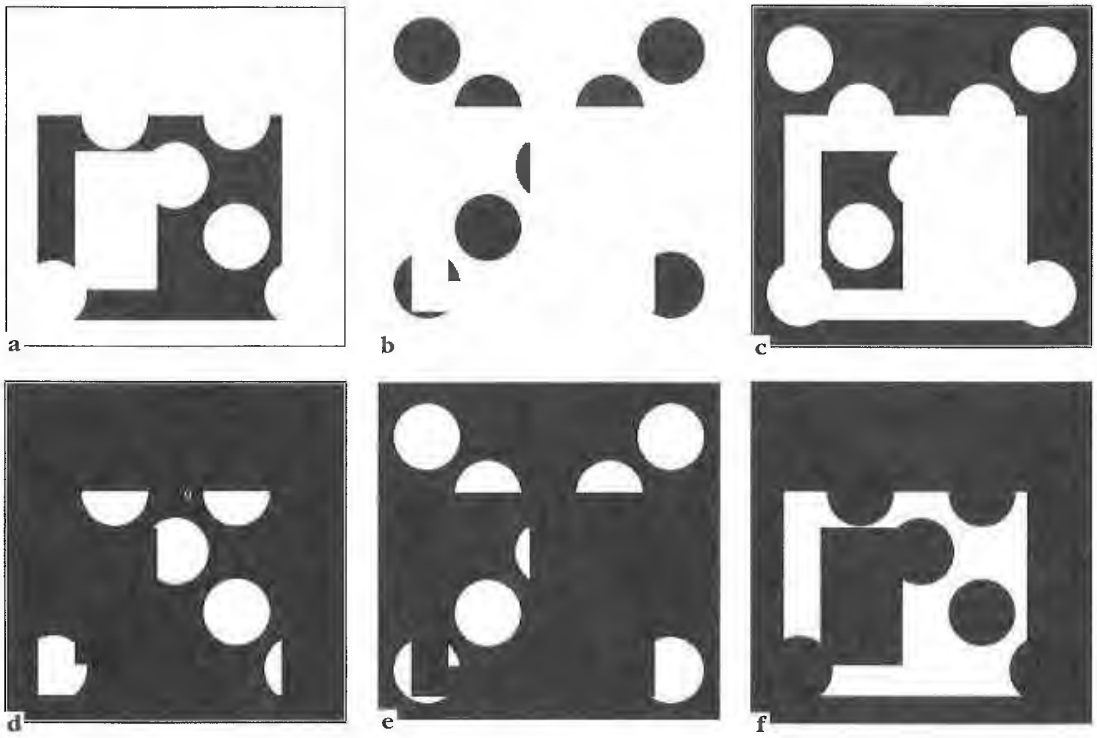


Figure 3. Combined Boolean operations:
(a) *(NOT A) AND B;*
(b) *A AND (NOT B);*
(c) *(NOT A) AND (NOT B);*
(d) *NOT (A AND B);*
(e) *(NOT A) OR B;*
(f) *A OR (NOT B)*



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 (NOT A) AND B will produce an image containing pixels that lie within B but outside A. This is quite different from NOT (A AND B), which selects pixels that are not ON in both A and B. It is also different from A AND (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-discernible 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

(Fe OR Cr OR Al) AND (NOT (Ti OR S)) (1)

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-

Figure 4. SEM results from a mineral:
(a) *backscattered electrons;*
(b) *secondary electrons;*
(c) *silicon (Si) x-ray map;*
(d) *iron (Fe) x-ray map;*
(e) *copper (Cu) x-ray map;*
(f) *silver (Ag) x-ray map.*

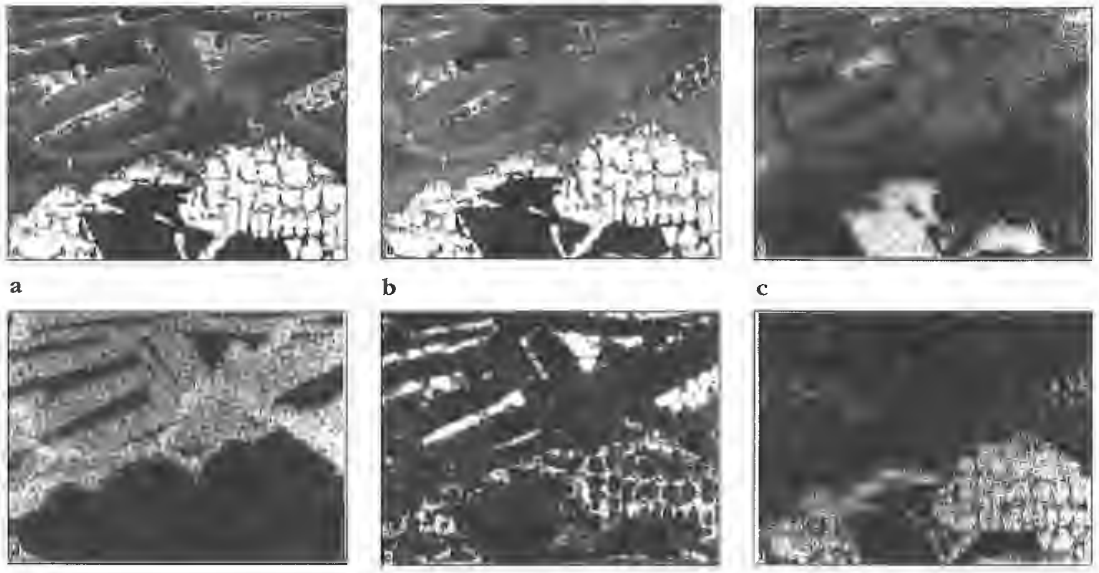




Figure 5. (a) *Iron;* ***(b)*** *iron AND NOT silicon*



a b c

*Figure 6. (a) Silver; (b) bright levels from backscattered electron image; (c) image **a** AND image **b**.*



Figure 7. Further combination to delineate structure: *(Cu OR Ag) AND NOT (Fe).*

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

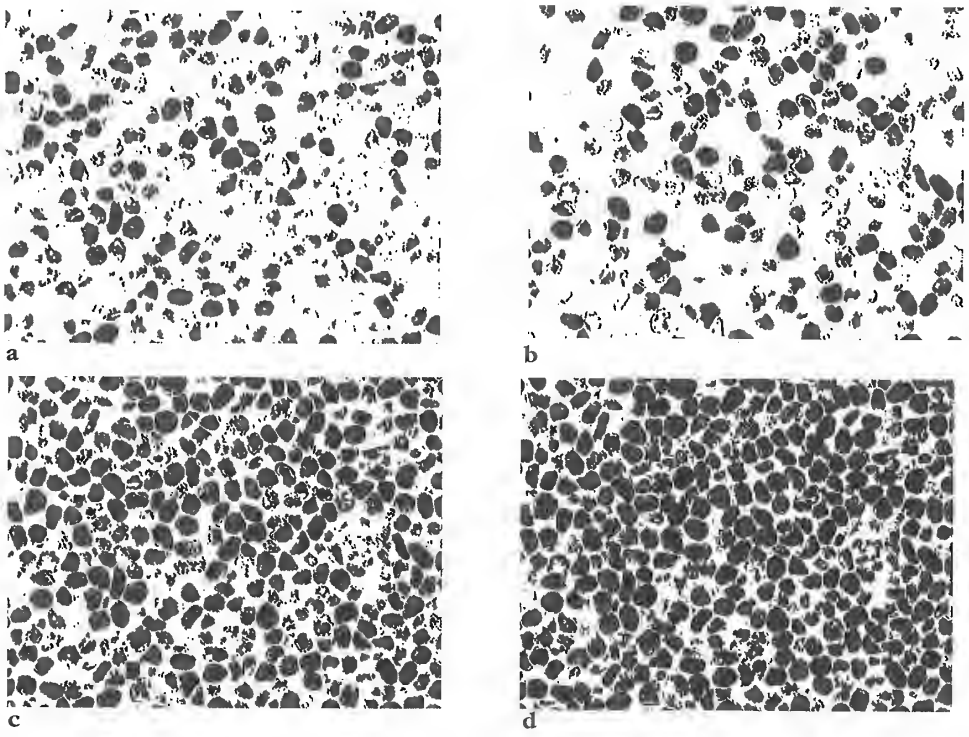


Figure 8. Combining multiple binary images.
*(a, b) binary images obtained by thresholding two of the polarized light images of a petrographic thin section of a sandstone (**Figure 81** in **Chapter 4**);*
(c) the result of ORing together six such images from different rotations of the analyzer;
(d) comparison binary image produced by thresholding the grey scale image obtained by combining the same six color images to keep the brightest pixel at each location.

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 aluminum alloy are masked by thresholding the magnitude of the gradient to isolate only the boundaries.

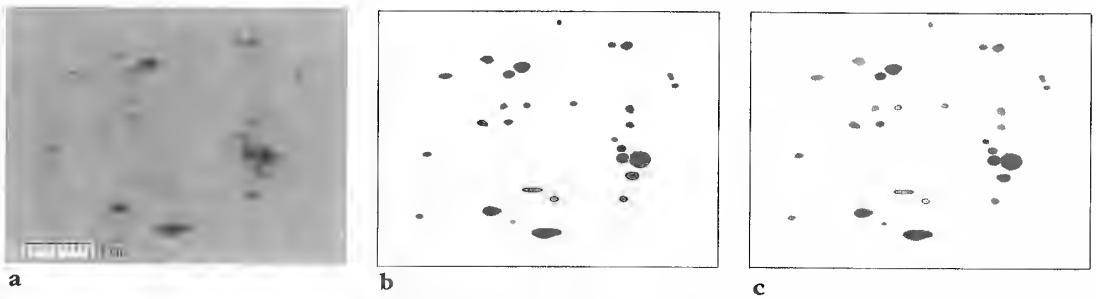


***Figure 9. Preserving feature intensity values:** (a) original 2D gel; (b) thresholded spots; (c) masked image in which pixels within the spots retain their original brightness values.*



a b c

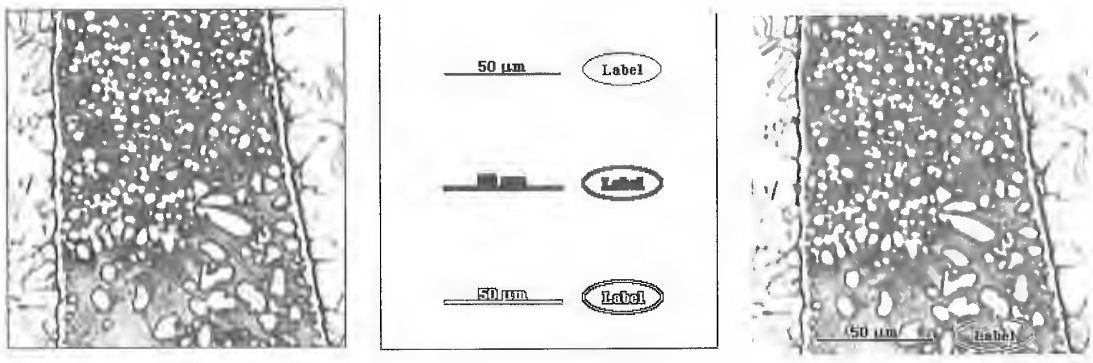
Figure 10. *Masking one image with another. The direction of a Sobel gradient applied to the light microscope image of an aluminum alloy is shown only in the regions where the magnitude of the gradient is large.*

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



a b c

Figure 11. *Using a mask to apply a label to an image. The original image contains both white and black areas, so that simple superimposition of text will not be visible. A mask is created by dilating the label and Ex-ORing that with the original. The composite is then superimposed on the grey-scale image.*

Figure 12. *Hudsonian Godwits searching for a nesting site on an SEM image of an alumina fracture surface.*



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

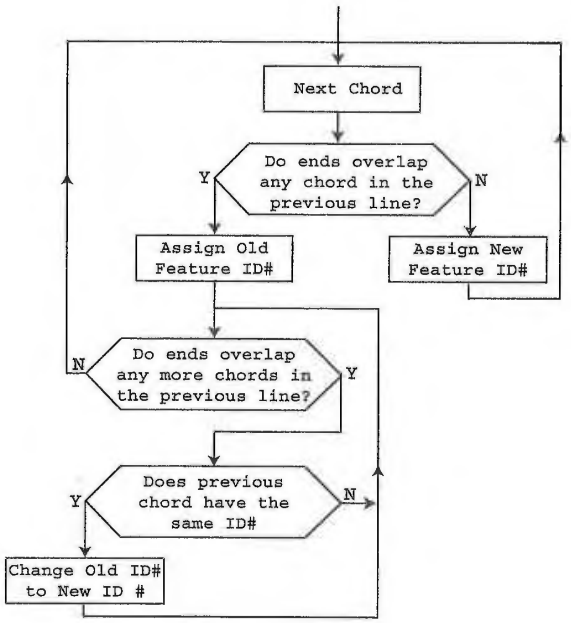


Figure 13. *Flow chart for grouping touching pixels in a run-length or chord-encoded array into features and assigning ID numbers.*

(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.

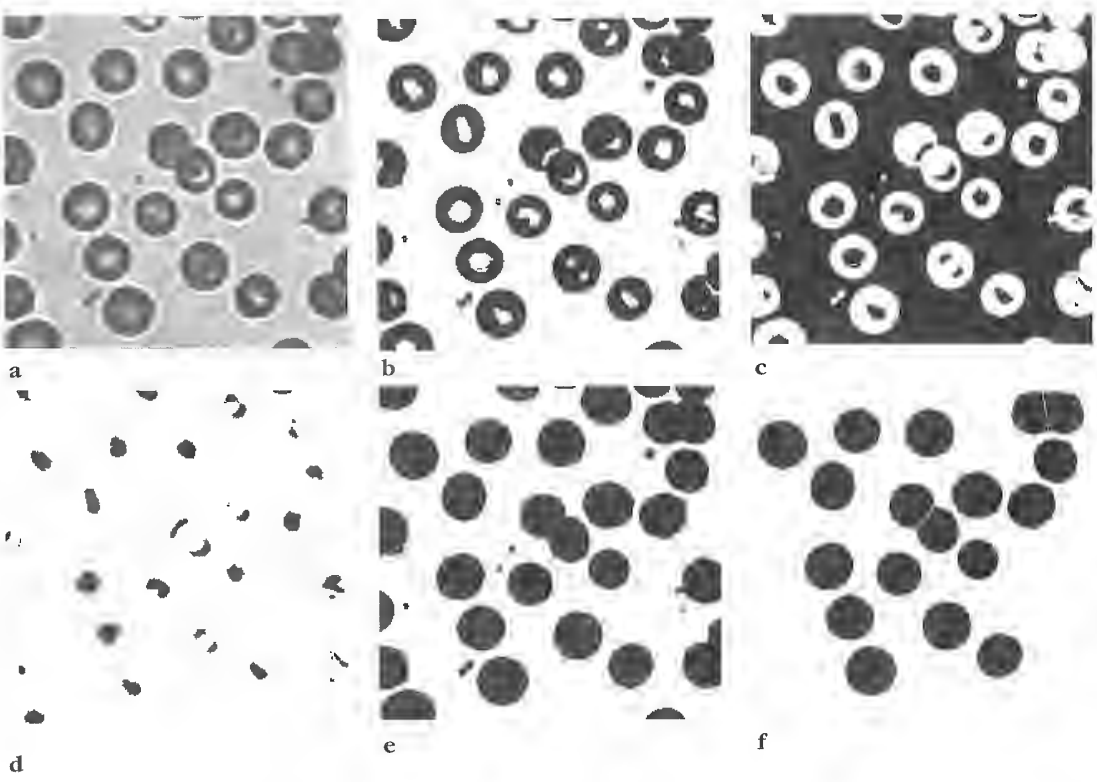
One very simple example of the application of this technique is shown in **Figure 15**. In this image of spherical particles, the center of each feature has a brightness very close to that of the substrate due to the lighting. Thresholding the brightness values gives a good delineation of the outer boundary of the particles, but the centers have holes. Filling them as described produces a corrected representation of the particles, which can be measured. This type of processing is commonly required for SEM images, whose brightness varies as a function of local surface slope so that particles frequently appear with bright edges and dark centers.

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****.*



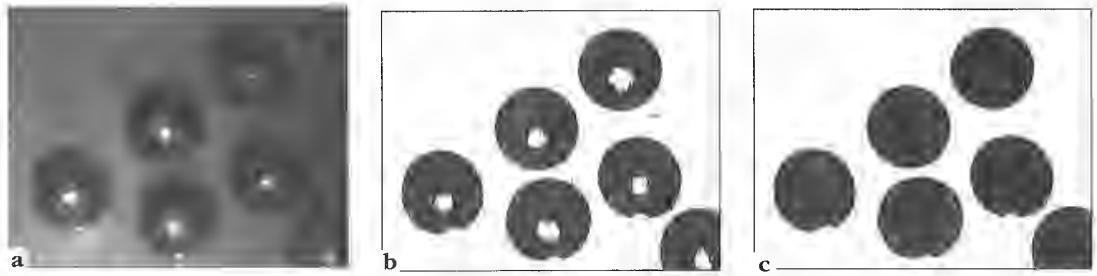


Figure 15. Image of buckshot with near-vertical incident illumination;
(a) *original grey-scale image;*
(b) *brightness thresholded after leveling illumination;*
(c) *internal holes filled and small regions (noise) in background removed by erosion.*

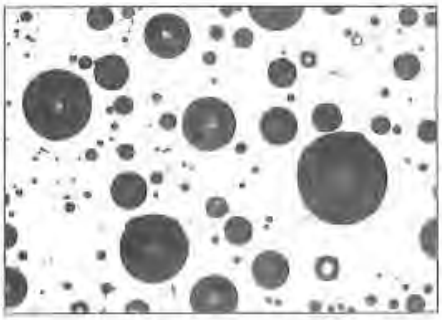


Figure 16. *Light microscope image of a polished section through an enamel coating on steel (courtesy V. Benes, Research Institute for Metals, Panenské Brezany, Czechoslovakia) shows bright spots of reflected light within many pores (depending on their depth).*

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****.*

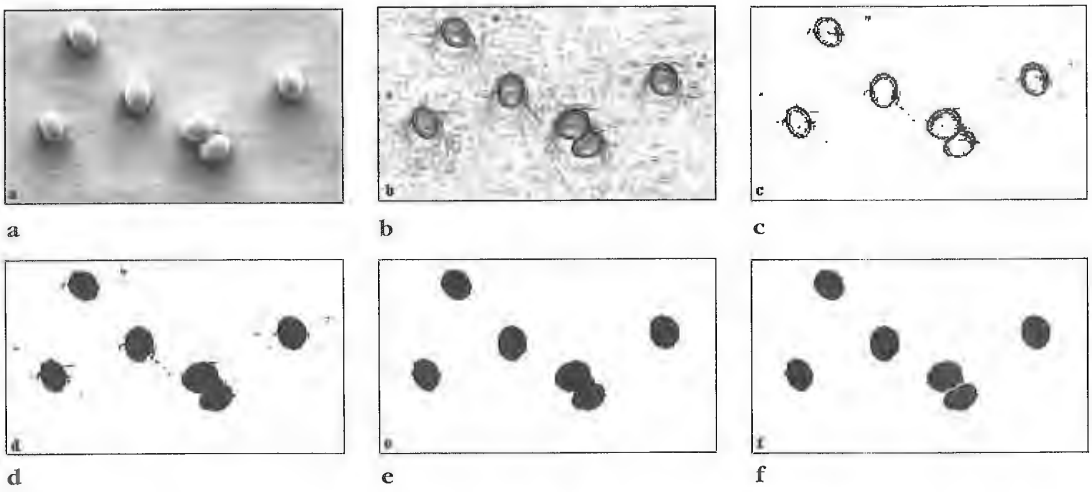
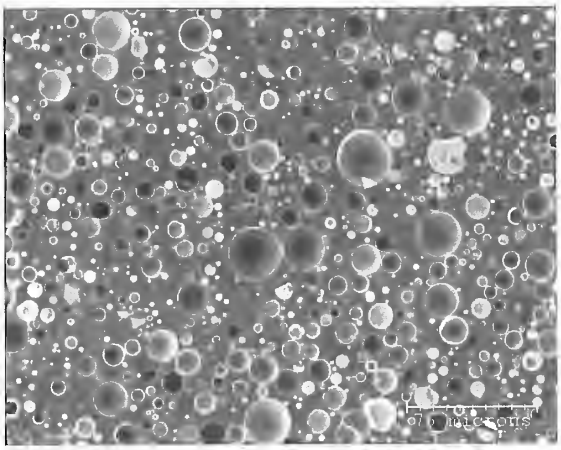
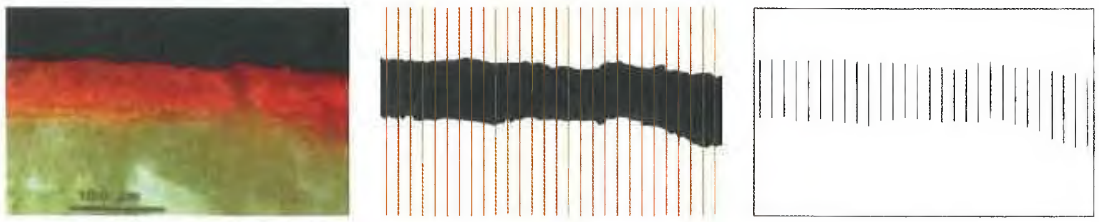


Figure 18. *Section through an epoxy resin containing bubbles. To delineate the bubbles for measurement, the bright, dark, and outlined pores must be processed in different ways and the results combined with a Boolean OR.*





a b c

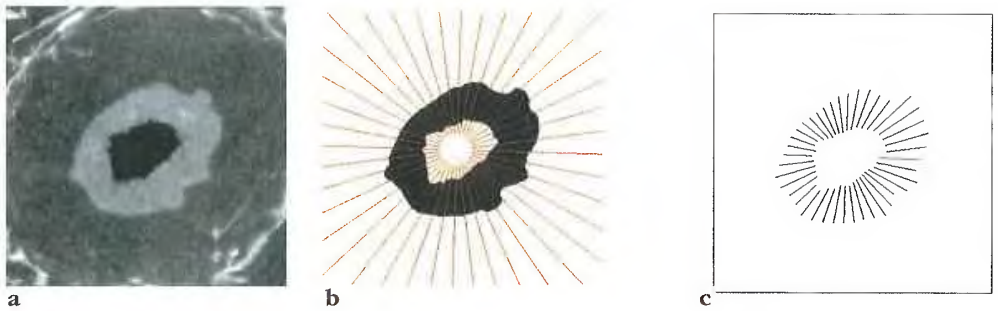
Figure 19. Measurement of layer thickness:
(a) *paint layer viewed in cross section;*
(b) *thresholded layer, with superimposed grid of vertical lines;* ***(c)*** *AND of lines with layer producing line segments for measurement.*

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.

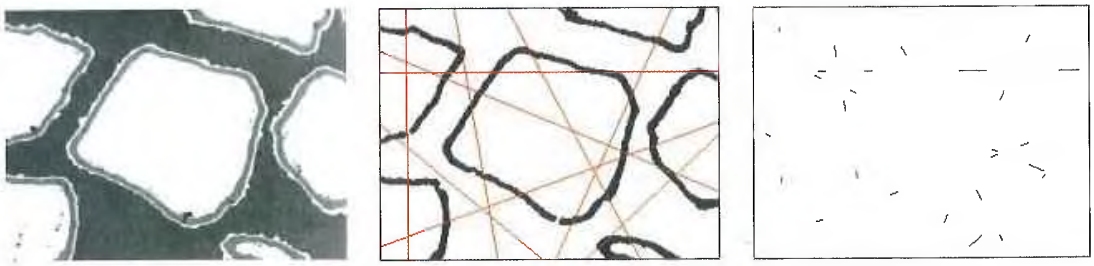
In the example of **Figure 20**, the vein is approximately circular in cross section and the lines do not perpendicularly intersect the wall, introducing a cosine error in the measurement which may or may not be acceptable. That the cross section is not round may indicate that the section plane is not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



a b c

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).*



a b c

Figure 21. Measuring coating thickness on particles:
(a) original grey-scale image of a random section through embedded, coated particles;
(b) thresholded binary image of the coating of interest with superimposed grid of random lines;
(c) AND of the lines with the coating producing line segments for measurement.

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 a 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.

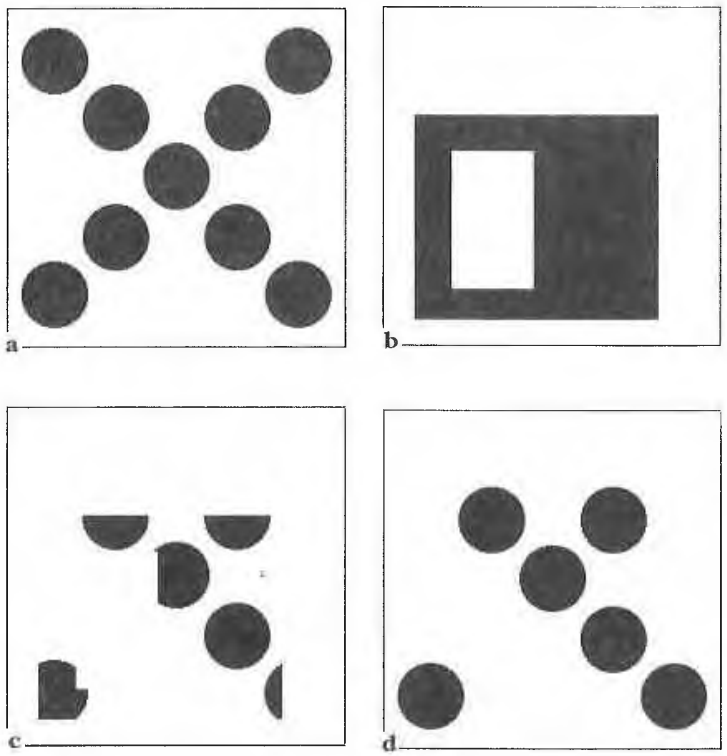


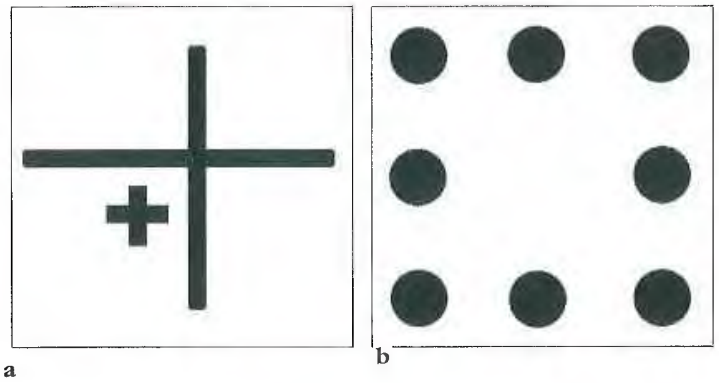
Figure 22. *Schematic diagram of feature-based AND: **(a, b)** test images; **(c)** pixel-based Boolean AND of images a and b; **(d)** feature based AND of image **a** with image **b**.*

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



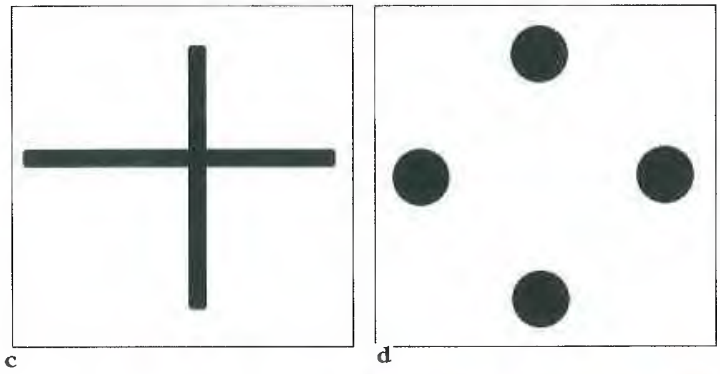


***Figure 23. Feature-based Boolean logic used to combine two test images (a, b): (c) a** Feature-AND **b**; (d) **b** Feature-AND **a**.*

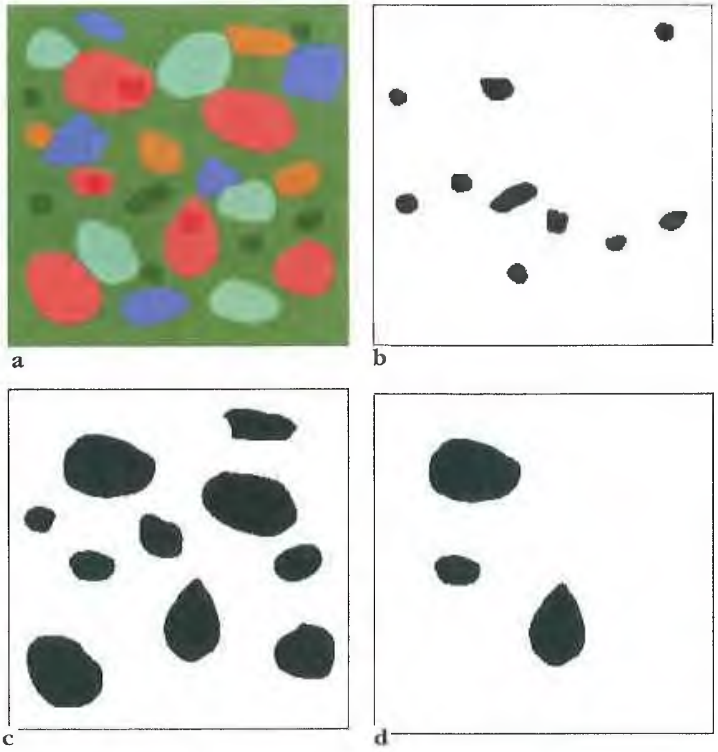


***Figure 24.** Example of feature selection using markers. The red features and the dark spots in the original image **a** are thresholded to produce separate binary images **b** and **c**. The dark spots are used as markers to select those red features which contain dark markers **d**.*

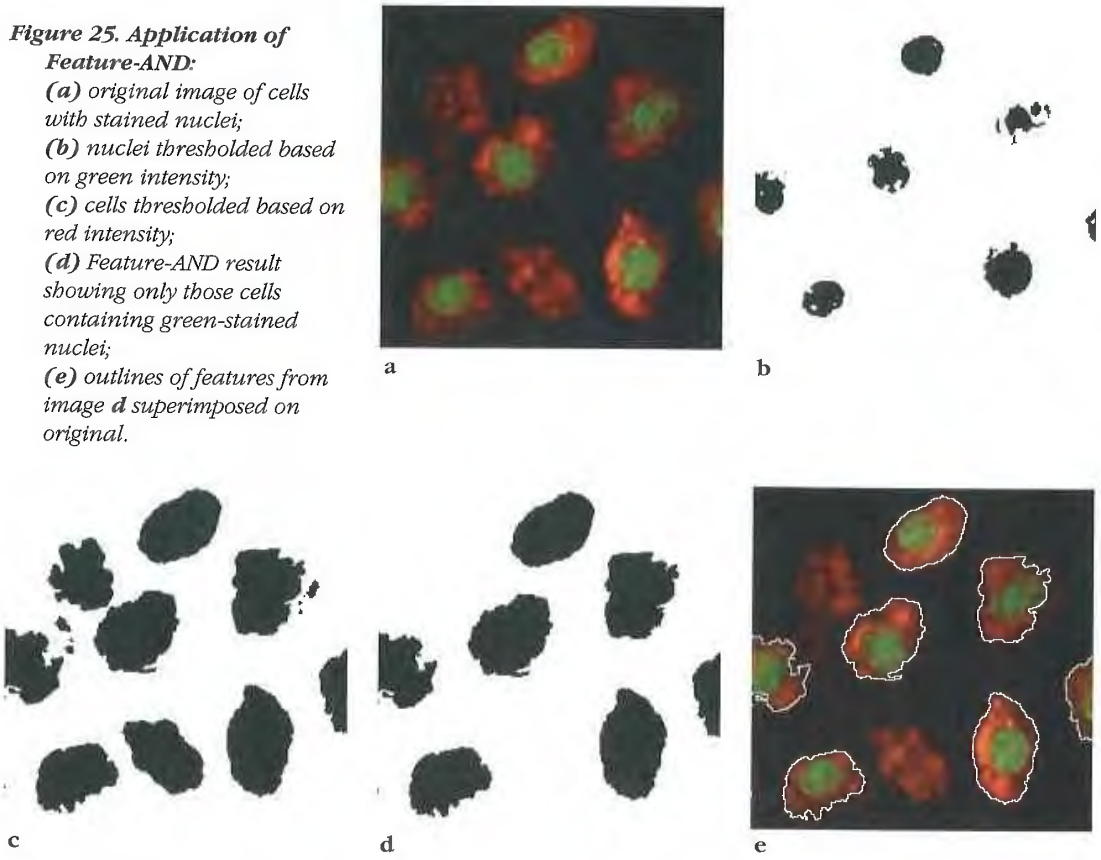


Figure 25. Application of Feature-AND: *(a) original image of cells with stained nuclei; (b) nuclei thresholded based on green intensity; (c) cells thresholded based on red intensity; (d) Feature-AND result showing only those cells containing green-stained nuclei; (e) outlines of features from image*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

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.

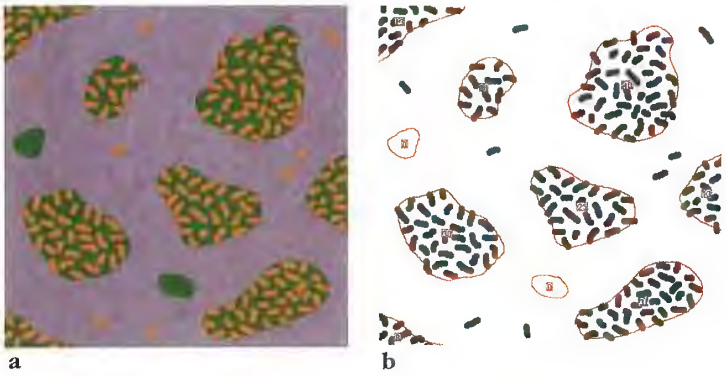
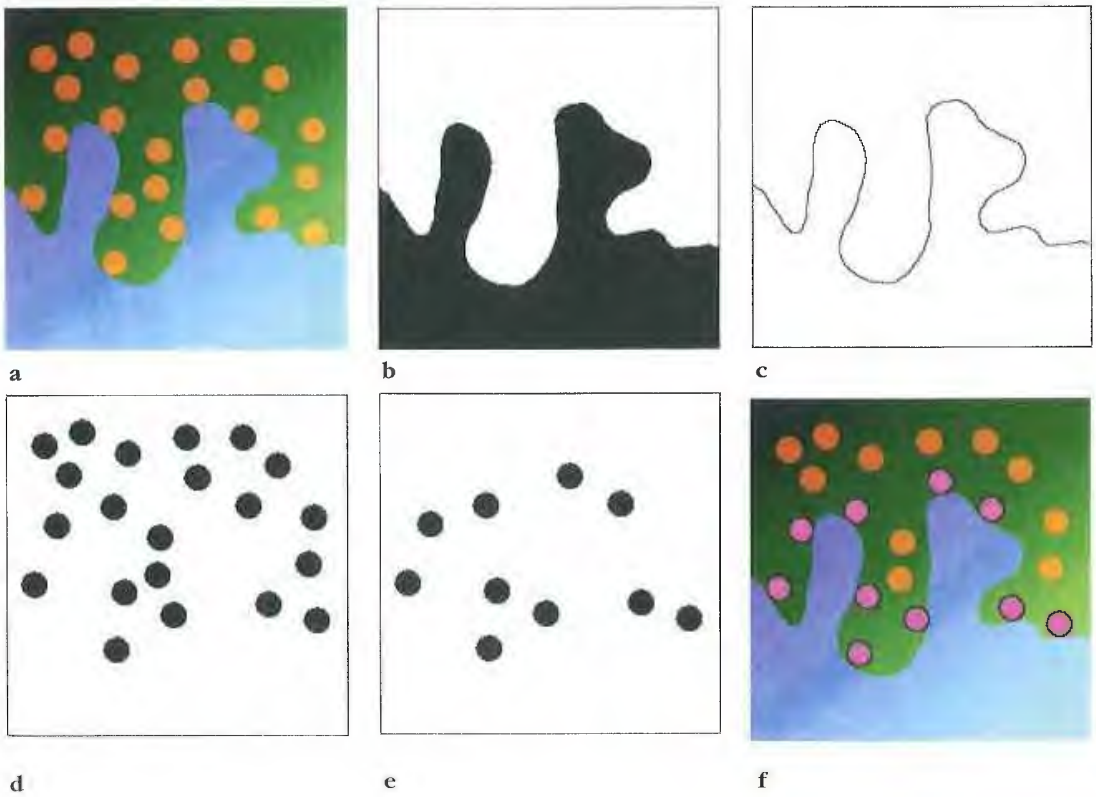


Figure 26. Colony counting:
(a) image representing colonies of bacterial cells, some of which extend beyond the stained area;
(b) counted results showing the number of cells in each colony.

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 custer, 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

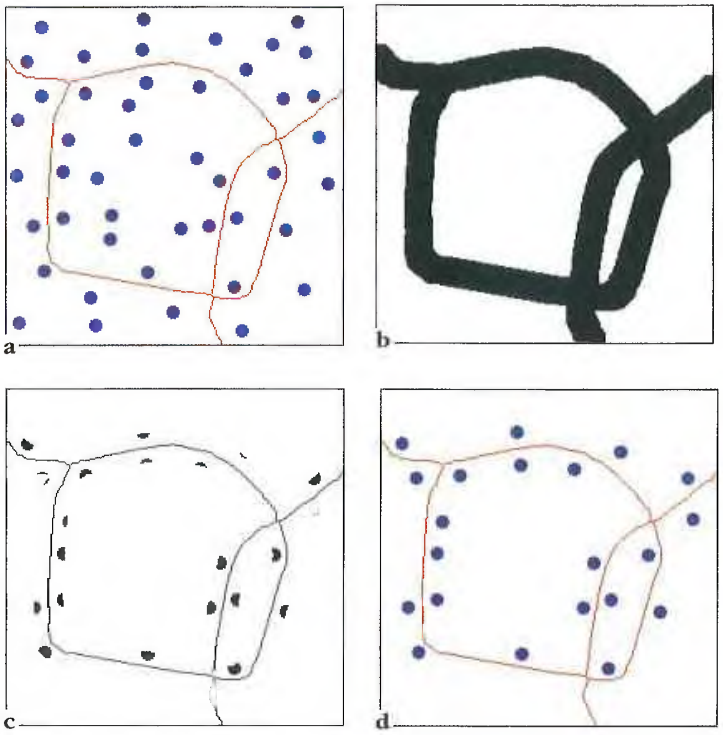


Figure 28. Comparison of pixel- and feature-AND:
(a) *diagram of an image containing features and a boundary;*
(b) *the boundary line, made thicker by dilation;*
(c) *pixel-based AND of image* ***b*** *and* ***a*** *(incomplete features and one divided into two parts);*
(d) *feature-AND of images* ***b*** *and* ***a*** *(all features within a specified distance of the boundary).*

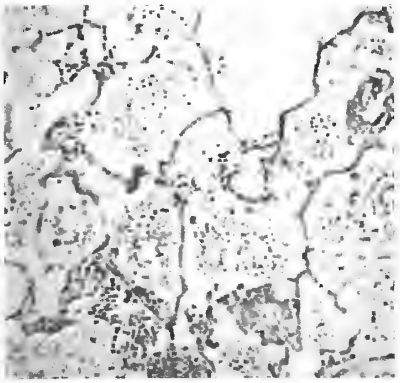
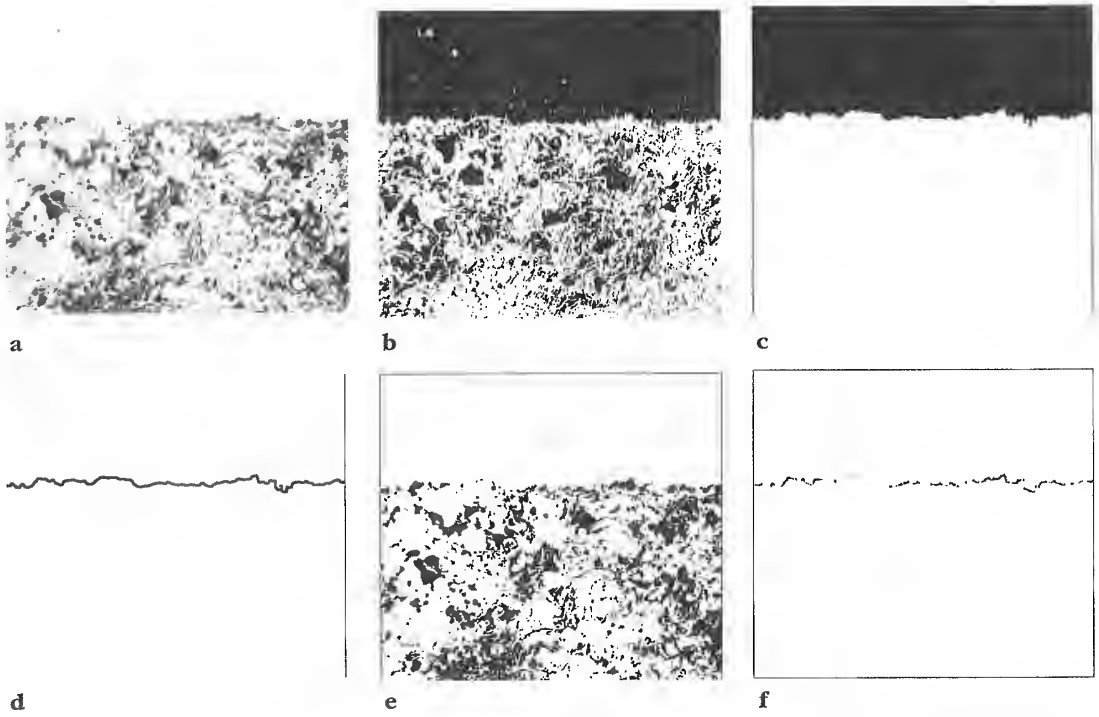


Figure 29. *Light microscope image of polished section through a steel used at high temperature in boiler tubes. Notice the depletion of carbides (black dots) in the region near grain boundaries. This effect can be measured using procedures described in the text.*

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.*



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 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.

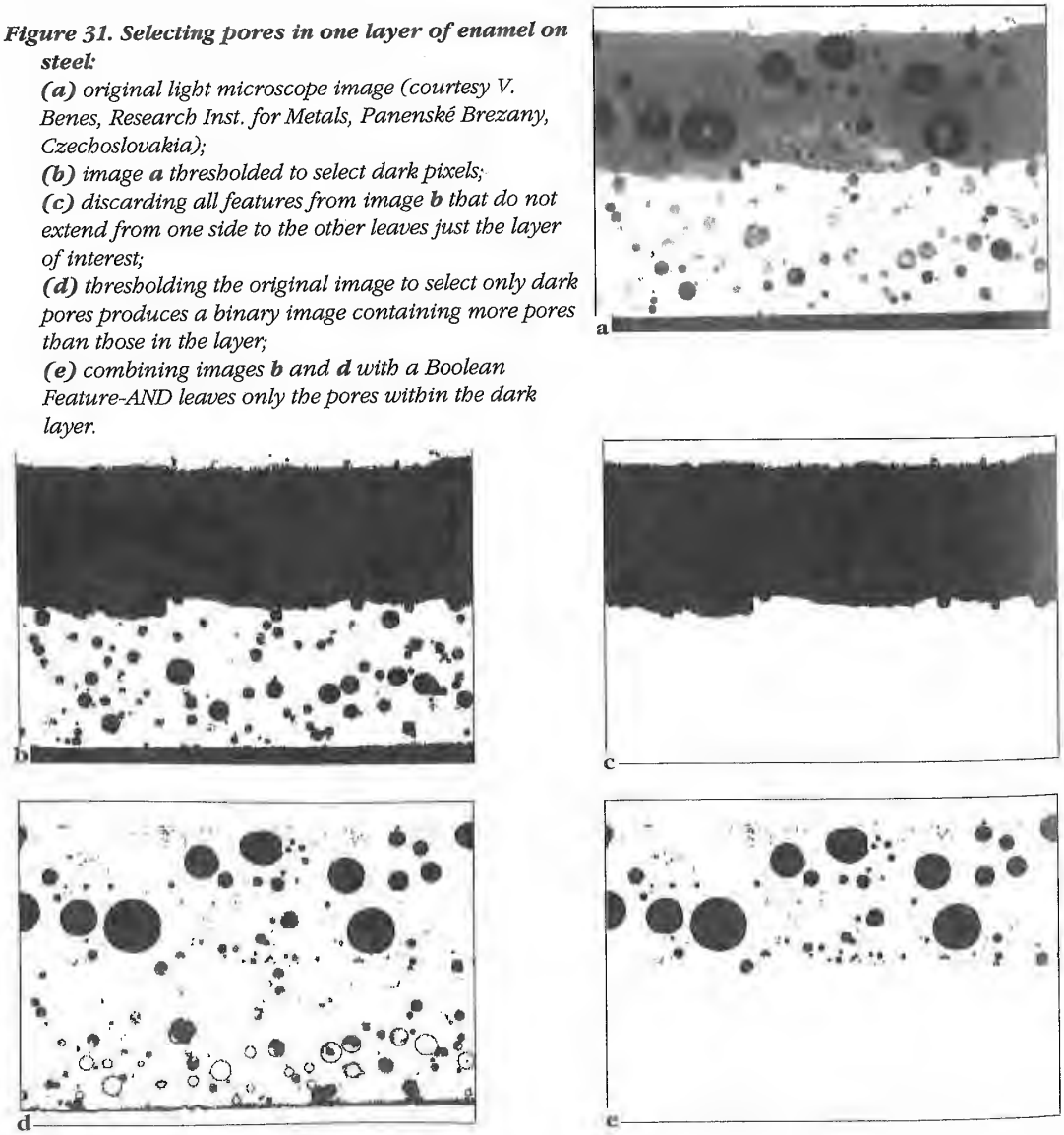


Figure 31. Selecting pores in one layer of enamel on steel:
(a) *original light microscope image (courtesy V. Benes, Research Inst. for Metals, Panenské Brezany, Czechoslovakia);*
(b) *image* ***a*** *thresholded to select dark pixels;*
(c) *discarding all features from image* ***b*** *that do not extend 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

$$N_V = \frac{Count}{2 \cdot Area \cdot T} \tag{2}$$

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 (N_1 and N_2). 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

$$N = N_1 + N_2 - 2 \cdot N_{common} \tag{3}$$

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: (#1 NOT F-AND #2) and (#2 NOT 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.

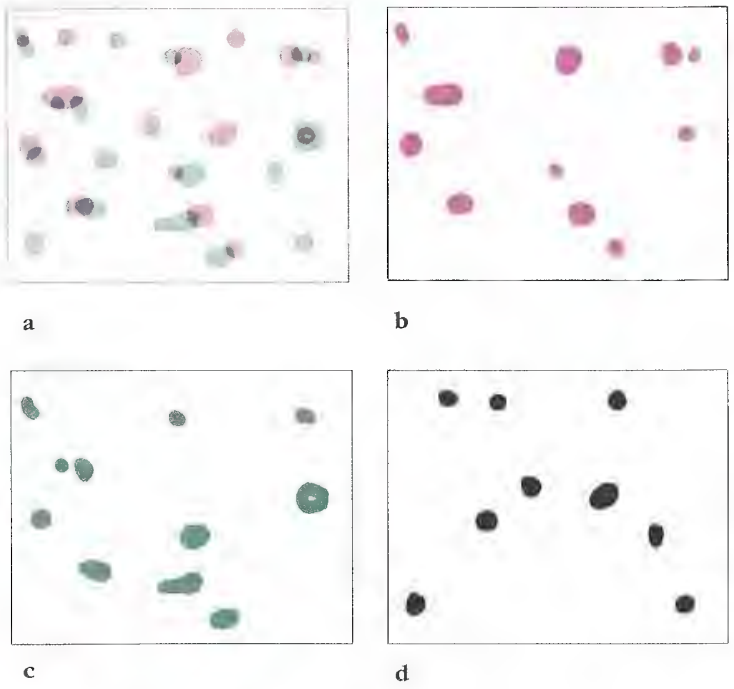


Figure 32. Implementation of the Disector:
(a) *two section images, overlaid in different colors to show matching features;*
(b) *1 F-AND 2 showing features in plane 2 matched with plane 1;*
(c) *2 F-AND 1 showing the features matched in the other plane;*
(d) *ORing together the 1 NOT F-AND 2 with 2 NOT F-AND 1 leaves just the unmatched features in both planes that area to be counted.*

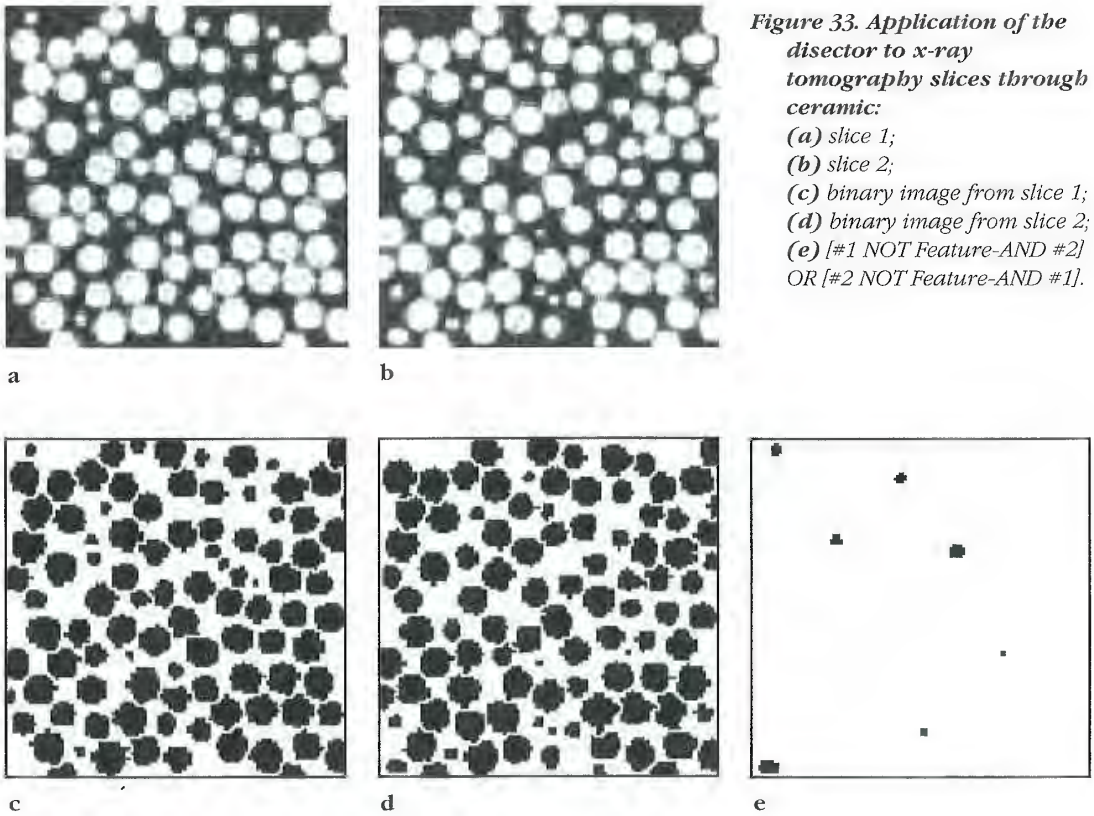


Figure 33. Application of the disector to x-ray tomography slices through a ceramic:
(a) *slice 1;*
(b) *slice 2;*
(c) *binary image from slice 1;*
(d) *binary image from slice 2;*
(e) *[#1 NOT Feature-AND #2] OR [#2 NOT Feature-AND #1].*

A technique for dealing with such situations has been described as "double thresholding" by Olsson (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).

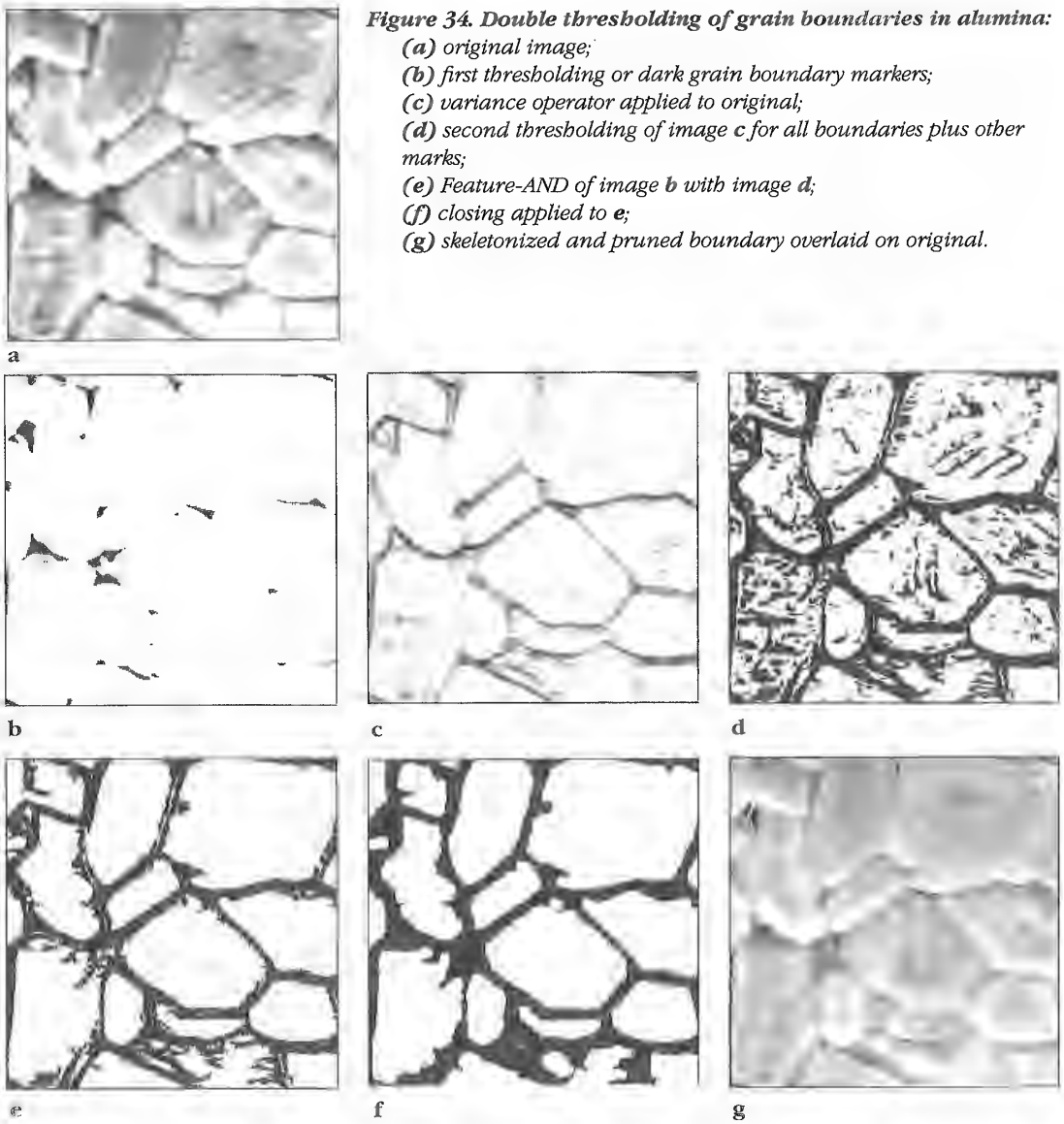
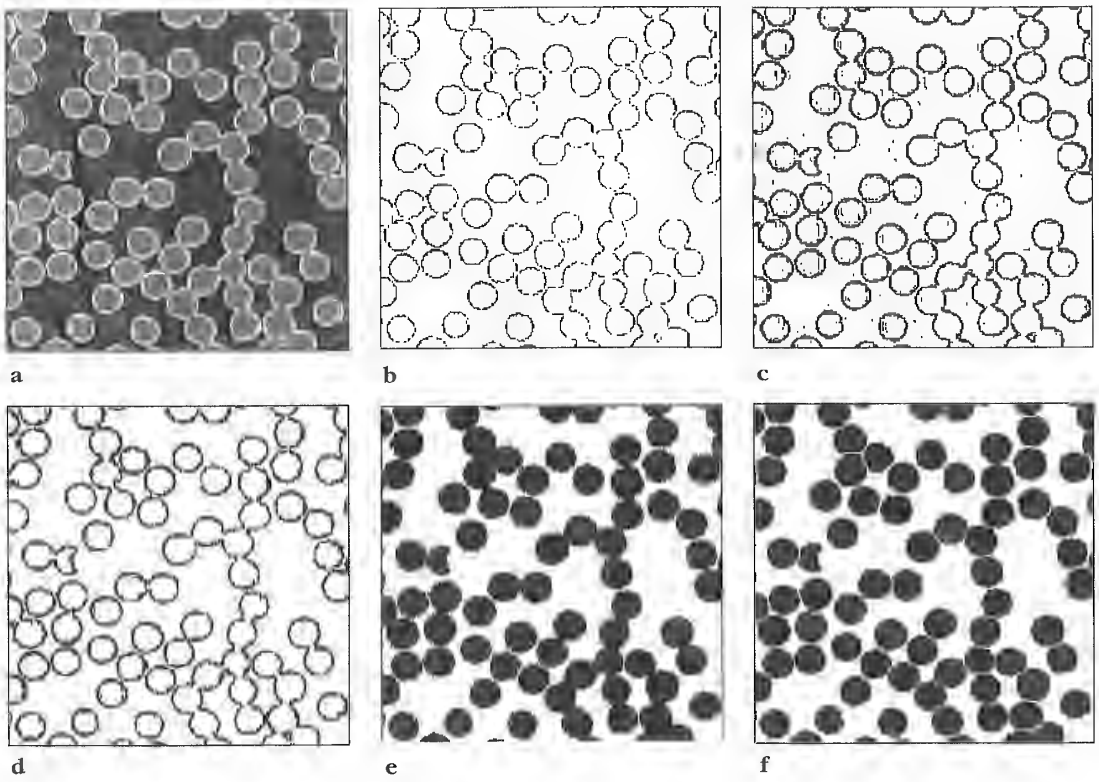


Figure 34. Double thresholding of grain boundaries in alumina:
(a) *original image;*
(b) *first thresholding or dark grain boundary markers;*
(c) *variance operator applied to original;*
(d) *second thresholding of image* **c** *for all boundaries plus other marks;*
(e) *Feature-AND of image* **b** *with 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 tesselation 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.

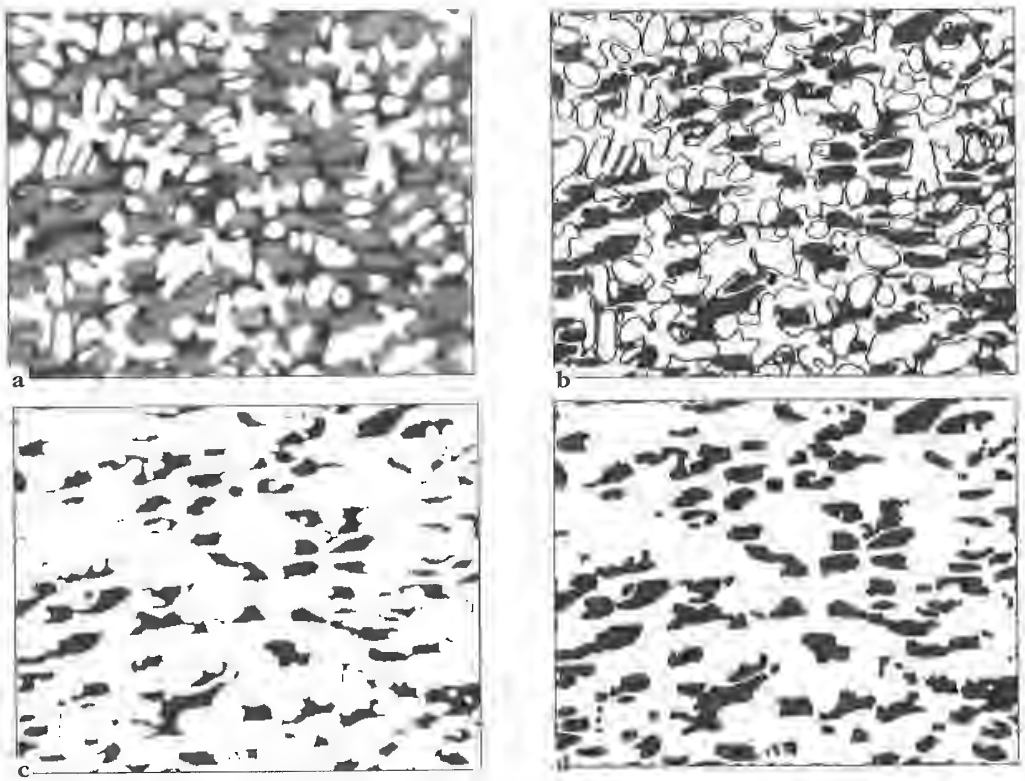


Figure 36: Removal of lines of pixels that straddle a boundary:
(a) *original grey-scale microscope image of a three-phase metal;*
(b) *binary image obtained by thresholding on the intermediate grey phase;*
(c) *erosion of image* ***b*** *using two iterations;*
(d) *dilation of image* ***c*** *using the same two iterations, restoring the feature size but without the lines.*

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.

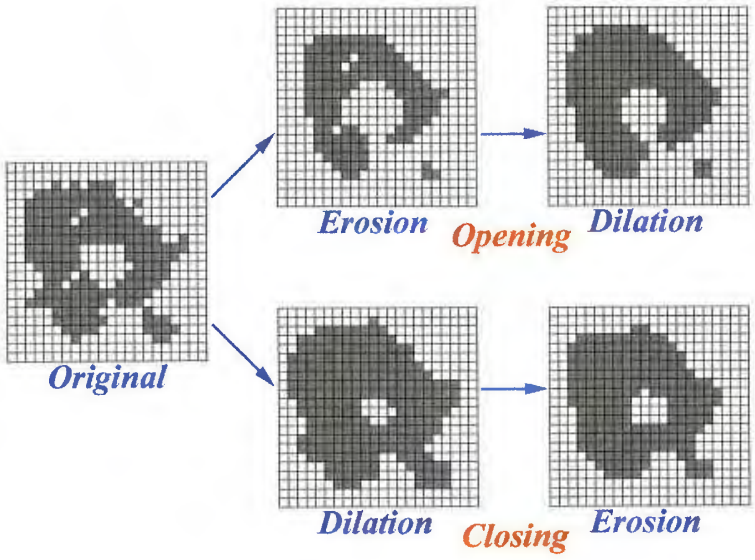


Figure 37. *Combining erosion and dilation to produce an opening or a closing. The result is different depending on the order of application of the two operations.*

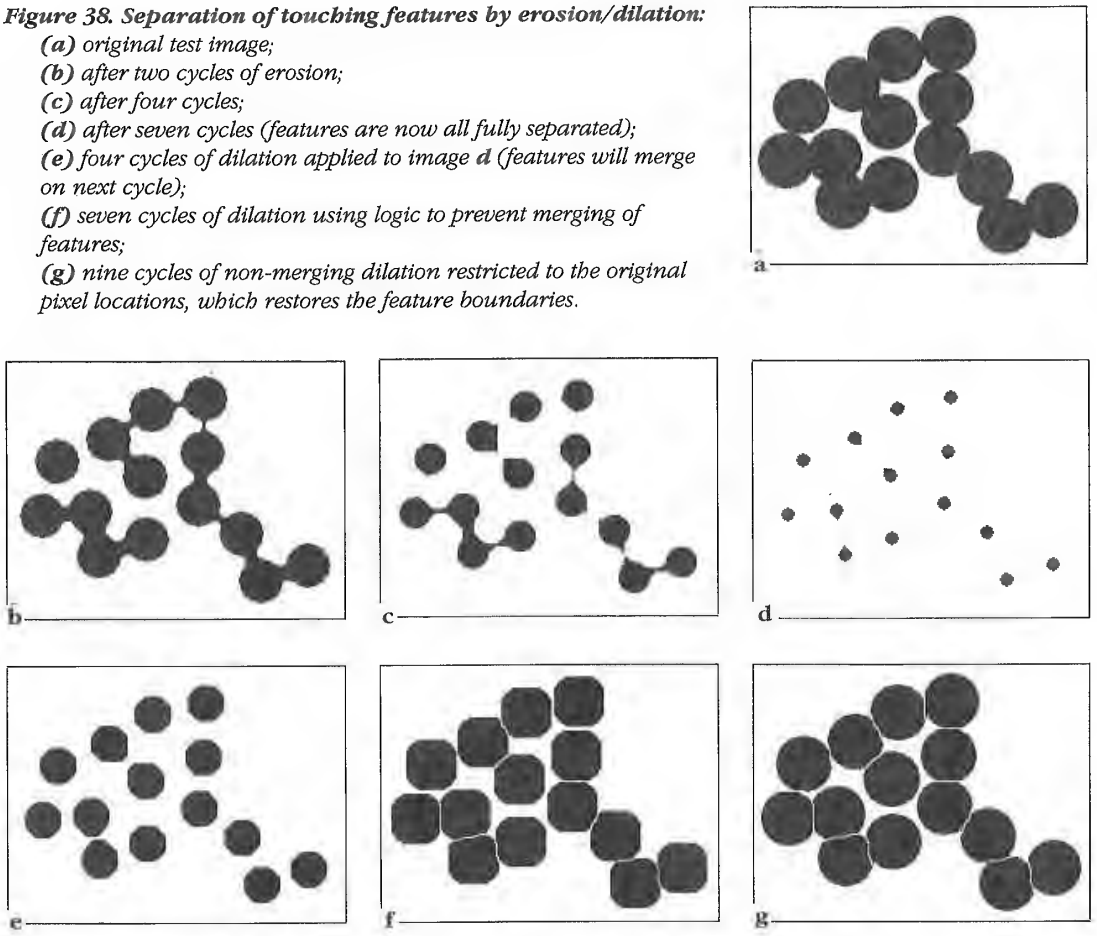
Openings can be used in some cases to separate touching features. In the example shown in **Figure 38**, the features are all similar in size. This fact makes it possible to continue the erosion until all features have separated but none have been completely erased. After the separation is complete, 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.

Figure 38. Separation of touching features by erosion/dilation:
(a) *original test image;*
(b) *after two cycles of erosion;*
(c) *after four cycles;*
(d) *after seven cycles (features are now all fully separated);*
(e) *four cycles of dilation applied to image* ***d*** *(features will merge on next cycle);*
(f) *seven cycles of dilation using logic to prevent merging of features;*
(g) *nine cycles of non-merging dilation restricted to the original pixel locations, which restores the feature boundaries.*



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).

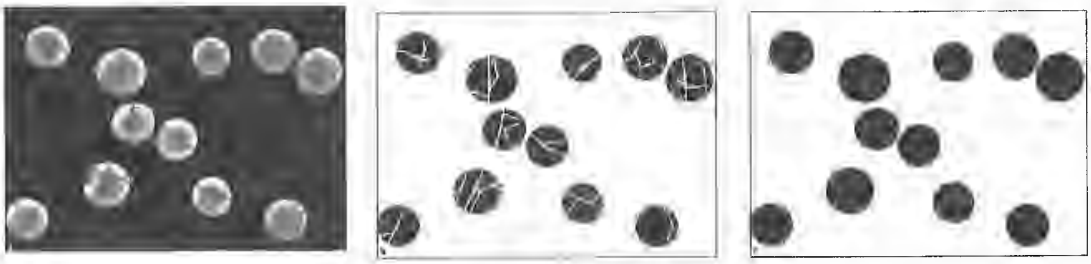


Figure 39. Joining parts of features with a closing:
(a) original image, cross section of cracked glass fibers;
(b) brightness thresholding, showing divisions within the fibers;
(c) after application of a closing.

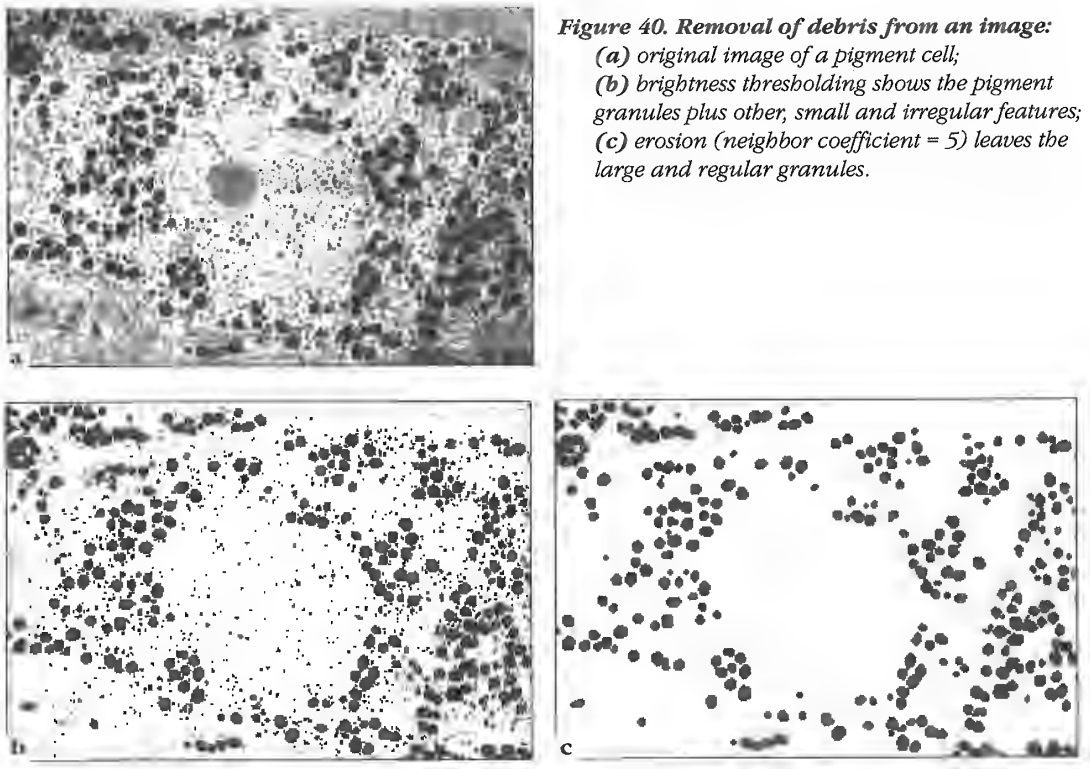


Figure 40. Removal of debris from an image:
(a) original image of a pigment cell;
(b) brightness thresholding shows the pigment granules plus other, small and irregular features;
(c) erosion (neighbor coefficient = 5) leaves the large and regular granules.

Isotropy

It is not possible for a small 3 × 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 41. 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.*

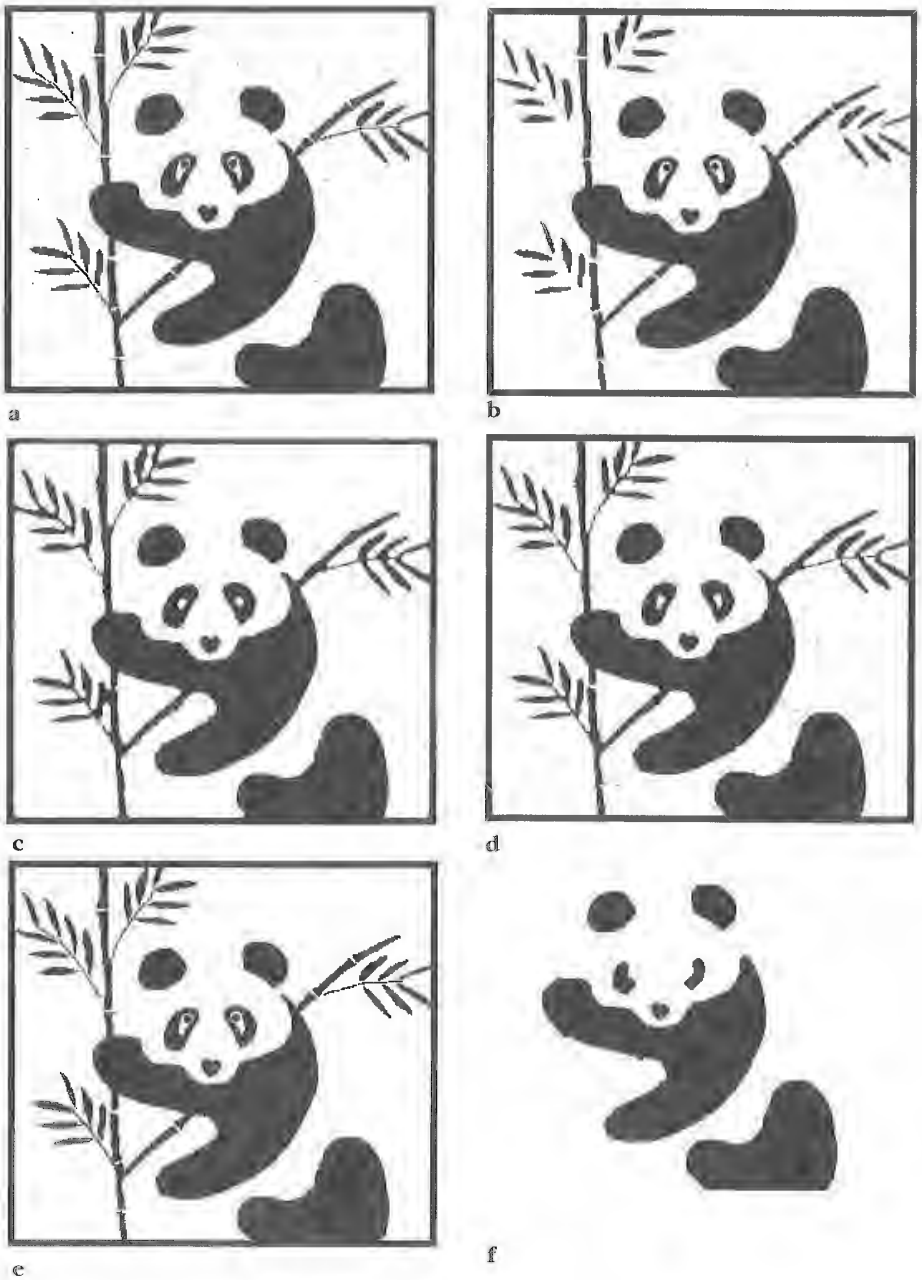




Figure 42. Testing the isotropy of classical (neighbor coefficient = 0) dilation and erosion:
(a) *original circle;*
(b) *after 50 iterations of dilation;*
(c) *after 25 iterations of erosion.*

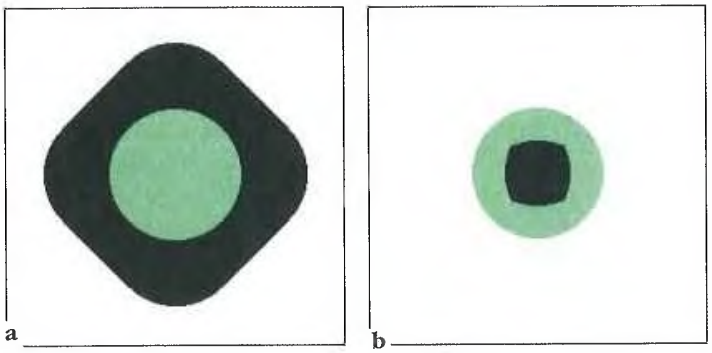


Figure 43. Isotropy tests using a coefficient of 1:
(a) *the circle after 50 iterations of dilation;*
(b) *the circle after 25 iterations of erosion.*

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.

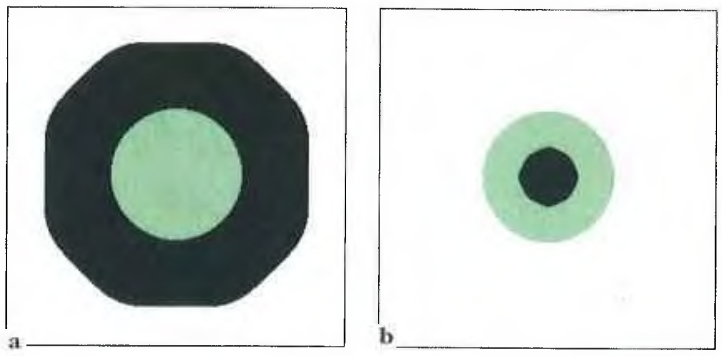


Figure 44. Improved isotropy using by alternating neighbor coefficients of 0 and 1: ***(a)*** *the circle after 50 iterations of dilation;* ***(b)*** *after 25 iterations of erosion.*

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.

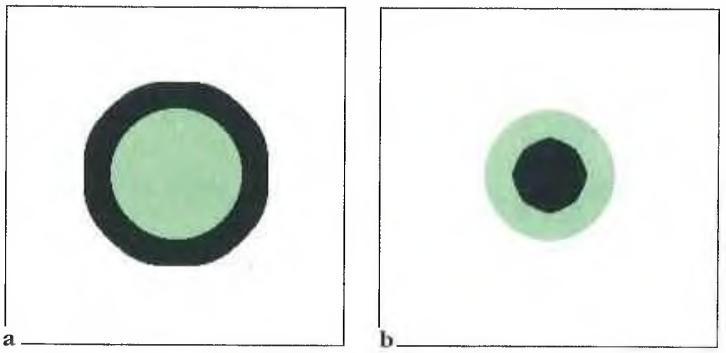


Figure 45. *Classical erosion and dilation using ten iterations and a larger neighborhood size (a 5-pixel-wide approximation to a circle).*

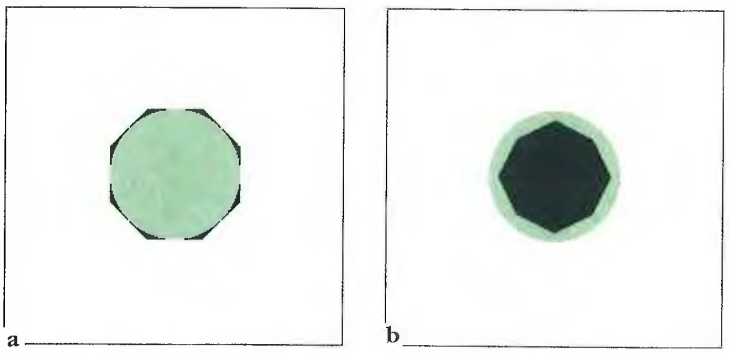


Figure 46. Octagonal shape and slow rate of addition or removal using a coefficient of 3: ***(a)*** *original circle after 50 iterations of dilation (no further changes occur);* ***(b)*** *circle after 25 iterations of erosion.*

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."

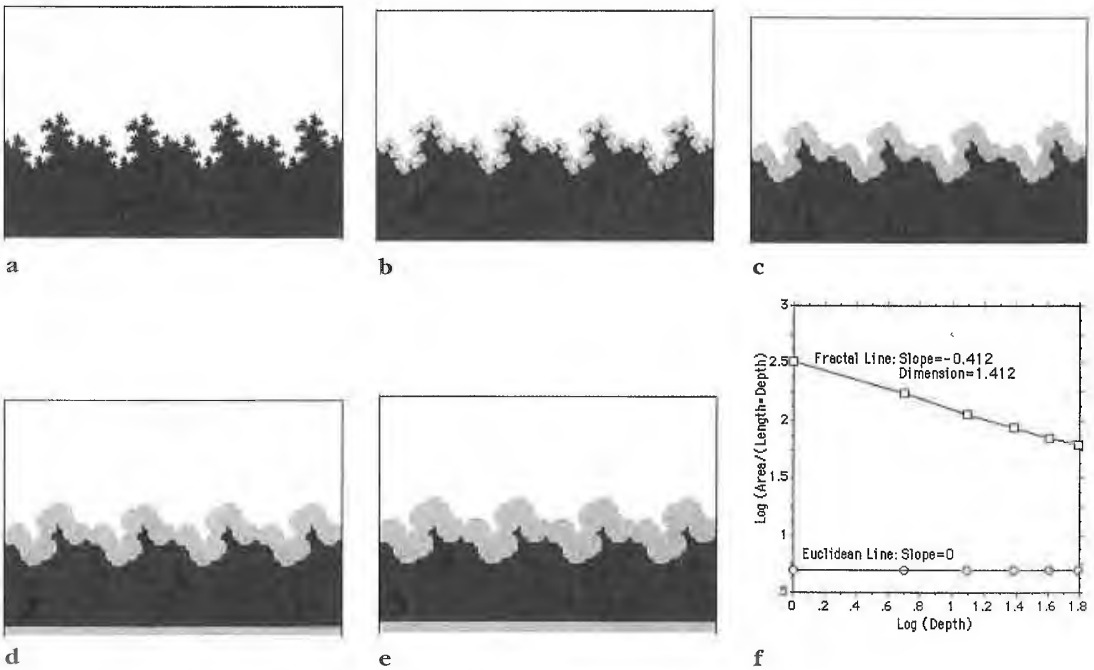
To measure the fractal dimension, the operation is repeated with different depths of erosion and dilation (Flook, 1978), and the effective width (total number of pixels divided by length and

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).*



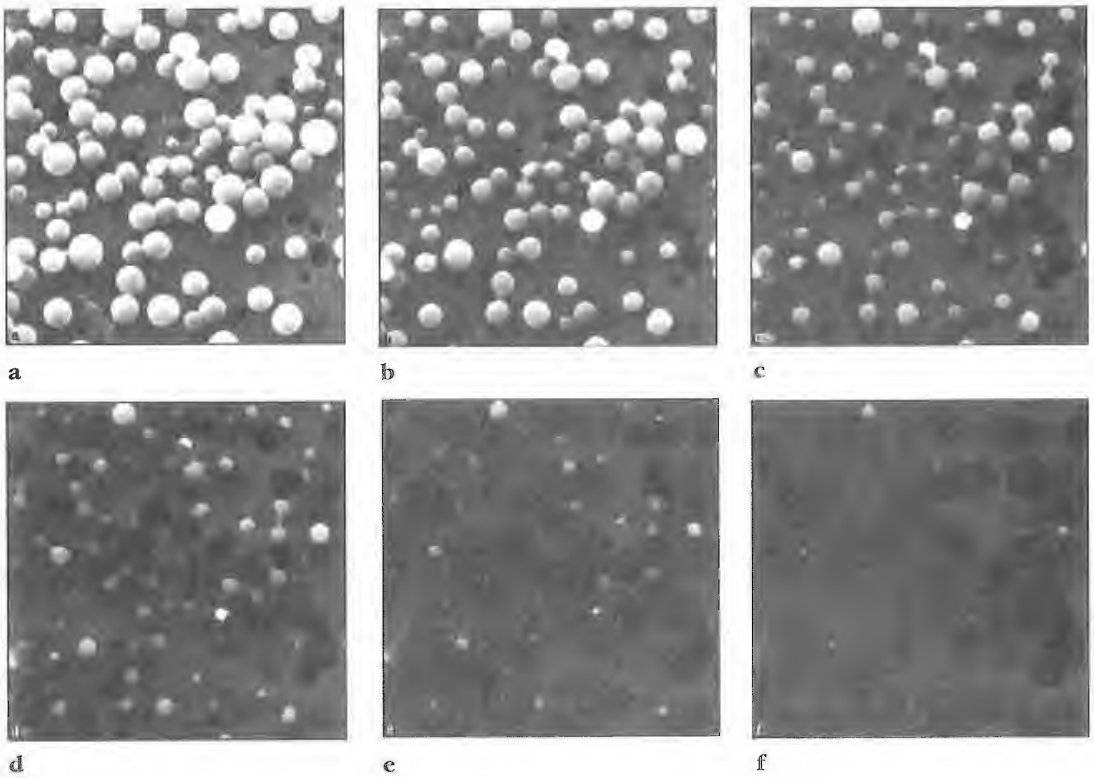
Extension to grey-scale images

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

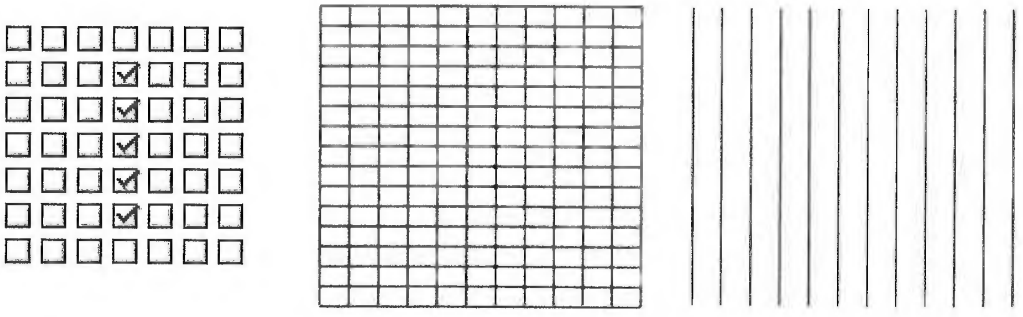
The important parameters for erosion and dilation are the neighborhood size and shape, the comparison test that is used and the number of times the operation is repeated. The use of a simple test coefficient based on the number of neighbors, irrespective of their location in the neighborhood, provides 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



a b c

Figure 49. Example of specifying the neighborhood pixels for morphological operations:
(a) *a vertical neighborhood for erosion;*
(b) *original pattern;*
(c) *eroded result.*

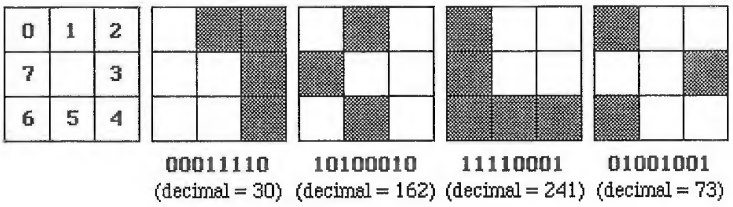


Figure 50. *Constructing an address into a fate table by assigning each neighbor position to a bit value.*

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.

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.

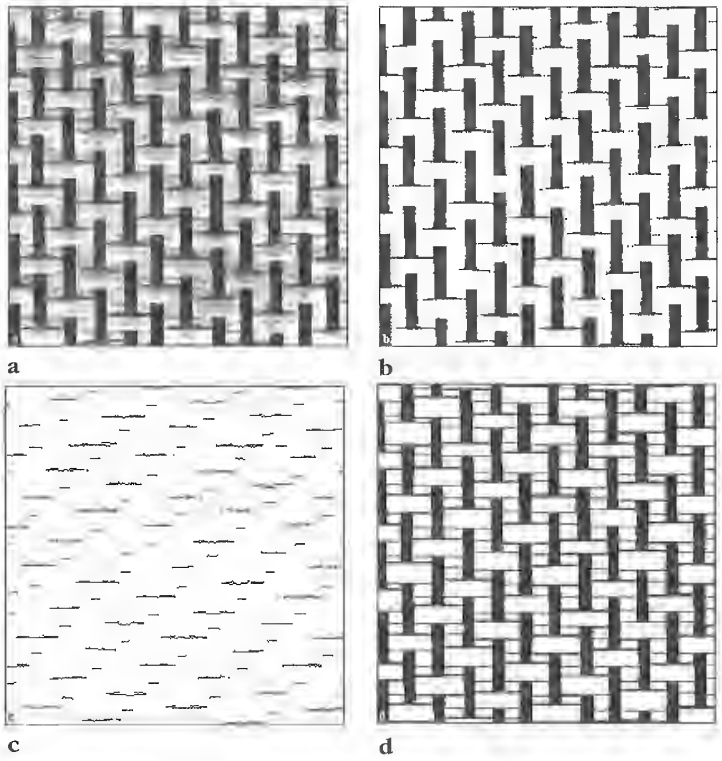


Figure 51. Using directional erosion and dilation to segment an image: *(a) original grey-scale image of a woven textile; (b) brightness thresholding of image* ***a****; (c) end pixels isolated by performing a vertical erosion and ExORing with the original; (d) completed operation by repeated horizontal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

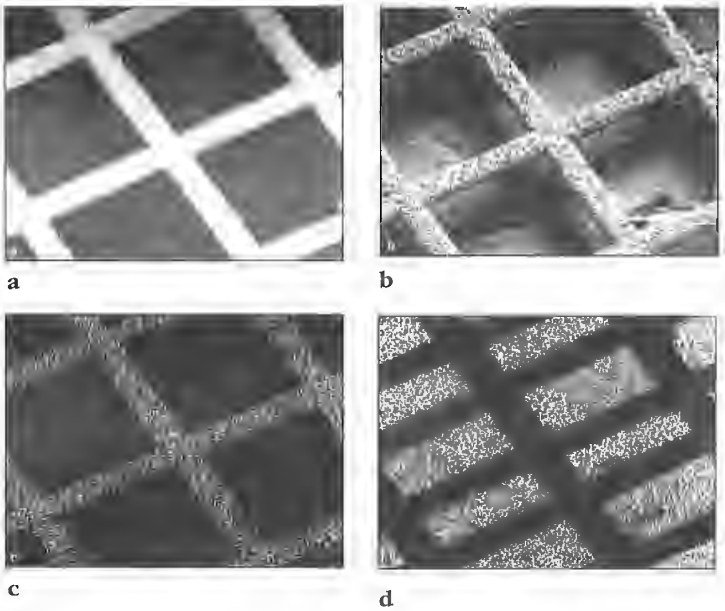


Figure 52. X-ray "dot" maps from the SEM: *(a) backscattered electron image of a gold grid above an aluminum stub; (b) secondary electron image; (c) gold x-ray dot image; (d) aluminum x-ray image (notice the shadows of grid).*

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.

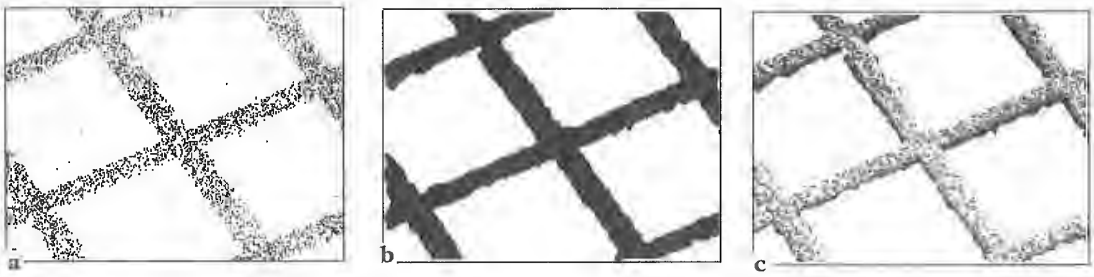


Figure 53. Delineating the gold grid:
(a) *thresholded x-ray map;*
(b) *image* ***a*** *after two repetitions of closing;*
(c) *the backscattered electron image masked to show the boundaries from image* ***b*** *(notice the approximate location of edges).*

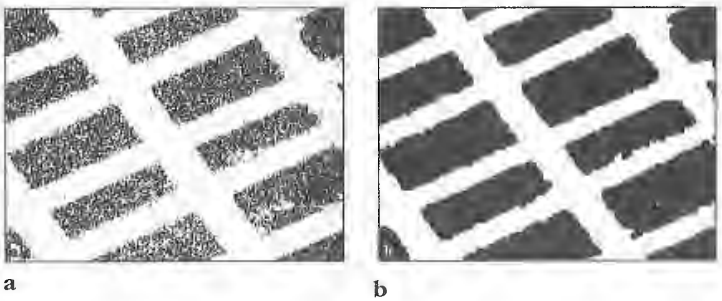


Figure 54. Delineating the aluminum map:
(a) *thresholding (notice the isolated continuum x-rays recorded within the grids);* ***(b)*** *after erosion with a neighborhood coefficient of 7 to remove the isolated pixels and dilation (two cycles) to fill the regions.*

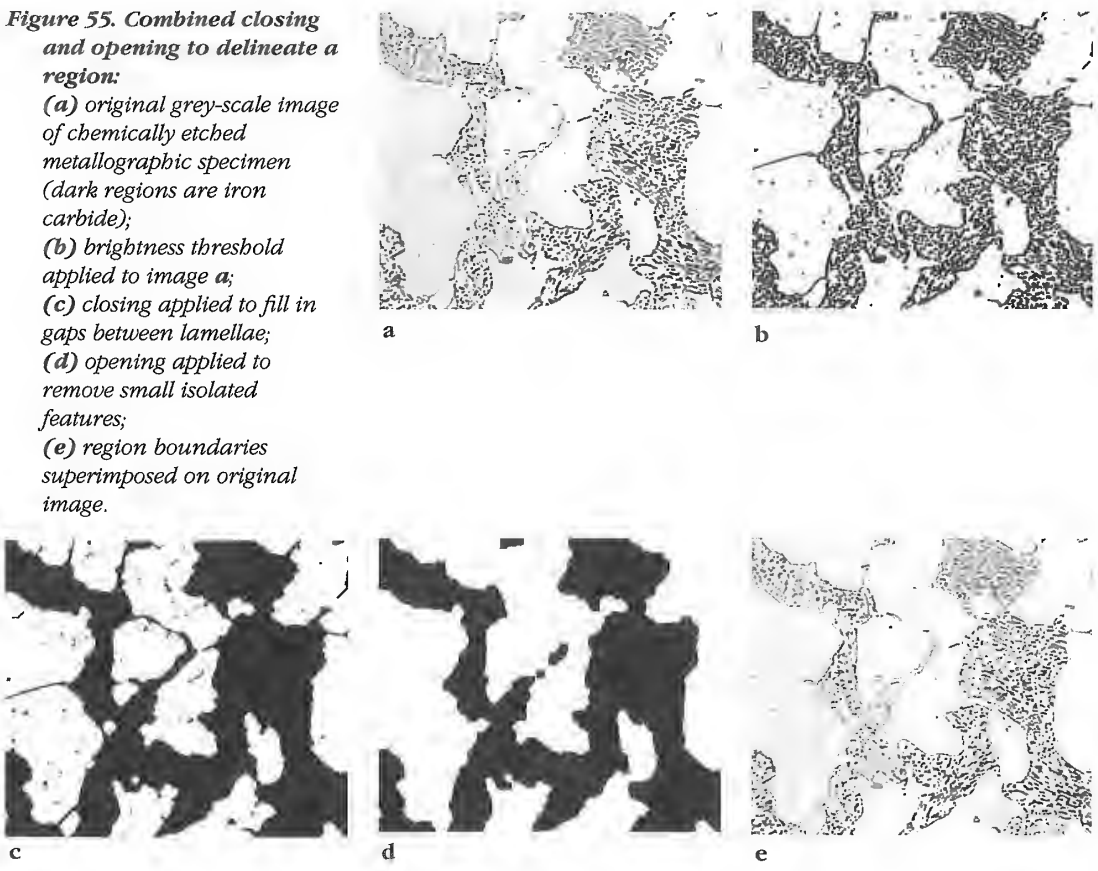


Figure 55. Combined closing and opening to delineate a region: *(a) original grey-scale image of chemically etched metallographic specimen (dark regions are iron carbide); (b) brightness threshold applied to image a; (c) closing applied to fill in gaps between lamellae; (d) opening applied to remove small isolated features; (e) region boundaries superimposed on original image.*

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.

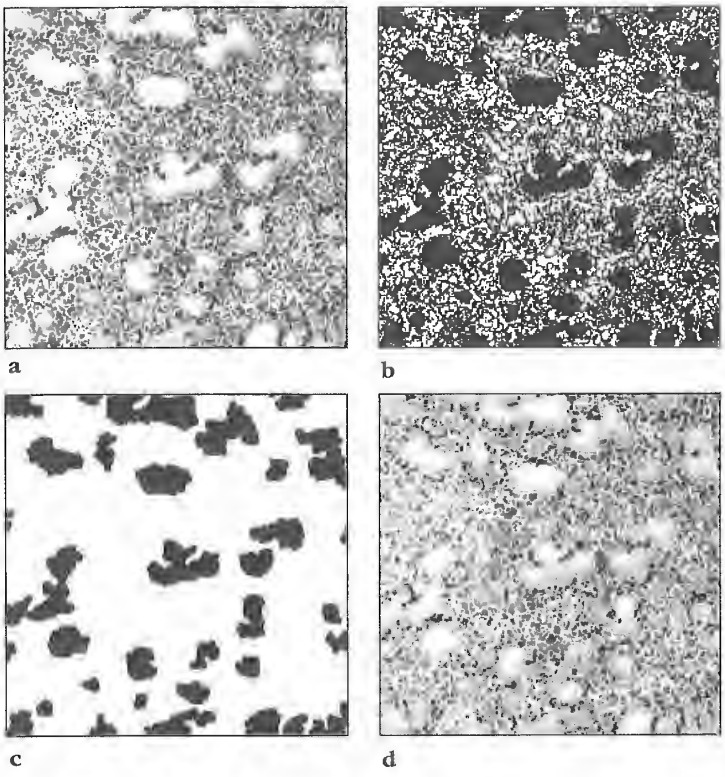


Figure 56. Segmenting the curds and whey image by erosion/dilation: *(a) original image; (b) thresholded; (c) morphological opening; (d) outlines superimposed on original image.*

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

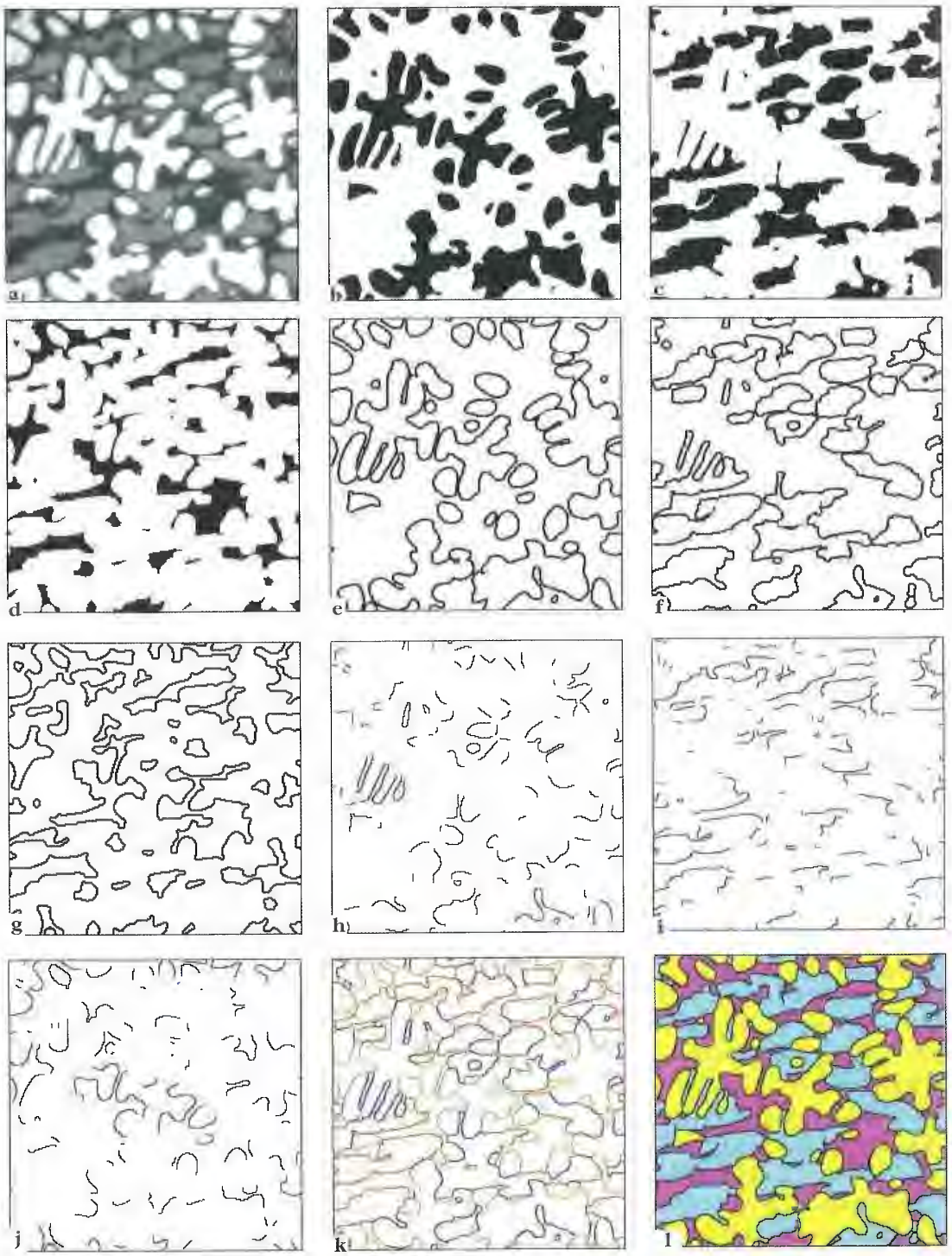


Figure 57. Use of Boolean logic to measure neighbor relationships: (a) *an original light microscope image of a three-phase metal;* ***(b–d)*** *thresholded white, grey, and black phases;* ***(e–g)*** *surrounding outlines of each phase produced by dilation and ExOR with original;* ***(h–j)*** *AND of 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:

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3.0	3.2	3.6	4.2
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2.0	2.2	2.8	3.6
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1.0	1.4	2.2	3.2
$\sqrt{9}$	$\sqrt{4}$	$\sqrt{1}$	0	1.0	2.0	3.0
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1.0	1.4	2.2	3.2
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2.0	2.2	2.8	3.6
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3.0	3.2	3.6	4.2

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.*

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 $\sqrt{2}$ = 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 3 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, 7/5 = 1.400 and 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 × 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 close approximation to the Pythagorean distance (92/41 = 2.243; $\sqrt{5}$ = 2.236) and produces more isotropic results.

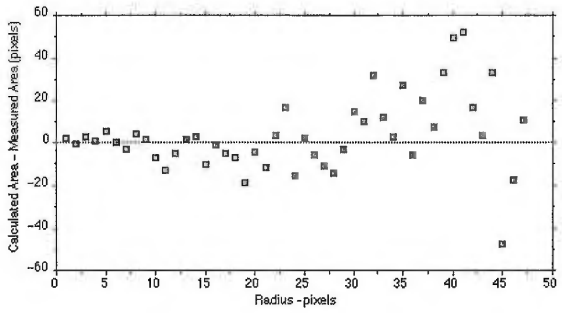


Figure 59. *Difference between theoretical area value (πr^2) and the actual area covered by the EDM as a function of brightness (distance from boundary) shows increasing but still small errors for very large distances.*

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

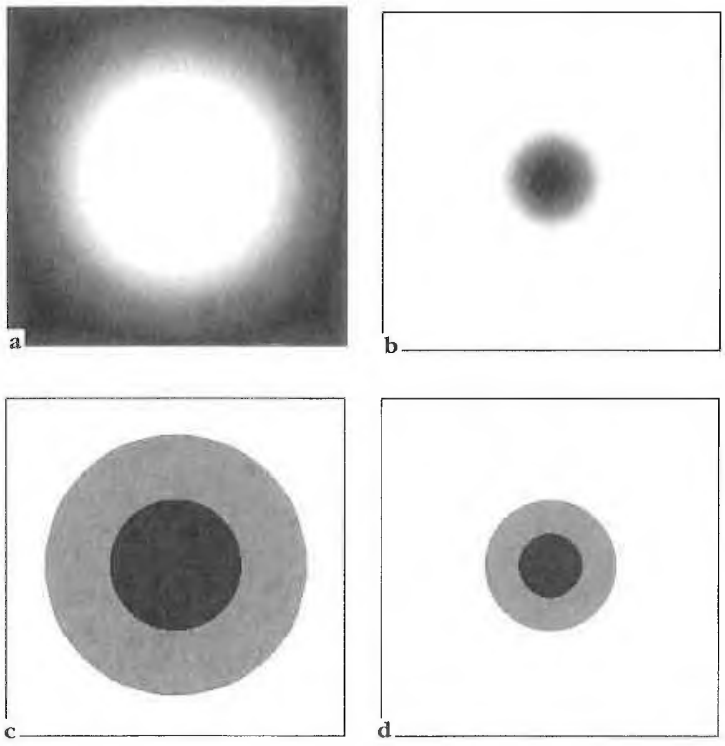


Figure 60. Isotropic erosion and dilation achieved by using the Euclidean distance map for dilation and erosion (compare to Figures 42–46): *(a) the EDM of the background around the circle; (b) the EDM of the circle; (c) dilation achieved by thresholding the background EDM at a value of 50; (d) erosion achieved by thresholding the circle EDM at a value of 25.*

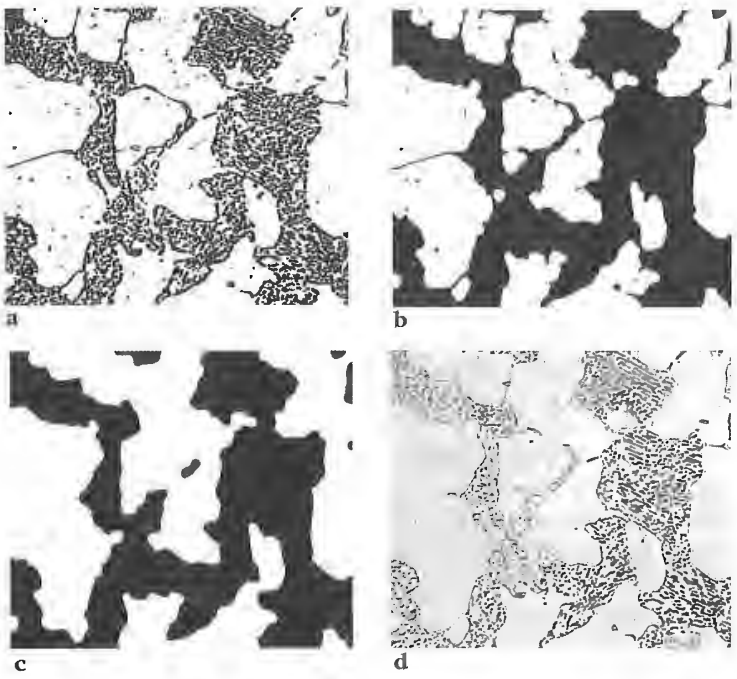


Figure 61. Closing and opening using the EDM:
(a) *thresholded original image (same as* ***Figure 55****);*
(b) *closing, distance of 5.5 pixels;*
(c) *opening; distance of 5.5 pixels;*
(d) *outlines superimposed on original image.*

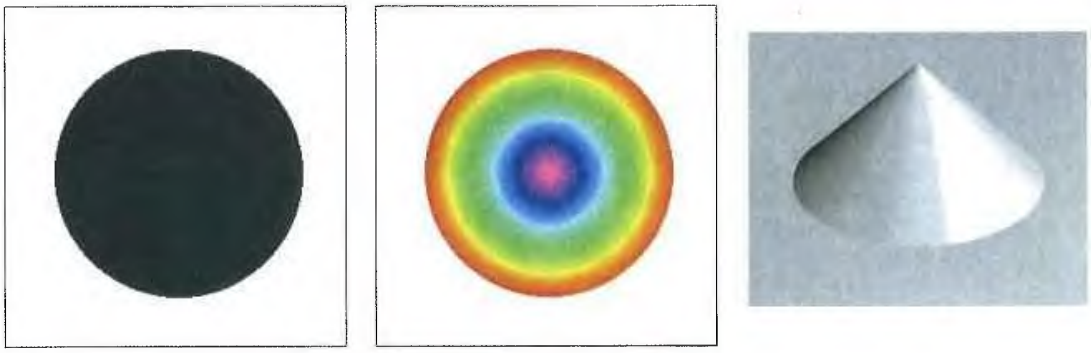
a thick section in transmission, where actual feature overlap may occur, or when particles resting on a surface tend to agglomerate and touch each other. One method for separating touching, but mostly convex, features in an image is known as watershed segmentation (Beucher and Lantejoul, 1979; Lantejoul 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.

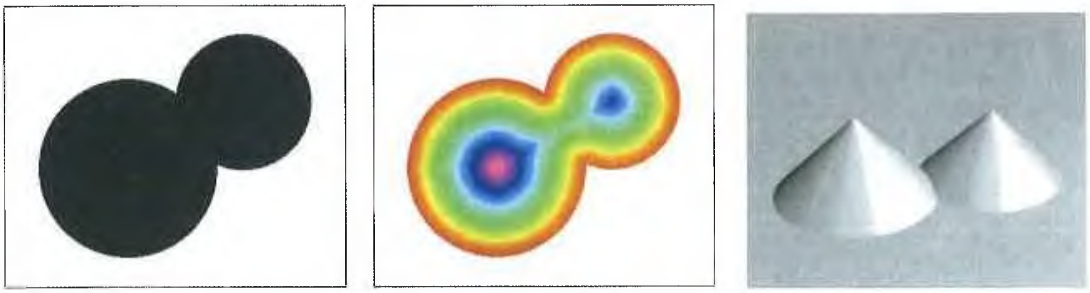
The method just described has two practical drawbacks: the iterative process is slow, requiring each pixel in the image to be processed many times, and the amount of storage required for all of the intermediate images is quite large. The same result can be obtained more efficiently using an EDM. Indeed, 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



a b c

Figure 62. Interpreting the Euclidean distance map as the height of pixels:
(a) *binary image of a circular feature;*
(b) *Euclidean distance map with pixels color coded to show distance from boundary;*
(c) *rendered display showing pixel heights.*



a b c

Figure 63: EDM for touching features:
(a) *binary image of two touching circular features;*
(b) *Euclidean distance map with pixels color coded to show distance from boundary;*
(c) *rendered display showing pixel heights. Note the boundary between the two cones.*

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 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.

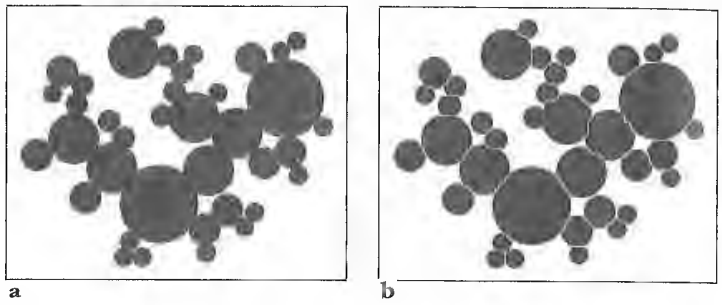


Figure 64. Watershed segmentation on an image of touching circles of different sizes.

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 a image of real particles, as shown in **Figure 65**, the method works subject to the assumption that the features are sufficiently convex 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 not 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.

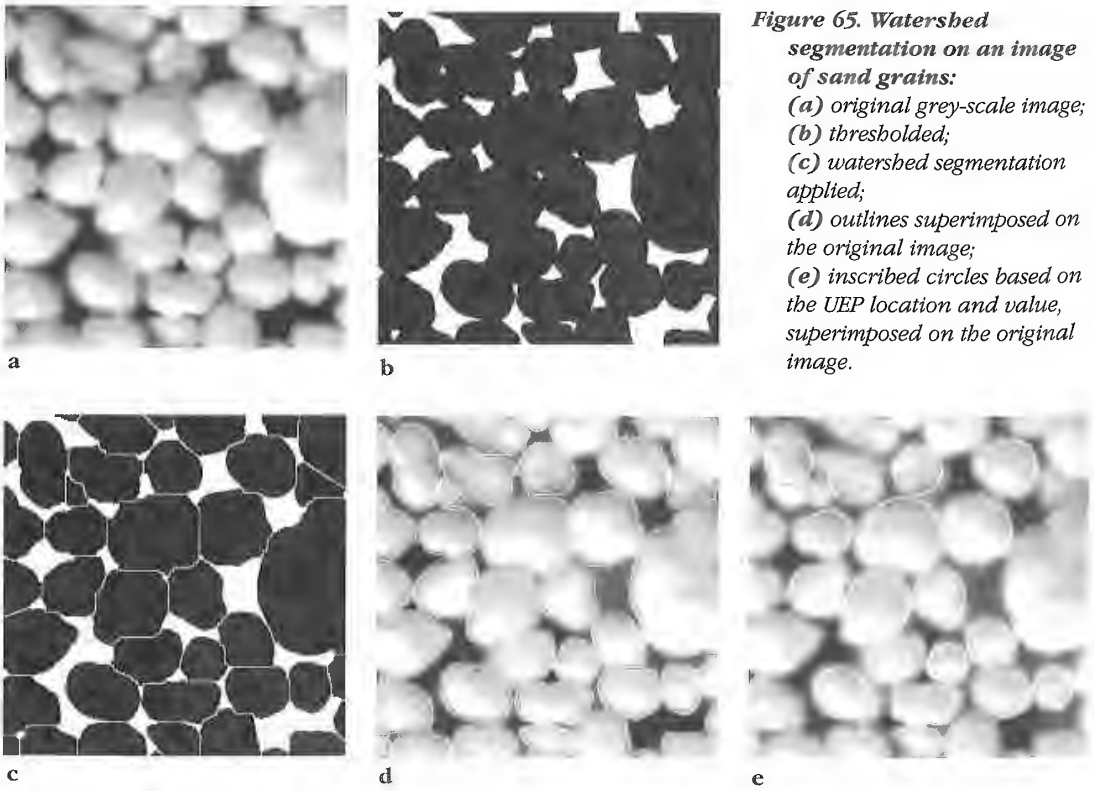


Figure 65. Watershed segmentation on an image of sand grains: *(a) original grey-scale image; (b) thresholded; (c) watershed segmentation applied; (d) outlines superimposed on the original image; (e) inscribed circles based on the UEP location and value, superimposed on the original image.*

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.

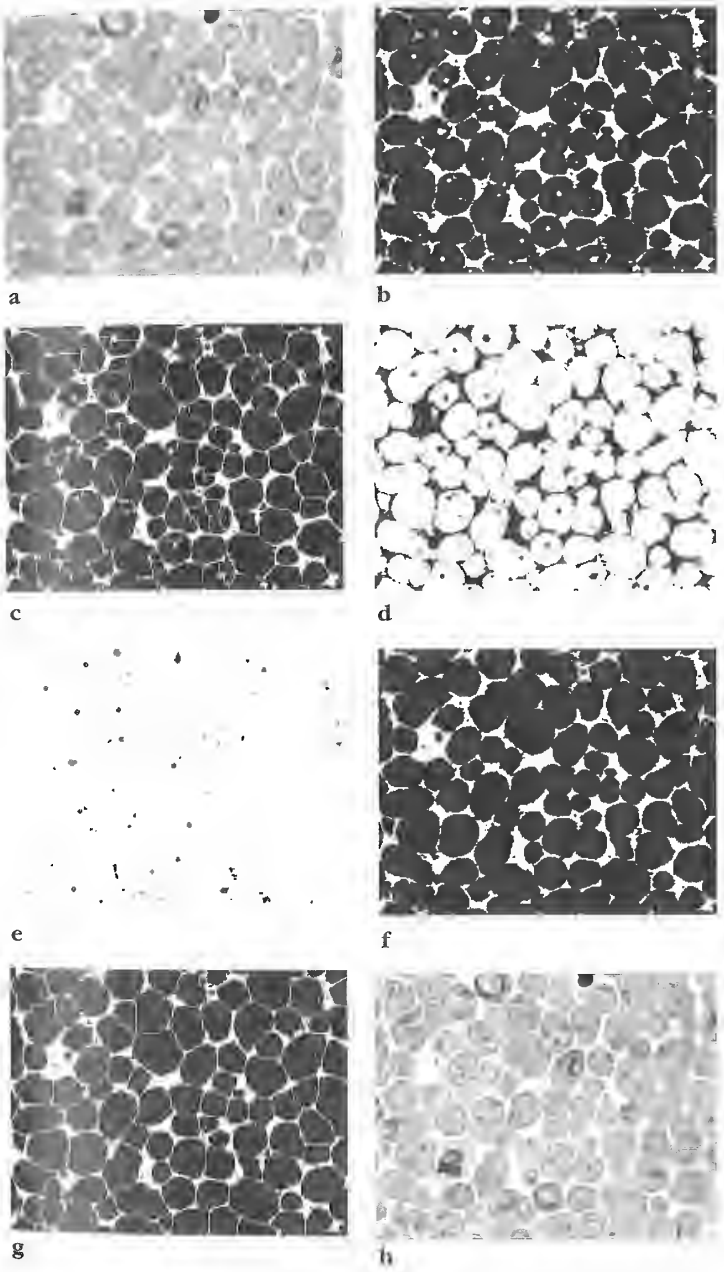


Figure 66. Separation of touching cells:
(a) *original grey-scale image;*
(b) *thresholded;*
(c) *erroneous watershed segmentation produced by holes within cells;*
(d) *inverting image* ***b*** *to show holes within and between cells;*
(e) *holes within cells selected by their rounder shapes;*
(f) *combining image* ***b*** *with* ***e*** *using a Boolean OR;*
(g) *watershed segmentation of image* ***f****;*
(h) *outlines superimposed on original image.*

Ultimate eroded points

The ultimate eroded points (UEPs) 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.

$$\{U: \quad \forall \text{ Uj neighbors of Ui, } |Uj| \le |Ui| \quad \text{AND} \quad \forall \text{ Uj neighbors of Ui such that } |Uj| = |Ui|, \; Ui \in U\} \tag{5}$$

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.

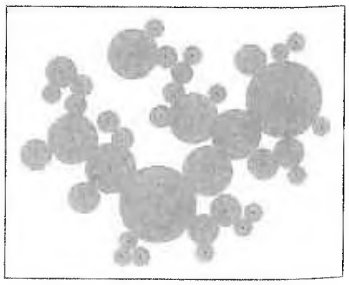
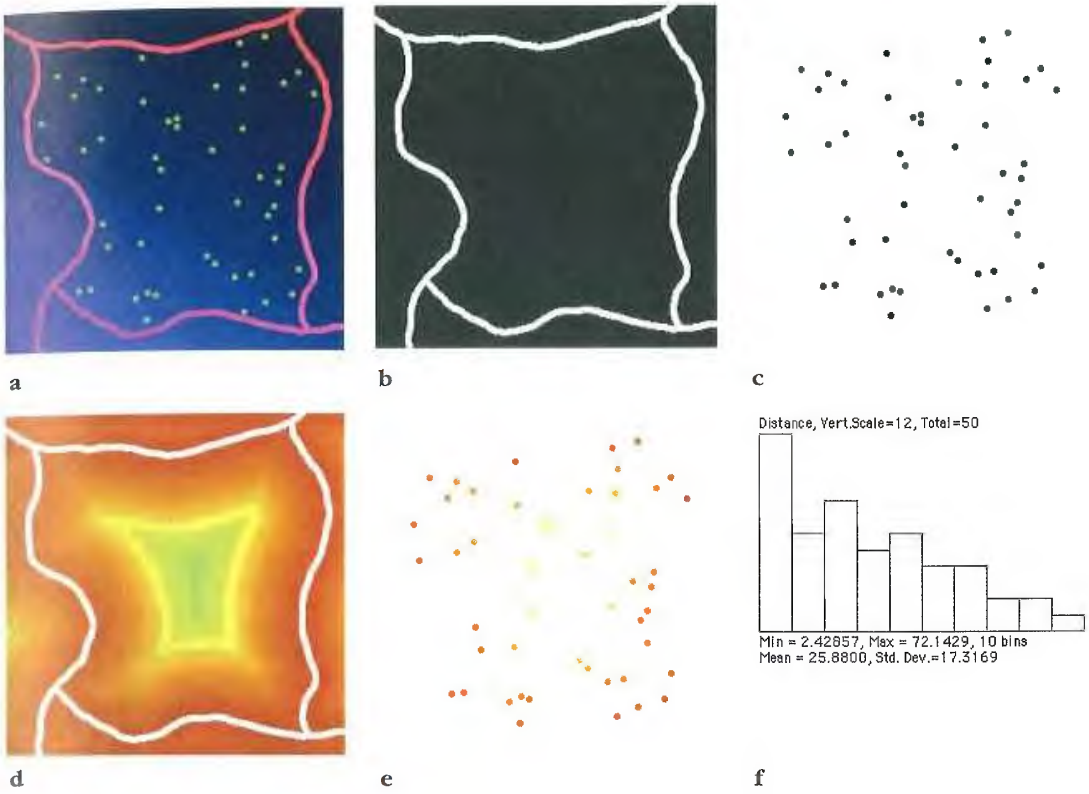


Figure 67. *The ultimate points for the touching features from* ***Figure 64.***

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 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

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:

$$\#\ \text{Loops} = \#\ \text{Branches} - \#\ \text{Ends} - \#\ \text{Nodes} + 1 \tag{4}$$

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

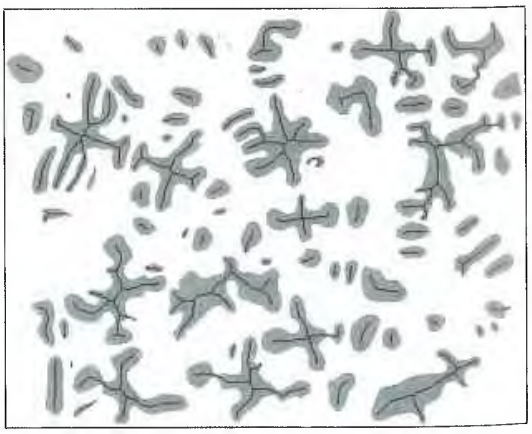


Figure 69. *A binary image containing multiple features, with their skeletons superimposed.*

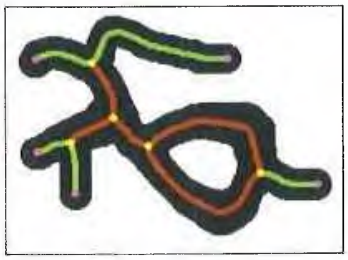


Figure 70. *The skeleton of a feature with five end points, five nodes, five external branches, five internal branches, and one loop (skeleton has been dilated for visibility).*

Figure 71. Skeletons for a ring and a circle, as discussed in the text.





Figure 72. Labeling star-shaped features according to the number of end points in the skeleton.

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.



Figure 73. Simplification of a fingerprint image (a) *by thresholding and skeletonization* ***(b)***.

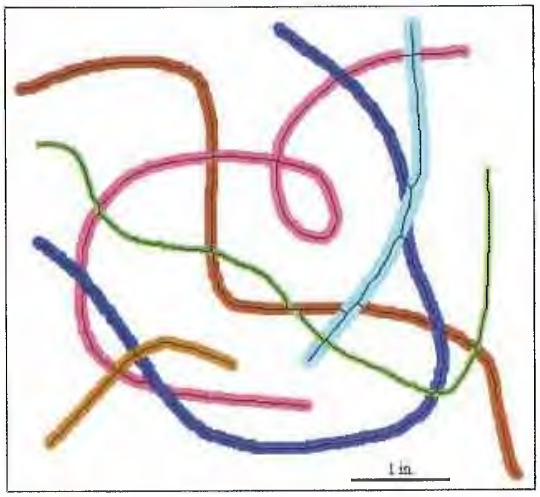
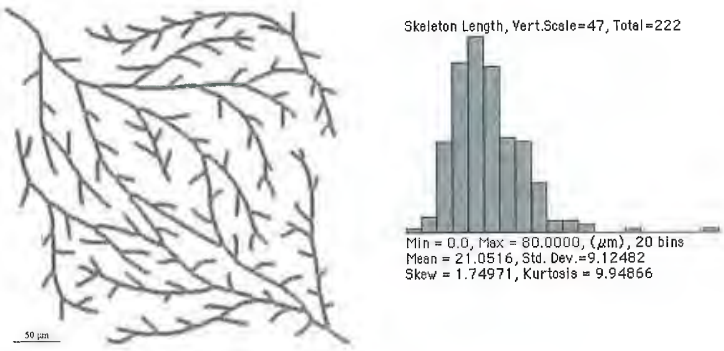


Figure 74. *Example of crossing fibers, as discussed in the text, with superimposed skeleton lines.*

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.



a b

Figure 75. *Separating the branches of the skeleton of a network for measurement of their lengths.*

Figure 76. *Detail of the skeleton from* ***Figure 75*** *showing nodes. Removing the red node points does not disconnect all of the branches; the green "near-node" points must also be deleted to assure that no eight-connections remain between different branches. In the figure, end points are shown as blue.*

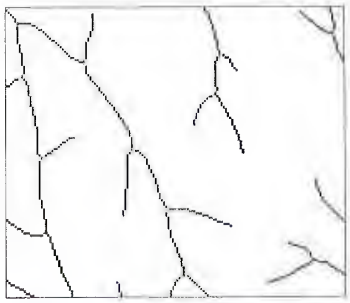
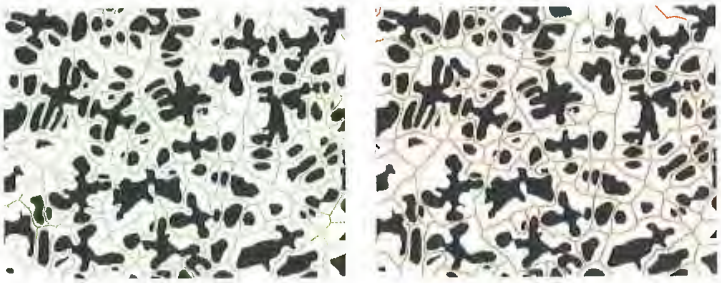


Figure 77. The skiz of the same image shown in Figure 69: ***(a)*** *complete skeleton of the background;* ***(b)*** *pruned skeleton.*



a b

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.

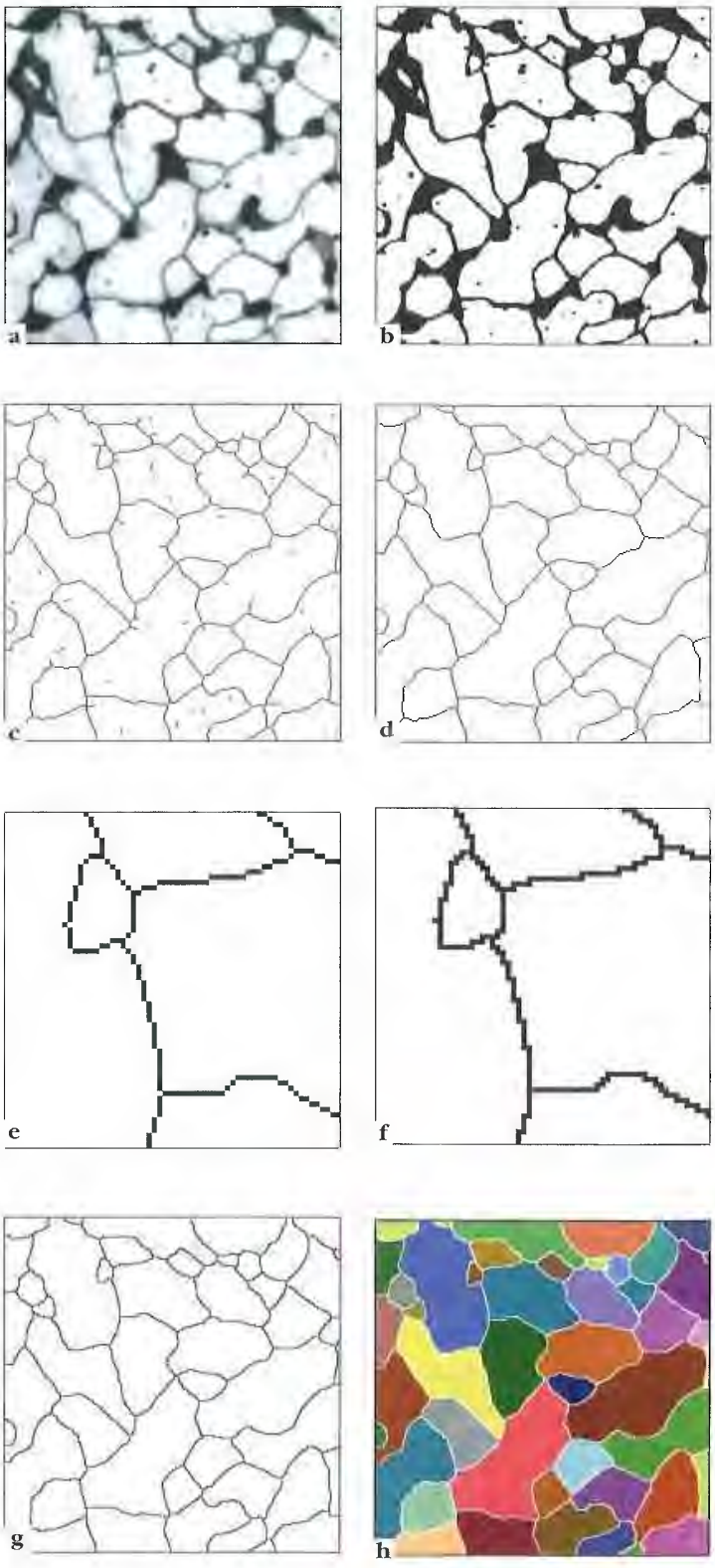


Figure 78. Skeletonization of grain boundaries: *(**a**) metallographic image of etched 1040 steel; (**b**) thresholded image showing boundaries and dark patches of iron carbide (and pearlite); (**c**) skeletonized from image **b**; (**d**) pruned from image **c**; (**e**) enlarged to show eight-connected line; (**f**) converted to four-connected line; (**g**) grains separated by thickened lines; (**h**) identification of individual grains.*

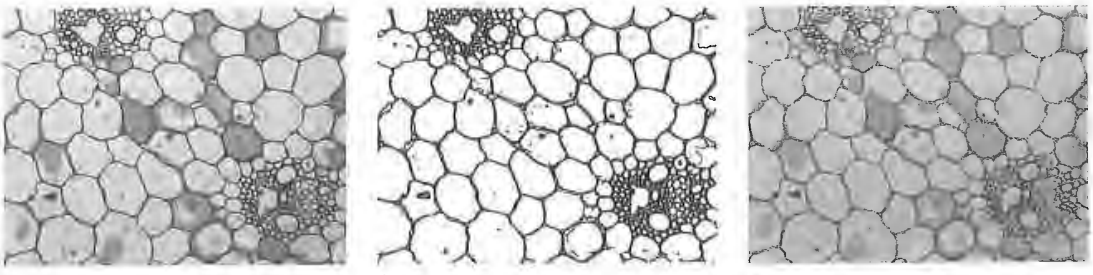
Figure 78 shows an example. The original polished and etched metal sample has dark and wide grain boundaries, as well as dark patches corresponding to carbides and pearlite. Thresholding the image produces broad lines, which can be skeletonized to reduce them to single-pixel width. Since this is properly 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 ORed 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 dirt or particulates within the grains, can cause difficulties.



a b c

Figure 79. Light microscope image of cells in plant tissue:
(a) *original;*
(b) *thresholded;*
(c) *skeleton superimposed on original (image courtesy Data Translations, Inc.)*

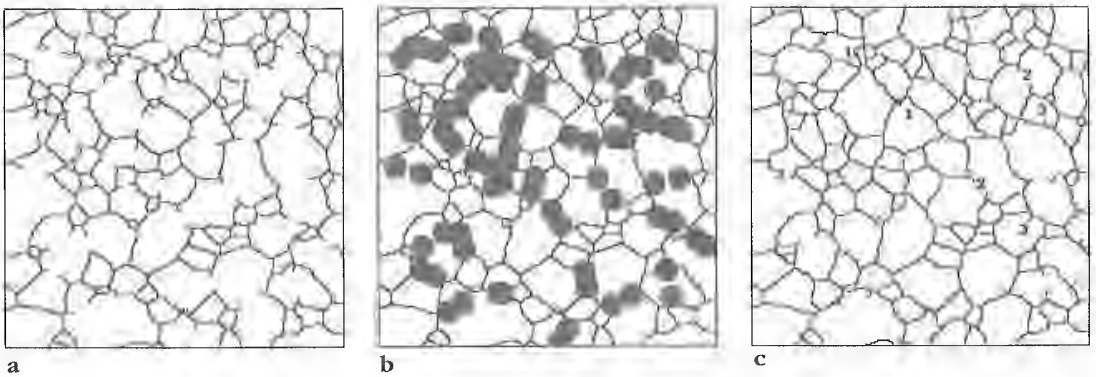


Figure 80. Dilation method for completing grain boundary tesselation:
(a) incomplete network;
(b) dilation of end point by arbitrary radius, shown as circles overlaid on the original;
(c) re-skeletonization of network, showing typical errors such as removal of small grains (1), large gaps still not joined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

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.

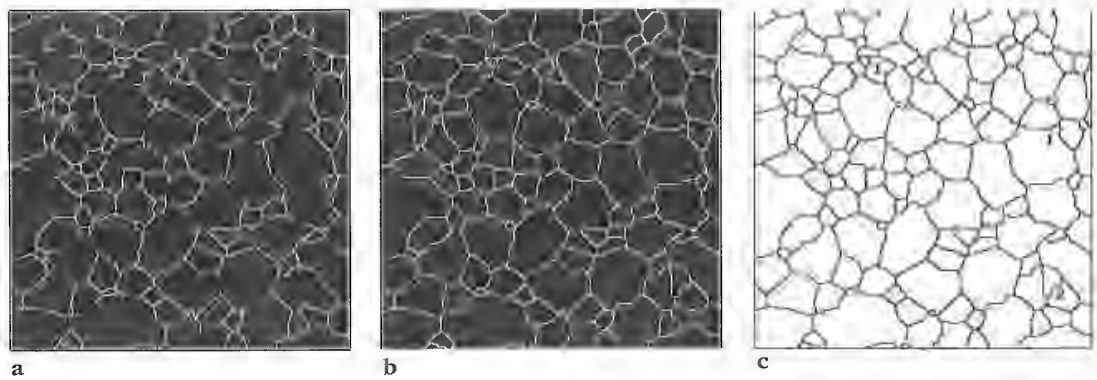


Figure 81. Watershed segmentation applied to the same image as Figure 80:
(a) the image is inverted to deal with the grains rather than the boundaries;
(b) watershed lines are drawn in, connecting most of the broken boundaries;
(c) in the re-inverted result typical errors appear such as large gaps not joined (1) and false segmentation of irregular shaped grains (2).

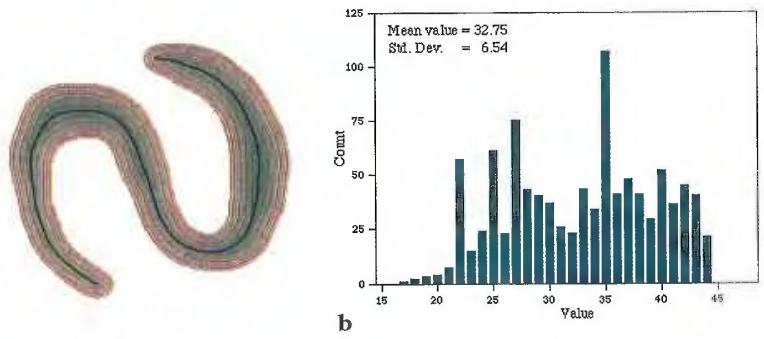


Figure 82. *An irregular feature shown with its skeleton superimposed on the EDM (using pseudo-color), and the histogram of the EDM values selected by the skeleton.*

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

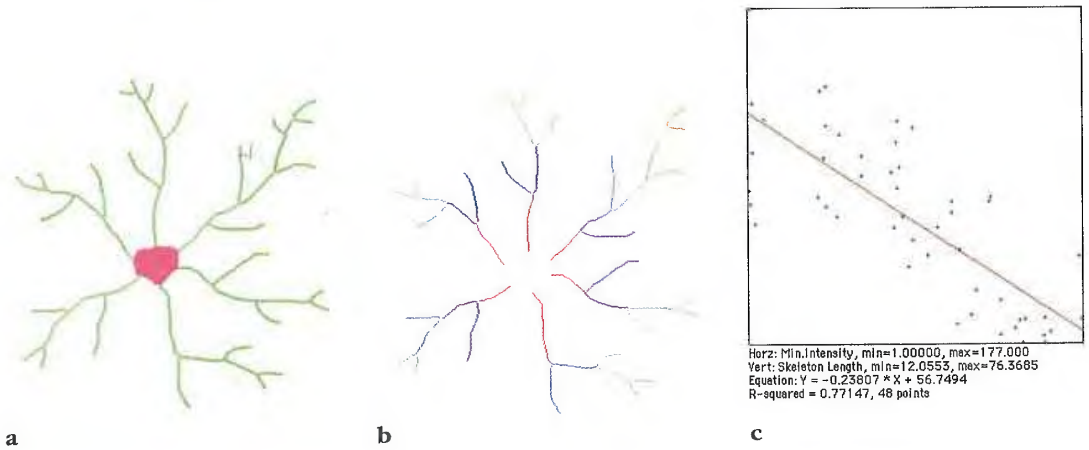


Figure 83. Relating distance to length:
(a) *diagram of a neural cell;*
(b) *skeleton segments of the neurites, color coded according to distance from the cell body (values obtained from the EDM as described in the text);*
(c) *plot of distance vs. length.*

distance of points from the cell body. The skeleton of the neurites can be separated into its component 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 distance 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.

8

Global Image Measurements

The distinction between image processing, which has occupied most of the preceding chapters, and image analysis lies in the extraction of information from the image. As mentioned several times previously, image processing, like word processing (or food processing) is the science of rearrangement. Pixel values may be altered according to neighboring pixel brightnesses, or shifted to another place in the array by image warping, but the sheer quantity of pixels is unchanged. So in word processing it is possible to cut and paste paragraphs, perform spell-checking, or alter type styles without reducing the volume of text. And food processing is also an effort at rearrangement of ingredients to produce a more palatable mixture, not to boil it down to the essence of the ingredients. Image analysis, by contrast, attempts to find those descriptive parameters, usually numeric, that succinctly represent the information of importance in the image.

The processing steps considered in earlier chapters are in many cases quite essential to carrying out this task. Defining the features to be measured frequently requires image processing to correct acquisition defects, enhance the visibility of particular structures, threshold them from the background, and perform further steps to separate touching objects or select those to be measured. And we have seen in several of the earlier chapters opportunities to use these processing methods themselves to obtain numeric information.

Global measurements and stereology

There are two major classes of image measurements: ones performed on the entire image field (sometimes called the "scene"), and ones performed on each of the separate features present. The latter feature-specific measurements are covered in the next chapter. The first group of measurements are most typically involved in the characterization of three-dimensional structures viewed as section planes in the microscope, although the methods are completely general and some of the relationships have been discovered and are routinely applied by workers in earth sciences and astronomy. The science of stereology relates the measurements that can be performed on two-dimensional images to the three-dimensional structures that are represented and sampled by those images. It is primarily a geometrical science, whose most widely used rules and calculations have a deceptive simplicity. Reviews of modern stereological methods can be found in (Russ and Dehoff, 2001; Howard and Reed, 1998; Kurzydlowski and Ralph, 1995), while classical methods are described in (Dehoff and Rhines, 1968; Underwood, 1970; Weibel, 1979; Russ, 1986).

The simplest and perhaps most frequently used stereological procedure is the measurement of the volume fraction that some structure occupies in a solid. This could be the volume of nuclei in cells, a particular phase in a metal, porosity in ceramic, mineral in an ore, etc. Stereologists often use the word "phase" to refer to the structure in which they are interested, even if it is not a phase in the chemical or thermodynamic sense. If a structure or phase can be identified in an image, and that image is representative of the whole, then the area fraction which the phase occupies in the image is a measure of the volume fraction that it occupies in the solid. This relationship is, in fact, one of the oldest known relationships in stereology, used in mineral analysis 150 years ago.

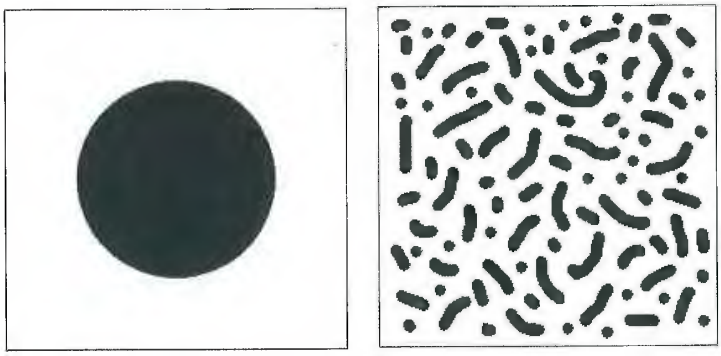
Of course, this requires some explanation and clarification of the assumptions. The image must be representative in the sense that every part of the solid has an equal chance of being examined, so the sections must be uniformly and randomly placed in the solid. Trying to measure the volume fraction of bone in the human body requires that head, torso, arms, and legs all have an equal chance of being viewed; that is the "uniform" part of the assumption. Random means, simply, that nothing is done to bias the measurements by including or excluding particular areas in the images. For example, choosing to measure only those images in which at least some bone was visible would obviously bias the measurements. More subtle, in many cases, is the tendency of microscopists to select areas for imaging that have some aesthetic quality (collecting pretty pictures). Almost certainly this will tend to bias the results. A proper random stereological sampling procedure does not allow the human to select or shift the view.

Many published papers in all fields of science that use microscopy include images with the caption "representative microstructure" or "typical microstructure," and in no case is this likely to be true. Either the particular image selected has been chosen because it shows most clearly some feature of the structure that the author believes is important, or it displays the best qualities of specimen preparation and image contrast, or some other characteristic that makes it (almost by definition) non-typical. In most real structures, there is no such thing as one typical field of view in a true statistical sense. That is why it is important to collect many images from multiple fields of view, spread throughout the specimen in an unbiased way. Data are collected from many fields and combined to represent the entire structure.

Assuming that one image could be a uniform, random sample of the structure, then the "expected value" of the area fraction of the structure is equal to the volume fraction. Of course, in any given image that may not be the result. In some images, the phase or structure of interest may not even be present, while in others it may occupy the entire field of view. In general it is necessary to select an appropriate magnification so that the structures are visible, and to examine multiple fields of view and to average the measurements. The average then approaches the true value as more measurements are included.

One way to measure the area fraction of a structure is, of course, to use the image histogram. If the phase has a unique grey scale or color value, then the area of peak in the histogram provides a direct measure of the number of pixels covered, and hence the total area, regardless of whether it occupies one large or many small regions in the image. As shown in previous chapters, however, it is common to require image processing before thresholding can selectively delineate a structure, and to require editing of the binary image after thresholding. These steps also affect the area measurement, so in most cases the determination of area fraction must be made from the final binary image. All that is required is to count the black and white pixels.

Although this is a simple procedure, it is difficult to assess the accuracy of the measurement. Pixels along the boundaries of features or regions present a challenge because thresholding and subsequent morphological processing may include or exclude them from the total. An image consisting of a single large, compact region has many fewer edge pixels (and thus less potential



a b

Figure 1. Two images with the same area fraction (26%) of black pixels: *(a) one large compact region; (b) many small irregular features. The measurement precision depends on the total periphery of the black-white boundary. It is also interesting to note that most human observers do not estimate area fractions very accurately, nor judge that these two images have the same area of black pixels.*

measurement error) than an image in which the same total area is distributed as many small or irregular features (**Figure 1**).

There is a preferred way to determine area fraction (and hence volume fraction) that is very efficient and does allow an estimation of the measurement precision. Traditionally, this method has been performed manually, but it is also easy to accomplish using a computer. A grid of points is superimposed on the image, and the fraction of the points that fall on the structure of interest is counted. The expected value of this point fraction is also the volume fraction. Often, the number of points in the grid is very small so that manual counting can be done at a glance. The grid may be on an eyepiece reticule in the microscope, or overlaid on a video screen or photograph, or generated within the computer. The points should be far enough apart that they provide independent measures of the structure (in other words, at the image magnification being used, two points should rarely fall into the same feature). If the structure is random, then any grid of points can be used and a regular square grid is convenient. If the structure is highly regular, then the grid itself should be randomized to prevent bias as discussed above.

When a grid is superimposed on the original grey-scale or color image (with whatever enhancement has been provided by processing), the human viewer can use independent knowledge and judgment to decide whether each point is within the region of interest. People are very good at this. But they are not very good at counting, so it may still be useful to have the human mark the points and the computer perform the counting operation as shown in **Figure 2**.

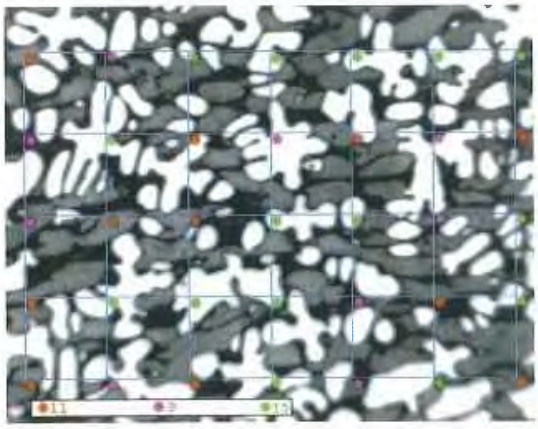


Figure 2. *Computer counting of marks. A human has placed color marks near the grid points (intersection of the lines) that fall on each phase of interest, and the computer has counted them. For the white phase, the estimated volume fraction is 15/35 = 42.8%.*

The points become a probe into the three-dimensional microstructure, and the number of points that "hit" the phase of interest allows an estimate of the measurement precision, since for independent events the standard deviation is just the square root of the count. This permits making a quick estimate of the number of images (multiple fields of view on multiple sections) that will be needed to achieve the desired final measurement precision. For example, if a grid of 25 points is used (a 5 × 5 array), and about 5 points, on average, lie on the phase of interest, that corresponds to a volume fraction of 20%. To determine the actual value with a relative precision of (say) 5% (in other words 20±1% volume fraction) it is only necessary to apply the grid to multiple fields of view until a total of 400 hits have been tallied (the square root of 400 is 20, or 5%). This would require about 80 fields of view (400/5).

A grid of points can also be efficiently applied to a thresholded and processed binary image using Boolean logic as shown in Chapter 7. If the grid is combined with the binary image using a Boolean "AND," and the surviving points are counted, the result is just those points that fell onto the phase of interest as shown in **Figure 3**. The accepted notation for these relationships is

$$V_V = A_A = P_P \tag{1}$$

meaning that the volume of the phase of interest per unit volume of material, V_V, equals (or more precisely is measured by) the area fraction A_A or the point fraction P_P.

Volume fraction is a dimensionless ratio, so the magnification of the image need not be known exactly. However, it is also possible to measure the volume of a specific structure by cutting a series of sections through it and measuring the area in each section (Gundersen, 1986). As shown in **Figure 4**, the volume of the sampled space is defined by the size and spacing of the sections, and the number of "hits" made by the grid points provides an absolute measure of the feature volume. Each grid point samples the structure and represents a volume in space equal to the area of a grid square times the section spacing. In this case, of course, the magnification must be calibrated.

In many cases, as shown in **Figure 5**, it is necessary to measure two volumes, that of the structure of interest and of an enclosing structure. In the example, a sectioned rat lung, the area of lung

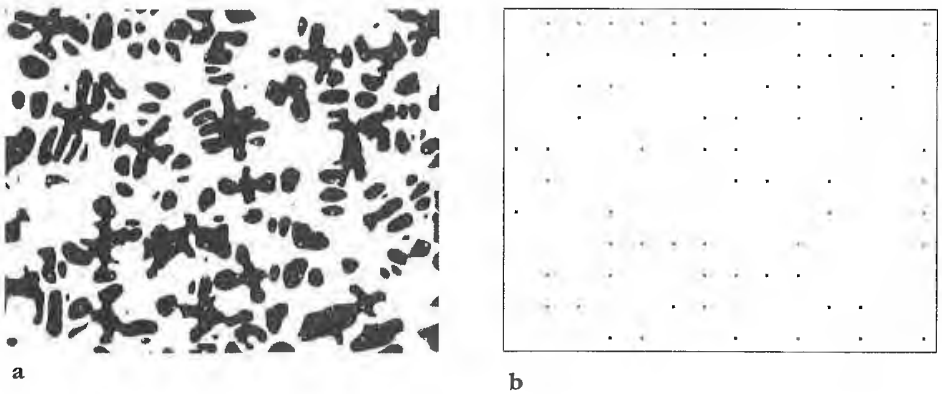
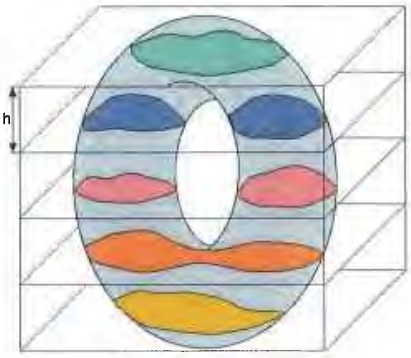


Figure 3. *Binary image of the white phase in the metal structure from* ***Figure 2****, with a superimposed point grid (points enlarged for visibility). A Boolean AND of the two images leaves just the points that lie on the phase* ***(b)****, which the computer then counts to determine* $P_P = 64/154 = 41.5\%$*. The density of points illustrated in this image is somewhat too high for optimum precision estimation, because cases occur in which multiple points fall on the same feature.*



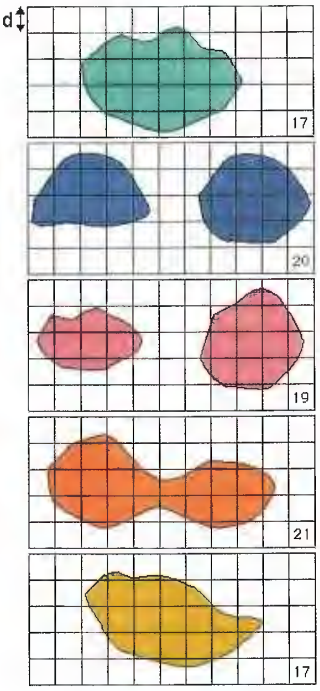


Figure 4. *Cavalieri's method for measuring the volume of an object. The grids on each section plane divide the object into cells. Counting the grid points that fall on the structure of interest provides a measure of the number of cells within the structure, and hence a measure of the volume. In the example, if the plane spacing* ***(b)*** *is 5 µm and the grid spacing* ***(d)*** *is 2 µm, then each cell has a volume of* $5 \times 2 \times 2 = 20\ \mu m^3$*, and the total number of hits (17 + 20 + 19 + 21 + 17 = 94) estimates the total volume as 1880 µm3. Measuring the area of each intersection and integrating the volume with Simpson's rule gives an estimate of* $1814\ \mu m^3$.

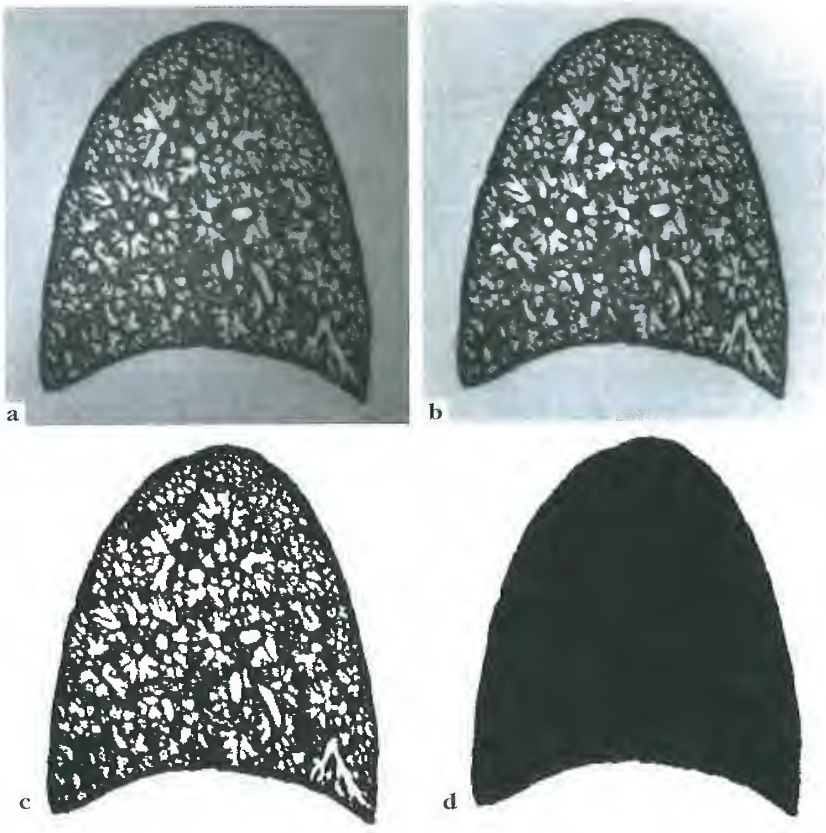


Figure 5. One section of rat lung tissue: ***(a)*** *original image;* ***(b)*** *leveled;* ***(c)*** *thresholded;* ***(d)*** *internal gaps filled. The areas of images* **c** *and* **d** *are used to determine the void volume within the lung.*

tissue in each section is added including internal voids, and the net area is also summed. The void area within the lung is then calculated as

$$Volume = \frac{\sum Filled\ Area - \sum Net\ Area}{\sum Filled\ Area} \tag{2}$$

Surface area

Another global parameter that is easily measured stereologically is the surface area. Surfaces can be the boundary between two structures, such as the surface area of the nucleus, the total area of cell wall in plant tissue, the total grain boundary area in a metal, or the surface of porosity in a ceramic. Note that in some of these examples the surface separates two different phases or structures, while in others it separates different regions (cells or grains) that are the same in structure and composition. Surfaces are usually very important in structures, because they are the interfaces where chemistry and diffusion take place, and control many properties such as strength, fracture, light scattering, etc.

In a two-dimensional section image through a three-dimensional structure, boundaries and surfaces are seen as lines. The total length of these lines is proportional to the amount of surface area present in the three-dimensional solid. The surface may not intersect the section plane perpendicularly, in fact in general it will not. If the surfaces are isotropic (have an equal probability of being oriented in any direction), then the relationship between total surface area per unit volume of sample and the total length of line per unit area of image is

$$S_V = \tfrac{4}{\pi} B_A \tag{3}$$

where S_V is the accepted notation for the surface area per unit volume, and B_A denotes the total length of boundary per unit area of image. Notice that both terms have dimensions of 1/length, and that it is consequently important to know the image magnification. **Figure 6** shows an example. The boundary lines (colored) have a total length of 9886 μm, and the image area is 15,270 μm^2. This gives a calculated surface area per unit volume of 0.0824 $\mu m^2/\mu m^3$, or 82.4 mm^2/mm^3.

Once again, it is difficult to specify the precision of such a measurement. Measuring the length of a boundary line in a digitized image is one of the most error-prone tasks in image measurement because of the pixellation of the image and the common observation that as magnification is increased, more irregularities in the boundary become visible and the measured boundary length increases. The preferred method for determining surface area is consequently to place a grid of lines on the image and to count the number of intersection points which they make with the line representing the surface of interest. The relationship between the surface area per unit volume (S_V, square micrometers per cubic micrometer) and the number of intersections (P_L, number per micrometer) is just

$$S_V = 2 \cdot P_L \tag{4}$$

where the factor 2 compensates for the various angles at which the grid lines can intersect the boundary. Again, because this is now a counting experiment, the measurement precision can be estimated from the square root of the number of intersection points, provided that the lines (which are the probes into the microstructure in this measurement) are far enough apart that the intersections are independent events.

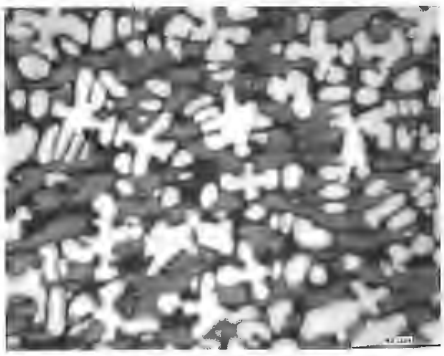


Figure 6. *The boundary lines around the white phase in the same image as* ***Figures 2*** *and* ***3*** *can be isolated using the techniques from Chapter 7, and measured to determine the surface area as described in the text.*

Generating a grid of lines on the same image as in **Figure 6** and counting intersections (**Figure 7**) produces a similar measurement result as the boundary line technique. The total length of the grid lines is 4646 μm. Performing a Boolean AND and counting the intersections gives 197 hits (some are a single pixel and some more than one, depending on the angle between the boundary and the line). Using **Equation 4**, this corresponds to a surface area per unit volume of 0.0848 $\mu m^2/\mu m^3$. Based on the number of counts, the estimated relative precision is ±7.2% (0.0848 ± 0.0061 $\mu m^2/\mu m^3$).

Placing grid lines on images and counting intersections can be performed in all of the same ways (manually, manual marking with computer counting, or automatically using Boolean logic) as discussed above for point counting. The problem with this method as described is that it relies on the assumption that the surfaces being measured are isotropic. In real structures this criterion is rarely met. Consequently, if the structure is not isotropic, then it is necessary to construct a grid of lines that does sample the structure isotropically (in addition to the requirements of uniform and random sampling noted above). Many of the developments in modern stereology are aimed at finding practical ways to meet this isotropic, uniform, random (IUR) requirement. To visualize the meaning of isotropic directions, consider a hemisphere as shown in **Figure 8**. Each direction in space is represented by a point on the sphere. Points should be distributed evenly across the spherical surface.

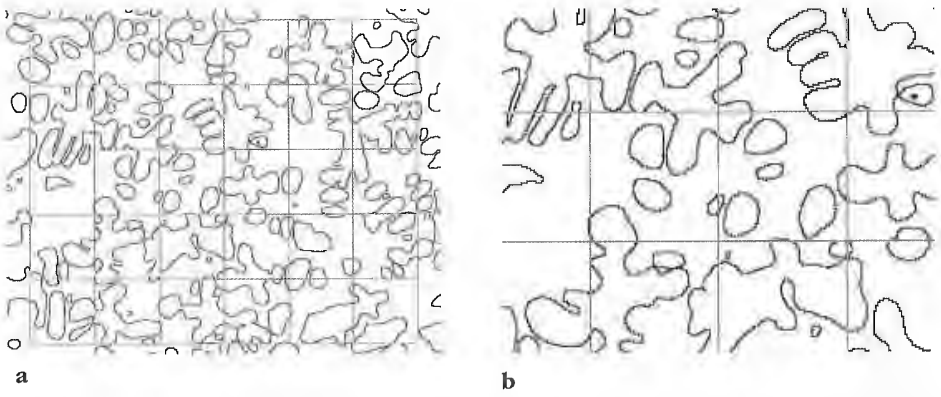


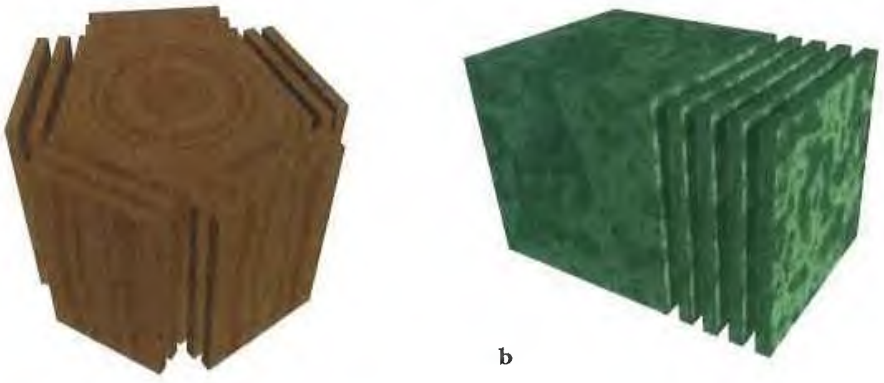
Figure 7. *ANDing a grid of lines with the boundary lines from* ***Figure 6*** *and counting the intersections also measures the surface area, as described in the text.*



Figure 8. *A hemisphere showing a direction in three-dimensional space represented by a point on the sphere.*

The most successful approach to generating isotropic, uniform and random lines to probe the surfaces in a three dimensional structure is called "vertical sectioning" (Baddeley et al., 1986). It requires selecting some direction in the structure that can always be identified. This might be the axis of the backbone in an animal, or the normal to a surface in a rolled metal sheet, or the direction of gravity for sedimentary layers of rock. Then all sections are cut parallel to this direction, but rotated uniformly and randomly about it, as shown in **Figure 9**. Note that this is emphatically not the way most sections are cut when biological tissue is embedded and microtomed (those sections are all perpendicular to the same direction). It would also be possible to cut the vertical sections as a pie is normally cut, with radial slices, but these would oversample the center of the pie compared to the periphery.

The vertical axis direction is present on all sections, so these cut planes are directionally biased. If lines were drawn on these sections with uniform directions in the plane, as shown in **Figure 10**, they would cluster near the north pole on the hemisphere. Directions near the equator would be undersampled. That bias can be compensated by drawing lines that are sine-weighted, as shown in **Figure 11**. Instead of drawing lines in uniform angle steps, they are drawn with uniform steps in the sine of the angle. This produces more directions around the equator and spreads out the directions near the north pole so that the points are uniformly distributed on the sphere. While a set of radial lines as shown in the figure is isotropic, it is not uniform (the center is oversampled and the corners undersampled).



a b

Figure 9. Method for cutting vertical sections:
(a) *correct — all sections are parallel to and rotated uniformly about a single selected axis;*
(b) *incorrect — sections are all perpendicular to a common direction.*

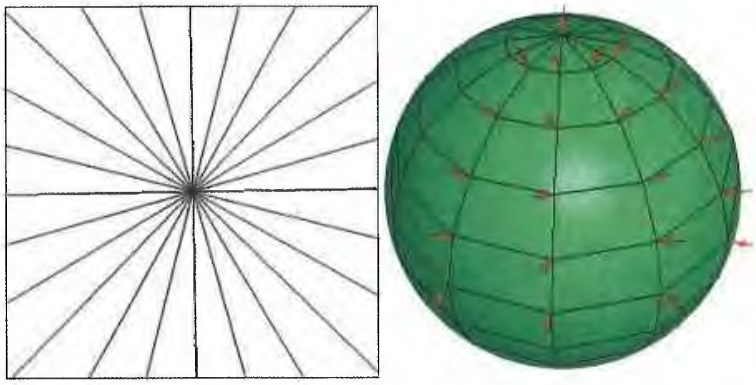
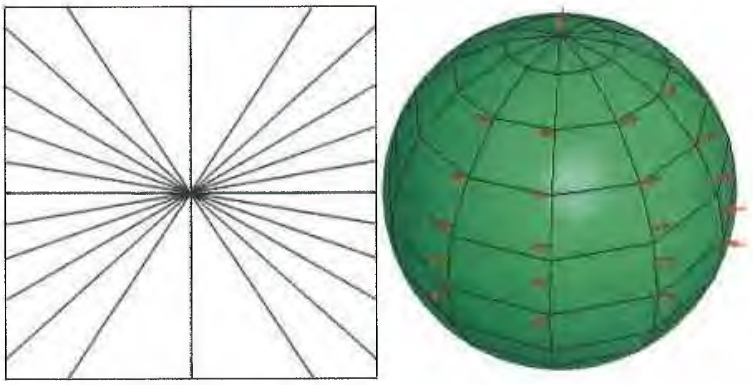


Figure 10. *Drawing lines with uniform angles on each plane, and the resultant clustering of directions near the north pole of the hemisphere of orientations.*

Figure 11. *Drawing lines with sine-weighted angles on each plane, and the resultant uniform distribution of orientations in three-dimensional space.*



The most convenient way to draw sine-weighted lines that uniformly sample the area is to generate cycloids as shown in **Figure 12**. The cycloid is the path followed by a point on the circumference of a rolling circle. It is sine-weighted and when drawn on vertical sections has exactly the correct set of orientations to provide isotropic sampling of directions in the three-dimensional solid from which the sections were cut. Counting the intersections made by the cycloid grid with the boundary lines in the image provides the P_L value needed to calculate the surface area per unit volume. The length of each quarter arc of the cycloid line is just twice its height.

If the specimen is actually isotropic, the method of cutting vertical sections and counting intersections using a grid of cycloids will produce the correct answer, at the expense of doing a bit more work (mostly in sample preparation) than would have been needed if the isotropy was known beforehand. But if the specimen has any anisotropy, the easy method of cutting parallel sections and drawing straight line grids would produce an incorrect answer with an unknown amount of bias, while the vertical section method gives the true answer.

ASTM Grain Size

More than 100 years ago, when naval gunnery was introducing larger and larger guns onto battleships, brass cartridge casings were used to hold the gunpowder and projectile, essentially an enlarged version of a modern rifle cartridge. It was observed that after firing it was sometimes difficult to extract the used brass cartridge from the gun chamber, because the material fractured when it was pulled from the rear rim. The metal flowed and thinned when the charge was detonated, and clearly some difference in material structure was related to the tendency to tearing.

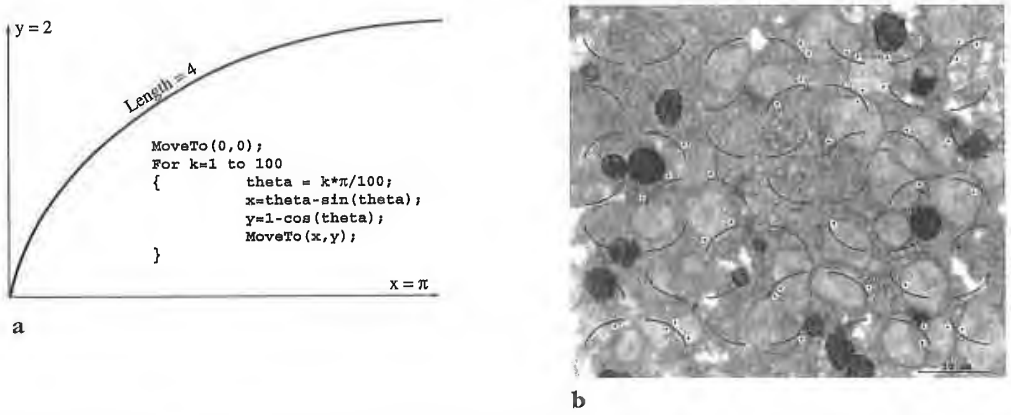


Figure 12. *Procedure for drawing an arc of the cycloid* ***(a)****, and a cycloid grid superimposed on a vertical section image* ***(b)****. The intersections (marked in color) of the cycloid lines with the membranes provide an unbiased measure of the total surface area per unit volume,* $S_V = 2\,P_L = 2 \cdot 39 / 331.5\mu m = 0.235\ \mu m^2/\mu m^3$.

From that observation, a method for measuring a microstructural parameter known as the ASTM (American Society for Testing and Materials) "grain size" was developed, and used for quality control of naval brass, and later many other metals and nonmetals. In fact, several different standard techniques are used, which have been adjusted with appropriate constants so that they agree approximately with each other's numeric values; but, in fact, the methods measure two entirely different characteristics of microstructure (neither of which is the "size" of the grains).

The first method proposed was to count the number of visible grains per square inch on a polished section at 100x magnification. The number of grains N is then related to the "grain size number" G by

$$N = 2^{(G-1)} \tag{5}$$

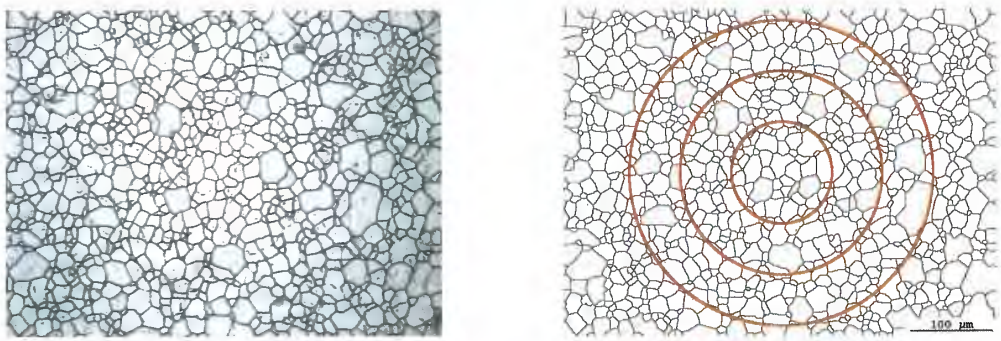
where the value G is never reported with a precision better than 0.5. An equivalent version adjusted to metric dimensions is also in use.

A second method for determining the value G is based on drawing a grid of lines on the image. Because many metals are anisotropic due to the forming processes, the grains are elongated in one direction. In order to avoid directional bias in the measurement, a circular grid is used. The number of intersections made by the line with the grain boundaries per unit length of the line (N_L where length is in millimeters) can is used to calculate

$$G = 6.6457 \cdot \log_{10} N_L - 3.298 \tag{6}$$

The constants are needed to convert the metric units and to make the results approximately agree with the grain count method. Again, the results are rounded to the nearest 0.5; **Figure 13** shows an example.

There are also other methods that are part of the ASTM standard (E112), such as counting the number of triple points (places where three grains meet at a point) per unit area, and most of these methods may be implemented either by hand count or by computer.



a b

Figure 13. *Light microscope image of a low carbon steel* ***(a)****, which is prepared by leveling, thresholding and skeletonizing to delineate the grain boundaries* ***(b)****. Counting the number of grains inside the largest circle (445, counting each one that intersects the circle as 1/2) gives a grain size number G = 8.7. Counting the intercepts with the concentric circle grids (147 with a grid length of 2300 µm) gives a grain size number G = 9.1. Both values would normally be reported as a grain size number of 9.*

The intercept count method (**Equation 6**) in reality measures the total surface area of grain boundary. Note the use of N_L which, similar to the P_L value in **Equation 4**, is proportional to the surface area per unit volume. The importance of grain boundary area makes sense from a mechanical property standpoint since creep in materials occurs due to grain boundary sliding and grain rotation, and grain deformation occurs by motion of dislocations which start and end at these same boundaries. The grain count and triple-point count methods actually measure the total length per unit volume of the edges of grains, which is a parameter related primarily to diffusion. Measurement of the length of structures is discussed in the following paragraphs.

The fact that one of these parameters can be even approximately tied to the other is due to the limited application of the method to metals that have been heat-treated so that they are fully recrystallized. In this condition, the distribution of actual three-dimensional grain sizes approaches a constant, more-or-less log-normal state in which the structure may coarsen with time (large grains grow and small ones vanish) but the structure remains self similar. Hence there is an approximately consistent relationship between the length of edges and the area of surfaces.

The same relationship does not apply to metals without this heat treatment, and so the notion of a single numerical value that can characterize the microstructure is flawed, even for quality control purposes, and the measurement method used does matter. But 100 years of use have sanctified the technique and most practitioners are not aware of what their procedure actually measures in the microstructure.

Actually measuring the "size" (usually the volume) of individual grains or cells in a three-dimensional solid is quite difficult. It has been done in a few cases by literally taking the material apart (e.g., by chemically dissolving a thin layer along the boundaries so the grains are separated), and measuring each one. Other researchers have used an exhaustive serial sectioning technique that produces a complete three-dimensional volumetric image of the structure, from which measurements can be made. Both of these approaches are far too laborious for routine practical use. Determining the mean volume of cells can be done using the Disector method described below. This actually measures the number per unit volume, but the inverse of that quantity is the mean volume per feature. This method relies heavily on computer-based image processing to deal with the large number of images. **Figure 14**, below, will show an example.



Figure 14. *The three-phase aluminum-zinc alloy from* ***Figure 2****, with the phase boundaries shown in color.*

It is also possible to measure the variance of the size distribution of the features using a point-sampled intercept method applied to IUR section planes, described below. But if the actual size distribution of the three-dimensional features is needed, then the more intensive methods must be undertaken.

Multiple types of surfaces

In most kinds of real samples, there are several different phases or structures present, and consequently many different types of boundary surfaces. In a three-phase material, containing regions which for convenience can be identified as types α, β and γ, there are six possible interfaces (α-α, α-β, α-γ, β-β, β-γ, and γ-γ) and the number goes up rapidly in more complex structures. Some of the possible interface types may be absent, meaning that those two phase regions never touch (e.g., nuclei in separate cells do not touch each other, it makes no sense to consider a surface between two pores, etc.). Measuring the amounts of each different type of interface can be very useful in characterizing the overall three-dimensional structure.

Figure 14 shows a three-phase metal structure. Notice that no white dendritic region touches another one, for example. Chapter 7 showed how Boolean logic with the dilated region outlines can be used to delineate each type of interface. Once the image of the boundaries has been isolated, counting intersections with a grid provides the measurement of surface area per unit volume.

Figure 15 shows a simpler, idealized, two-phase microstructure. By thresholding each phase, using morphological operations such as a closing, and Boolean logic to combine two derived images, each of the three distinct boundary types can be isolated (in the image, they have been combined



a b c

Figure 15. Idealized drawing of a two-phase microstructure
(a) *the three different types of interface boundaries present*
(b) *and the result of counting the intersections of these boundaries with a cycloid grid*
(c)*. The measurements and calculated results are detailed in the text.*

as color channels for visualization purposes). Assuming this is a vertical section, a cycloid grid such as the one shown can be used to estimate the surface area per unit volume as shown in the figure, using **Equation 4**.

It may be instructive to illustrate and compare the various measurement procedures described above for volume surface area measurement, and using the magnification calibration shown on the image in **Figure 15**. Counting pixels estimates the area fraction, and hence the volume fraction of the grey phase at 17.3%. Placing a square grid of 90 points produces 14 hits, for an volume fraction estimate using **Equation 1** of 15.6%. Measuring the length of the boundary lines for each type of interface and applying equation 3 produces the results listed below. Counting intersections of the grid lines with each type of boundary and applying equation 4 gives another estimate of surface area, also shown in the table. The image area is 5453.6 μm^2 and the grid used had a total length of 614 μm.

Boundary	Length (μm)	$S_V = (4/\pi) \bullet$ Length/Area	Grid counts	$Sv = 2 P_L$
White-white	441.96	103.2 mm^2/mm^3	28	101.8 mm^2/mm^3
Grey-white	551.24	128.7	37	134.5
Grey-grey	123.30	28.8	8	29.1

The agreement between these different methods is quite satisfying, but only the counting methods allow a simple estimate of the measuring precision be made. Note that it is possible to squeeze a considerable amount of surface area into a small volume.

Length

Length measurement is usually applied to structures that are elongated in one direction, such as neurons or blood vessels in tissue, fibers in composites, or dislocations in metals. It is also possible, however, to measure the total length of edges, such as the edges of polyhedral grains in a metal (as mentioned earlier in connection with the ASTM grain size measurement), or any other line that represents the intersection of surfaces.

When a linear structure intersects the sampling plane, the result is a point. The number of such points is proportional to the total length of the line. If the linear structures in the three-dimensional solid are isotropic, then the relationship between the total length of line per unit volume L_V (with units of micrometers per cubic micrometer or length^{-2}) and the total number of points per unit area P_A (with units of number per square micrometer or length^{-2}) is

$$L_V = 2 \cdot P_A \tag{7}$$

where the constant 2 arises, as it did in the case of surface area measurements, from considering all of the angles at which the plane and line can intersect. In some cases a direct count of points can be made on a polished surface. For example, fibers in a composite are visible, and dislocations can be made so by chemical etching.

In the example of **Figure 13**, the triple points where three grains meet represent the triple lines in space that are the edges of the grains. These are paths along which diffusion is most rapid. To isolate them for counting, the skeleton of the grain boundaries can be obtained as shown in Chapter 7. The branch points in this tesselation can be counted as shown in **Figure 16** because they are points in the skeleton that have more than two neighbors. Counting them is another way (less commonly used) to calculate a "grain size" number.

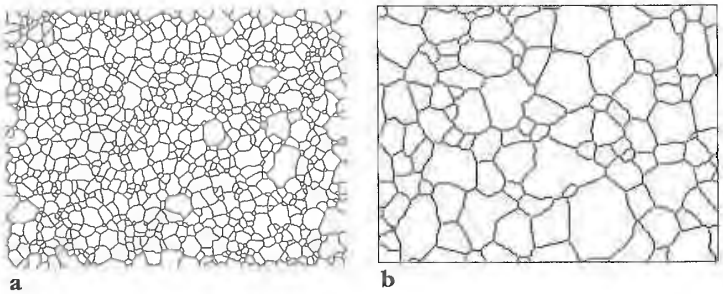


Figure 16. The image from Figure 13, thresholded and skeletonized with the 1732 branch points in the skeleton color coded (with an enlarged portion for visibility).

One convenient way to measure the length of a structure, independent of its complexity or connectedness, is to image a thick section (considerably thicker than the width of the linear portions of the structure). In this projected image, the length of the features is not truly represented because they may incline upwards or downwards through the section. Any line drawn on the image represents a surface through the section, with area equal to the product of its length and the section thickness. Counting the number of intersections made by the linear structure with the grid line, and then calculating the total length using the relationship in **Equation 7**, provides a direct measurement of the total length per unit volume.

If the sample is isotropic, any grid line can be used. If not, then if the sections have been taken with isotropic sectioning, a circle grid can be used since that samples all orientations in the plane. But the most practical way to perform the measurement without the tedium of isotropic section orientation or the danger of assuming that the sample is isotropic is to cut the sections parallel to a known direction (vertical sectioning) and then use cycloidal grid lines, as shown in **Figure 17**. In this case, since it is the normal to the surface represented by the lines that must be made isotropic in three-dimensional space, the generated cycloids must be rotated by 90° to the "vertical" direction.

In the example of **Figure 17**, the intersections of the grid lines with the projections of the microtubules allow the calculation of the total length per unit volume using **Equation 7**. There are 41 marked intersections; of course, these could also be counted by thresholding the image and ANDing it with the grid, as shown previously in this chapter and in Chapter 7. For an image area of 12,293 μm^2, an assumed section thickness of 3 μm, and a total length of grid lines of 331.2 μm, the length calculation from **Equation 7** gives:



Figure 17. Application of a cycloid grid to a TEM image of a section containing filamentous microtubules. Counting the number of intersections made with the grid allows calculation of the total length per unit volume, as discussed in the text.

$$L_V = 2 \cdot 41 / (12293 * 3)\ \mu m/\mu m^3 = 2.22\ mm / mm^3$$

It is rare to find a real three-dimensional structure that is isotropic. If the structure is (or may be) anisotropic, it may be necessary to generate section planes that are isotropic. Procedures for doing so have been published, but in general they are tedious, wasteful of sample material, and hard to make uniform (that is, they tend to oversample the center of the object as compared to the periphery).

Another way to generate isotropic samples of a solid, either for surface area or length measurement, is to subdivide the three-dimensional material into many small pieces, randomly orient each one, and then make convenient sections for examination. This is perhaps the most common approach that people really use. With such isotropic section planes, it is not necessary to use a cycloid grid. Any convenient line grid will do. If there is preferred directionality in the plane, circular line grids can be used to avoid bias. If not, then a square grid of lines can be used. If the structure is highly regular, then it is necessary to use randomly generated lines (which can be conveniently performed by the computer). **Figure 18** shows examples of a few such grids. Much of the art of modern stereology lies in performing appropriate sample sectioning and choosing the appropriate grid to measure the desired structural parameter.

Sampling strategies

Several of the examples shown thus far have represented the common situation of transmission microscopy in which views are taken through thin slices, generally using the light or electron microscope. The requirement is that the sections be much thinner than the dimensions of any

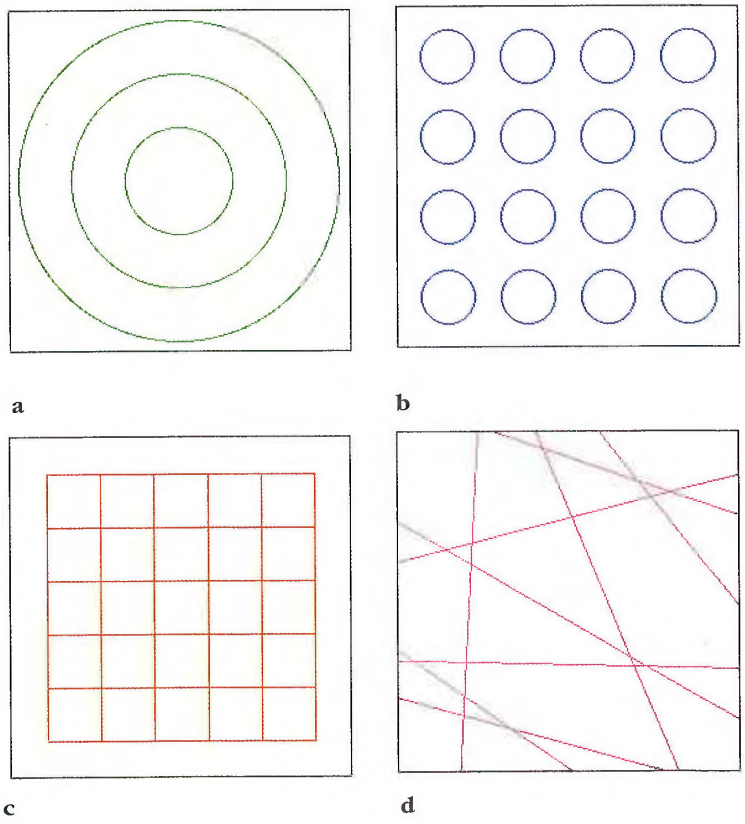


Figure 18. Examples of line grids:
(a) *concentric circles;*
(b) *array of circles;*
(c) *square line grid;*
(d) *random lines.*

structures of interest, except for the case above of thick sections used for length measurement. For opaque materials, the same strategies are used for sectioning but the images are obtained by reflected light, or by scanning electron microscopy, or any other form of imaging that shows just the surface. Many imaging techniques in fact represent the structure to some finite depth beneath the surface, and most preparation methods such as polishing produce finite surface relief in which the software structures are lower than the harder ones. Again, the criterion is that the depth of information (e.g., the depth from which electrons are emitted in the scanning electron microscope, or the depth of polishing relief) is much less than the dimensions of any structures of interest. Corrections to the equations above for measuring volume, area and length can be made for the case of finite section thickness or surface relief, but they are complicated and require exact measurement of the thickness.

Although the procedures for obtaining isotropic sampling have been outlined, little has been said thus far about how to achieve uniform, random sampling of structures. It is of course possible to cut the entire specimen up into many small pieces, select some of them by blind random sampling, cut each one in randomly oriented planes, and achieve the desired result. However, that is not the most efficient method (Gundersen and Jensen, 1987). A systematic or structured random sample can be designed that will produce the desired unbiased result with the fewest samples and the least work. The method works at every level through a hierarchical sampling strategy, from the selection of a few test animals from a population (or cells from a petri dish, etc.) to the selection of tissue blocks, the choice of rotation angles for vertical sectioning, the selection of microtomed slices for viewing, the location of areas for imaging, and the placement of grids for measurement.

To illustrate the procedure, consider the problem of determining the volume fraction of bone in the human body. Clearly, there is no single section location that can be considered representative of such a diverse structure: a section through the head would show proportionately much more bone than one at the waist. Let us suppose that, in order to obtain sufficient precision, we have decided that eight sections are sufficient for each body. For volume fraction, it is not necessary to have isotropic sections so we could use transverse sections. In practice, these could be obtained nondestructively using a computerized tomography (CT) scanner or magnetic resonance imaging (MRI). If the section planes were placed at random, there would be some portions of the body that were not viewed while some planes would lie close together and oversample other regions. The most efficient sampling would space the planes uniformly apart, but it is necessary to avoid the danger of bias in their placement, always striking certain structures and avoiding others.

Systematic random sampling would proceed by generating a random number to decide the placement of the first plane somewhere in the top one-eighth of the body. Then position the remaining planes with uniform spacing, so that each has the same offset within its one-eighth of the body height. These planes constitute a systematic random sample. A different random number placement would be used on the next body, and the set of planes would be shifted as a unit. This procedure guarantees that every part of the body has an equal chance of being measured, which is the requirement for random sampling.

When applied to the placement of a grid on each image, two random numbers are generated which specify the position of the first point (the upper left corner) of the grid. The other grid points then shift as a unit with uniform spacing. The same logic applies to selecting measurement positions on a slide in the microscope. If for adequate statistical sampling it has been decided to acquire images from, for example, 8 fields of view on slide, then the area is divided into 8 equal areas as shown in **Figure 19**, two random numbers are generated to position the first field of view in the first rectangle, and the subsequent images are acquired at the same relative position in each of the other areas.

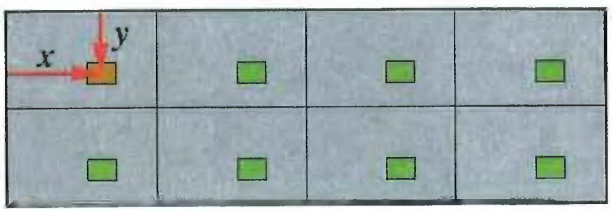


Figure 19. *Systematic random sampling applied to selecting fields of view on a slide. The area is divided into as many regions as the number of images to be acquired. Two random numbers are used to select the location for the first image in the first region (orange). Subsequent images (green) are then captured at the same relative position in each region.*

For rotations of vertical sections, if it is decided to use five orientations as shown earlier in **Figure 9**, then a random number is used to select an angle between 0 and 72° for the first cut, and then the remaining orientations are placed systematically at 72-degree intervals. For selection of animals from a population, if 7 animals are to be examined from a group of 100, a random number selects one of the first 14, and then each fourteenth animal after that is selected.

The method can obviously be generalized to any situation. It is uniform and random because every part of the sampled population has an equal probability of being selected. Of all of the possible slices and lines that could be used to probe the structure, each one has an equal probability of being used.

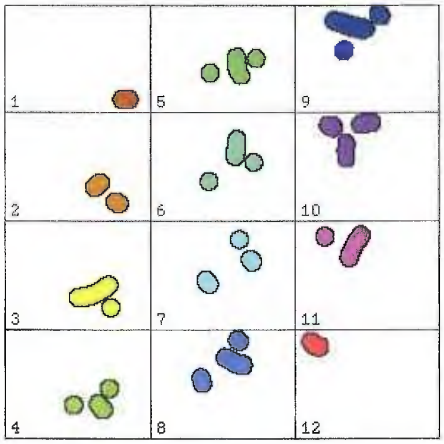
The mantra of the stereologist in designing an experiment is to achieve IUR sampling of the structure. For some types of probes the requirement of isotropy can be relaxed because the probe itself has no directionality, and thus is not sensitive to any anisotropy in the specimen. Using a grid of points, for example, to determine the volume fraction of a structure of interest, does not require the vertical sectioning strategy described previously because the point probe has no orientation. Any convenient set of planes that uniformly and randomly sample the structure can be probed with a uniformly and randomly placed grid to obtain an unbiased result.

The disector, described in the next section, is a volume probe and hence also has no directionality associated with it. Pairs of section planes can be distributed (uniformly and randomly) through the structure without regard to orientation. Being able to avoid the complexity of isotropic sectioning is a definite advantage in experimental design.

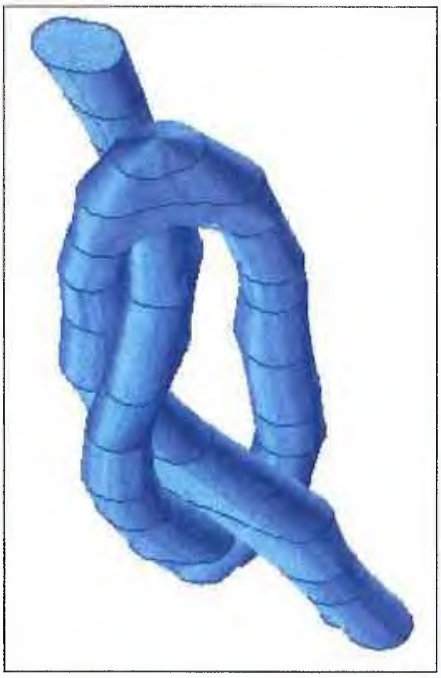
As noted previously, for measurement of length or surface area the probes (surfaces and lines) have orientation, so they must be placed isotropically, as well as uniformly and randomly. It is in these cases that a strategy such as vertical sectioning and the use of cycloidal grids becomes necessary. Using a systematic random sampling approach reduces the number of such sections that need to be examined, as compared to a fully random sampling, but it does not alleviate the need to guarantee that all orientations have an equal probability of being selected. It bears repetition that unless it can be proven that the sample itself is isotropic (and uniform and random), then unless an appropriate IUR sampling strategy is employed the results will be biased and the amount of the bias cannot be determined.

Determining number

Figure 20 shows several sections through a three-dimensional structure. Subject to the caveat that the sections must be IUR, it is straightforward from the areas of the intersections or by using a point count with a grid to determine the total volume of the tubular structure (**Equation 1**). From the length of the boundary line, or by using intercept counts of the boundary with a suitable line grid, the total surface area of the structure can be determined (**Equations 3** and **4**). From the number of separate features produced by the intersection of the object with the planes, the length of the



a



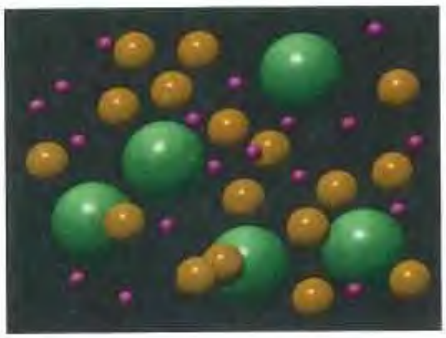
b

Figure 20.
(a) *Multiple sections through a tubular structure, from which the volume, surface area and length can be determined. Only by arranging them in order, aligning them, and reconstructing the volume*
(b) *can the presence of a single string arranged in a right-handed knot be recognized.*

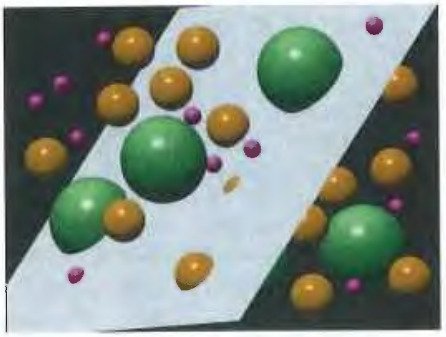
tube can be estimated (**Equation 7**). But there is nothing in the individual section planes that reveals whether this is one object or many, and if it is one object whether the tube is branched or not. The fact that the single tube is actually tied into a knot is of course entirely hidden. It is only by connecting the information in the planes together, by interpolating surfaces between closely spaced planes and creating a reconstruction of the three dimensional object, that these topological properties can be assessed.

Volume, surface area, and length are metric properties. The plane surfaces cut through the structure and the grids of lines and points placed on them represent probes that sample the structure and provide numerical information that can be used to calculated these metric properties, subject to the precision of the measurement and to the need for uniform, random and (except for volume measurement) isotropic sampling. Topological properties of number and connectedness can not be measured using plane, line or point probes. It is necessary to actually examine a volume of the sample.

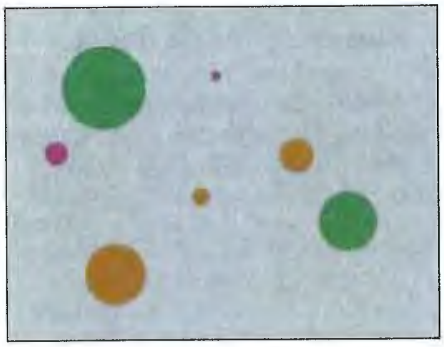
In the limit, of course, this could be a complete volumetric imaging of the entire structure. Nondestructive techniques such as confocal light microscopy, medical CT, MRI or sonic scans, or stereoscopic viewing through transparent volumes, can be used in some instances. They produce dramatic visual results when coupled with computer graphics techniques (which are shown in Chapters 11 and 12); however, these are costly and time consuming methods that cannot easily be applied to many types of samples. Serial sectioning methods in which many sequential thin sections are cut, or many sequential planes of polish are prepared and examined, are much more difficult and costly because of the problems of aligning the images and compensating for distortion or variation in spacing or lack of parallel orientation.



a



b



c

Figure 21.
(a) *A volume containing large (green), medium (orange) and small (purple) objects;*
(b) *an arbitrary section through the volume;*
(c) *the appearance of the objects in the plane. The sizes of the intersections are not the size of the objects, and the number of intersections with each does not correspond to the number per unit volume because large objects are more likely to be intersected by the plane.*

It is a very common mistake to count the number of features visible on a section plane and from that try to infer the number of objects in the three-dimensional volume. The fact that the units are wrong is an early indication of trouble. The number of features imaged per unit area does not correspond to the number per unit volume. In fact, the section plane is more likely to strike a large object than a small one so the size of the objects strongly influences the number that are observed to intersect the sectioning plane, as shown in **Figure 21**.

There is a relationship between number per unit area and number per unit volume based on the concept of a mean diameter for the features. For spheres, the mean diameter has a simple meaning. The number per unit volume N_V can be determined from the number per unit area N_A as

$$N_V = \frac{N_A}{D_{mean}} \tag{8}$$

More precisely, the expected value of N_A is the product of $N_V \cdot D_{mean}$. For convex but not spherical shapes, the mean diameter is the mean caliper or tangent diameter averaged over all orientations. This can be calculated for various regular or known shapes, but is not something that is known *a priori* for most structures. So determining something like the number of grains or cells or nuclei per unit volume (whose inverse would give the mean volume of each object) is in fact rarely practical.

For objects that are not convex, have protrusions or internal void, or holes and bridges (multiple connected shapes), the concept of a mean diameter becomes even more nebulous and less useful. It turns out that the integral of the mean surface curvature over the entire surface of the ob-

ject is related to this mean diameter. As noted below, the surface curvature is defined by two principal radii. Defining the mean local curvature as

$$H = \frac{1}{2}\left(\frac{1}{r_1} + \frac{1}{r_2}\right) \tag{9}$$

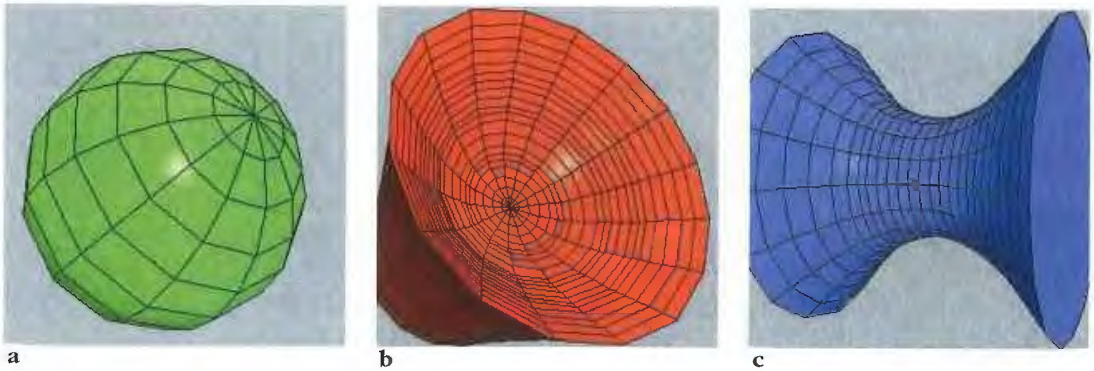
and integrating over the entire surface gives a value called the integral mean curvature M, which equals $2\pi \cdot D_{mean}$. In other words, the feature count N_A on a section plane is actually a measure of the mean curvature of the particles and hence the mean diameter as it has been defined previously. If the number per unit volume can be independently determined, for instance using the disector method described in the next section, it can be combined with N_A to calculate the mean diameter.

Curvature, connectivity, and the disector

Surfaces and lines in three-dimensional structures are rarely flat or straight, and in addition to the relationship to mean diameter introduced above, their curvature may hold important information about the evolution of the structure, or chemical and pressure gradients. For grain structures in equilibrium, the contact surfaces are ideally flat but the edges still represent localized curvature. This curvature can be measured by stereology.

Surface curvature is defined by two radii (maximum and minimum). If both are positive as viewed from "inside" the object bounded by the surface, the surface is convex. If both are negative the surface is concave, and if they have opposite signs, the surface has saddle curvature (**Figure 22**). The total integrated curvature for any closed surface around a simply connected object (no holes, bridges, etc.) is always 4π. For more complicated shapes, the total curvature is $4\pi(N - C)$ where N is the number of separate objects and C is the connectivity.

For a simple array of separate objects, regardless of the details of their shape, C is zero (they are not connected) and the total integrated curvature of the structure gives a measure of how many objects are present. To measure this topological quantity, a simple plane surface is not adequate as explained above. The total curvature can be measured by considering a sweeping plane probe moving through the sample (this can be physically realized by moving the plane of focus of a confocal microscope through the sample, or by using a medical imaging device) and counting the events when the plane probe is tangent to the surface of interest. Convex and concave tangent points (T^{++} and T^{--}, respectively) are counted separately from saddle tangencies (T^{+-}). Then the



a b c

Figure 22. Surface curvature: (a) *convex;* ***(b)*** *concave;* ***(c)*** *saddle*

net tangent count is ($T^{++} + T^{--} - T^{+-}$) gives the quantity (N – C), also called the Euler characteristic of the structure, as

$$(N-C) = \tfrac{1}{2}\left(T^{++} + T^{--} - T^{+-}\right) = \tfrac{1}{2}T_{net} \tag{10}$$

There are two extreme cases in which this is useful. One is the case of separate (but arbitrarily and complexly shaped) objects, in which C is 0 and N is one-half the net tangent count. This provides a straightforward way to determine the number of objects in the measured volume. The other situation of interest is the case of a single object, typically an extended complex network with many branchings such as neurons, capillaries, textiles used as reinforcement in composites, etc. In this case, N is 1 and the connectivity can be measured. Connectivity is the minimum number of cuts that would be needed to separate the network. For flow or communication through a network, it is the number of alternate paths, or the number of blockages that would be required to stop the flow or transmission.

In many situations, such as the case of an opaque matrix, it is not practical to implement a sweeping tangent plane. The simplest practical implementation of a volume probe that can reveal the important topological properties of the structure is the disector (Sterio, 1984). This consists of two parallel planes of examination, which can be two sequential or closely spaced thin sections examined in transmission. For many materials samples, it is implemented by polishing an examination plane which is imaged and recorded, polishing down a small distance (usually using hardness indentations or scratches to gauge the distance and also to facilitate aligning the images), and acquiring a second image.

The disector logic compares the images in the two parallel planes and ignores any feature that continues through both planes, even if the position and shape have changed somewhat. The planes must of course be close enough together that this determination can be made unequivocally. Usually this means that the spacing must be significantly less than any characteristic dimension of the features or structure of interest, and so most features that intersect one plane will intersect both. **Figure 23** shows the possible combinations of events that may be found. Any features that are present in one plane and not the other imply that the three-dimensional structure had a end point somewhere between the two planes. This represents a convex curvature and is counted as a T^{++} event.

Typically, much more rare is the case in which a hollow feature in one plane becomes solid in the other plane. This represents the end of a void or, more generally, a hollow with concave curvature, and is counted as a T^{--} event. Any branching in the structure, in which one feature in one of the two planes can be presumed to connect to two features in the other, represents a saddle surface and is counted as a T^{+-} event. The tangent events from the disector are summed in the same way shown in **Equation 10** to determine the net tangent count and the Euler characteristic of the structure.

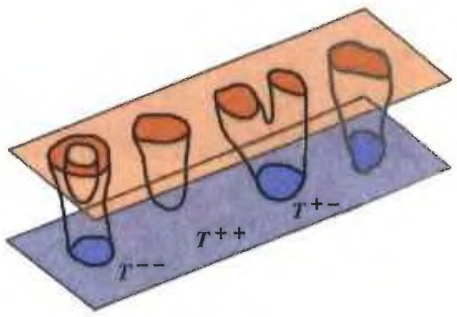


Figure 23. *Possible combinations of features in the two planes of the disector. Features that continue through with no topological change are ignored. Feature ends are counted as convex (T^{++}) events and void ends are counted as concave (T^{--}) ones. Branching indicates the presence of saddle surface with T^{+-} curvature.*

Because most of the features seen in the disector images continue through both sections and so contribute no information, it is necessary to examine many pairs of sections, which of course must be aligned to compare intersections with the features. When the matrix is transparent and viewed in a confocal microscope, examining and comparing the parallel section planes is straightforward. When conventional serial sections are used, it is attractive to use computer processing to align and compare the images. The Feature-AND logic described in Chapter 7 is particularly useful in this regard, to find features that continue through both sections. It is also necessary to sample a large image area, well distributed to meet the usual criteria of random, uniform sampling (since the disector is a volume probe, isotropy is not an issue).

For opaque matrices, the problems are more severe. Generally it is necessary to prepare a polished section, image it, place some fiducial marks on it for reference, and then polish down to some greater depth in the sample and repeat the process. If the fiducial marks are small pyramidal hardness indentations (commonly used to measure hardness of materials), the reduction in their size provides a measure of the distance between the two planes. Aligning and comparing the two images to locate new features can be quite tedious.

Figure 24 shows an example. The specimen is a titanium alloy in which the size of colonies of Widmanstatten laths is of interest. Each colony is identified by its lath orientation. After one set of images was acquired on a metallographically polished plane, the material was polished down a further 5 μm (as measured from the size change of a microhardness indentation) and additional corresponding images acquired. In the example, another colony (identified by the red arrow) has appeared. From the total number of positive tangent events and the volume sampled by the disector, the mean colony volume was determined.

Anisotropy and gradients

There is much emphasis above on methods for performing sectioning and applying grids that eliminate bias in those structures that may not be IUR. Of course, when the specimen meets those criteria then any sampling method and grid can be used, but in most cases it is not possible to assume

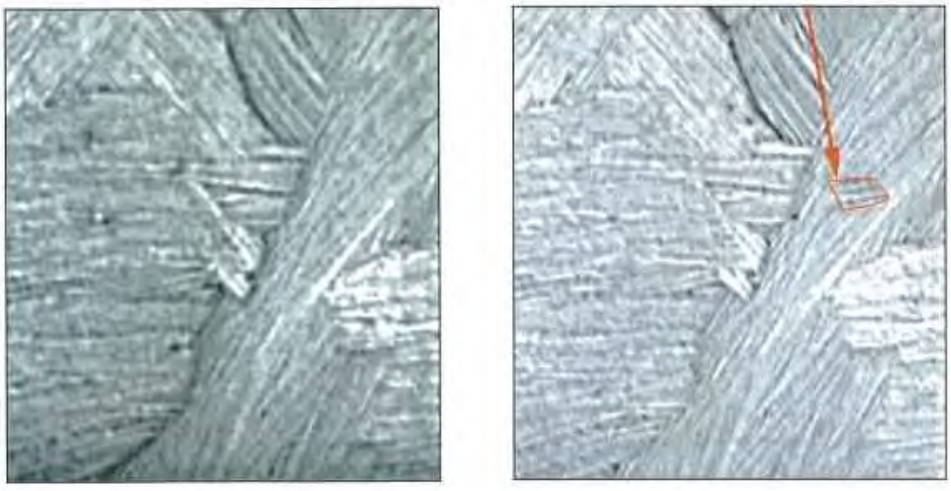


Figure 24. *Scanning electron micrographs of colonies of laths in a titanium alloy (images courtesy Dr. H. Fraser, Dept. of Materials Science and Engineering, Ohio State University, Columbus, OH). The two images represent parallel planes with a measured spacing of 5 μm. A new colony (red arrow and outline) appears in one image, producing a single positive tangent count (T^{++}).*

an IUR structure. Thus, techniques such as vertical sectioning and cycloid grids are used, since they will provide unbiased results whether the specimen is IUR or not, at the cost of somewhat more effort in preparation and measurement.

In some situations, it may be important to actually measure the anisotropy (preferred orientation) in the specimen, or to characterize the nonuniformity (gradients) present in the structure. This requires more effort. First, it is usually very important to decide beforehand what kind of anisotropy or gradient is of interest. Sometimes this is known *a priori* based on the physics of the situation. Deformation of materials in fabrication and the growth of plants typically produce grains or cells that are elongated in a known direction, for example. Many physical materials and biological tissues have structures that vary markedly near surfaces and outer boundaries, so a gradient in that direction may be anticipated. Anisotropy can be qualitatively determined visually, by examining images of planes cut in orthogonal directions, as shown in **Figures 25** and **26**. It is usually necessary to examine at least two planes to distinguish the nature of the anisotropy as shown in **Figure 27**.

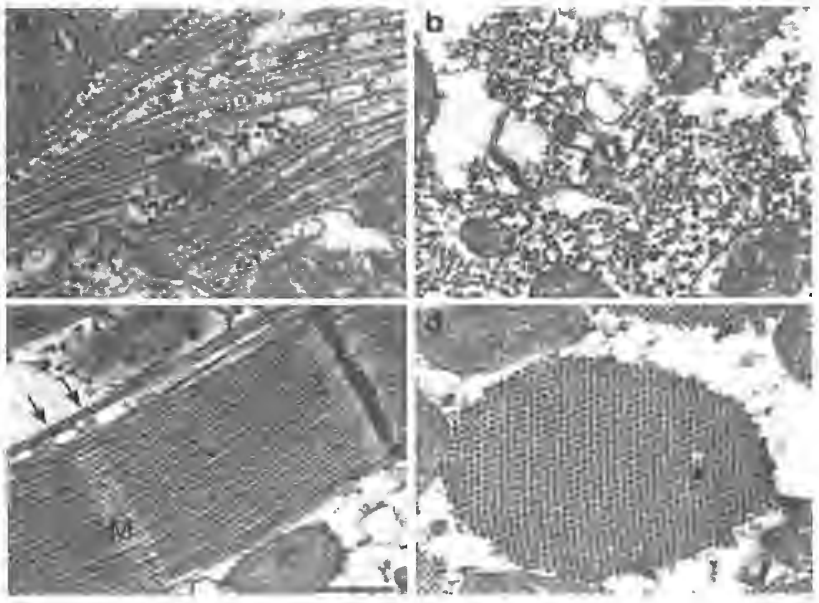


Figure 25. *Anisotropy in muscle tissue, visible as a difference in structure in transverse* ***(b,d)*** *and longitudinal* ***(a,c)*** *sections.*

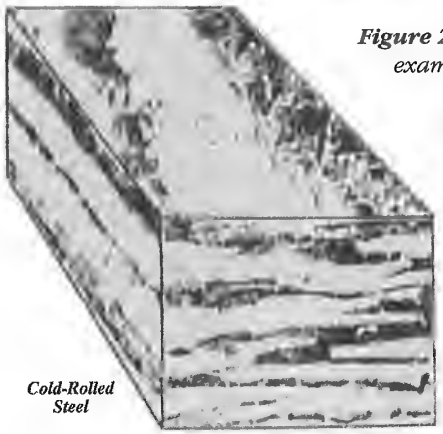


Figure 26. *Anisotropy in rolled steel, shown by microscopic examination of three orthogonal polished surfaces.*

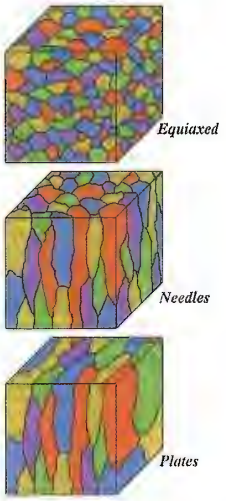


Figure 27. *The need to examine at least two orthogonal surfaces to reveal the nature of anisotropy. In the example, the same appearance on one face can result from different three-dimensional structures.*

When there is expectation that directionality is present in the structure, the classic approach is to use grids to measure selectively in different directions. For example, in the case shown in **Figure 28**, the number of intercepts per unit line length (which measures the surface area per unit volume) can be measured in different directions by using a rotated grid of parallel lines. Since each set of lines measures the projected surface area normal to the line direction, the result shows the anisotropy of surface orientation. Plotting the reciprocal of number of intersections/line length (called the intercept length) as a function of orientation produces a polar (or rose) plot as shown in the figure, which characterizes the mean shape of the grains or cells.

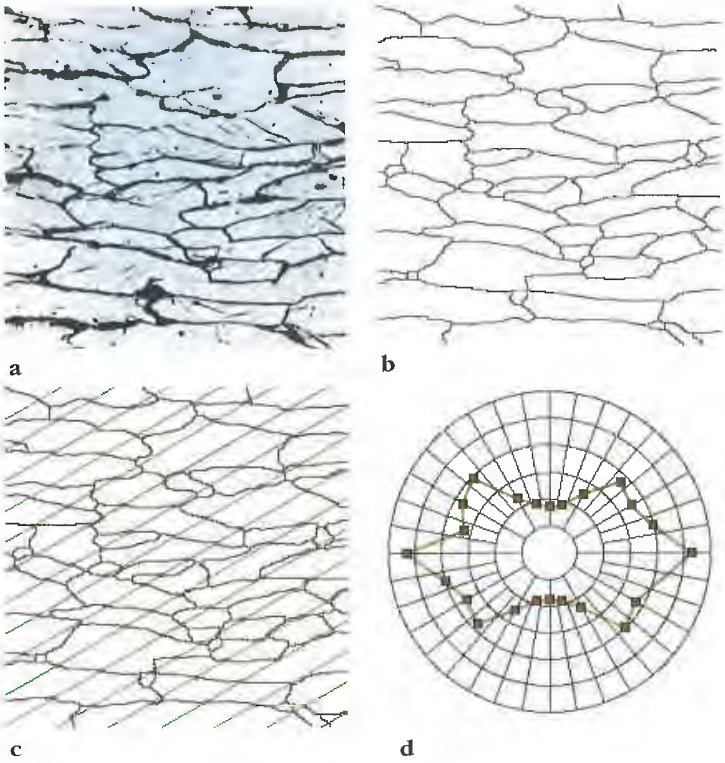


Figure 28. Characterizing anisotropy: ***(a)*** *an anisotropic metal grain structure;* ***(b)*** *the idealized grain boundaries produced by thresholding and skeletonization;* ***(c)*** *measuring the number of intercepts per unit line length in different orientations by rotating a grid of parallel lines;* ***(d)*** *the results shown as a polar plot of mean intercept length.*

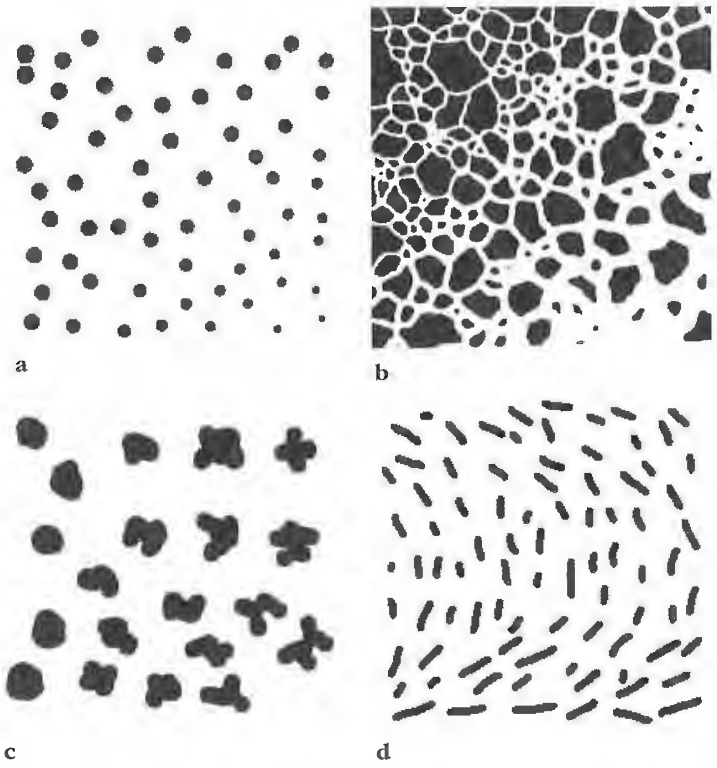


Figure 29. Schematic examples of gradients in:
(a) *size;*
(b) *area fraction;*
(c) *shape;*
(d) *orientation.*

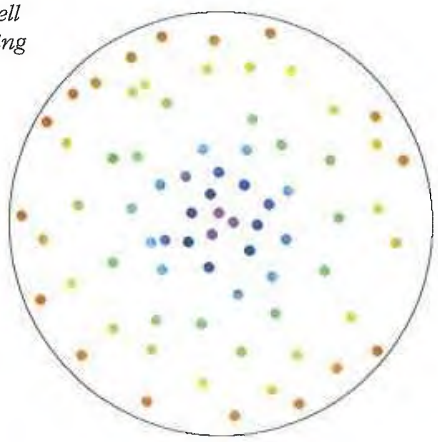
Gradients can sometimes be observed by comparing images from different parts of a specimen, but because visual judgment of metric properties such as length of boundary lines and area fraction of a phase is not very reliable it is usually safest to actually perform measurements and compare the data statistically to find variations. The major problem with characterizing gradients is that they are rarely simple. An apparent change in the number of features per unit area or volume may hide a more significant change in feature size or shape, for example. **Figure 29** shows some examples in which gradients in size, number, shape and orientation are present. A secondary problem is that once the suspected gradient has been identified, a great many samples and fields of view may be required to obtain adequate statistical precision to properly characterize it.

Figure 30 shows a simple example of the distribution of cell colonies in a petri dish. Are they preferentially clustered near the center, or uniformly distributed across the area? By using the method shown in Chapter 7 to assign a value from the Euclidean distance map (of the area inside the petri dish) to each feature, and then plotting the number of features as a function of that value, the plot of count vs. radial distance shown in **Figure 31** is obtained. This would appear to show a decrease in the number of features toward the center of the dish; but this is does not correctly represent the actual situation, because it is not the number but the number per unit area that is important, and there is much more area near the periphery of the dish than near the center. If the distribution of number of features as a function of radius is divided by the number of pixels as a function of radius (which is simply the histogram of the Euclidean distance map), the true variation of number per unit area is obtained as shown in the figure.

Size distributions

Much of the contemporary effort in stereological development and research has been concerned with improved sampling strategies to eliminate bias in cases of non-IUR specimens. There have also

Figure 30. *Distribution of features within an area (e.g., cell colonies in a petri dish), with each one labeled according to distance from the periphery.*



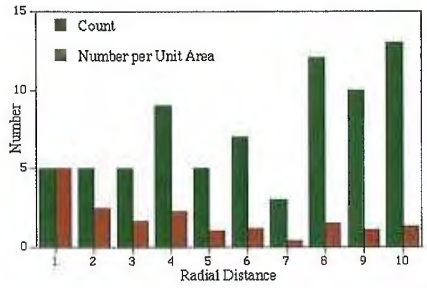


Figure 31. *Data from* ***Figure 30****. Plot of number and number per unit area as a function of distance.*

been interesting developments in so-called second order stereology that combine results from different types of measurements, such as using two different types of grids to probe a structure or two different and independent measurements. One example is to use a point grid to select features for measurement in proportion to their volume (points are more likely to hit large than small particles). For each particle that is thus selected, a line is drawn (in a random direction) through the selection point to measure the radius from that point to the boundary of the particle.

Figure 32 shows this procedure applied to the white dendritic phase in the image used earlier. The grid of points selects locations. For each point that falls onto the phase of interest, a radial line in uniformly randomized directions is drawn to the feature boundary. For convex shapes, the length of this line may consist of multiple segments which must be added together. For isotropic structures or isotropic section planes, the line angles uniformly sample directions. For vertical sections through potentially anisotropic structures, a set of sine-weighted angles should be used.

The length of each line gives an estimate of particle volume:

$$V = \tfrac{4}{3}\pi r^3 \tag{11}$$

which is independent of particle shape. Averaging this measurement over a surprisingly small number of particles and radius measurements gives an estimate of the volume-weighted mean volume (in other words, the volume-weighted average counts large particles more often than small ones, in proportion to their volume).

A more common way to define a mean volume is on a number-weighted basis, where each particle counts equally. This can be determined by dividing the total volume of the phase by the number of particles. We have already seen how to determine the total volume fraction using a point

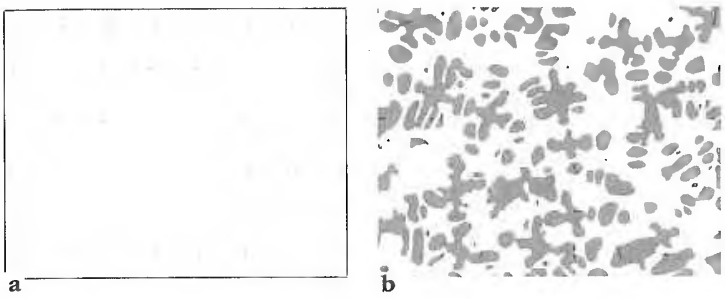


Figure 32. Measuring point-sampled intercepts: *(a) an array of grid points, each having a direction associated with it; (b) applying the grid to the image from* ***Figure 2*** *and measuring the radial distance from those grid points that fall on the phase of interest to the boundary in the chosen direction. Note that, for convex shapes, the radial distance is the sum of all portions of the intercept line.*

count, and the number of particles per unit volume using the disector. Dividing the total by the number gives the number-weighted mean volume.

Once both the mean volumes have been determined, they can be combined to determine the standard deviation of the distribution of particle sizes, independent of particle shape, since for the conventional number-weighted distribution the variance is given by the difference between the square of the volume-weighted mean volume and the square of the number-weighted mean volume

$$\sigma^2 = V_V^2 - V_N^2 \tag{12}$$

Although this may appear to be a rather esoteric measurement, in fact it gives an important characterization of the size distribution of particles or other features that may be present in a solid, which as we have seen are not directly measurable from the size distribution of their intersections with a plane section.

Classical stereology (unfolding)

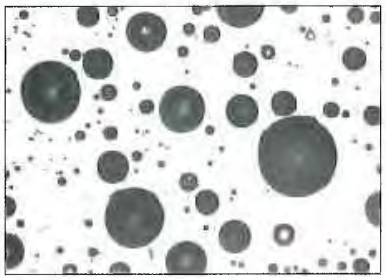
Historically, much of classical stereology was concerned with exactly that problem, determining the size distribution of three-dimensional features when only two-dimensional sections through them were measurable (Weibel, 1979; Cruz-Orive, 1976, 1983). The classical method used was to assume a shape for the particles (i.e., that all of the particles were the same shape, which did not vary with size). The most convenient shape to assume is (of course) a sphere. The distribution of the sizes of circles produced by random sectioning of a sphere can be calculated as shown in **Figure 33**. Knowing the sizes of circles that should be produced if all of the spheres were of the same size makes it possible to calculate a matrix of α values that enable solving for the distribution of sphere sizes that must have been present in the three-dimensional structure to produce the observed distribution of circle sizes, as shown in **Figure 34**.

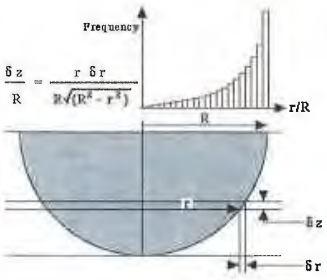
$$N_{V_{sphere,i}} = \sum \alpha_{i,j} \cdot N_{A_{circle,j}} \tag{13}$$

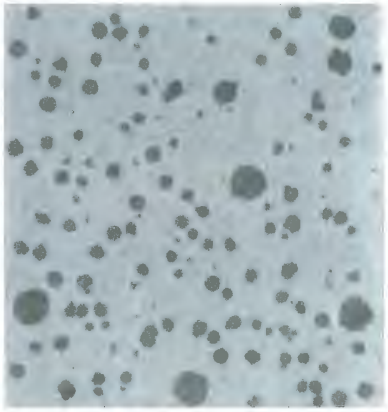
Other shapes have also been proposed, as a compromise between mathematical ease of calculation and possible realistic modeling of real world shapes. These have included ellipsoids of revolution (both oblate and prolate), disks, cylinders, and a wide variety of polyhedra, some of which can be assembled to fill space (as real cells and grains do). Tables of alpha coefficients for all



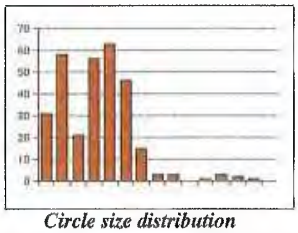
Figure 33. *Sectioning a sphere at random produces a distribution of circle sizes.*



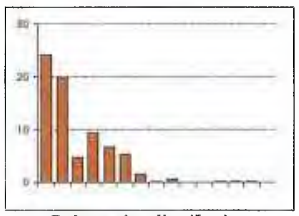




Graphite nodules in cast iron



Circle size distribution



Sphere size distribution

Figure 34. *Unfolding the size distribution of spheres that must have been present to produce an observed distribution of circle sizes. The sample is a cast iron containing graphite nodules; note that the features are not perfectly circular, even though a sphere model has been used in the unfolding.*

these exist and can be rather straightforwardly applied to a measured distribution of the size of two-dimensional features viewed on the section plane.

However, there are several critical drawbacks to this approach. First, the ideal shapes do not usually exist. Nuclei are not ideal spheres, cells and grains are not all the same polyhedral shape (in fact, the larger ones usually have more faces and edges than smaller ones), precipitate particles in metals are rarely disks or rods, etc. Knowledge of the shape of the three-dimensional object is critical to the success of the unfolding method, since the distribution of intersection sizes varies considerably with shape (**Figure 35** shows an example of cubes vs. spheres). It is very common to find that size varies with shape, which strongly biases the results. Small errors in the shape assumption produce very large errors in the results. For example, the presence of a few irregularities or protrusions on spheres results in many smaller intersection features, which are mis-interpreted as many small spherical particles.

The second problem is that from a mathematical point of view, the unfolding calculation is ill-posed and unstable. Variations in the distribution of the number of two-dimensional features as a function of size result from counting statistics. The variations are typically greatest at the end of the

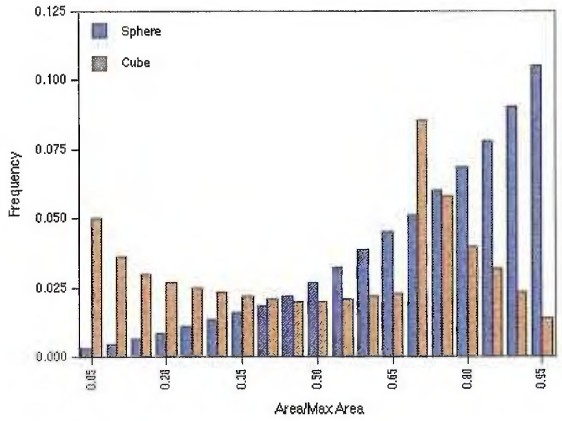


Figure 35. *Size distribution of intercept areas produced by random sectioning of a sphere and a cube.*

distribution, for the few largest features which are rarely encountered. But in the unfolding process these errors are magnified and propagated down to smaller sizes, often resulting in negative numbers of features in some size classes, and greatly amplifying the statistical uncertainty.

For these reasons, although unfolding was a widely used classical stereological method for decades, it has fallen out of favor in the last 15–20 years. Instead there has been increased emphasis on finding unbiased ways to measure structural parameters using more robust tools. Nevertheless, in spite of these limitations (or perhaps because their magnitude and effect are not sufficiently appreciated), unfolding of size distributions is commonly performed. When the shape of the three-dimensional objects is not known, it is common to assume they are simple spheres, even when the observed features are clearly not circles. The argument that this gives some data that can be compared from one sample to another, even if it is not really accurate, is actually quite wrong and simply betrays the general level of ignorance about the rules and application of stereological procedures.

9

Feature-Specific Measurements

Measurements that can be performed on each of the individual features in images can be grouped into four classes: brightness (including color values and parameters such as density that are related to brightness), location (both absolute position and relative to other features present), size, and shape. For each class, quite a variety of different specific measurements can be made, and there are a variety of different ways to perform the operations. Most image analysis systems offer at least a few measures in each class. Users find themselves at some time or another having to deal with several different measurement parameters. The problem is frequently to decide which of the measured parameters is most useful or appropriate for solving a particular problem.

Most of these techniques produce a numeric output suitable for statistical analysis or presentation graphics. Frequently, the interpretation of the data is left to a separate program, either a simple spreadsheet relying on the user's programming ability, or a dedicated statistics package. In a few cases, the numbers are converted to go/no go decisions or become the basis for classification, discussed in the next chapter. Examples might include quality control testing of the size and placement of holes in a part, medical pathology decisions based on the identification of the presence of cancerous cells, or recognition of different objects in the field of view.

Brightness measurements

Normally, in the kinds of images we have been discussing here, each pixel records a numeric value that is often the brightness of the corresponding point in the original scene. Several such values can be combined to represent color information. The most typical range of brightness values is from 0 to 255 (8-bit range), but depending on the type of camera, scanner or other acquisition device a large range of 10 or more bits, up to perhaps 16 (0 to 65,535) may be encountered. Most programs that accept images from sources such as cooled cameras and scanners having more than 8 bits of depth store them as 16 bit images (occupying two bytes per pixel for grey-scale images, or 6 bytes per pixel of red, green, and blue [RGB] values for color images), even if the actual grey-scale or tonal resolution is less than 16 bits. Some systems use the 0–255 value range for all bit depths but report decimal values for images having more than 8 bits of depth. This has the advantage of permitting easy comparisons between images acquired with different depths. Rarely, the stored values may be real numbers rather than integers (for instance, elevation data). In most

cases, however, these images are still stored with a set of discrete integer "grey" values because it is easier to manipulate such arrays and convert them to displays. In such cases, a calibration table or function is maintained to convert the pixel values to meaningful real numbers when needed.

The process of creating such a calibration function for a particular imaging device, or indeed for a particular image, is far from trivial (Inoue, 1986; Chieco et al., 1994; Swing, 1997; Boddeke, 1998). Many of the cameras and other devices that have been mentioned for acquiring images are neither perfectly linear nor exactly logarithmic, nor completely consistent in the relationship between the pixel's numeric value and the input signal (e.g., photon intensity). Many video cameras have an output function that varies with the overall illumination level. The presence of automatic gain circuits, or user-adjustable gamma controls, make it more difficult to establish and maintain any kind of calibration. In general, any kind of automatic gain or dark level circuitry, automatic color balancing, etc., will serve to frustrate calibration of the camera. With consumer-level video and still cameras, it is not always possible to turn such "features" off.

For color imaging, cameras may incorporate automatic white balance adjustments (which are intended for a type of scene much different than the majority of scientific images). Control of the color temperature of the light source and maintaining consistency of the camera response to perform meaningful color measurement is rarely possible with video type cameras. Chapter 1 showed examples of correcting color values using measured RGB intensities from a standard color chart. It is important to note that the typical color camera has RGB sensors that respond to a relatively broad range of wavelengths, and that many different combinations of actual photon colors can produce identical stored RGB values. In general, it is not practical to attempt to perform colorimetry (the detailed measurement of color) using such cameras, and of course complications arising from light sources, viewing geometry, optics, etc., make the problem even more difficult.

Even when the use of a stable light source and consistent camera settings can be assured, the problem of grey scale calibration remains. Some standards are reasonably accessible, for example, density standards in the form of a step-wedge of film. Measurement of the brightness of regions in such a standard can be performed as often as needed to keep a system in calibration. In scanning large area samples such as electrophoresis gels or x-ray films, it is practical to incorporate some density standards into every sample so that calibration can be performed directly. In other cases, separate standard samples can be introduced periodically to check calibration.

Optical density is defined as

$$O.D. = -\log_{10}\left(\frac{I}{I_0}\right) \tag{1}$$

where I/I_0 is the fraction of the incident light that penetrates through the sample without being absorbed or scattered. If a camera or scanner and its light source are carefully adjusted so that the full range of linear brightness values covers the range of optical density from 0.1 (a typical value for the fog level of unexposed film) to 2.5 (a moderately dense exposed film), the resulting calibration of brightness vs. optical density would be as shown in **Figure 1**. The shape of the curve is logarithmic.

At relatively low-density/bright pixel values the sensitivity is quite good. A difference of 1 pixel brightness value at the bright end of the scale is only about 0.002. At the high-density/dark-pixel end of the scale, however, a difference of one pixel brightness value corresponds to an optical density change of 0.3, which is very large. This is an indication that when trying to apply a linear camera or scanner to optical density reading, more than 8 bits of grey scale are needed. An input device with more precision can be used with a lookup table that converts the values to the logarithmic optical density scale and then stores that in an 8-bit image. This problem becomes more severe as higher density values are encountered. Films containing high amounts of silver can easily reach densities of 3.5. Satisfactorily digitized such films requires using scanners with at least 12, and preferably 14 bits per channel precision.

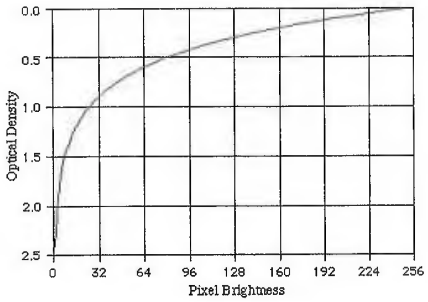
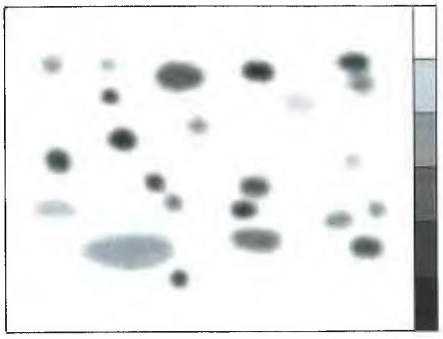


Figure 1. *Calibration of optical density vs. pixel brightness if the latter is adjusted linearly to span the range from 0.1 (fog level) to 2.5 (typical exposed film).*

It is not uncommon to include a grey wedge in a film to be scanned for one- or two-dimensional separation of organic molecules. **Figure 2** shows an example. The grey wedge of known density values allows the construction of a calibration scale (**Figure 3**) that can then be used to measure the optical density of each object (**Figure 4**). In this application, the total amount of protein in each spot is proportional to the integrated optical density, which is the product of the average density and the area.

Figure 2. *Scanned image of a 2D electrophoresis gel with a grey-scale calibration wedge.*



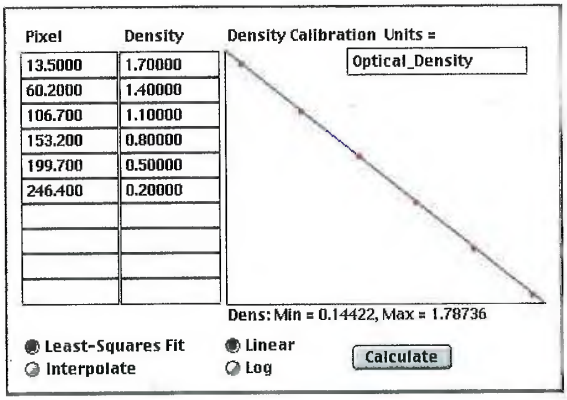
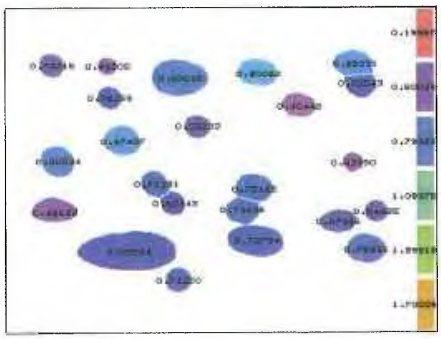


Figure 3. *Calibration plot from the grey-scale wedge in **Figure 2**, showing the relationship between optical density and pixel brightness for this particular scanner.*

Figure 4. *Measured data from **Figure 2**, using the calibration plot from **Figure 3** to compute the integrated optical density of each spot.*



The use of a calibration scale raises another point. It is common for features to contain some variation in pixel brightness. The average density (or whatever other quantity is calibrated against pixel brightness) is not usually recorded linearly. It thus becomes important to take this into account when determining an average value for the density, or an integrated total dose, or any other calibrated quantity. If all of the pixels within the feature are simply averaged in brightness, and then that value is converted using the calibration scale, the wrong answer will be obtained. It is instead necessary to convert the value from each pixel to the calibrated values, and then sum or average those. It is also important not to include adjacent background pixels that are not part of the feature, which means that the delineation should be exact and not some arbitrary shape (a circle or square) drawn around the features. Reliance on manual outlining in not usually acceptable, since humans usually draw outlines larger than the boundaries around features. The solution is to use the same routines that threshold and segment the image into discrete features. The pixels in that binary image define the features and can be used as a mask to locate the corresponding pixels in the grey-scale array.

For one-dimensional measurements of density, the procedure is the same. **Figure 5** shows an example of a film exposed in a Debye–Scherer x-ray camera. The total density of each line, and its position, are used to measure the crystal structure of materials. Knowing that the lines extend vertically, it is reasonable to average the values in the vertical direction to reduce noise and obtain a more precise plot of the density variations along the film (**Figure 6**). Converting the individual pixel values to density before summing them is the correct procedure that gives the proper values for the relative line intensities. The linear plots of intensity values may be subtracted to find differences or used with statistical routines to detect peaks.

Figure 7 shows a similar procedure for tracks or columns in an electrophoresis separation. Using the brightness profile measured between the tracks allows a simple correction to be made for background variations due to nonuniformities in thickness or illumination. The brightness values

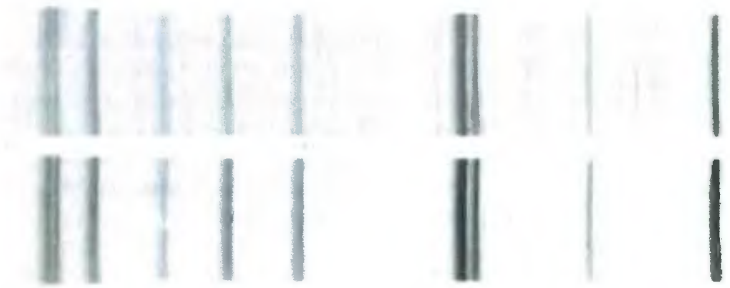


Figure 5. *Two Debye–Scherer x-ray films. The vertical lines occur at angles that are related to the atomic spacing in the lattice structure of the target material. Small differences in intensity and the presence of additional lines indicate the presence of trace amounts of additional compounds.*

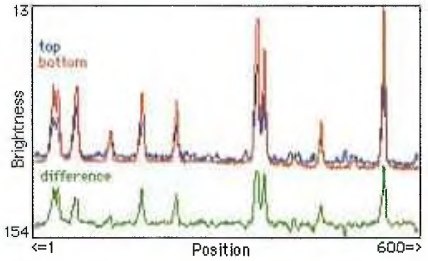


Figure 6. *Integrated intensity plots along the films from* ***Figure 5****, with the difference between them.*

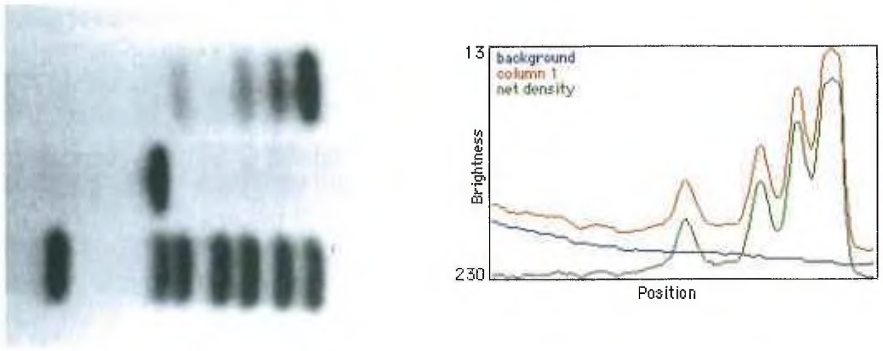


Figure 7. *A 1D electrophoresis separation. Intensity scans along the center of each track are leveled by subtracting scans between the tracks (the example shows the top track).*

are averaged across the center of the columns, avoiding the edges. Calibration of the position of bands within the columns is usually done with respect to markers of known molecular weight that are placed in adjacent columns. This is necessary because the distance scale is generally not very linear or constant in such specimens.

Of course, not all images have this convenient relationship between intensity and a property of the sample being imaged, such as density. In microscope images, phase differences can also produce contrast, and diffraction effects or polarization effects may also be present. In real-world images, the surface orientation and characteristics of objects influence the brightness, as does the interplay between the light source color and the surface color. In most of these cases, the brightness cannot be measured to provide information about the sample. It is difficult enough to use the relative brightness values to delineate regions that are to be grouped together or separated from each other.

When color images are digitized, the problem of having enough bits to give enough precision for each of the color planes is amplified. Many of the higher-end scanners acquire more than 8 bits, often as many as 12, for each of the RGB planes, use these to determine the color information, and then may create an optimized 8-bit representation to send to the computer. Even with this increase in precision, it is difficult to achieve accurate color representation. As pointed out in Chapter 1, the digitization of color requires far greater control of camera settings and lighting than can normally be achieved with a video camera. Even a flat bed scanner, which offers greater consistency of illumination and typically uses a single charge coupled device (CCD) linear array with various filters to obtain the RGB information, is not a good choice for measurement of actual color information, however, such scanners can be calibrated to reproduce colors adequately on particular printers.

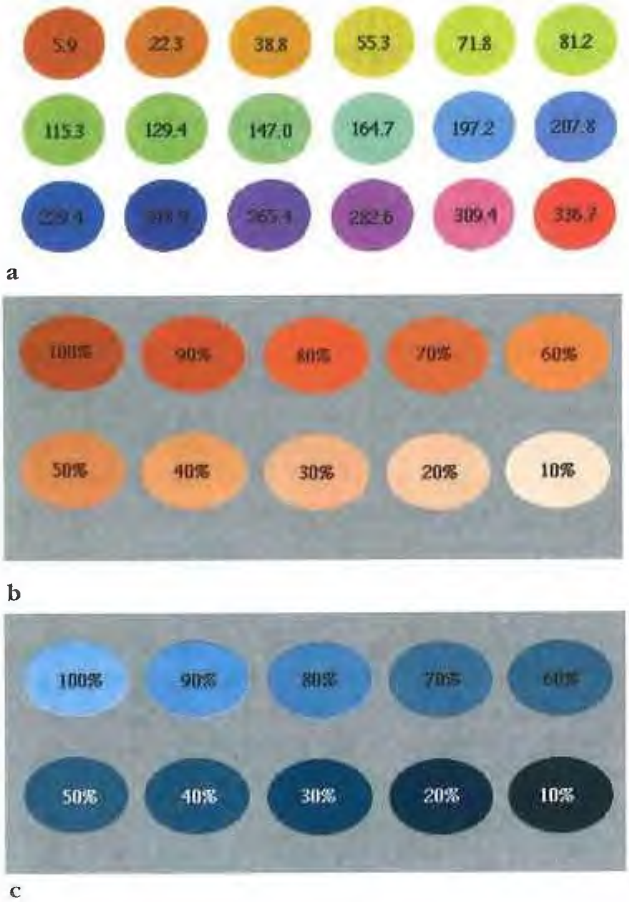
The pixel brightness value need not be optical density or color component information, of course. Images are so useful to communicate information to humans that they are used for all kinds of data. Even within the most conventional expression of the idea of imaging, pixel values can be related to the concentration of dyes and stains introduced into a specimen. In an x-ray image from the scanning electron microscope (SEM), the brightness values are approximately proportional to elemental concentration. These relationships are not necessarily linear nor easy to calibrate. x-ray emission intensities are affected by the presence of other elements. Fluorescence intensities depend not only on staining techniques and tissue characteristics, but also on time, since bleaching is a common phenomenon.

In infrared imaging, brightness is a measure of temperature. Backscattered electron images from the scanning electron microscope have brightness values that increase with the average atomic

number, so that they can be used to determine chemical composition of small regions on the sample. In a range image, the pixel brightness values represent the elevation of points on the surface and can be calibrated in appropriate units.

To support this range of applications, it is generally useful to be able to measure the mean intensity or a calibrated "density" value, as well as finding the brightest or darkest pixel values in each region or feature, and perhaps the standard deviation of the brightness values as a measure of variation or texture. For color images, it may be useful to report the RGB components (usually as values ranging from 0 to 255), but in most cases the hue, saturation and intensity are more directly useful. The saturation and intensity are generally measured on either the 0–255 scale or reported as percentages, but the hue may be reported as an angle from 0 to 360°, as is shown in **Figure 8**. Note that several different hue, saturation and intensity (HIS) spaces are shown, as discussed in Chapter 1, and it is important to know which one is being used to convert the stored RGB pixel values, particularly for the saturation values.

These examples are at best a tiny sample of the possible uses of pixel values, but the measurement of the stored values and conversion to some calibrated scale is a broadly useful technique. Statistical analysis of the data provides mean values and standard deviations, trends with position, comparisons between locations, within or between images, and so forth. For such procedures to work, it is important to establish useful calibration curves, which requires standards or fundamental knowledge and is a subject beyond the scope of this text.



a

b

c

Figure 8. Mean color values for a series of color spots:
(a) *Hue, reported as angles from 0 to 360°, for features with full saturation;* ***(b)*** *saturation, reported as percent, for features with constant hue and intensity;* ***(c)*** *intensity, reported as percent, for features with constant hue and saturation.*

Determining location

In several of the examples for measuring brightness values, the location of features was also needed for interpretation of the results. For a typical irregular feature extending over several pixels, there can be several different definitions of location, some easier to calculate than others. For instance, the *x,y* coordinates of the midpoint of a feature may be determined simply as halfway between the minimum and maximum limits of the pixels comprising the feature. Normally, the pixel addresses themselves are just integer counts of position, most often starting from the top left corner of the array. This convention arises from the way that most computer displays work, using a raster scan from the top left corner. There may be some global coordinate system of which the individual image is just a part, and these values may also be integers, or real number values that calibrate the pixel dimensions to some real world units such as latitude and longitude, or millimeters from the edge of a microscope sample.

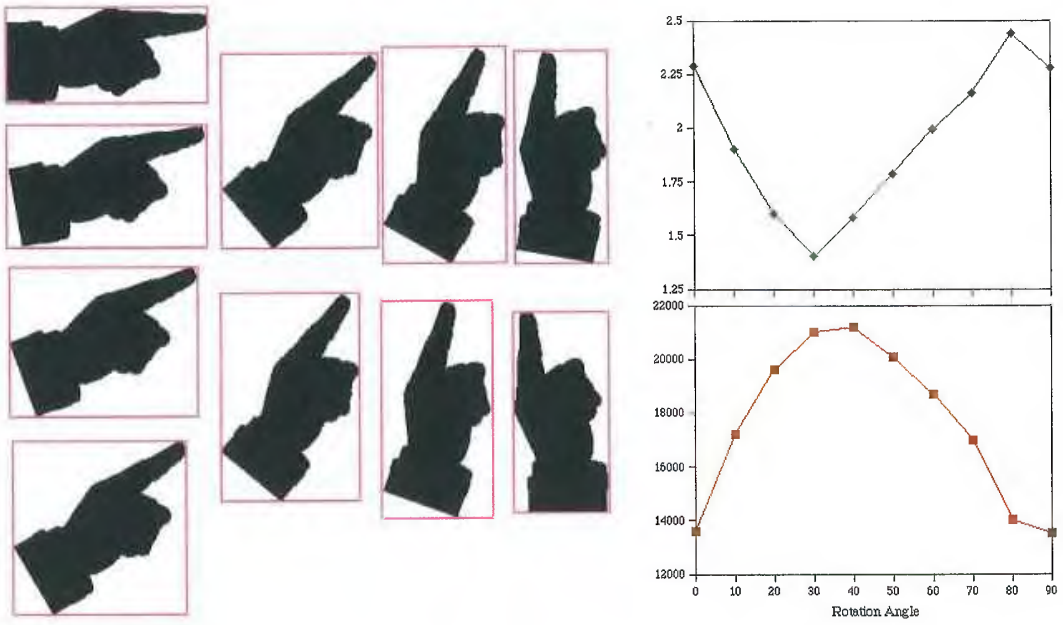
Establishing a real-world calibration for pixel dimensions is often quite difficult, and maintaining them while shifting the specimen or the camera (by moving the microscope stage or the satellite, for instance) presents a variety of challenges. Often, locating a few known features in the image itself can serve to establish fiduciary marks that allow this sort of calibration, so that all of the other features in the image can be accurately located. Shifting the sample, stage or camera so that successive viewing frames overlap allows position calibration to be transferred from one field to the next, but the errors are cumulative and may grow rapidly due to the difficulty of accurately locating the same feature(s) in each field to a fraction of a pixel. This also assumes that there are no distortions or scan nonlinearities in the images. Relying on the stage motion for position information depends upon the precision of that mechanism (which may be much worse if directions are reversed due to backlash), and the fidelity with which the sample motion corresponds to that of the stage.

The feature's minimum and maximum limits are easy to determine by finding the pixels with the largest and smallest coordinates in the horizontal and vertical directions. These limiting coordinates define a bounding rectangle around the feature, and the midpoint of the box is then taken as a location for the feature. The midpoint is not usually the preferred representation of location, however, because it is too easily biased by just a few pixels (for instance a whisker sticking out from the rest of the feature). One application in which these box coordinates are used, however, is in computer drawing programs. Many such programs allow the user to select a number of drawn objects and then move them into alignment automatically. The options are typically to align the objects vertically by their top, center, or bottom, and horizontally by their left, center, or right edges. These are exactly the box coordinates and midpoint.

The center of the so-called bounding box is useful in a few situations such as the gel scan, because the *x* and *y* coordinates have meaning in terms of how the sample was created. For most real images, this is not the case. When the *x-y* axes are arbitrary, the bounding box is very biased and sensitive to object orientation, as shown in **Figure 9**. More representative of the geometric center of a feature is the location of the center of a bounding or circumscribed circle. Fitting such a circle to the points on the object periphery is straightforward, and made more efficient by using only those points that represent the vertices of a bounding polygon, fitted as described later in this chapter by rotating axes to a limited number of directions (typically about every 10°) and finding the bounding box limits for each orientation. Of course, the size and center location of the bounding circle is not sensitive to changes in the orientation of the axes or the feature.

For a non-convex feature or one containing internal holes, the geometric center may not even lie within the feature bounds. Nor is it sensitive to the actual feature shape. For irregularly shaped features, it is usually preferable to take into account the feature shape and location of all the pixels present. This approach defines the centroid of the feature, a unique *x, y* point that would serve to

Figure 9. *Rotation of a figure and the changes in the size and shape of the bounding box.*



balance the feature on a pinpoint if it were cut out of a rigid, uniform sheet of cardboard. The coordinates of this point can be determined by averaging the coordinates of each pixel in the object.

$$C.G._x = \frac{\sum_i x_i}{Area} \qquad C.G._y = \frac{\sum_i y_i}{Area} \tag{2}$$

where the *Area* is just the total number of pixels present. Notice that this equation provides a set of coordinates that are not, in general, integers. The center of gravity or centroid of an object can be determined to sub-pixel accuracy, which can be very important for locating features accurately in a scene. **Figure 10** compares the geometric center and the centroid for an irregular feature.

If the centroid is calculated according to **Equation 2** using only the boundary pixels, as is sometimes done (especially if features are encoded with boundary representation as described in Chapter 7), the result is quite wrong. The calculated point will be biased toward whichever part of the boundary is most complex and contains the most pixels (**Figure 11**). This bias can even vary with the orientation of the boundary with respect to the pixel array because square pixels are larger in the diagonal direction than in their horizontal and vertical dimension.

The centroid location can be calculated correctly from a boundary representation such as chain code. The correct calculation uses the pairs of coordinates x_i, y_i for each point in the boundary,



Figure 10. *Centroid (green), density weighted centroid (red) using the pixel density values, and geometric center (blue) determined as the center of the bounding circle, for an irregular feature.*



Figure 11. *Effect of boundary irregularities on centroid determination. The right hand side of the circle is irregular while the left hand side is smooth. The centroid location determined by using just the boundary pixels (magenta) is shifted to the right, as compared to the location determined using all of the pixels in the feature.*

where x_0, y_0, x_n, and y_n are the same point (i.e., the boundary representation is a closed loop with the two ends at the same place).

$$C.G._x = \frac{\sum_i (x_i + x_{i-1})^2 \cdot (y_i - y_{i-1})}{Area} \qquad (3)$$

$$C.G._y = \frac{\sum_i (y_i + y_{i-1})^2 \cdot (x_i - x_{i-1})}{Area}$$

and it is now necessary to calculate the area as

$$Area = \frac{\sum_i (x_i + x_{i-1}) \cdot (y_i - y_{i-1})}{2} \qquad (4)$$

Parenthetically, it is worth noting here that some of the attractiveness of chain code or boundary representation as a compact way to describe a feature is lost when you try to use that data to calculate things like the area or centroid of the feature.

The definition of centroid or center of gravity just given treats each pixel within the feature equally. For some purposes, the pixel brightness, or a value calculated from it using a calibration curve, makes some pixels more important than others. For example, the accurate location of the spots and lines in the densitometric examples shown earlier would benefit from this kind of weighting. That modification is quite easy to introduce by including the brightness-derived value in the summations in **Equation 2**. **Figure 10** shows the density-weighted centroid in comparison to the conventional centroid in which all pixels are treated as equal.

$$C.G._x = \frac{\sum_i Value_i \cdot x_i}{\sum_i Value_i}$$

$$C.G._y = \frac{\sum_i Value_i \cdot y_i}{\sum_i Value_i} \qquad (5)$$

The denominator is now the integrated density (or whatever the parameter related to brightness may be). Of course, this kind of calculation requires access to the individual pixel brightness values, and so cannot be used with a boundary representation of the feature.

For a convex shape, the centroid may not lie within the bounds of the feature. When a unique representative point that always lies within the bounds of a potentially irregular shape is required, the ultimate eroded point or (UEP) determined from the EDM is the center of the largest inscribed circle in the feature and can often be used.

Orientation

Closely related to the location of the centroid of a feature is the idea of determining its orientation. There are a number of different parameters that are used, including the orientation of the longest dimension in the feature (the line between the two points on the periphery that are farthest apart, also known as the maximum Feret's diameter), and the orientation of the major axis of an ellipse fitted to the feature boundary. Just as the centroid is a more robust descriptor of the feature's location than is the midpoint, however, an orientation defined by all of the pixels in the image is often better than any of these. This is due to the fact that it is less influenced by the presence or absence of a single pixel around the periphery, where accidents of acquisition or noise may make slight alterations in the boundary.

The moment axis of a feature is the line around which the feature, if it were cut from rigid, uniform cardboard, would have the lowest moment of rotation. It can also be described as the axis which best fits all of the pixels in the sense that the sum of the squares of their individual distances from the axis is minimized. This is the same criterion used to fit lines to data points when constructing graphs. Determining this axis and its orientation angle is straightforward, and just involves summing pixel coordinates and the products of pixel coordinates for all of the pixels in the image. As for the example in **Equation 5**, it is possible to weight each pixel with some value such as the density, instead of letting each one vote equally. The most convenient procedure for the calculation is to add up a set of summations as listed in **Equation 6**.

$$
\begin{aligned}
S_x &= \sum x_i \\
S_y &= \sum y_i \\
S_{xx} &= \sum x_i^2 \\
S_{yy} &= \sum y_i^2 \\
S_{xy} &= \sum x_i y_i
\end{aligned}
\tag{6}
$$

Once these sums have been accumulated for the feature, the net moments about the x and y axes, and the angle of the minimum moment are calculated as shown in **Equation 7**.

$$
\begin{aligned}
M_x &= S_{xx} - \frac{S_x^2}{Area} \\
M_y &= S_{yy} - \frac{S_y^2}{Area} \\
M_{xy} &= S_{xy} - \frac{S_x \cdot S_y}{Area} \\
\Theta &= \tan^{-1}\left\{\frac{M_{xx} - M_{yy} + \sqrt{(M_{xx} - M_{yy})^2 + 4 \cdot M_{xy}^2}}{2 \cdot M_{xy}}\right\}
\end{aligned}
\tag{7}
$$

Figure 12 shows an example of the measurement of orientation angle on features. Notice that features that intersect the edges of the image field are not measured as will be discussed in the next section. The measurement can also be applied to lines, as shown in **Figure 13**. In this case the axons were thresholded and skeletonized, and the nodes removed leaving line segments. To characterize the distribution of orientations for the axons, it is more meaningful to plot the total length of segments as a function of angle rather than the number. **Figure 14** shows an example combining position and angle. The orientation of the cells in the tissue varies with vertical position. This is an example of a gradient as discussed in Chapter 8.

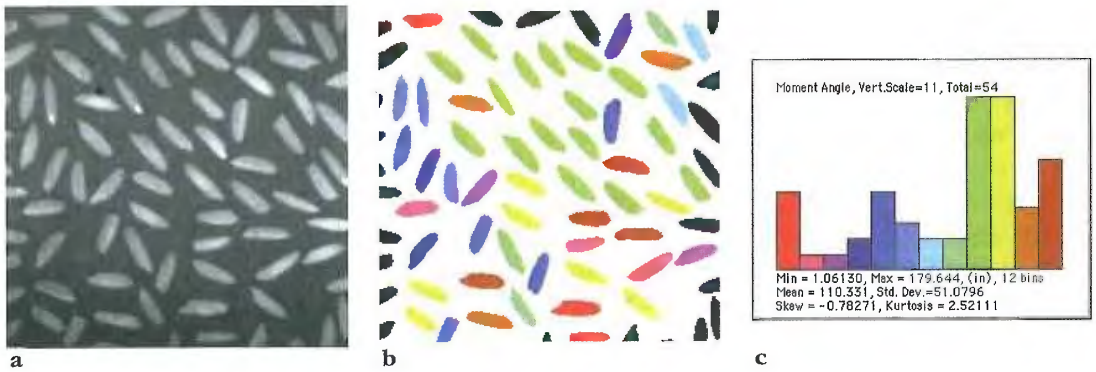


Figure 12. Measurement of feature orientation: ***(a)*** *rice grains;* ***(b)*** *color coded according to angle;* ***(c)*** *distribution of grains according to angle.*

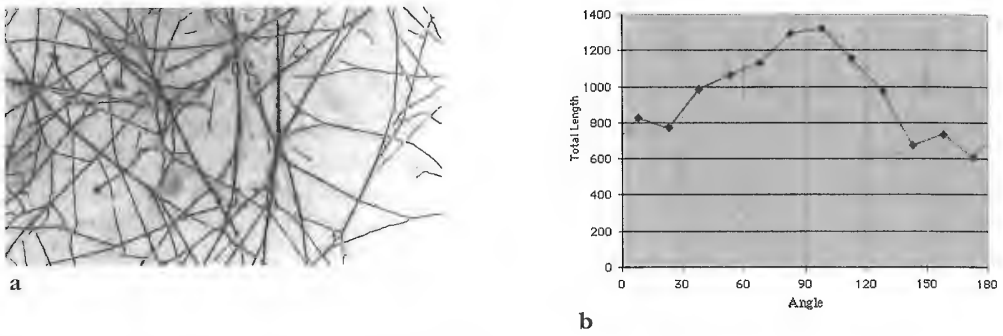


Figure 13. Measurement of fiber orientation:
(a) *image of axons, with skeleton segments superimposed (nodes removed as described in Chapter 7);*
(b) *cumulative length of fibers as a function of their angles, showing preferred orientation.*

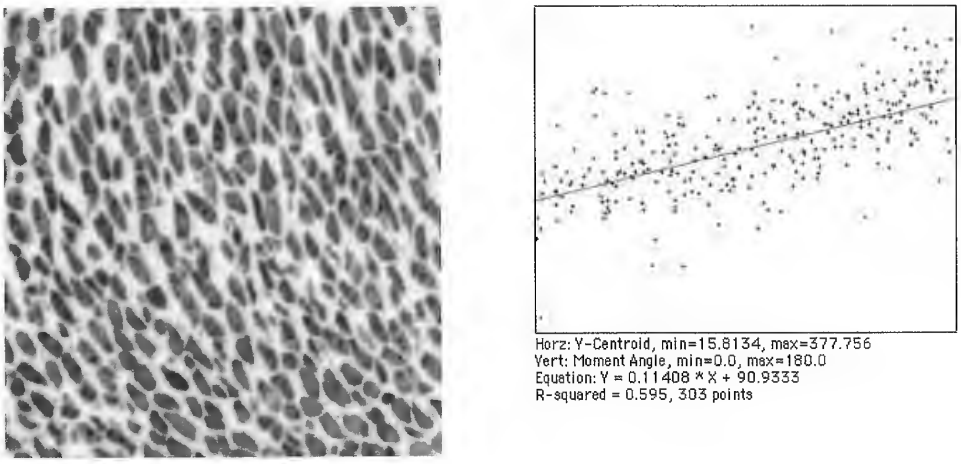
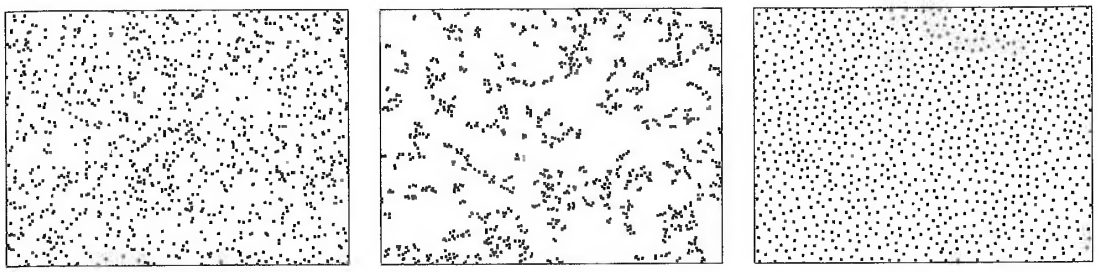


Figure 14. *Cells in tissue, showing a gradient of moment angle with vertical position.*

Neighbor relationships

The location of individual features may be less important in some applications than the relationships between neighboring features. For instance, **Figure 15** shows several distributions of features. How can such distributions be compactly described to reveal the extent to which they are random, regularly spaced, or clustered?

Schwarz and Exner (1983) showed that a histogram of the distribution of the distances between nearest neighbors can provide an answer. Actually, the distance between any pair of neighbors, second nearest, etc., can be used as well, but in most cases the nearest-neighbor pairs are the easiest to identify. Once the coordinates of the centroid points representing each feature have been determined, sorting through the resulting table to locate the nearest neighbor for each point is a straightforward task (best left to the computer). The straight line distances between these points are calculated and used to construct the histogram. This in turn can be characterized by the mean and variance (or standard deviation) of the distribution. A word of caution is needed in dealing with feature points located adjacent to the edge of the field of view (Reed and Howard, 1997): if the distance to the edge is less than the distance found to the nearest neighbor within the field of view, the distance should not be used in the distribution, because it may cause bias. It is possible that



a b c

Figure 15. Feature distributions illustrating:
(a) *random;*
(b) *clustered;*
(c) *spaced or self-avoiding arrangements.*

another feature outside the field of view would actually be closer. For large images (fields of view) containing many features, this problem is only a minor concern. The nearest-neighbor distance method also generalizes to three dimensions, in the case of three-dimensional (3D) imaging; the method has been used with a confocal microscope (Baddeley et al., 1987; Russ et al., 1989; Reed et al., 1997). A related statistical test on all neighbor pairs can also be used (Shapiro et al., 1985).

Consider now the particular distributions shown in **Figure 15**. The image in **Figure 15a** is actually a random distribution of points, often called a Poisson random distribution because the histogram of nearest-neighbor distances is in fact a Poisson distribution. This is the sort of point distribution you might observe if you just sprinkled salt on the table. Each point is entirely independent of the others, thus "random." For such a distribution, the mean distance between nearest neighbors is just

$$Mean = \frac{0.5}{\sqrt{\frac{N}{Area}}} \tag{8}$$

where N is the number of points within the area of the field of view. For a Poisson distribution, the variance is equal to the mean (the standard deviation is equal to the square root of the mean). The consequence is that for a random distribution of points, the number of points per unit area of the surface is all that is needed to determine the mean and variance of the histogram of nearest-neighbor distances.

When clustering is present in the point distribution, most points have at least one neighbor that is quite close by. Consequently, the mean nearest-neighbor distance is greatly reduced. In most cases, the variance also becomes less, as a measure of the uniformity of the spacing between the clustered points. As shown in the image in **Figure 15b**, this clustering produces a histogram (**Figure 16**) that is much narrower and has a much smaller mean value than the Poisson distribution obtained for the random case. Examination of stars in the heavens indicates that they strongly cluster (into galaxies and clusters of galaxies). People also cluster, gathering together in towns and cities.

When the points are self-avoiding as shown in **Figure 15c**, the nearest-neighbor distances are also affected. The mean value of the histogram increases to a larger value than for the random case. The variance also usually drops, as a measure of the uniformity of the spacings between points. Self-avoiding or regular distributions are common in nature, whether you look at the arrangement of mitochondria in muscle tissue or precipitate particles in metals, because the physics of diffusion plays a role in determining the arrangement. The mitochondria are distributed to provide energy as uniformly as practical to the fibers. Forming one precipitate particle

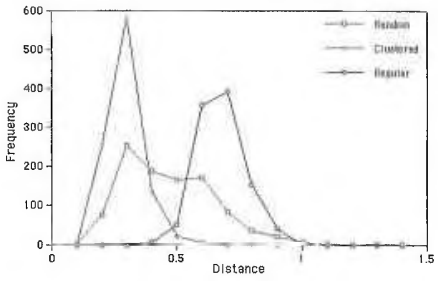


Figure 16. *Histogram of nearest-neighbor distances for each of the point distributions in* ***Figure 15****. The mean value of the clustered distribution is less than, and that for the self-avoiding distribution is greater than that for the random one.*

depletes the surrounding matrix of that element. The same effect occurs in the growth of cacti in the desert. Even the location of shopping centers is to some degree self-avoiding, in order to attract a fresh market of customers.

The ratio of the mean value of the nearest-neighbor distance distribution to that which would be obtained if the same number of points were randomly distributed in the same area provides a useful measure of the tendency toward clustering or self-avoidance for the features, and the variance of the distribution provides a measure of the uniformity of the tendency. Finding nearest-neighbor pairs using the centroid coordinates of features can also be used to characterize anisotropy in feature distributions. Instead of the distance between nearest neighbors, we can measure the direction from each feature to its nearest neighbor. For an isotropic arrangement of features, the nearest-neighbor directions should be a uniform function of angle. Plotting the histogram as a rose plot shows any deviations from this uniform function and indicates the degree of anisotropy.

For instance, if the regularly spaced distribution of feature points in **Figure 15c** is measured, the rose plot is reasonably circular and indicates that the distribution is isotropic. If the image is stretched 10% in the horizontal direction and shrunk 10% in the vertical direction, the total number of features per unit area is unchanged. The visual appearance of the image (**Figure 17**) does not reveal the anisotropy to a casual observer, but a plot of the rose of nearest-neighbor directions (**Figure 18**) shows that most of the features now have a nearest neighbor that is situated above or below. The elongation of the rose plot is a sensitive indicator of this type of anisotropy.

Measuring nearest-neighbor distances using the feature centroids is fine when the features are small compared to the distances between them. When features are large compared to the distances that separate them, and particularly when they are irregular in shape, vary in size, and have different orientations, it may be more appropriate to measure the distance from edge to edge rather than between centroids (Yang et al., 2001, suggest that it is the variance of the distribution of these distances that is most sensitive to inhomogeneity and to anisotropic clustering). The minimum separation distance is the shortest edge-to-edge distance between a point on one feature and that on its neighbor. As shown in **Figure 19**, this may be much different from the centroid-to-centroid distance, and in fact may involve a different neighbor.

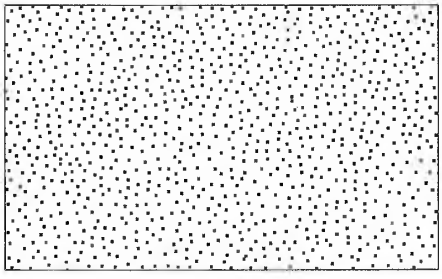


Figure 17. *Stretching the point distribution in* ***Figure 15c*** *horizontally and compressing it vertically introduces nonuniformity in the nearest-neighbor directions.*

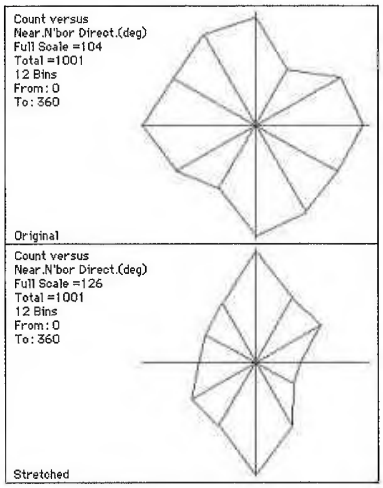


Figure 18. *Rose plot of nearest-neighbor directions for the point distributions in* ***Figures 15c*** *and* ***17****.*

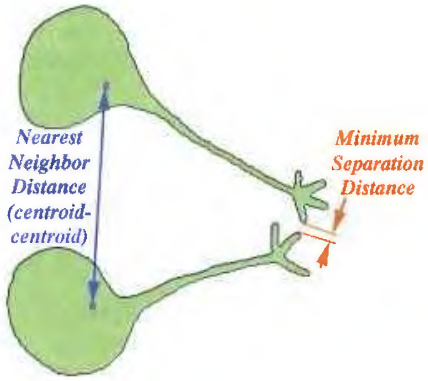


Figure 19. *Centroid-to-centroid distance and minimum separation distance for two features.*

The Euclidean distance map (EDM) introduced in Chapter 7 on binary images can be used to determine those edge-to-edge distances as shown in **Figure 20**. The skiz, or skeleton of the background between features, is a Voronoi tesselation of the image with one feature in each cell. The EDM of the cells in this tesselation measures the distance of every pixel from the lines. Assigning these EDM values to the features (the Boolean masking operation used in Chapter 7) gives each feature a minimum brightness of distance value that is exactly half the distance to the feature with the closest boundary point, because the skiz passes halfway between features.

A second method for measuring minimum separation distance is to construct the EDM of the background and find the local minima along the points in the skiz. These are midway between points of closest approach, and provide another measurement of minimum separation distance between features. With this technique, it is possible to measure multiple minimum separations between different points on one feature and those on one or several neighbors.

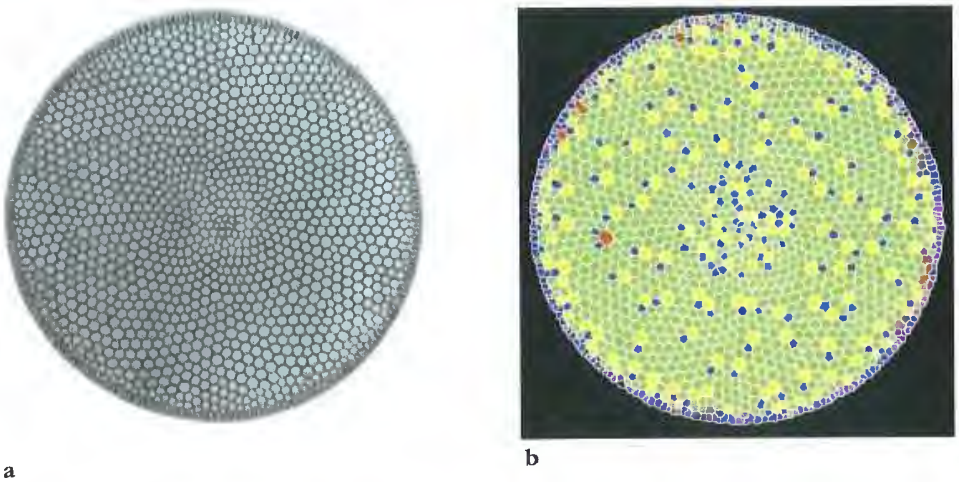
In the case of a space-filling structure such as cells in tissue, grains in a metal, or fields in an aerial survey, the number of adjacent neighbors is also of interest. The features are separate, so there must be a line of background pixels that separate them. In many cases, this line is produced by skeletonizing the original thresholded image of the boundaries and then inverting it to define the pixels. Counting the number of neighbors for each feature can be accomplished by checking the feature identification number of pixels that touch each of the background pixels in the boundaries. Building a table of the features that abut each other feature allows counting the number of such neighbors.



a b c

Figure 20. Determining minimum separation distance:
(a) features;
(b) the skiz of the features and the EDM of the cells in the resulting Voronoi tessellation;
(c) EDM values assigned to pixels in the features (shown in false color). The minimum value for each feature is half the distance to its nearest neighbor.

Labeling the features according to the number of neighbors can reveal some interesting properties of the structure. **Figure 21** shows the labeling of the number of neighbors in a diatom, an essentially two-dimensional (2D) structure. The pairing of the five- and seven-neighbor cells is apparent in the colored image, but might go unnoticed in the original. **Figure 22** shows a similar labeling for grains in a 3D metal structure, as revealed on a 2D section. For an equilibrium structure such as a fully recrystallized metal, the distribution of nearest neighbors should be log normal (as it is for the example shown).



a b

Figure 21. An image of a diatom
(a) with the holes in the structure color-coded according to the number of sides
(b) The predominant shape is six-sided, but some five- and seven-sided holes generally lie adjacent to each other.

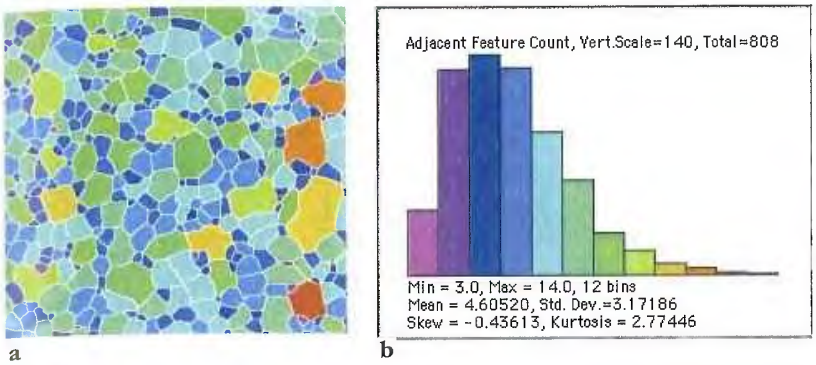


Figure 22. *Grains in a steel alloy, color coded according to the number of nearest neighbors. The plot of the frequency distribution of number of neighbors is log-normal in both two and three dimensions for an equilibrium grain structure.*

Alignment

One thing that people are very good at seeing in images that requires a special computer algorithm is the alignment and arrangement of features. We are so good at it that we sometimes find such alignments and arrangements when they do not really exist. Constellations in the sky are a good example of this tendency to bring order to disorder.

Many image analysis situations include an algorithmic procedure for determining if an alignment or arrangement is needed. One of the most common is completing broken lines, especially straight lines. Such a procedure is useful at all magnifications, from trying to locate electric transmission lines in reconnaissance photos (which show the towers but not the wires) to trying to delineate atomic lattices in transmission electron microscopy. In general, the points along the line are not spaced with perfect regularity and may not lie precisely on the line. This irregularity, and the sensitivity to noise in the form of other points of features near but not part of the line, make frequency-domain techniques difficult to apply and poor in resulting precision.

Transforming the image into a different space provides the key. The use of Fourier space was shown in Chapter 5, but many alternative spaces are less common. Transformation into Hough space can be used to find alignments (Hough, 1962; Duda and Hart, 1972; Ballard, 1981). Different Hough spaces are used to fit different kinds of shapes, and it is necessary in most cases to have a pretty good idea of the type of line or other arrangement that is to be fit to the data. We will start with the straight line, because it is the simplest case. The conventional way to fit a line to data points on a graph is the so-called "least-squares" method where the sum of the squares of the vertical deviations of each point from the line is minimized. The Hough method is superior to this because it minimizes the deviations of points from the line in a direction perpendicular to the line, and it deals correctly with the case of the points not being uniformly distributed along the line.

Two parameters are required to define a straight line. In Cartesian coordinates, the equation of a straight line is

$$y = m \cdot x + b \tag{9}$$

where m is the slope and b the intercept. Because m becomes infinitely large for lines that are nearly parallel to the y-axis, this representation is not usually used in the Hough transform. Instead, the polar coordinate representation of a line is used. This consists of a radius ρ and angle φ (the length and angle of a normal to the line from the origin). These define the line as shown schematically in **Figure 23**. It is possible either to allow the angle to vary from 0 to 2π and to keep ρ

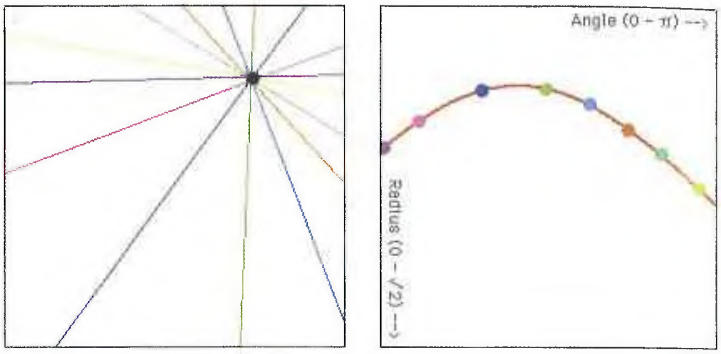


Figure 23. Principle of the Hough transform. *Each point in the real-space image* ***(a)*** *produces a sinusoidal line in Hough space* ***(b)*** *representing all possible lines that can be drawn through it. Each point in Hough space corresponds to a line in real space. The real-space lines corresponding to a few of the points along the sinusoid are shown, with color coding to match them to the points.*

positive, or to allow ρ to be either positive or negative and to restrict the angle φ to a range of 0 to π. The latter convention is used in the examples that follow. The black point shown in **Figure 23a** generates the sinusoid in Hough space shown in **Figure 23b**. Each point along this sinusoid corresponds to the ρ-φ values for a single line passing through the original point. Several of these are shown in color, where the color of the point in Hough space matches the corresponding line in real space.

Hough space is an accumulator space. This means that it sums up the votes of many pixels in the image, and points in Hough space that have a large total vote are then interpreted as indicating the corresponding alignment in the real-space image. For the linear Hough transform, used to fit straight lines to data, the space is an array of cells (pixels, since this is another image) with coordinates of angle and radius. Every point in Hough space defines a single straight line with the corresponding angle φ and radius ρ that can be drawn in the real-space image.

To construct the Hough transform, every point present in the real-space image casts its votes into the Hough space for each of the lines that can possibly pass through it. As shown in **Figure 23**, this means that each point in the real-space image generates a sinusoid in Hough space. Each point along the sinusoid in Hough space gets one vote added to it for each point in the real-space image, or perhaps a fractional vote based on the density of the point in real space. The superposition of the sinusoids from several points in real space causes the votes to add together where they cross. These crossing points, as shown in **Figure 24**, occur at values of ρ and φ that identify the lines that go through the points in the real-space image.

In this example, the duality of lines and points in real and Hough space is emphasized. Each of the five original points (labeled A–E) in real space produces a sinusoid (similarly labeled) in Hough space. Where these sinusoids cross, they identify points in Hough space (labeled 1–6). These points correspond to lines back in real space (similarly labeled) that pass through the same points. Notice for instance that three lines pass through point A in real space, and that the sinusoid labeled A in Hough space passes through the points for each of those three lines.

If several such alignments of points in the real-space image exist, then there will be several locations in the Hough transform that receive numerous votes. It is possible to find these points either by thresholding (which is equivalent to finding lines that pass through a selected minimum number of points) or by looking for local maxima (peaks in the Hough transform), for example by using a top-hat filter.

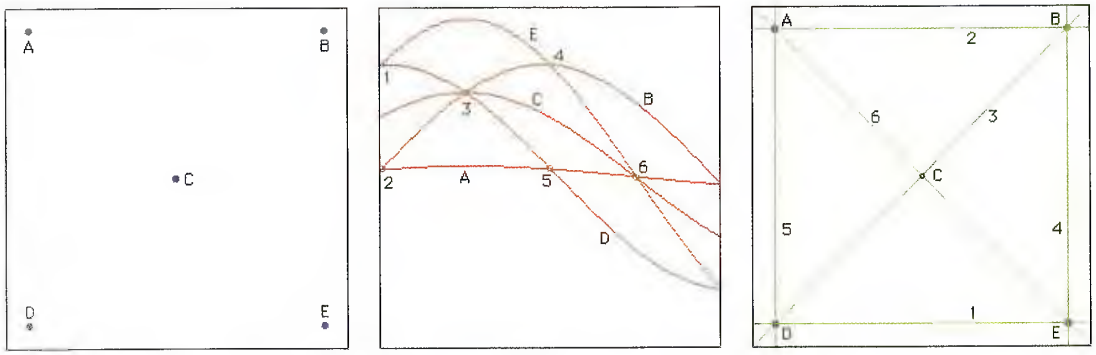


Figure 24. Duality of lines and points in Hough and real space. *Each of the labeled points in* ***(a)*** *generates a sinusoidal line in the Hough space shown in* ***(b)****. The numbered crossing points in this space generate the lines in the real-space image shown in* ***(c)****.*

If the original real-space image contains a step in brightness, and an image-processing operation such as a gradient (e.g., Sobel filter) has been applied, then an improved detection and fit of a line to the edge can be achieved by letting each point in the image vote according to the magnitude of the gradient. This means that some points have more votes than others, according to how probable it is that they lie on the line. The only drawback to this approach is that the accumulator space must be able to handle much larger values or real numbers when this type of voting is used, than in the simpler case where pixels in a binary image of feature points either have one vote or none. **Figure 25** shows an example of fitting a line to a noisy edge using an gradient operator and

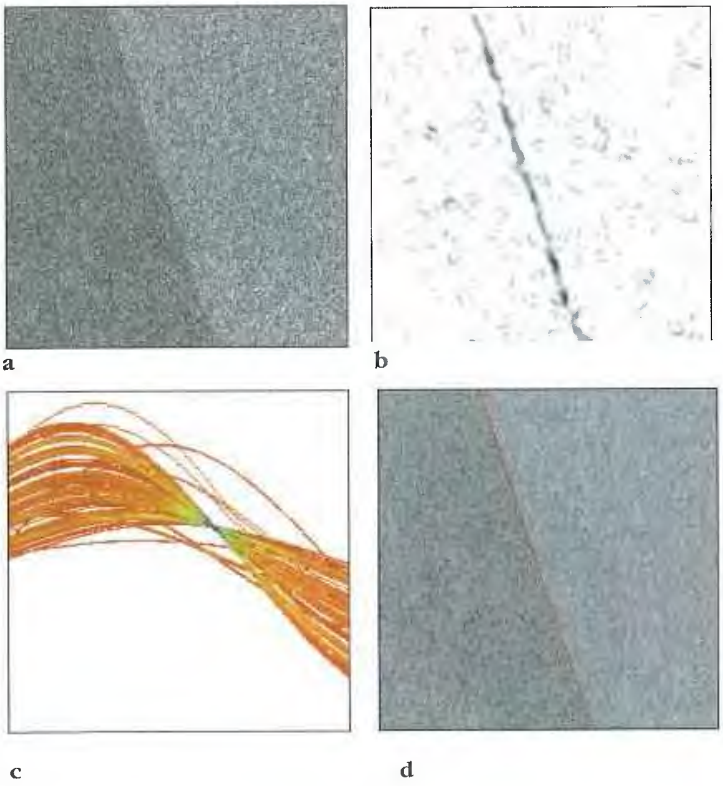


Figure 25. Fitting a line to a noisy edge using the Hough transform: ***(a)*** *the original noisy image;* ***(b)*** *smoothing to reduce the noise and application of a Sobel gradient filter;* ***(c)*** *Hough transform produced from the gradient image;* ***(d)*** *line defined by the maximum point in the Hough transform.*

the Hough transform. Notice that the location of the maximum point in Hough space involves the same requirements for using the pixel values to interpolate a location to sub-pixel dimensions as discussed at the start of this chapter.

For measurement of the spacings of parallel lines, Hough spaces reduce the problem to measuring the vertical distance between peaks in the transform. As shown in **Figure 26**, the application of a derivative to an SEM image of a line on an integrated circuit produces peaks at the same φ value, and measuring the distance in ρ provides direct measures of the width and spacing of the lines. Similarly, to measure the angular orientation of lines it is straightforward to determine the φ values in the transform.

This proportional voting system is useful when there are many points to be fit, with some points being more important than others. Summing the votes according to brightness (or some value obtained from brightness) allows some points to be weighted more and produces an improved fit.

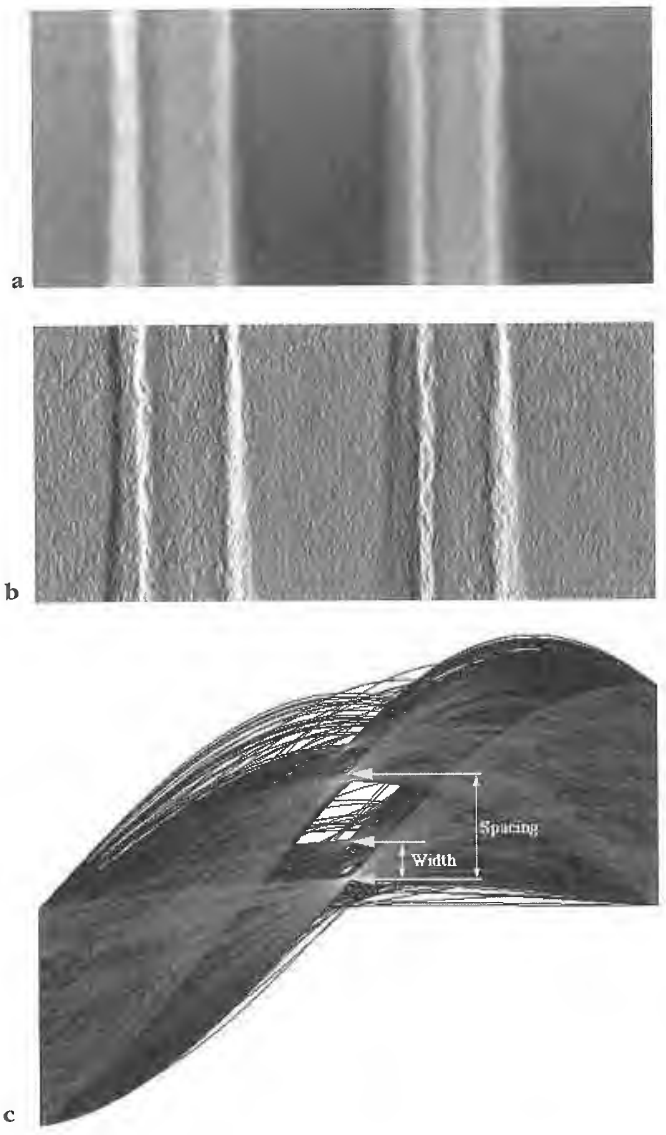


Figure 26. Measuring line width on an integrated circuit: *(a) original SEM image; (b) horizontal derivative applied; (c) Hough transform, showing peaks corresponding to line edges and the distances that measure width and spacing.*

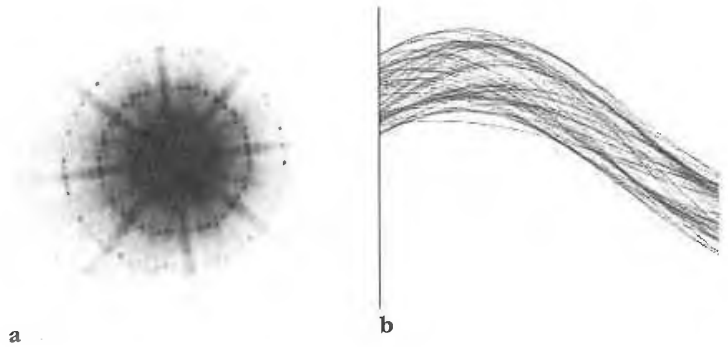


Figure 27. Convergent beam electron diffraction pattern
(a) *and its Hough transform* ***(b)*** *Each peak in the transform space identifies an alignment of points in the original pattern.*

Figure 27 shows a Hough transform of a convergent beam electron diffraction pattern. The most intense spots in the Hough transform identify each of the spot alignments in the pattern.

The Hough transform can be adapted straightforwardly to other shapes, but the size and dimensionality of the Hough space used to accumulate the votes increases with the complexity of the shape. Fitting a circle to a series of points is used to measure electron diffraction patterns and to locate holes in machined parts. A generalized circular Hough transform requires a three-dimensional space, because three parameters are required to define a circle (the x,y coordinates of the center and the radius). Each point in the real-space image produces a cone of votes into the Hough space, corresponding to all of the circles of various radii and center positions that could be drawn through the point.

If one or two of these values are known (for instance, the center of the electron diffraction pattern is known, or the radius of the drilled hole) or at least can vary over only a small range of possible values, then the dimensionality or size of the Hough space is reduced and the entire procedure becomes quite rapid. Accurately fitting a circle to an irregularly spaced set of points of varying brightness is a good example of the power that the Hough approach brings to image measurement. Because the brightness of the point in Hough space that defines the circle is also the summation of the votes from all of the points on the corresponding circle in real space, this approach offers a useful way to integrate the total brightness of points in the electron diffraction pattern as well.

Figure 28 shows an example of using a circular Hough transform to locate the "best fit" circles in a selected area diffraction pattern. The pattern itself has bright spots with irregular locations around each circle. A human has little difficulty in estimating where the circles lie, but this is generally a difficult thing for computer algorithms that examine only local pixel regions to accomplish. The Hough transform locates the circles and also provides a measure of the integrated brightness around each circle. This approach has been used to identify asbestos fibers from the very spotty electron diffraction patterns obtained from a few fibers (Russ et al., 1989).

Figure 29 shows an example from machine vision used to locate edges and holes accurately for quality control purposes or robotics guidance. Despite overall noise in the image, the fit is quite rapid and robust. A separate region of interest (ROI) is set up covering the area where each feature is expected. In each ROI, a gradient operator is applied to identify pixels on the edge. These are then used to perform a Hough transform, one for a straight line and one for a circle. The resulting feature boundaries are shown, along with the measured distance between them. Many pixels have contributed to each feature, so the accuracy of location is much better than the pixel dimensions.

The Hough transform approach can be used to fit other alignment models to data as well. The limitation is that as the algebraic form of the model changes, each of the adjustable constants requires another dimension in Hough space. Constructing the transform and then finding the maximum points becomes both memory-intensive and time-consuming.

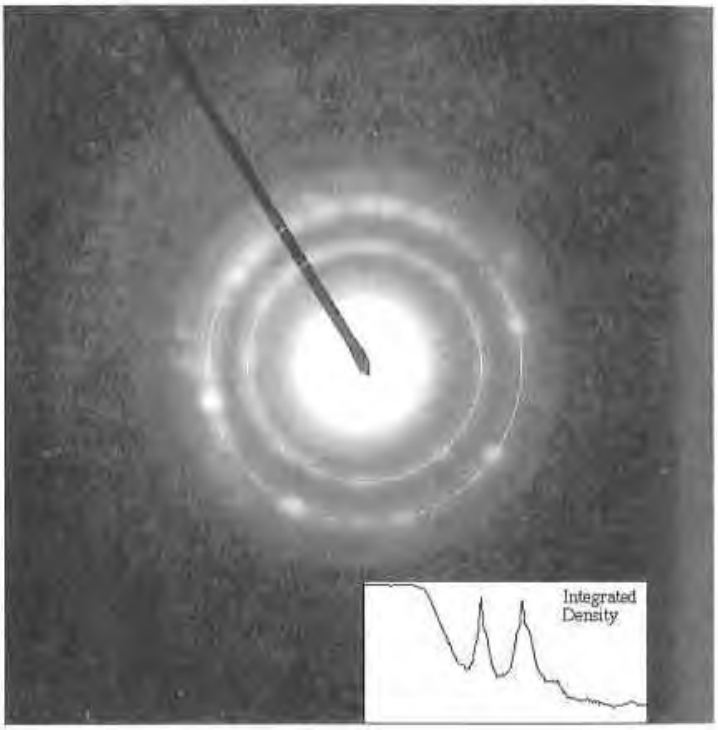


Figure 28. *Selected area electron diffraction pattern with principal circles located by a circular Hough transform, and the integrated circular density of the pattern.*

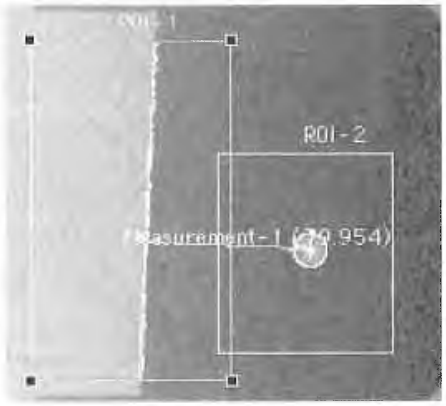


Figure 29. *Use of the Hough transform to locate an edge and a circle in a machined part, in order to measure the distance between them to sub-pixel accuracy.*

Counting features

Counting the number of features present in an image or field of view is one of the most common procedures in image analysis. The concept seems entirely straightforward, and it is surprising how many systems get it wrong. The problem has to do with the finite bounds of the field of view. In the case in which the entire field of interest is within the image, there is little difficulty. Each feature has simply to be defined by some unique point and those points counted. Depending on how features are recognized, some algorithms may ignore features that are entirely contained within holes in other features. This problem most often arises when boundary representation is used to define the feature boundaries in the binary image. Determining whether it is appropriate to count features that lie inside other features is a function of the application, and consequently must be left up to the user.

When the field of view is a sample of the entire structure or universe to be counted, the result for the number of features is generally given as number-per-unit-area. When features intersect the edge of the field of view, it is not proper to count all features that can be seen. The most common solution to produce an unbiased result is to count those features that touch two adjacent edges, for instance the top and left, and to ignore those that touch the other two edges, for instance the right and bottom. This is equivalent to counting each feature by its lower right corner. Each feature has one and only one lower right corner, therefore, counting those points is equivalent to counting the features. (**Figure 30**).

This method is the same as determining the number of people in a room by counting noses. Since each person has one nose, the nose count is the same as the people count. If a smaller region was marked within the room, so that people might happen to straddle the boundary, counting the noses would still work. Anyone whose nose was inside the region would be counted, regardless of how much of the person lay outside the region. Conversely, any person whose nose was outside would not be counted, no matter how much of the person lay inside the region. It is important to note that in this example, we can see the portions of the people that are outside the region, but in the case of the features in the image, we can usually cannot (unless the counting is performed within a reduced area inside the full image, another possible solution). Any part of the feature outside the field of view is by definition not visible, and we cannot know anything about the amount or shape of the feature outside the field of view. That is why it is important to define a unique point for each feature to use in counting.

The convention of counting features that touch two edges only is not implemented in all systems. Some software packages offer a choice of counting all features regardless of edge touching, or counting only those features that do not touch any edge. Note that when measuring features, as opposed to counting them, a more complicated procedure will be needed. A feature that intersects any edge cannot be measured, because it is not all imaged and therefore no size, shape or position information can be correctly obtained. If only the features in **Figure 31** that did not touch any edge were counted, the proportions of large and small features would be wrong. It is more likely that a large feature will touch an edge, and so a disproportionate fraction of the large features intersect an edge of the field of view and cannot be measured.

This bias can be corrected in two ways. The methods produce the same result, but are implemented differently. The older method is to set up a "guard frame" within the image, as shown in **Figure 32**. In this case, features that touch the lower and right edges of the field of view are not counted or measured, as before. Features that cross the top and left edges of the guard frame are counted and measured in their entirety. Features that lie only within the guard frame are not counted, whether they touch the actual edge of the field of view or not. The number of features counted is then an accurate and unbiased measure of the number per unit area, but the area

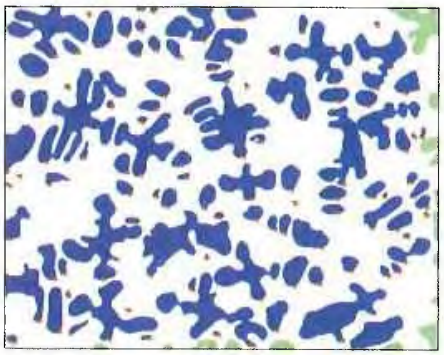


Figure 30. *Counting features by a unique point. The red dots mark the right end of the lowest line of pixels in each feature. Features that touch the upper and left boundaries are counted, but those that touch the bottom and right edges are not.*

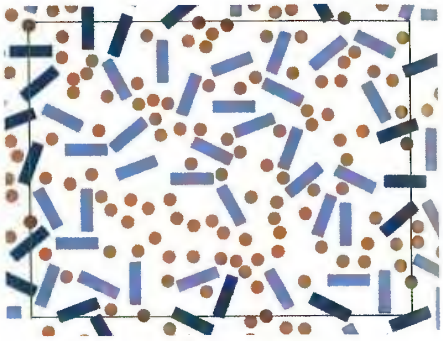


Figure 31. *When an image area (shown by the green frame) lies within a larger field of objects to be measured, some will be intersected by the edge. These cannot be measured because their extent outside the image area is unknown. In the example shown, although it includes three times as many small red features as long blue ones, the frame intersects (and removes from the measurement) twice as many blue ones, biasing the result.*

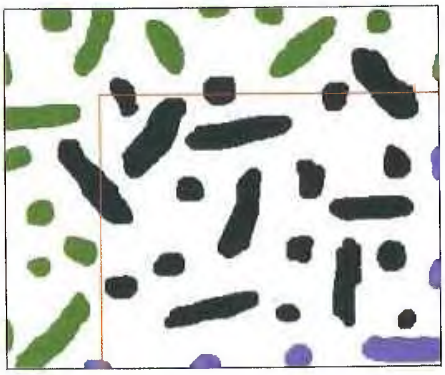


Figure 32. *Measurement using a guard frame. Features that lie partially or entirely within the inner region are counted. Features that touch the outer edge of the image or lie entirely in the guard frame are not. The width of the guard frame must be large enough that no feature can span it.*

reported for the measurement is the area within the guard frame, not the entire area of the image. It is necessary for the guard region to be wide enough that no feature can extend from within the active region across the guard region to the edge of the field, so the active region may be reduced to a fraction of the total image area.

The second method uses the entire image area and measures all of those features that do not touch any of the edges. In order to compensate for the bias arising from the fact that larger features are more likely to touch the edge and be bypassed in the measurement process, features are counted in proportion to the likelihood that a feature of that particular size and shape would be likely to touch the edge of a randomly placed image frame. The so-called adjusted count for each feature is calculated as shown in **Figure 33** as:

$$Count = \frac{W_x \cdot W_y}{(W_x - F_x)\cdot(W_y - F_y)} \tag{10}$$

where W_x and W_y are the dimensions of the image in the x and y directions (in pixels), and F_x and F_y are the maximum projected dimensions of the feature in those directions. These are simply the same bounding-box coordinates as discussed previously in connection with finding a feature's location. When the feature dimensions are small compared to the dimensions of the field of view, the fraction is nearly 1.0 and simple counting is unaffected. This is the case for counting small features. When the feature extends across a larger fraction of the field of view in either direction, it is more likely that a random placement of the field of view on the sample will cause it to intersect an edge; thus the features that can be measured must be counted as more than one to correct for those that have been overlooked. The adjusted count factor makes that compensation.

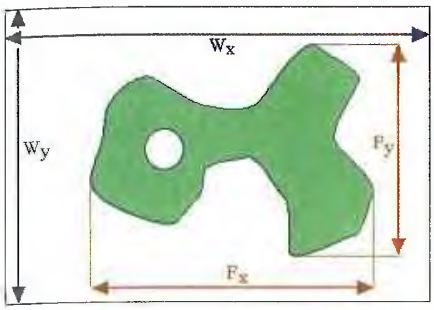
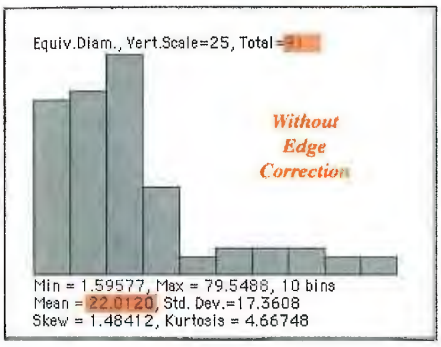


Figure 33. Dimensions of the image and the feature used to adjust the count for correct results.

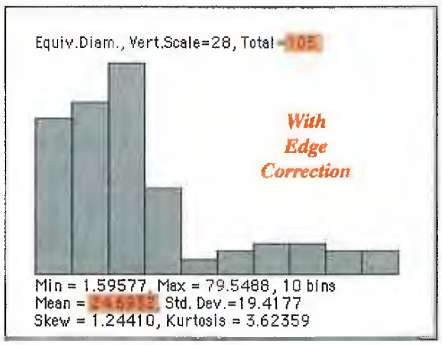
Figure 34. Example of the adjusted count for features that vary in size:
(a) adjusted count for each feature;
(b) size distribution data with no adjusted count to correct for bias due to features intersecting the edge;
(c) using the adjusted count — note the difference in the total number of features and mean feature size.



a



b



c

Figure 34 shows an example of an image containing features of varying sizes and shapes, with the adjusted count for each. A plot of the size distribution shows that using the adjusted count significantly affects the mean size, the shape of the distribution, and the total number of features.

Special counting procedures

The examples of counting in the previous section make the tacit assumption that the features are separate and distinct. In earlier chapters, procedures for processing images in either the grey-scale or binary format were shown whose goal was to accomplish the separate delineation of features to permit counting and measurement; however, these methods are not successful in numerous situations, or they are, at least, very difficult. Sometimes, if there is enough separate knowledge about the specimen, counting can be accomplished even in these difficult cases.

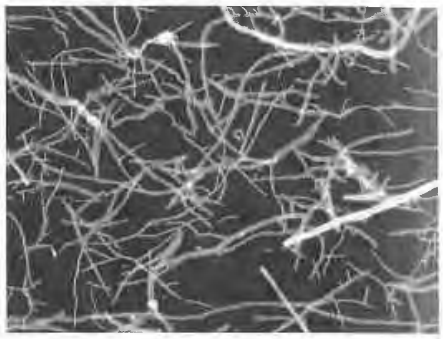


Figure 35. Image of crossing fibers, with the skeleton superimposed, and a total of 222 end points.

As an example, **Figure 35** shows a collection of crossing fibers. Such structures are common in biological samples, wood fibers used in paper production, food technology, textiles, and many more situations. Because the fibers cross, from the point of view of the image analysis algorithms, there is only a single feature present, and it touches all sides of the field of view. It is possible, however to estimate the number of fibers per unit area, and the average fiber length. By thresholding the fibers and skeletonizing the resulting binary image, a series of crossing midlines is revealed. As discussed in Chapter 7, the end points of fibers can be counted as those pixels in the skeleton that have exactly one touching neighbor. This is not a perfect method, as there may be some ends of real fibers that are hidden because they lie exactly on another fiber, but in principle it is possible to detect these occurrences as well because they produce pixels with exactly three neighbors, and one of the angles at the junction will be about 180°. Even without this refinement, however, half of the end point count gives a useful approximation for the number of fibers.

Recall that, in the preceding section, we decided to count each feature by one unique point. Counting the end points of fibers uses two unique points per fiber. It is a little like the old joke about how to count the number of cows in a field ("Count the number of hooves and divide by four."). In this case, we count the number of end points and divide by two, to get the number of fibers. If a fiber has one end in the field of view and the other end out, it is counted (correctly) as ½ fiber, because the other end would be counted in a different field of view. Dividing by the image area gives the number of fibers per unit area. If the total length of fiber (the total length of the skeletonized midline) is divided by the number of fibers (one half the number of ends), the result is the average fiber length. If the fibers are much longer than the field of view so that there are no end points present in most fields of view, it then becomes necessary to combine the data from many fields of view to obtain a statistically meaningful result.

In the earlier chapter on binary image processing, several techniques for separating touching features were discussed. One of the more powerful, known as watershed segmentation, utilizes the EDM. This operation assigns a grey-scale value to each pixel within a feature proportional to the distance from that pixel to the nearest background point. The valleys, or points that lie between two higher pixels in this distance map, are then used to locate boundaries that are ultimately drawn between features in order to separate them. Because there is a built-in assumption in this method that any indentation around the periphery of the feature cluster indicates a separation point, this technique is also known as convex segmentation.

If the purpose of the processing and segmentation is to count the features, a short-cut method saves much of the computation. In the EDM, every local maximum point (pixels equal to or greater than all eight of their neighbors) is a unique point that represents one feature that will eventually be separated from its touching companions. Finding these so-called ultimate eroded points provides another rapid way to count the features present. In addition, the grey-scale value of that point in the EDM is a measure of the size of the feature, since it corresponds to the radius of the

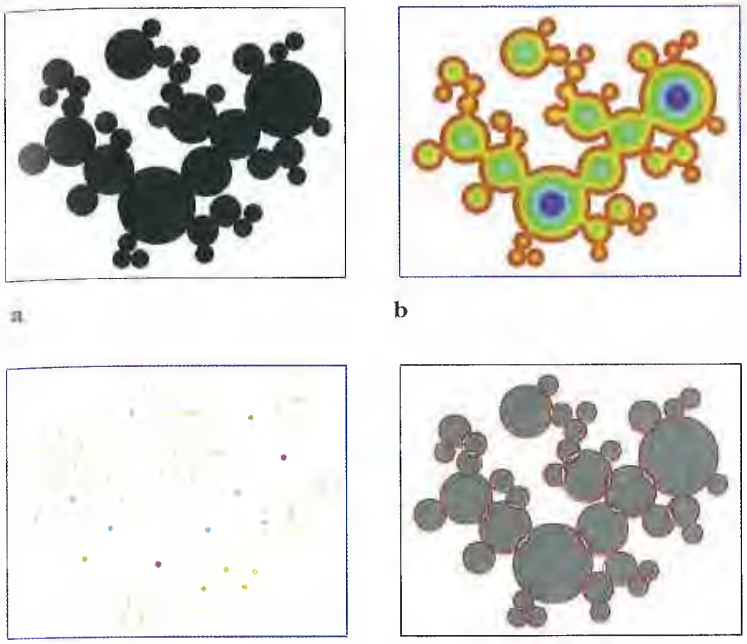


Figure 36. Counting and sizing touching particles:
(a) *binary image of touching circles;*
(b) *EDM (color indicates distance of pixel from nearest background point);*
(c) *ultimate points (local maxima in the EDM), whose color gives the radius;*
(d) *intersecting circular particle boundaries based on the ultimate points.*

circle that can be inscribed within it. **Figure 36** shows a simple example of some touching circles, with the ultimate points found from the EDM.

Counting touching particles presents many difficulties for image analysis algorithms. If the particles are not simple convex shapes that lie in a single plane and happen to touch at their boundaries, watershed segmentation is not usually able to separate them successfully. A typical example of a more difficult problem is shown in **Figure 37**. The clay particles that coat this paper sample form

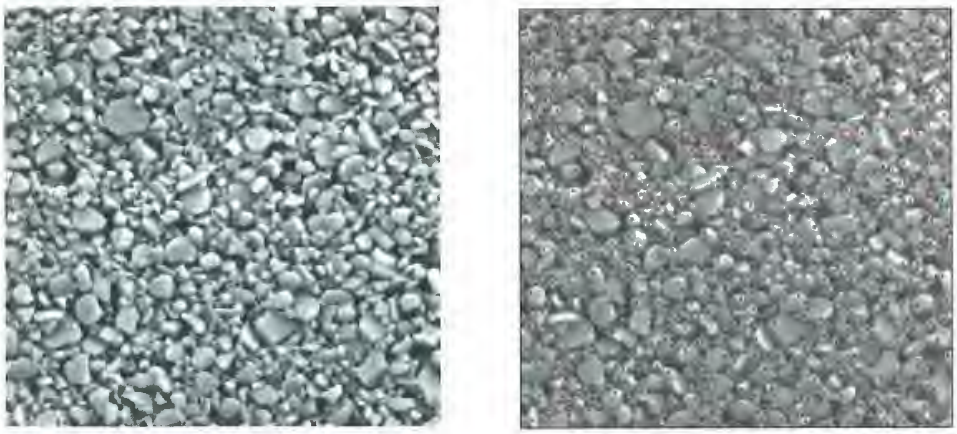


Figure 37. Counting overlapping particles:
(a) *SEM image of clay particles on paper;*
(b) *superimposed points from a top-hat filter used to locate a local brightest point, which can be used to estimate the number of particles.*

a layer that is essentially one particle thick, but the overlaps are quite extensive. In order to obtain a useful estimate of the number of particles per unit area of the paper, it is again necessary to use the idea of finding one unique point per particle.

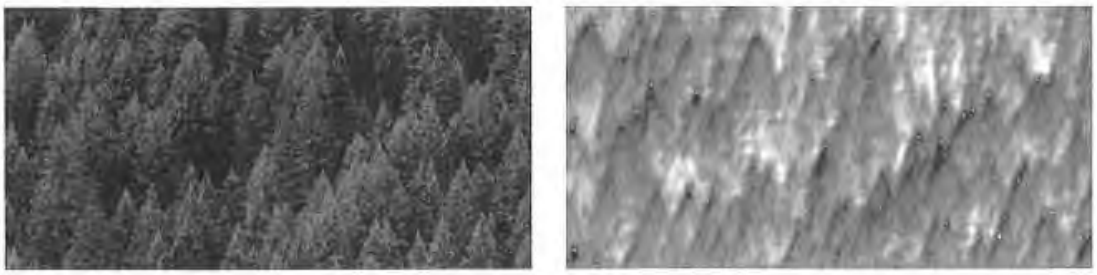
In this case, the way the image is formed by the scanning electron microscope provides an answer. Most of the particles are somewhat angular and have a single highest point that appears bright in the secondary electron image. These points (shown in the figure) can be isolated using a top hat filter, as discussed in Chapter 4. In the example, the top-hat filter finds one unique point with a local maximum in brightness for most of the clay particles. A few are missed altogether because they have a shape or orientation that does not produce a characteristic bright point, and a few particles have sufficiently irregular shapes that they produce more than a single characteristic point. This means that counting the points gives only an estimate of the particle density on the paper, but for many quality control purposes this sort of estimate is sufficient to spot changes.

The top-hat filter used in the preceding example is a specific case of a feature-matching or template operation, that looks throughout the image for a particular pattern of bright and dark pixels. A more general case uses cross correlation, as discussed in Chapter 5. The example in **Figure 38** shows an aerial photograph of trees in the forests on Mount Mitchell, North Carolina. It is desired to count the trees to monitor damage from acid rain. A useful short-cut method is to use the image of the pointed top of one tree as a target, cross correlate with the entire image, and count the resulting spots. Of course, this method is only approximate; it does a good job of finding trees that are partially hidden or in front of another tree, but will miss trees that do not have the characteristic pointed top of the target example.

Another approach to counting particles in clusters is shown in **Figure 39**. This is a transmission electron microscope (TEM) image of carbon black particles, which are present in the form of clusters of varying sizes. If it can be assumed that the particles in the clusters are similar in size and density to the isolated one, then it becomes possible to estimate the number of particles in a cluster from the integrated density of the cluster. Measuring the average value of the integrated optical density of a isolated particles, and dividing that value into the integrated density of each cluster, provides a number that is an estimate of the number of particles present.

Feature size

The most basic measure of the size of features in images is simply the area. For a pixel-based representation, this is the number of pixels within the feature, which is straightforwardly determined by counting. For boundary representation, the area can be calculated as discussed earlier (**Equation 4**). Of course, it must be remembered that the size of a feature in a 2D image may be related to the size of the corresponding object in 3D space in various ways depending on how the image



a b

Figure 38. Counting with cross-correlation:
(a) *aerial photograph of trees — the red outline marks the tree top selected as the target for matching;*
(b) *cross-correlation result, with identified tree tops marked.*

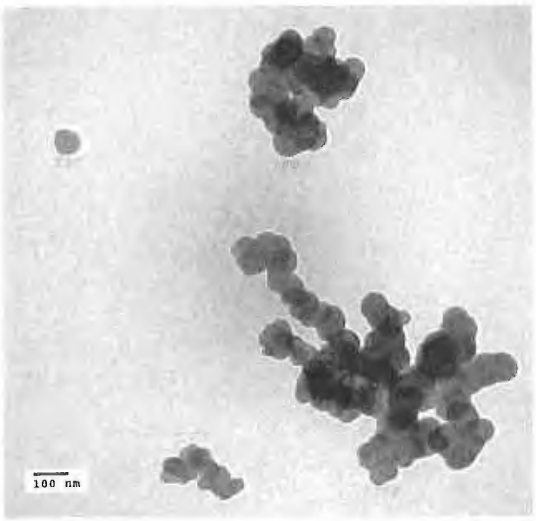


Figure 39. *Clusters of carbon black particles viewed in the transmission electron microscope. Dividing the integrated optical density of each cluster by the integrated density for the single particle provides an estimate of the number of particles in each cluster (10, 54, and 162 particles in the clusters shown).*

was obtained. The most common type of images are projections, in which the features show the outer dimensions of the objects, or planar sections, in which the features are slices across the objects. In the latter case, it is possible to estimate the volume of the objects using stereological methods as discussed in the preceding chapter. In this chapter, we will assume that the feature size is directly represented in the image.

Figure 40 shows a projected image of spherical particles dispersed on a flat substrate. The diameters can be measured straightforwardly from such an image subject to the usual restriction that there must be enough pixels in each feature to give a precise measure of its size. When the particles cover a large size range, as they do in this image, this creates a problem. With the resolution of a typical video camera, the smallest features are only one or a few pixels in size, and are not well defined (in fact many are not even thresholded). A high-resolution digital camera provides superior results. The alternative would be to increase the optical magnification to enlarge the small particles, but then the large ones would be likely to intersect the edges of the screen and could not be measured. Multiple sets of data taken at several magnifications would be required, unless a high-resolution camera can be used.

Even for such a simple idea as counting pixels to determine feature area, some decisions must be made. For instance, consider the feature shown in **Figure 41**. Should the pixels within internal holes be included in the area or not? Of course, this depends on the intended use of the data. If the hole is a section through an internal void in an object, then it should be included if the area of the feature is to be related to the object volume; but it is hard to know whether the hole may be a section through a surface indentation in that object, in which case it would be more consistent to also include in the area those pixels in fjords around the feature boundary. As shown in the figure, this produces three different possible area measurements, the net area, the filled area, and the convex area.

Measuring the first two can be accomplished as a simple counting exercise. In the process of labeling the pixels that touch each other and comprise the feature, the presence of internal holes can be detected and the pixels within them counted. These pixels can be added back to the original image to fill in the holes, if desired. Determining whether to include those pixels in the area then becomes a user decision based on other knowledge.

The convex area is a slightly more difficult proposition. In some cases, a combination of dilation and erosion steps can be used to construct a convex hull for the feature and fill any boundary

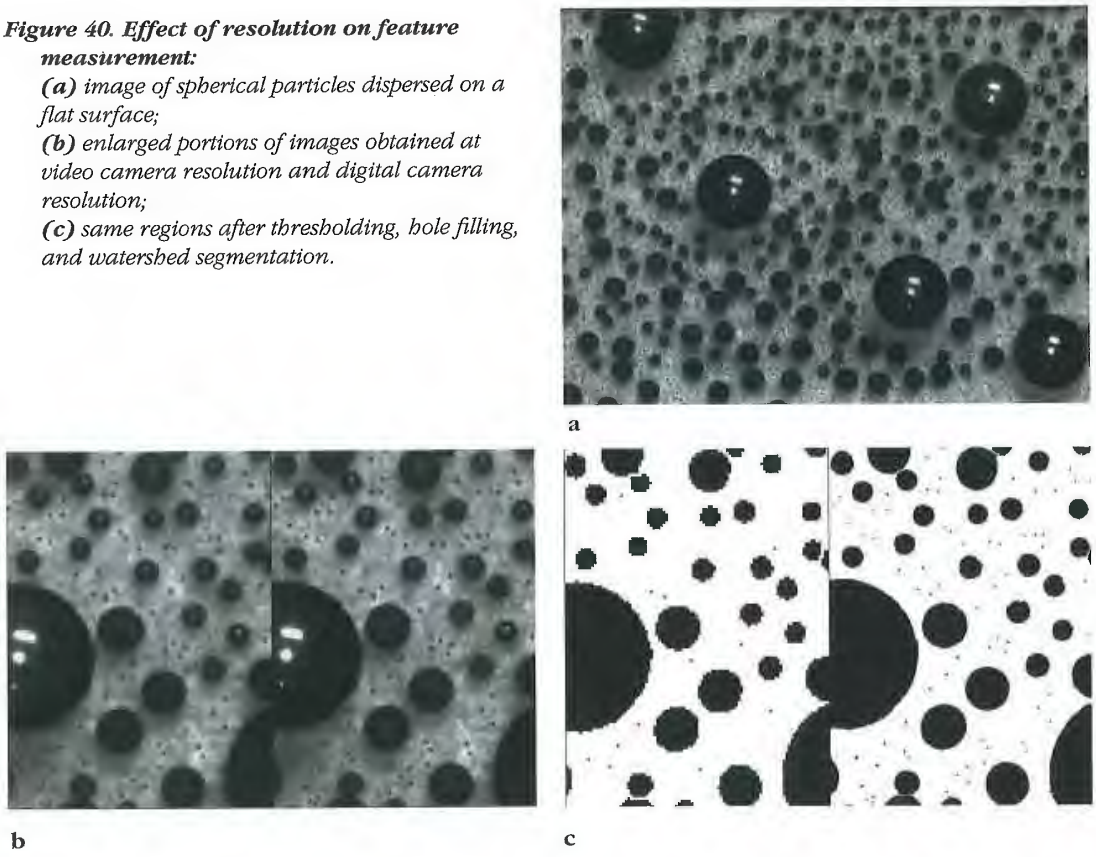


Figure 40. Effect of resolution on feature measurement:
(a) *image of spherical particles dispersed on a flat surface;*
(b) *enlarged portions of images obtained at video camera resolution and digital camera resolution;*
(c) *same regions after thresholding, hole filling, and watershed segmentation.*

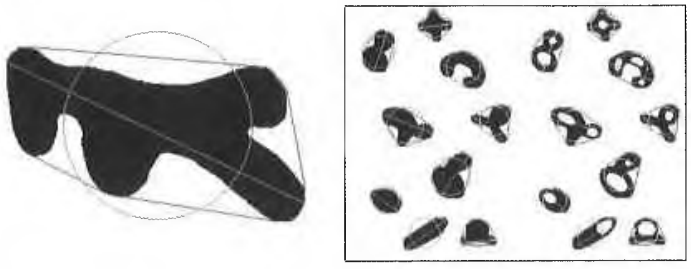


Figure 41. *Three possible measures for feature area: net, filled, and convex.*

irregularities, so that pixel counting can be used to determine the area. On a square pixel grid, however, these methods can cause some other distortions of the feature shape, as was shown in Chapter 7. Another approach that constructs an n-sided polygon around the feature is sometimes called the taut-string or rubber-band boundary of the feature, because it effectively defines the minimum area for a convex shape that will cover the original feature pixels.

By rotating the coordinate axes, it is similarly possible to locate the minimum and maximum points in any direction. The procedure of rotating the coordinate system, calculating new x',y' values for the points along the feature boundary, and searching for the minimum and maximum values, is simple and efficient. For any particular angle of rotation α, the sine and cosine values are needed. In most cases these are simply stored in a short table corresponding to the specific angles used in the program. Then the new coordinates are calculated as

$$x' = x \cdot \cos\alpha + y \cdot \sin\alpha$$
$$y' = y \cdot \sin\alpha - x \cdot \cos\alpha \qquad (11)$$



a b

Figure 42. *Comparison of the bounding polygon (red), the longest chord (yellow) and equivalent circle (green — the circle centered at the feature centroid having the same area) for an irregular feature, and an assortment of features.*

When this process is carried out at a series of rotational angles, the points with the largest difference in each rotated coordinate system form the vertices of the bounding polygon discussed above. For purposes of constructing the convex or taut string outline, using a modest number of rotation steps will produce a useful result. For instance, with 18 steps (rotating the axes in 10-degree steps) a bounding polygon with 36 vertices and sides is obtained. **Figure 42** shows an example, comparing the bounding polygon to the equivalent circle and longest dimension for various feature shapes. It is only for extremely long and narrow features that more sides might be required to produce a bounding polygon that is a good approximation to the outer boundary.

No matter how the area is defined and determined, it of course requires that a conversion factor between the size of the pixels and the dimensions of the real-world structures be established. Calibration of dimension is usually done by capturing an image of a known standard feature and measuring it with the same algorithms as later used for unknown features. For macroscopic or microscopic images, a measurement scale may be used. Of course, it is assumed that the imaging geometry and optics will not vary; this is a good assumption for glass lenses in light microscopes, but not necessarily for electron lenses used in electron microscopes. For some remote sensing operations, the position and characteristics of the camera are known and the magnification is calculated geometrically.

Most modern systems use square pixels that have the same vertical and horizontal dimensions, but for distances to be the same in any direction on the image, it is also necessary that the viewing direction be normal to the surface. If it is not, then image warping as discussed in Chapter 3 is required. Some systems, particularly those that do not have square pixels, allow different spatial calibrations to be established for the horizontal and vertical directions. For area measurements based on pixel counting, this does not matter; but for length measurements and the shape parameters discussed in the next section, this discrepancy creates serious difficulties.

Circles and ellipses

Once the area has been determined, it is often convenient to express it as the equivalent circular diameter. This is linear size measure, calculated simply from the area as

$$Eq.Diam. = \sqrt{\frac{4}{\pi} Area} \tag{12}$$

Figure 42b shows several features of different sizes and shapes with the equivalent circle diameter shown based on the net feature area (pixel count). Features of different shape or orientation can fool the eye and make it difficult to judge relative size. The equivalent diameter values offer a simple and easily compared parameter to characterize size.

Circles are widely used as size measures. Besides the equivalent circle with the same area as the feature, the inscribed and circumscribed circle can be used (**Figure 43**). The circumscribed circle is determined by using the corners of the bounding polygon, sorting through them to find the two



Figure 43. *Circumscribed and inscribed circles for an irregular feature. The centers of these circles do not generally coincide with each other or with the feature centroid.*

or three that define the circle that encloses all of the others (a concise algorithm for finding the circle is in Arvo, 1991). The inscribed circle is easily found as the maximum value of the EDM of the feature; the maximum pixel marks the center and its value gives the radius.

Because most real features have quite irregular shapes, it is not easy to find size measures that compactly and robustly describe them and allow for their classification and comparison. The use of equivalent circular diameter is one attempt. Recognizing that not all features are equiaxed or evenly approximately round, some systems also provide for the use of an ellipse to describe the feature. The fact that the ellipse has two axes seems to allow describing both a size, a degree of departure from circularity, and even an orientation.

The major and minor axes of the ellipse may be determined in several different ways, however. These actually represent some quite different aspects of the feature size and shape, and may be more misleading than helpful unless the user is fully aware of (and careful with) their various biases.

One definition of the ellipse axes can be taken from the minimum and maximum caliper dimensions of the feature, discussed previously. The maximum caliper dimension does a good job of representing a maximum dimension, and indicates the feature orientation at least within the step size of the search (e.g. 10°). If this is taken as the major dimension of the ellipse, then the minor axis could assigned to the minimum caliper dimension. This approach can lead to several difficulties. First, as pointed out previously, this value may seriously overestimate the actual minimum dimension for a long, narrow feature. Second, the direction of the minimum dimension is not, in general, perpendicular to the maximum dimension. Third, the resulting ellipse area is not the same as the area of the feature.

Basing the ellipse breadth on the minimum caliper dimension is suspect. As a result, a modification of this approach uses the maximum caliper dimension as the ellipse major axis, and determines the minor axis in order to make the ellipse area agree with the feature area. The area of an ellipse is $(\pi/4)\cdot a\cdot b$ where a and b are the axes; therefore, once the major axis has been determined, the minor axis can be adjusted to agree with the feature area. The orientation angle can be either the approximate value determined from the steps used in the maximum caliper dimension search, or the orientation angle determined from the moment calculation shown earlier (**Equation 7**). This tends to produce ellipses that have a longer and narrower shape than our visual perception of the feature.

The moments from **Equation 7** can also be used to produce a fitted ellipse. This is perhaps the most robust measure, although it requires the most calculation. Surprisingly, it seems to be little used. Instead, many systems fit an ellipse not to all of the pixels in the feature area, but instead to the pixels along the feature boundary. This procedure is computationally simple but very hard to justify analytically, and irregularities along any portion of the feature's boundary will significantly bias the ellipse.

Caliper dimensions

Caliper dimensions represent another description of feature size. The maximum caliper or maximum Feret's diameter is sometimes called the feature length, because it is the longest distance between any two points on the periphery. A projected or shadow dimension in the horizontal or vertical direction can be determined simply by sorting through the pixels or the boundary points to find the smallest and largest coordinates, and then taking the difference. These dimensions were introduced above in terms of the bounding box around the feature, used to correct for its probability of intersecting the edge of a randomly placed image field.

The extreme points determined in a variety of directions were used to construct the bounding polygon described above. With such a polygon, the minimum and maximum caliper diameters can be found simply by sorting through the pairs of corner points. The pair of vertices with the greatest separation distance is a close approximation to the actual maximum dimension of the feature. The worst-case error occurs when the actual maximum chord is exactly halfway between the angle steps, and in that case the measured length is short by

$$Meas.Value = True\ Value \cdot \cos\left(\frac{\alpha}{2}\right) \tag{13}$$

For the example mentioned previously of 10-degree steps, the cosine of 5° is 0.996. This means that the measurement is less than one-half percent low. For a feature whose actual maximum dimension is 250 pixels, the value obtained by the rotation of coordinate axes would be one pixel short in the worst-case orientation.

If the same method is used to determine a minimum caliper dimension for the feature (sometimes called the breadth), this is equivalent to finding the pair of opposite vertices on the bounding polygon that are closest together. The error here can be much greater than for the maximum caliper dimension, because it depends on the sine of the angle and on the length of the feature rather than its actual breadth. A narrow feature of length L and actual width W oriented at an angle the same 5° away from the nearest rotation step would have its breadth estimated as $L \cdot \sin(5) = 0.087 \cdot L$. This does not even depend on the width W, and if the actual length L is large and the width W is small, the potential error is very great.

Of course, it is also possible to search directly for the maximum chord by sorting through the points to find the two with maximum separation. There are various ways to speed up this search, which would otherwise have to calculate the sum of squares of the x- and y- coordinate differences between all pairs of boundary points. One is to consider only those points that lie outside the equivalent circle. Determining the area by pixel counting (in this case, the area including any internal holes) and converting it to the equivalent diameter with **Equation 12**, provides a circle size. Locating that circle with its center on the feature centroid will cover any points inside the circle, and leave just those points that lie farther away. These are candidates for the most widely separated pair.

An even more restrictive selection of the points to be searched can be obtained from the dimensions of the bounding rectangle. As shown in **Figure 44**, arcs drawn tangent to each side with their center in the middle of the opposite side will enclose most of the boundary of the feature. Only points that lie on or outside the arcs need to be used in the search, and points lying outside one arc need only to be combined with those that lie outside the opposite arc. For a reasonably complex feature whose boundary may contain thousands of points, these selection algorithms offer enough advantage to be worth their computational overhead when the actual maximum dimension is needed.

Consider a feature shaped like the letter "S" shown in **Figure 45**. If this is a rigid body, say a cast-iron hook for use in a chain, the length and breadth as defined by the minimum and maximum caliper dimensions may have a useful meaning. On the other hand, if the object is really a worm

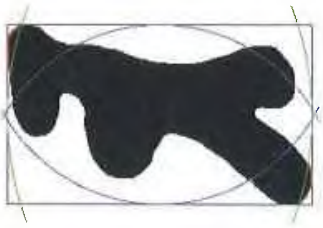


Figure 44. *Construction used to limit the search for boundary points giving the maximum chord. The blue arcs are drawn with centers in the middle of the horizontal edges of the bounding box, and the green arcs are drawn with centers in the middle of the vertical edges. Only the red points along the perimeter, which lie outside those arcs, are candidates for endpoints of the longest chord.*

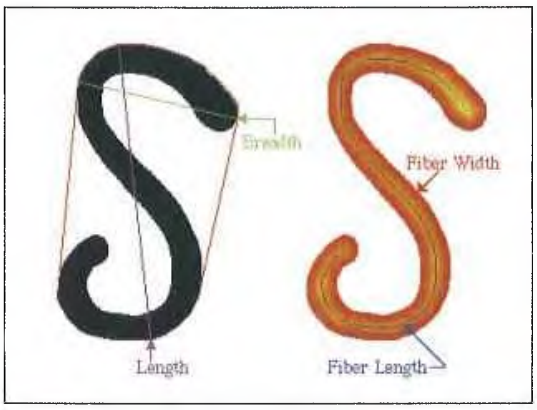


Figure 45. *Comparison of caliper dimensions (length and breadth) with values determined from the skeleton and the EDM (fiber length and fiber width)*

or fiber that is flexible, and the overall shape is an accident of placement, it would be much more meaningful to measure the length along the fiber axis and the width across it. To distinguish these from the maximum and minimum caliper dimensions, often called length and breadth, these are sometimes called the fiber length and fiber width.

Two quite different approaches are used to measure these dimensions. Of course, it is still up to the user to determine which set of parameters offers the most useful values for describing the features in a particular situation. One approach goes back to the binary image processing discussed in Chapter 7. If the feature is skeletonized, the length of the skeleton offers a measure of the length of the fiber. The skeleton is shorter than the actual length of the fiber, but this can be corrected by adding the value of the Euclidean distance map at each end point of the skeleton.

The length of the skeleton itself can be estimated by counting the pixel pairs that are present, keeping track of those that touch along their edges and their corners (because on a square pixel grid the distance between pixels that touch diagonally is greater by $\sqrt{2}$ than the distance orthogonally). This produces a slight overestimate of the length. Instead of using the strictly geometric distances of 1.0 and 1.414, Smeulders (1989) has shown that calculating the length as

$$\begin{aligned} Length = 0.948 \cdot (num.orthogonal\ nbors) \\ +1.340 \cdot (num.diagonal\ nbors) \end{aligned} \tag{14}$$

gives a mean error of only 2.5% for lines that run in all directions across the square pixel grid, but, of course, larger errors can be encountered for specific worst-case orientations.

A more accurate measurement can be performed by fitting a smooth curve to the pixels. Several ways are used to accomplish this, including polynomial and Bezier curves. The most efficient method is to use the same technique described below for perimeter measurement, in which the image is smoothed and a super-resolution line interpolated through the pixels.

As described in Chapter 7, combining the skeleton with the EDM also provides a tool to measure the fiber width. Assigning the grey-scale values from the EDM to the skeleton pixels along the midline represents the radius of the inscribed circle centered at each point. Averaging these values for all of the points in the midline gives the mean fiber width, and it is also possible to measure the variation in width.

An older and less accurate (but still used) approach to estimating values for the length and width of a fiber is based on making a geometric assumption about the fiber shape. If the feature is assumed to be a uniform-width ribbon of dimensions F (fiber length) and W (fiber width), then the area (A) and perimeter (P) of the feature will be $A = F \cdot W$ and $P = 2 \cdot (F + W)$. The area and perimeter parameters can both be measured directly. As we will see in the next section, the perimeter is one of the more troublesome values to determine. For a ribbon with smooth boundaries as assumed here, however, the perimeter can be measured with reasonable accuracy. Then the fiber length and width can be calculated from the measured perimeter and area as

$$F = \frac{P - \sqrt{P^2 - 16 \cdot A}}{4} \qquad (15)$$

$$W = \frac{A}{F}$$

Minor modifications to this model can be made, for example by assuming that the ends of the ribbon are rounded instead of square, but the principle remains the same. The difficulty with this approach is its sensitivity to the shape model used. If the feature is not of uniform width or is branched, for instance, the average width obtained from the skeleton and EDM still produces a consistent and meaningful result, while the calculation in **Equation 15** may not.

Perimeter

The perimeter of a feature seems to be a well-defined and familiar geometrical parameter. Measuring a numerical value that really describes the object turns out to be less than simple, however. Some systems estimate the length of the boundary around the object by counting the pixels that touch the background. Of course, this underestimates the actual perimeter because, as noted earlier for the measurement of skeleton length, the distance between corner-touching pixels is greater than it is for edge-touching pixels. Furthermore, this error depends on orientation, and the perimeter of a simple object like a square will change as it is rotated under the camera. **Figure 46** compares the variation in perimeter that is obtained by counting the edges of pixels to the more accurate value obtained from the chain code, as a square is rotated. The sensitivity of measurement values to orientation is often used as a test of system performance.

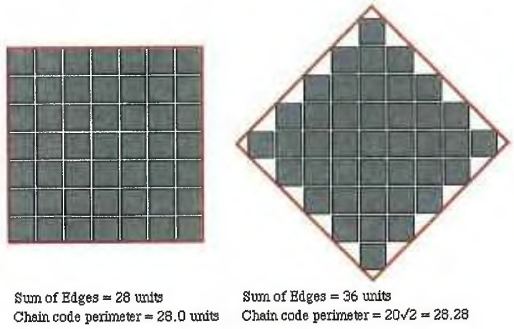


Figure 46. *Comparison of perimeter estimation by chain code vs. summation of pixel edge lengths, as a function of orientation.*

If boundary representation is used to represent the feature, then the Pythagorean distance between successive points can be summed to estimate the perimeter, as

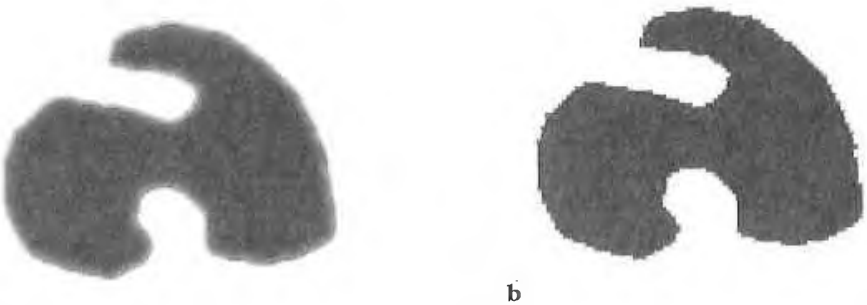
$$Perim. = \sum_i \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad (16)$$

In the limiting case of chain code, the links in the chain used for boundary representation are either 1.0 or 1.4142 pixels long, and can be used to estimate the perimeter. It is only necessary to count the number of odd chain code values and the number of even chain code values, since these distinguish the orthogonal or diagonal directions. The same argument as used above for the irregularity of the pixel representation of the midline applies to the boundary line, and using the modified values from **Equation 14** may be applied to reduce this source of error.

The most accurate perimeter measurements, with the least sensitivity to the orientation of feature edges, are obtained by fitting smooth curves to the feature boundaries. The easiest way to do this is by antialiasing the edge so that the pixels near the edge have grey values that fill in the steps (Neal et al., 1998), as shown in **Figure 47**. The process uses the Laplacian of a Gaussian (LoG) filter, which is also used in operators such as the Canny edge locator discussed in Chapter 4 to most accurately define the location of feature boundaries.

The actual boundary is constructed by drawing and measuring straight line segments across the antialiased pixels. These line segments are drawn between points along the pixel edges that are linearly interpolated based on the grey scale values. The points correspond to the location where the midpoint value would lie, treating the pixels as spaced points. This is functionally equivalent to enlarging the image using super resolution and bilinear interpolation to fill in the new, tiny pixels, as shown in **Figure 48**.

Although the super-resolution boundary measurement technique is very accurate and extremely robust to feature rotation, the basic difficulty with perimeter measurements is that for most objects the perimeter itself is very magnification-dependent. Higher-image magnification reveals more boundary irregularities and hence a larger value for the perimeter. This is not the case for area, length, or the other size dimensions discussed previously. As the imaging scale is changed so that the size of individual pixels becomes smaller compared to the size of the features, measurement of these other parameters will of course change, but the results will tend to converge toward a single best estimate. For perimeter, the value usually increases.



a b

Figure 47. Fitting a smooth boundary for perimeter measurement:
(a) *contour line on a smoothed (antialiased) feature;*
(b) *the contour line superimposed on the original pixels.*

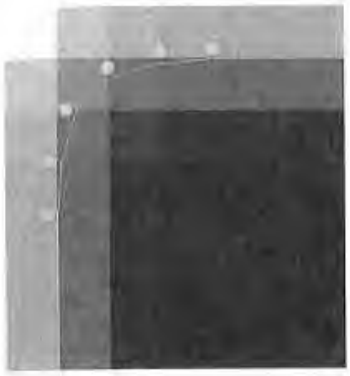
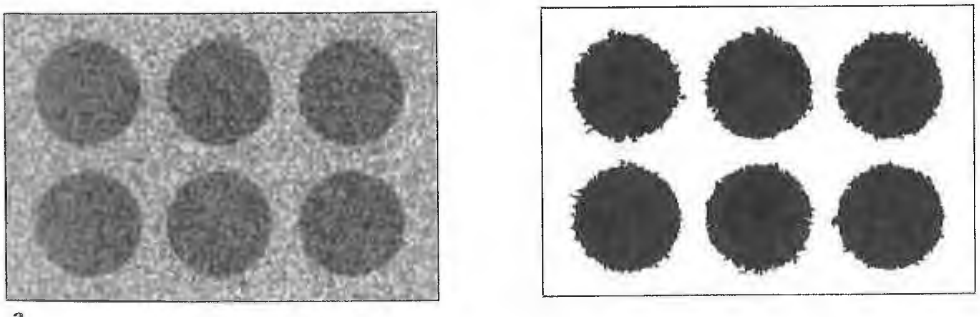


Figure 48. *Interpolating the boundary between points on the edges of the smoothed pixels, showing how an abrupt corner is smoothly rounded.*

In many cases, plotting the measured perimeter against the size of the pixels, or a similar index of the resolution of the measurement, produces a plot that is linear on logarithmic axes. This kind of self-similarity is an indication of fractal behavior, and the slope of the line gives the fractal dimension of the boundary. This will be used below as a measure of feature shape.

Even if the feature boundary is not strictly fractal (implying the self-similarity expressed by the linear log-log plot of perimeter vs. measurement scale), there is still often some increase in measured perimeter with increased imaging magnification. The only exceptions are smooth (Euclidean) objects such as membranes or surfaces in which tension or an energy term enforces local smoothness for physical reasons. This dependency on magnification makes the perimeter values suspect as real descriptors of the object, and at least partially an artefact of the imaging method and scale used. Further, any noise in the image may be expected to cause a roughening of the boundary and increase the apparent perimeter.

Figure 49 shows a simple example. Random grey-scale noise is superimposed on six circles. The original circles were drawn with a diameter of 80 pixels, and had a measured area of 5024 pixels and perimeter of 254 pixels. Measuring a series of noisy images will produce a value for the area that averages to the correct mean, as pixels are added to or removed from the feature, but the perimeter is always increased as shown by the results in **Table 1**. The variation in area measurements is only 0.5%, while that for the perimeter measurements is 8%, and the mean value is far too high. This bias introduced by noise in the imaging procedure is another cause for concern in perimeter measurements.



a b

Figure 49. Six identical circles superimposed on grey scale noise *(a), then thresholded, internal holes filled* *(b) and measured. The results are shown in Table 1.*

Table 1. Results from circle measurements in Figure 49

Circle			Area	Perimeter
1	5027	358		
2	5042	338		
3	4988	343		
4	5065	391		
5	5020	372		
6	5030	313		
average	5028.7	352.5		
Standard deviation.			25.4 (0.5%)	27.4 (8%)
original	5024	254		

If perimeter is measured, there is still the need to choose among the same three alternatives discussed for area measurements. The total perimeter includes the length of boundaries around any internal holes in the feature. The net or filled perimeter excludes internal holes and just measures the exterior perimeter. The convex perimeter is the length of the convex hull or bounding polygon and bridges over indentations around the periphery of the feature. The same considerations as mentioned in connection with area measurement apply to selecting whichever of these measures describes the aspect of the feature that is important in any particular application.

Describing shape

Shape is not something that human languages are well equipped to deal with. We have few adjectives that describe shape, even in an approximate way (e.g., rough vs. smooth, or fat vs. skinny). In most conversational discussion of shapes, it is common to use a prototypical object instead ("shaped like a . . ."). Of course, this assumes we all agree about the important aspects of shape found in the selected prototype. Finding numerical descriptors of shape is difficult because the correspondence between them and our everyday experience is slight, and the parameters all have a "made-up" character.

The oldest class of shape descriptors are simply combinations of size parameters, arranged so that the dimensions cancel out. Length/breadth, for example, gives us an "aspect ratio," and changing the size of the feature does not change the numerical value of aspect ratio. Of course, this assumes we have correctly measured a meaningful value for length and breadth, as discussed earlier.

Dozens of size parameters are possible; therefore, they can be combined in hundreds of ways into a formally dimensionless expression that might be used as a shape descriptor. In fact, only a few relatively common combinations are possible, but even these are plagued by total inconsistency in naming conventions. **Table 2** summarizes some of the most widely used shape parameters calculated as combinations of size measurements. Note that in any particular system the same parameter may be called by some quite different name, or the name shown for a different parameter in the table. Also, some systems define the parameters as the inverse of the formula shown, or may omit constant multipliers such as π.

The burden placed on the user, of course, is to be sure that the meaning of any particular shape descriptor is clearly understood and that it is selected because it bears some relationship to the observed changes in the shape of features, because it is presumably being measured in order to facilitate or quantify some comparison. **Figures 50** through **53** illustrate several of these parameters which distinguish between features. In general, each of them captures some aspect of shape, but of course none of them is unique. An unlimited number of visually quite different shapes can be created with identical values for any of these dimensionless shape parameters.

Table 2. Representative shape descriptors

$$Formfactor = \frac{4\pi \cdot Area}{Perimeter^2}$$

$$Roundness = \frac{4 \cdot Area}{\pi \cdot Maximum\ Diameter^2}$$

$$Aspect\ Ratio = \frac{Maximum\ Diameter}{Minimum\ Diameter}$$

$$Elongation = \frac{Fiber\ Length}{Fiber\ Width}$$

$$Curl = \frac{Length}{Fiber\ Length}$$

$$Convexity = \frac{Convex\ Perimeter}{Perimeter}$$

$$Solidity = \frac{Area}{Convex\ Area}$$

$$Compactness = \frac{\sqrt{\left(\frac{4}{\pi}\right)Area}}{Maximum\ Diameter}$$

$$Modification\ Ratio = \frac{Inscribed\ Diameter}{Maximum\ Diameter}$$

$$Extent = \frac{Net\ Area}{Bounding\ Rectangle}$$

Figure 50 shows four variations on one basic shape, which is stretched and smoothed by erosion and dilation. Notice that *formfactor* varies with surface irregularities, but not with overall elongation, while *aspect ratio* has the opposite behavior. **Figure 51** shows four variations on one basic shape with the values of several shape parameters. Examination of the values shows that the parameters vary quite differently from one shape to another. **Figure 52** shows several shapes with the values of their *curl*.

Figure 50. *Variations on a shape produced by erosion/dilation and by horizontal stretching. The numeric values for formfactor and aspect ratio (listed below) show that stretching changes the aspect ratio and not the formfactor, and vice versa for smoothing the boundary.*

Shape	Formfactor	Aspect ratio
A	0.257	1.339
B	0.256	2.005
C	0.459	1.294
D	0.457	2.017

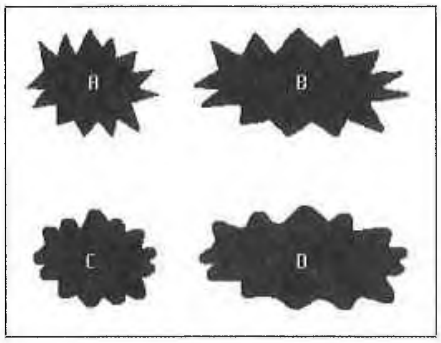
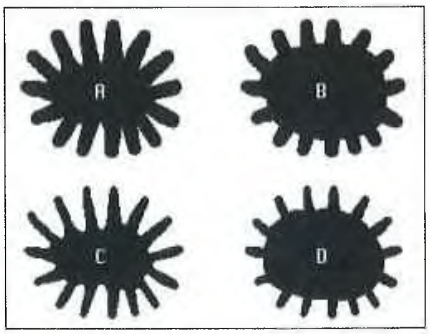


Figure 51. Another set of four related shapes, with the numeric values for their measured shape parameters.

Shape	Roundness	Convexity	Solidity	Compactness
A	0.587	0.351	0.731	0.766
B	0.584	0.483	0.782	0.764
C	0.447	0.349	0.592	0.668
D	0.589	0.497	0.714	0.768



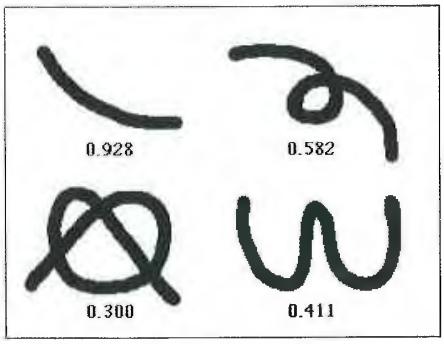


Figure 52. Four features with different values of curl, indicating the degree to which they are "curled up."

Figure 53 illustrates the differences between these parameters in a different way. The "features" are simply the 26 capital letters, printed in a font with a serif (Times New Roman). In each horizontal row, the colors code the value of a different measured shape parameter. The variation in each set is from red (largest numeric value) to magenta (smallest numeric value). The independent variation of each of the shape factors is remarkable. This suggests on the one hand that shape factors are a powerful tool for feature identification; however, the large variety of such factors, and the inability of human vision to categorize or estimate them, reminds us that people do not use these parameters to recognize letters.

Another group of simple dimensionless shape parameters can be calculated based on the various definitions of location introduced above. For a perfectly symmetrical feature, such as a circle, the

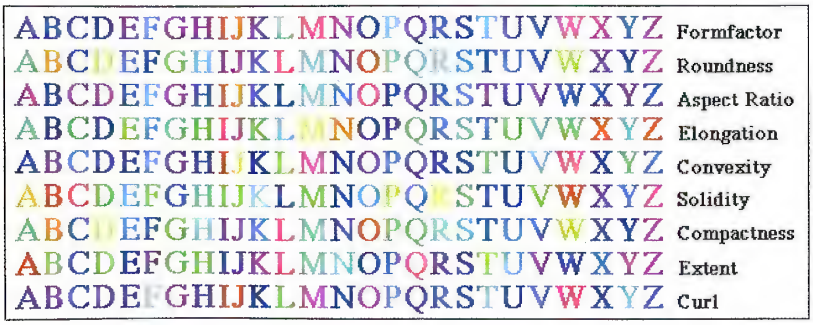


Figure 53. Measurement of a series of shapes (letters of the alphabet). In each row, the colors of the features code the relative numeric value (red = high, magenta = low) of a different shape parameter, corresponding to the labels.

centroid and geometric center (the center of the bounding circle) will coincide. If the feature is not symmetric, the distance between these two points, normalized by dividing by the radius of the bounding circle, produces a value between 0 and 1 that measures the asymmetry. The distance between the unweighted centroid (in which all pixels in the feature are counted equally) and the weighted centroid (in which pixels are counted according to some calibrated density function) can be used in the same way.

Similar ratios using other location measures are less useful. For example, the center determined as an average of the location of perimeter pixels is very resolution sensitive. As shown earlier in **Figure 11**, the roughness on the right side of the circle becomes visible only at high magnification, but would shift the apparent center of this visually symmetrical feature.

Likewise, the center of the largest inscribed circle in the feature also creates difficulties. Many nonconvex features have more than one maximum in their EDM. Each local maximum is the center of an inscribed circle (tangent to at least three points on the feature boundary). Using the largest as a measure of size is meaningful in some circumstances, but the location is rarely useful. As shown in **Figure 54**, the inscribed circle center is undefined when the feature has more than one equal maximum in the EDM, and even when it does not, the location is insensitive to many aspects of feature shape.

Fractal dimension

Quite a few of the shape parameters discussed previously and summarized in **Table 2** include the perimeter of the feature. As pointed out, this is often a problematic size parameter to measure, with a value that is an artefact of image magnification. In fact, the concept of perimeter may be fundamentally flawed when it is applied to many real objects. If a real-world object is actually fractal in shape, the perimeter is undefined. Measuring the object boundary at higher magnification will always produce a larger value of perimeter. Consider a cloud, for example. What is the length of the boundary around the projected image of the cloud? Measurements covering many orders of magnitude, from a single small cloud in the sky observed by visible light, up to entire storm systems viewed by radar or from weather satellites, show that the perimeter of clouds obeys fractal geometry. Presumably this trend would continue at smaller and smaller scales, at least down to the dimensions of the water molecules. What, then, does the perimeter really mean?

Objects that are fractal seem to be the norm rather than the exception in nature. Euclidean geometry with its well-defined and mathematically tractable planes and surfaces is usually only found as an approximation over a narrow range of dimensions where mankind has imposed it, or in limited situations where a single energy or force term dominates (e.g., surface tension). Roads, buildings,

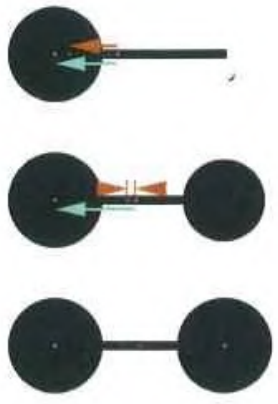


Figure 54. *Location of the geometric center (magenta), center of the inscribed circle (green), and centroid (red), showing the effects of asymmetry and shape.*

and the surface of the paper on which this book is printed are flat, straight and Euclidean, but only in the dimension or scale where humans perceive and control. Magnify the paper surface and it becomes rough. Look at the roads from space and they cease to be straight. The use of fractal dimensions to describe these departures from Euclidean lines and planes is a relatively recent conceptual breakthrough that promises a new tool for describing roughness.

Briefly, the fractal dimension is the rate at which the perimeter (or surface area) of an object increases as the measurement scale is reduced. A variety of ways are used to measure it. Some are physical, such as the number of gas molecules that can adhere to the surface in a monolayer, as a function of the size of the molecule (smaller molecules probe more of the small surface irregularities and indicate a larger surface area). Some are most easily applied in frequency space, by examining the power spectrum from a two-dimensional Fourier transform as discussed in Chapter 13. Others are relatively easily measured on a thresholded binary image of features.

Perhaps the most widely used fractal measurement tool is the so-called Richardson plot. This was originally introduced as a procedure applied manually to the measurement of maps. Setting a pair of dividers to a known distance, the user starts at some point on the boundary and strides around the perimeter. The number of steps multiplied by the stride length produces a perimeter measurement. As the stride length is reduced, the path follows more of the local irregularities of the boundary and the measured perimeter increases. The result, plotted on log-log axes, is a straight line whose slope gives the fractal dimension. Deviations from linearity occur at each end: at long stride lengths, the step may miss the boundary altogether, while at short stride lengths the finite resolution of the map limits the measurement.

When boundary representation is used for a feature, a Richardson plot can be constructed from the points by calculating the perimeter using the expression in **Equation 16**, and then repeating the same procedure using every second point in the list, every third point, and so forth. As points are skipped, the average stride length increases and the perimeter decreases, and a Richardson plot can be constructed. When every nth point in the list is used to estimate the perimeter, there are n possible starting positions. Calculating the perimeter using each of them, and averaging the result, gives the best estimate. The stride length for a particular value of n is usually obtained by dividing the total perimeter length by the number of strides. In all these procedures, it is recognized that there may be a partial stride needed at the end of the circuit, and this is included in the process.

When the x,y coordinates of the boundary points are digitized in a grid, a variability in the stride length is required. Unlike the manual process of walking along a map boundary, there may not be a point recorded at the location one stride length away from the last point, and so it is necessary either to interpolate from the actual data or to use some other point and alter the stride length. Interpolation makes the tacit assumption that the boundary is locally straight, which is in conflict with the entire fractal model. Altering the stride length may bias the plot. Combined with the general difficulties inherent in the perimeter measurement procedure, the classical stride length method is usually a poor choice for measuring digitized pixel images.

Using the super-resolution perimeter measurement, however, which naturally interpolates points as needed, another Richardson method can be employed. By applying smoothing kernels with progressively larger Gaussian standard deviations to blur the boundary, small irregularities are removed and the perimeter decreases. Plotting the measured perimeter vs. the standard deviation of the smoothing kernel, on log-log axes, produces a straight line whose slope gives the feature fractal dimension as shown in **Figure 55**. The dimension is a value greater than 1 (the topological or Euclidean dimension of a line) and 2 (the dimension of a plane), and may be thought of as the degree to which the line spreads out into the plane.

A second measurement technique was shown in the chapter on binary image processing. Dilating the boundary line by various amounts is equivalent to sweeping a circle along it. The area of the

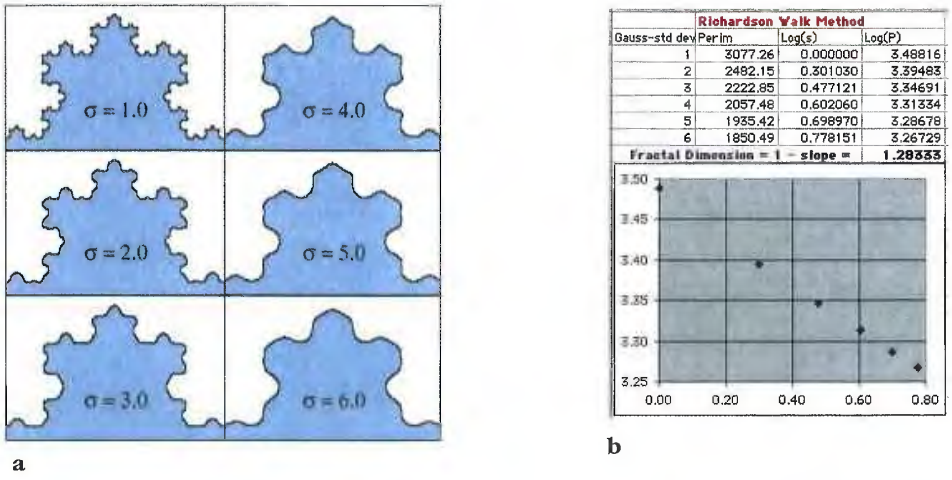


Figure 55. *Progressive Gaussian smoothing of the outline of a Koch snowflake (theoretical fractal dimension = 1.262) and perimeter measurement produces a Richardson measurement of the fractal dimension.*

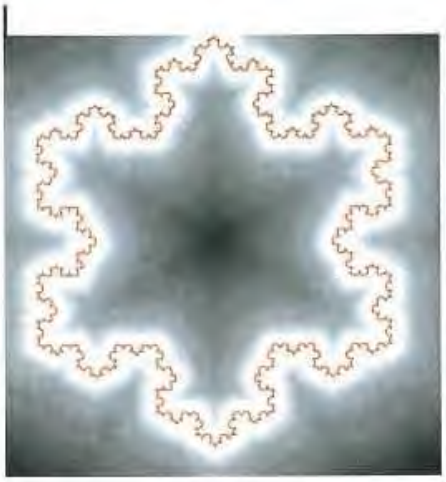
band swept over by the circle does not increase directly with the radius, because of the irregularities of the boundary. Plotting the area swept out by the circle (sometimes called the sausage) vs. the radius (again on log-log axes) produces a line whose slope gives the fractal dimension. This Minkowski technique is actually older than the Richardson method. It works well on pixel-based images, particularly when the EDM is used to perform the dilation. Using the cumulative histogram of the EDM of the region on either side of the feature outline to construct the log-log plot is the least sensitive to feature orientation, and much faster than iterative dilation, since it assigns to each pixel a grey scale value equal to the distance from the boundary. Examples of this method were shown in Chapter 7; **Figure 56** shows the measurement of the same Koch snowflake.

The Minkowski method produces a dimension that is not identical to the Richardson method (or to other fractal dimension procedures), and so it is important when comparing values obtained from different specimens to use only one of these methods. Neither method gives the exact result expected theoretically for the Koch snowflake fractal, due to the finite resolution available in the pixel image, but both are reasonably close.

A third approach to fractal dimension measurement produces another dimension, similar in meaning but generally quite different in value from the others discussed. The Kolmogorov dimension is determined manually by grid counting. A mesh of lines is drawn on the image and the number of grid squares through which the boundary passes is counted. When this number is plotted on log-log axes vs. the size of the grid, the slope of the line again gives a dimension, as shown in **Figure 57**. Automatic implementation and application to a pixel image can be performed by progressively coarsening the image into 2 × 2, 3 × 3, etc. blocks of pixels, and is sometimes called mosaic amalgamation. This is perhaps the fastest method but has the least numeric precision because it has the fewest possible steps and hence the fewest points to establish the line.

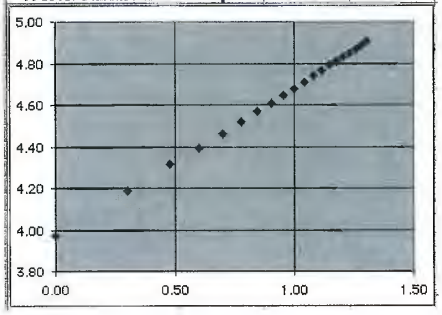
No matter how it is determined, the fractal dimension produces a single numeric value that summarizes the irregularity of "roughness" of the feature boundary. **Figure 58** shows examples of several natural fractals. The relationship between this boundary and what it may represent in three dimensions is not simple. For the case of a section through an isotropic surface, the boundary fractal dimension is smaller than the surface fractal dimension by exactly 1.0, the difference between the topological dimension of the sectioning plane and the volume in which the object resides. Few real

Figure 56. *The cumulative histogram of the Euclidean distance map around the outline of the Koch snowflake produces a Minkowski measurement of the fractal dimension.*



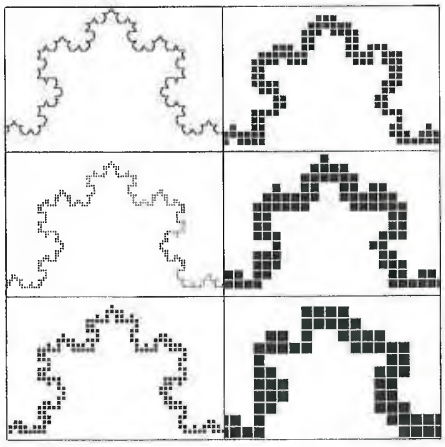
a

	Minkowski Sausage Method			
EDM Value	Pixel Count	Cumulative	Log(R)	Log(A)
1	9330	9330	0.00000	3.96988
2	5956	15286	0.30103	4.18429
3	5198	20484	0.47712	4.31141
4	4052	24536	0.60206	4.38980
5	4448	28984	0.69897	4.46216
6	4008	32992	0.77815	4.51841
7	4070	37062	0.84510	4.56893
8	3416	40478	0.90309	4.60722
9	3580	44058	0.95424	4.64402
10	3620	47678	1.00000	4.67832
11	3640	51318	1.04139	4.71027
12	3796	55114	1.07918	4.74126
13	3096	58210	1.11394	4.76500
14	3260	61470	1.14613	4.78866
15	3074	64544	1.17609	4.80986
16	3114	67658	1.20412	4.83032
17	2978	70636	1.23045	4.84903
18	3082	73718	1.25527	4.86757
19	3072	76790	1.27875	4.88530
20	3000	79790	1.30103	4.90195
Fractal Dimension = 2 - slope =			1.28257	



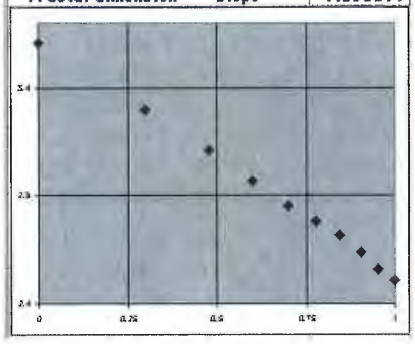
b

Figure 57. *Counting the boxes of increasing size through which the boundary of the Koch snowflake passes and the log-log plot. The plot is a straight line but the slope does not give an accurate fractal dimension for the shape.*



a

	Kolmogorov Box Count Method		
cell width	count	Log(w)	Log(N)
1	4020	0	3.60422605
2	1985	0.30103	3.29776051
3	1294	0.47712125	3.11193428
4	928	0.60205999	2.96754798
5	713	0.69897	2.85308953
6	603	0.77815125	2.78031731
7	521	0.84509804	2.71683772
8	432	0.90308999	2.63548375
9	363	0.95424251	2.55990663
10	322	1	2.50785587
Fractal dimension = - slope =			1.093811



b

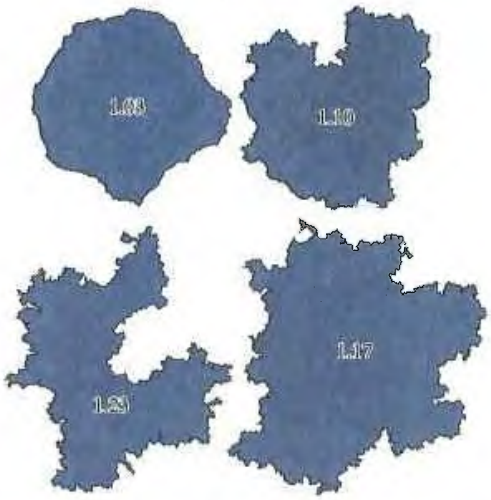


Figure 58. *Several fractal outlines with varying fractal dimensions.*

surfaces are perfectly isotropic, however, and in the presence of preferred orientations this simple relationship breaks down. Even more serious, if the boundary measured is a projected outline of a particle, the boundary irregularities will be partially hidden by other surface protrusions, and the measured fractal dimension will be too small by an amount that depends on the size of the particle and on its roughness. There is no general correction for this, and despite the fact that this procedure is very commonly used, the data obtained (and their subsequent interpretation) remain very open to question. Chapter 13 discusses the meaning and method of measurement for fractal surfaces, which have dimensions between 2 and 3.

Harmonic analysis

The fractal dimension attempts to condense all of the details of the boundary shape into a single number that describes the roughness in one particular way. There can, of course, be an unlimited number of visually different boundary shapes with the same fractal dimension or local roughness. At the other extreme it is possible to use a few numbers to preserve all of the boundary information in enough detail to effectively reconstruct the details of its appearance.

Harmonic analysis is also known as spectral analysis, Fourier descriptors, or shape unrolling (Schwartz and Shane, 1969; Ehrlich and Weinberg, 1970; Zahn and Roskies, 1972; Granlund, 1972; Beddow et al., 1977; Persoon, 1977; Flook, 1982; Kuhl and Giardina, 1982; Kaye et al., 1983; Rohlf and Archie, 1984; Barth and Sun, 1985; Ferson et al., 1985; Bird et al., 1986; Diaz et al., 1989; Rohlf, 1990; Diaz et al., 1990; Verscheulen et al., 1993; Lestrel, 1997). It begins by converting the boundary to a function of the form radius (angle) or $\rho(\varphi)$. As shown in **Figure 59**, a radius drawn from the feature centroid is drawn as a function of angle, and plotted to unroll the shape. This plot obviously repeats every 2π, and as a periodic or repeating function is straightforwardly subjected to Fourier analysis. This allows the determination of the *a* and *b* terms in the series expansion

$$\rho(\varphi) = a_0 + a_1\cos(\varphi) + b_1\sin(\varphi) + a_2\cos(2\varphi) + b_2\sin(2\varphi) + \ldots \quad (17)$$

It is often more convenient to represent the values as amplitude *c* and phase δ for each sinusoid:

$$c_i = \sqrt{a_i^2 + b_i^2}$$
$$\rho(\varphi) = \sum c_i \cdot \sin(2\pi i\varphi - \delta_i) \quad (18)$$

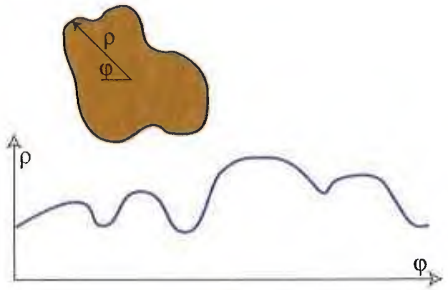


Figure 59. *Illustration of the procedure for unrolling a feature profile.*

This series continues up to an *n*th term equal to half the number of points along the periphery, although it is a characteristic of Fourier analysis that only the first few terms are needed to preserve most of the details of feature shape. As shown in **Figure 60**, with only one or two dozen terms in the series, the original shape can be redrawn to as good a precision as the original pixel representation. In many cases, the phase information δ_i for each term in the series can be discarded without much effect on the shape characterization, and a single coefficient c_i used for each frequency. The first few values of c in the harmonic or Fourier expansion of the unrolled boundary thus contain a great deal of information about the feature shape.

Of course, this method has serious problems if the shape is reentrant so that the radial vector is multivalued. In order to avoid that problem, an alternative approach to shape unrolling plots the slope of the line as a function of the distance along the line. This plot is also a repeating function, and can be analyzed in exactly the same way.

The chain code boundary representation of a feature already contains the slope vs. position information along the boundary, but with the problem that the data are not uniformly spaced (the diagonal links being longer than the horizontal and vertical ones). An effective way to deal with

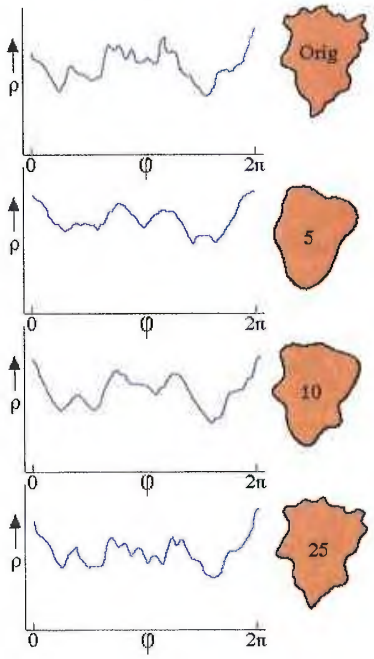


Figure 60. *Reconstruction of a real feature outline (an extraterrestrial dust particle) from the first 5, 10, and 25 terms in the Fourier expansion (from Kaye, 1983).*

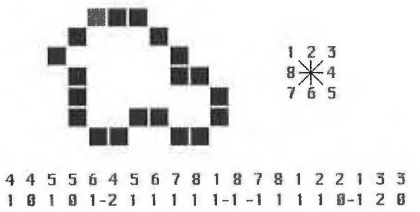


Figure 61. *Chain code representations for a feature outline, showing the conventional direction codes and below them the differential chain code.*

this is to replace each horizontal link with five shorter conceptual sublinks, each in the same direction, and each diagonal line with seven shorter sublinks. The ratio of 7:5 is close enough to $\sqrt{2}$ for practical purposes, and the total number of sublinks along the boundary is still reasonably short. A plot of the link value (which indicated direction) vs. position now gives the unrolled feature shape. Performing a Fourier analysis of the sequence of values along the chain code for the feature produces a list of c_i values that contain the shape information. The interpretation of these values is hampered because of the discontinuity in the direction numbers.

Chain code need not be recorded as a set of values corresponding to each of the eight possible pixel neighbor directions. The difficulty in analysis that results from the discontinuity between directions eight and one can be overcome by converting to the "first difference chain code" as shown in **Figure 61**. This is just the differences between values using modulo arithmetic, so that a zero indicates the edge proceeding straight ahead in whatever direction it was already, positive values indicate bends of increasing angle in one direction and negative values indicate the opposite direction. In this form, the shape is rotation-invariant (at least in 45-degree steps), except for the fact that the links in the original chain were not all of the same length. It is also easier to find corners, by defining them as a minimum net change in value over some distance (Freeman and Davis, 1977).

The most successful of the harmonic analysis techniques is to plot the x- and y- projections of the feature outline, treat them as real and imaginary parts of a complex number, and perform a Fourier transform on the resulting values as a function of position along the boundary, as shown in **Figure 62**. Again, this results in a set of c_i values that summarize the feature shape.

These can be compared between classes of objects using standard statistical tests such as stepwise regression or principal components analysis, to determine which of the terms may be useful for feature classification or recognition. In a surprising number of cases, this approach proves to be successful. The identification of particles in sediments with the rivers that deposited them, the correlation of the shape of foraminifera with the water temperature in which they grew, distinguishing the seeds from various closely related plant species, and the discrimination of healthy from cancerous cells in Pap smears, are but a few of the successes of this approach.

Despite its successes, the harmonic analysis approach has been little used outside of the field of sedimentation studies. In part, this neglect is due to the rather significant amount of computing needed to determine the parameters, and the need to apply extensive statistical analysis to interpret them. As computer power has continued to increase and cost to decrease, however, this cannot be the major reason any longer. The probable cause is that the frequency terms have no obvious counterpart in human vision. The shape information that we extract visually from images does not reveal these numeric factors. The distinction between two sediments based on the seventh harmonic coefficient can be understood intellectually to somehow represent the presence or amplitude of that frequency in the boundary irregularities of the object, but to a human observer that is masked by other variables. The success of the approach illustrates the power of computer-based measurement algorithms to surpass human skills not only quantitatively (in terms of accuracy and precision) but even qualitatively (in terms of the types of things that can be measured). But that doesn't mean that humans feel comfortable using such tools.

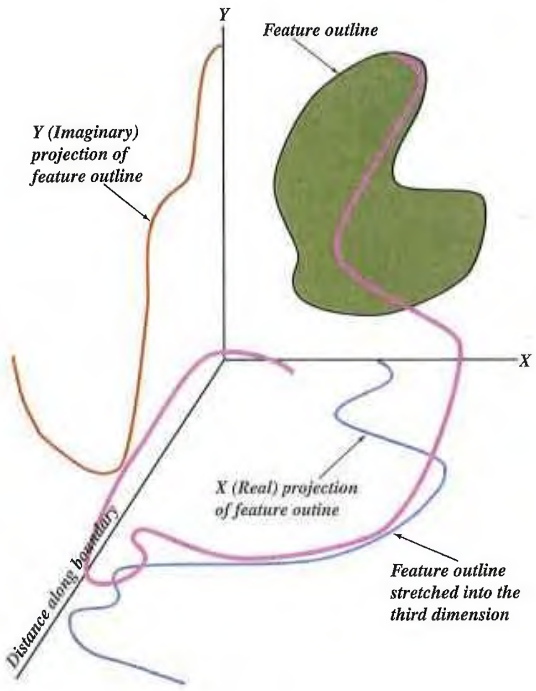


Figure 62. *Unrolling a feature shape by plotting the x- and y- coordinates of boundary points as a function of position along the boundary, and combining them as real and imaginary parts of complex numbers.*

Topology

Harmonic analysis uses characteristics of feature shape that are quite different from those that human vision selects. On the other hand, topological parameters are those which seem most obvious to most observers. When asked to differentiate the stars on the U.S., Australian, and Israeli flags we do not use color, dimension or angles, but the number of points. The most obvious difference between a disk and the letter "O" is not the slight ellipticity of the latter, but the presence of the central hole. Topological properties are quite different from metric ones. If the feature were drawn onto a sheet of rubber, stretching it to any size and with any distortion will not change the topology. Smoothing an irregular outline does not alter the topological information.

Some topological properties of features can be determined directly from the pixel array, such as the number of internal holes. Other properties, such as the number of points on the stars mentioned previously, are most readily determined using the skeleton. As discussed in Chapter 7, the skeleton consists of pixels along the midline of the features. Pixels with only a single touching neighbor are end points, while those with more than two neighbors are branch points. Counting the branch and end points, and the number of loops (holes), provides a compact topological representation of feature characteristics. **Figure 63** shows an example of using the end points to characterize shape.

These characteristics of features are important to visual classification and recognition, as can be easily shown by considering our ability to recognize the printed letters in the alphabet regardless of size and modest changes in font or style. Many character recognition programs that convert images of text to letters stored in a file make heavy use of topological rules.

Another topological property of a feature is its number of sides. In the case of a space-filling structure such as cells in tissue, grains in a metal, or fields in an aerial survey, the number of sides each feature has is the number of other features it adjoins. This was discussed previously in its relation to relative feature position. Labeling the features according to the number of neighbors can reveal some interesting properties of the structure, as was shown earlier in **Figures 21** and **22**.

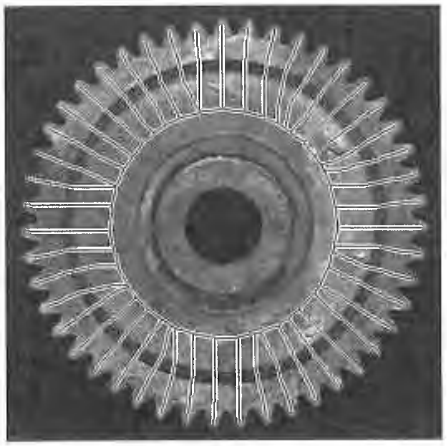


Figure 63. *Skeleton of an image of a gear used for topological characterization. The 47 end points efficiently count the number of teeth, and, of course, there is one hole.*

The number of sides that a feature has can also be described in another way. If corner points are defined as arbitrarily abrupt changes in boundary direction, then counting the number of sides can be accomplished directly from the chain code. A second approach that is less sensitive to minor irregularities in the boundary pixels uses the convex or bounding polygon. As described earlier, this polygon is usually constructed with some fixed and relatively large number of sides. For instance, performing rotation of axes in 10-degree steps would form a 36-sided polygon. Many of the vertices in this polygon may be close together if the feature shape has a relatively sharp corner. Setting an arbitrary limit on the distance between vertices that can be merged (usually expressed as a percentage of the total polygon perimeter) allows collecting nearby vertices together. Of course, counting the vertices is equivalent to counting the sides. Setting a threshold for the minimum length of the polygon side (again, usually as a percentage of the total perimeter) to consider it as a representation of a side of the feature can be used to count sides.

None of the four types of shape characterization (dimensionless ratios, fractal dimension, harmonic analysis, and topology) is completely satisfactory. One (harmonic analysis) is too unfamiliar and different from the human interpretation of shape. One (dimensionless ratios) is convenient for calculation but not very specific. One (fractal dimension) corresponds well with the human idea of boundary roughness but does not capture more macroscopic aspects of shape. Also, one (topology) is just the opposite in that it represents the gross aspects of feature shape but not the fine details. In real situations, several complementary techniques may be needed.

Three-dimensional measurements

Many of the measurement procedures discussed above for 2D images generalize directly into three dimensions. Images acquired by confocal light microscopy, seismic imaging, medical or industrial tomography, and even, in some cases, serial sectioning can be represented in many different ways, as discussed in Chapter 12. For image processing and measurement purposes, however, an array of cubic voxels (volume elements, the 3D analog to pixels or picture elements) is the most useful. Image processing in these arrays uses neighborhoods just as in two dimensions, although these contain many more neighbors and consequently take longer to apply (in addition to the fact that there are many more voxels present in the image).

Image measurements still require identifying those pixels that are connected to each other. In two dimensions, it is necessary to decide between an 8-connected and a 4-connected interpretation of touching pixels. In three dimensions, voxels can touch on a face, edge, or corner (6-, 18- or 26-connectedness); again, the rules for features and background cannot be the same. It is more difficult

to identify internal holes in features because the entire array must be tested to see if there is any connection to the outside, but the logic remains the same.

This consistency of principle applies to most of the measurements discussed in this chapter. Summing up the numeric values of voxels (or something calculated from them) to obtain the total density, or water content, or whatever property has been calibrated, is straightforward. So are location measurements using the voxel moments. Orientations in 3D space require two angles instead of one. Neighbor relationships (distance and direction) have the same meaning and interpretation as in two dimensions.

The 3D analog to feature area is feature volume, obtained by counting voxels. The caliper dimensions in many directions and the bounding polyhedron can be determined by rotation of coordinate axes and searching for minimum and maximum points. Of course, getting enough rotations in three dimensions to fit the polyhedron adequately to the sample is much more work than in two dimensions. In fact, everything done in 3D voxel arrays taxes the current limits of small computers: their speed, memory (to hold all of the voxels), displays (to present the data using volumetric or surface rendering), and the human interface. For instance, with a mouse, trackball, or other pointing device it is easy to select a location in a 2D image. How do you accomplish this in three dimensions? Various schemes have been tried (some requiring a real-time stereo display), none with wide acceptance.

It was noted earlier that perimeter is a somewhat troublesome concept and a difficult measurement in two dimensions. The analog to perimeter in three dimensions is the surface area of the feature, and it has all of the problems of perimeter plus some more. The idea that the surface area may be an artefact of voxel resolution remains and is exacerbated by the somewhat coarser resolution usually available in 3D images. Measuring the length of the perimeter accurately is difficult in two dimensions. For a 3D surface, the boundary representation is the list of coordinates of vertices of a polyhedron with triangular facets. Calculating the area of a triangle from its three corners is straightforward, but knowing which three points to use for any particular triangle is not. As a very simple example, consider four points as shown in **Figure 64**. The surface can be constructed between them two different ways, with different surface areas (usually both are calculated and the average used).

Boundary representation in two dimensions relies on the fact that there is only one path around any feature, no matter how complicated. This is not true in three dimensions. There is no unique order in which triangular facets between boundary voxels must or can be followed. In fact, for objects that are topologically shaped like a torus (have at least one open hole through them), there is no guarantee that a continuous surface path will completely cover the surface and reach all points on it. This means that there is no convenient analog to chain code, and many of the 2D measurements that were based on it become difficult to perform in three dimensions.

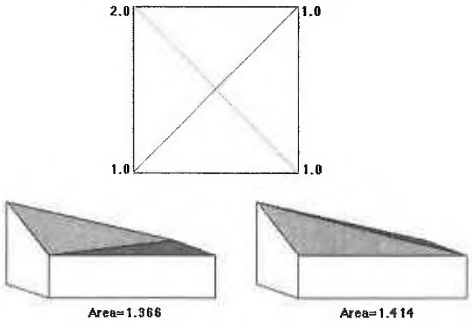


Figure 64. *Tiling a surface with triangular facets has two different solutions. In this simple example, the elevation of the four corners of a square are shown. Triangles can be constructed using either diagonal as the common line. As shown, this produces a surface that is either convex or concave, with a surface area of either 1.366 or 1.414 square units.*

Likewise, the 3D skeleton is harder to obtain. Actually, two "kinds" of skeleton can be calculated for voxel arrays. One is the set of midlines, and the other is a set of planes. The latter is formed by connecting the skeleton lines in each plane section, while the former is a connection between the junction points of the skeletons in each plane section. This set of midlines can also be obtained as a medial axis transform, by using the 3D analog of the EDM to find the centers of inscribed spheres. Neither type of skeleton by itself captures all of the topological properties of the feature. The planes can show twists and protrusions not visible in the line skeleton, for example.

Measuring the length of a voxel line is similar to a pixel line. The voxels can touch in three different ways; therefore, the rule shown earlier in **Equation 14** must be extended to become:

$$\begin{aligned} Length = {} & 0.877 \cdot (num.of\ face\ touching\ neighbors) \\ & + 1.342 \cdot (num.of\ edge\ touching\ neighbors) \\ & + 1.647 \cdot (num.of\ corner\ touching\ neighbors) \end{aligned} \tag{19}$$

although deciding what voxel patterns to count to identify the important topological properties of the structure is not entirely clear.

Most of the shape parameters calculated as ratios of dimensions have simple (although equally limited) generalizations to three dimensions. Harmonic analysis can also be performed in three dimensions, by expressing the radius as a function of two angles and performing the Fourier expansion in two dimensions. There is no analog to chain code or the description of slope as a function of position, so the method is restricted to shapes that are not re-entrant or otherwise multiple-valued (only one radius value at each angle). Fractal dimensions are very important for dealing with 3D surfaces and networks, although in most cases it remains easier to measure them in two dimensions.

In fact, this generalization holds for most image measurement tasks. The practical difficulties of working with 3D voxel arrays are considerable. The size of the array and the amount of computing needed to obtain results are significant. Because of memory restriction, or limitations in the resolution of the 3D-imaging techniques, the voxel resolution is usually much poorer than the available pixel resolution in a 2D image of the structure. Consequently, to the extent that the needed information can be obtained from 2D images and related by stereology to the 3D structure, that is the preferred technique. It is faster and often more precise.

This approach does not work for all purposes. Strongly anisotropic materials require so much effort to section in enough carefully controlled orientations, and are still so difficult to describe quantitatively, that 3D imaging may be preferred. Above all, topological properties of the structure such as the number of objects per unit volume, or the connectivity of a network, are simply not accessible on 2D sections (although a technique using two such parallel planes to sample 3D topology was presented in Chapter 8). Many aspects of 3D topology can only be studied in three dimensions, and consequently require 3D imaging, processing, and measurement.

The discussion of 2D images was based on an array of pixels, and so far all of the discussion of 3D images has assumed they consist of voxels. As will be discussed in Chapter 10, many 3D structures are studied by obtaining a set of parallel section planes that are relatively widely spaced. This is generally described as a "serial section" technique, although physical sectioning is not always required. Constructing the object(s) from the sections is discussed in Chapter 12.

Measuring the features from the sections is computationally straightforward. The volume is estimated as the summation of section area times section spacing, and the surface area can be estimated as the summation of the section perimeter times section spacing, but few objects have perfectly smooth surfaces. The section perimeter will reveal the roughness in the plane of the section,

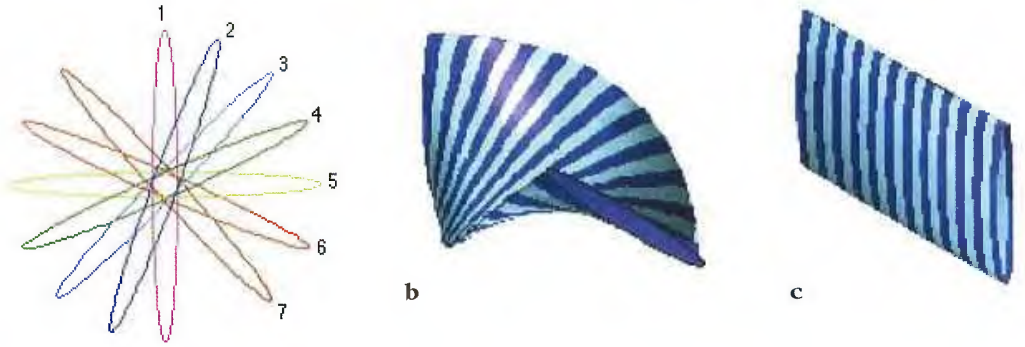


Figure 65. Reconstruction of a spiral structure:
(a) *the individual section outlines;*
(b) *the correct reconstruction of the surfaces;*
(c) *the incorrect reconstruction that results if the outlines are rotated for a "best fit."*

but the reconstruction technique generally connects the section profiles together with planes that are perfectly smooth, which underestimates the surface area. Even if the surface is smooth, the volume and surface area will be underestimated if it is curved.

Aligning the sections correctly is vital both to correctly understand the topology of the structure, and to measure the volume and surface. In the simple example shown in **Figure 65**, the actual structure is a spiral. If the serial sections are rotated to achieve a "best fit" from one section to the next, the result is a straight, untwisted structure with less volume and surface area than the actual structure.

Only when the sections are very closely spaced do these problems go away. In the limit when the section spacing is the same as the resolution within the individual section images, it is equivalent to a cubic voxel array. This is generally preferred for measurement purposes.

10

Feature Recognition and Classification

Template matching and cross-correlation

Recognition and classification are essentially complementary functions that lie at the "high end" (i.e., require the most complicated algorithms) in the field of image analysis (Duda and Hart, 1973). Classification is concerned with establishing criteria that can be used to identify or distinguish different populations of objects that may appear in images. These criteria can vary widely in form and sophistication, ranging from example images of representatives of each class to numeric parameters for measurement, to syntactical descriptions of key features. Recognition is the process by which these tools are subsequently used to find a particular feature within an image. It functions at very different levels, including such different processes as finding a face in an image, or matching that face to a specific individual.

Generally, computers tend to be better than humans at classification because they are not distracted by random variations in noncritical parameters and can extract useful statistical behavior to isolate groups. Sometimes these groups are meaningful. On the other hand, people are generally much better (or at least faster) at recognition than are computers, because they can detect the few critical factors that provide identification of familiar objects. But humans do not fare as well with unfamiliar objects or even with unfamiliar views of common ones.

Various recognition and classification techniques involve an extremely broad range of complexities. At the low end, operations such as scanning UPC bar codes (Universal Product Code) on supermarket packaging or automatic sorting and routing mail to the proper zip code, are based on careful design of the target to facilitate the reading function. These are normally detected by linear scans instead of acquiring full images as used elsewhere in this text, and UPC codes are symmetric (with black-and-white reversal) so that the scan can be used in either direction.

The basic statistics behind classification is independent of image analysis, since the parameters used for classification and identification can come from any source. A good introduction to the statistics can be found in textbooks such as (Divijver and Kittler, 1980; Schalkoff, 1992; Haykin, 1993; Stefik, 1995; Hand, 1981, Pao, 1989).

Probably the lowest level functionality that utilizes two-dimensional pixel images is the restricted optical character recognition (OCR) of the type applied to processing checks. The characters

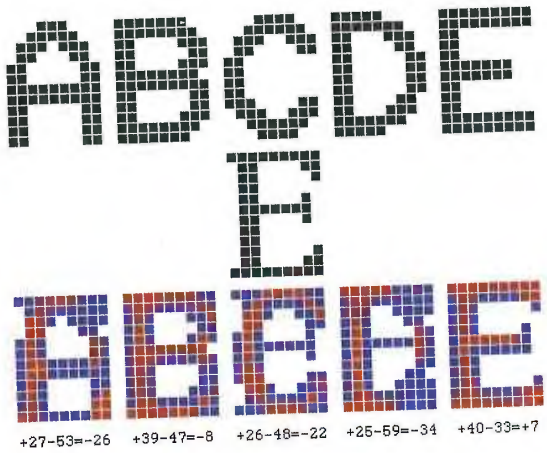


Figure 1. *Matching each of the five letter templates with a target letter produces a net score (sum of red matched pixels minus blue unmatched pixels) that is greatest for the correct match.*

printed on bank checks are restricted to numerals 0–9 and a few punctuation marks. Furthermore, the characters are in a fixed size, location and orientation, and in a special fixed font designed to make them easily distinguishable. In such highly constrained situations, a technique known as template matching provides fast results with minimum computation.

The example in **Figure 1** shows the application of template matching to the letters A through E. A template consisting of black pixels that cover the shape of each target character is stored in memory. Each letter to be recognized is combined with all of the stored templates using an exclusive-OR function to count the number of pixels that match and the number which do not. The template that gives the highest net score (number of matches minus number of misses) is selected as the identification of the character. In some implementations, the net score may be normalized by dividing by the number of black pixels in the template, if this varies substantially for the various characters.

In the example, the templates are not exact and have enough width and extra pixels to cover a modest range of variation in the target character, but obviously cannot handle widely divergent fonts or styles. The similarities in shape between the various letters (e.g., B and E) do not create a serious problem and the error rate for this approach is very low, provided that the characters meet the assumptions of size, orientation, font, etc., that are built into the templates. The presence of small amounts of noise due to dirt, etc., is tolerated as well.

This type of template matching is a rather specialized and restricted case of cross correlation, which can be used for grey scale or color images as well as binary pixel arrays. Chapter 5, on image processing in Fourier space, introduced the cross-correlation procedure as a technique for matching a target image with a second image to locate brightness patterns. This method can be applied to images of text or any other features, provided that a set of representative target images can be established.

It is not necessary to perform cross correlation using a Fourier transform. When the target is very small, it may be efficient to perform the mathematics in the spatial domain. The target pattern is moved to every possible position in the image and at each location i,j the cross-correlation value is summed according to **Equation 1**, to produce a result as shown in **Figure 2**.

$$\frac{\sum_{x=0}^{s}\sum_{y=0}^{s} B_{x+i,y+j} \cdot T_{x,y}}{\sqrt{\sum_{x=0}^{s}\sum_{y=0}^{s} B_{x+i,y+j}^2 \cdot \sum_{x=0}^{s}\sum_{y=0}^{s} T_{x,y}^2}} \quad (1)$$

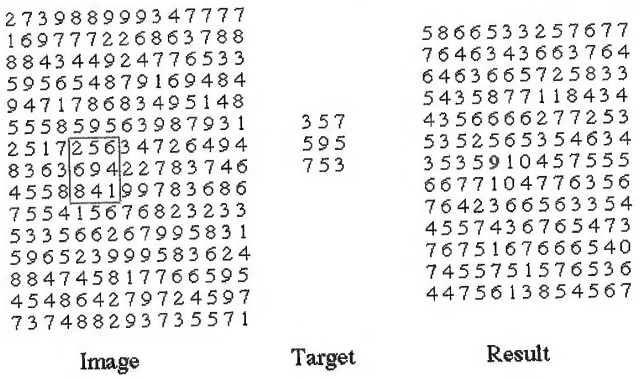


Figure 2. *Cross correlation using pixels with single digit values. The best match is marked in red (note that it is not an exact match).*

where B and T are the pixel brightness values for the image and target, respectively, and the summations are carried out over the size s of the target. This value is very high where bright pixels in the target and image align, and the denominator normalizes the result for variations in overall brightness of the target or the image. In addition to providing a quantitative measure of the degree of match, the correlation score can be scaled appropriately to produce another grey-scale image in which brightness is a measure of how well the target matches each location in the image, and can be processed with a top hat filter, or thresholded, to detect matches that are present.

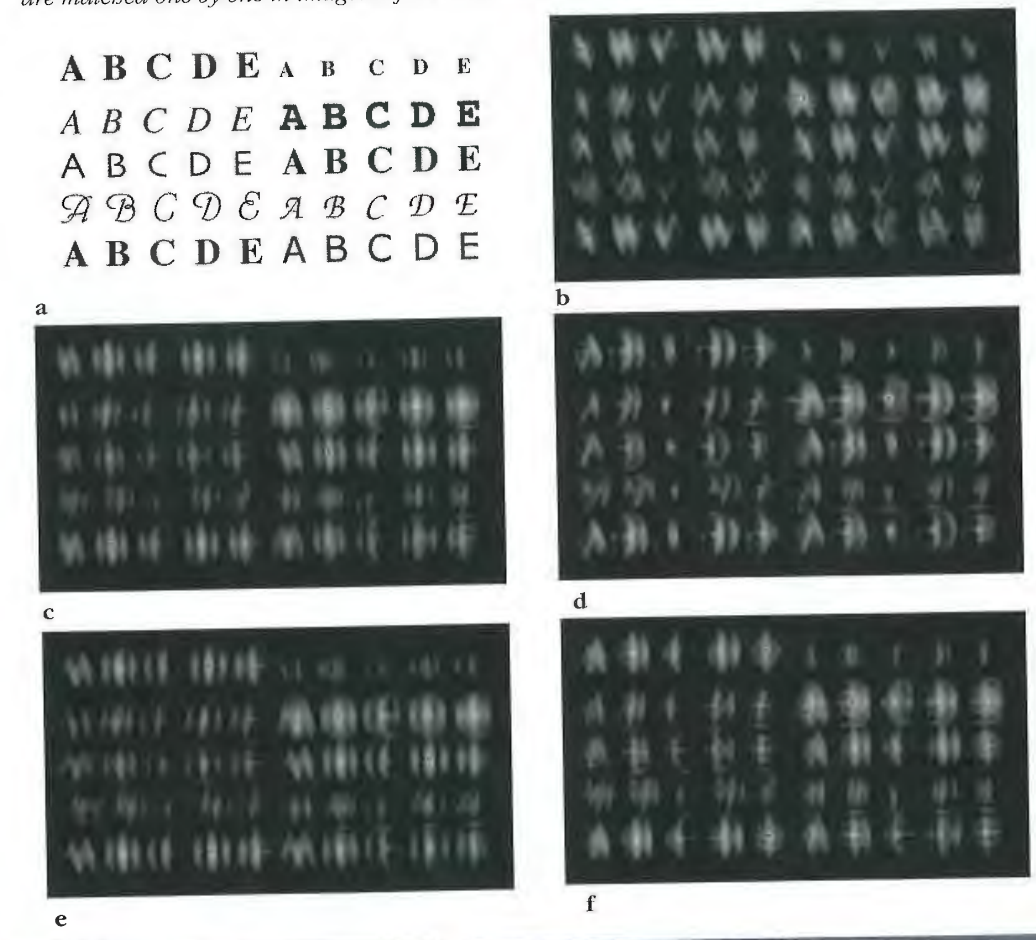
Figure 3 shows an example image from Chapter 5, in which several different fonts have been used for the letters A through E. Each of the letters in the first set (which is repeated in sets 6 and 9) was used as the target. The cross-correlation results are marked with red dots showing the maximum values of cross-correlation. The letter is found where it recurs in the same font, but most of the other fonts are not matched at all. The exception, font number 4, shows that the characters are similar enough to be matched but that they also generate false matches.

Cross-correlation looks for very specific details of feature shape and brightness, and while it is reasonably robust when a feature is partially obscured (by noise or camouflage), it is not very tolerant of changes in size or orientation. **Figure 4** shows an example of one of the major uses of the technique, the identification of military targets. The target image (an F-15 aircraft) produces a cross-correlation value that drops to 84% with a 10-degree rotation, and 80% for a 20-degree rotation. Both values are still higher than the correlation with different aircraft, which are 72% for an F-14 and 47% for a B-1. Of course, the images must be scaled to the same approximate size before the calculation is performed. In practice, it would be necessary to have target images at many different orientations. Views in other directions (from the side, front, oblique views, etc.) would also be required.

Parametric description

Recognition of features in images covers an extremely wide range of applications. In some cases the targets are fully known and can be completely represented by one or several images, so that cross-correlation is an appropriate method. In others, the goal is to have the computer "understand" natural three-dimensional scenes in which objects may appear in a wide variety of presentations. Applications such as automatic navigation or robotics require that the computer be able to extract surfaces and connect them hierarchically to construct three-dimensional objects, which are then recognized (see, for example, Roberts, 1982; Ballard and Brown, 1982; Ballard et al., 1984). The topics and goals discussed here are much more limited: for the image analysis system to be able to recognize discrete features in essentially two dimensional scenes. If the objects are three dimensional and can appear in different orientations, then each different two-dimensional view may be considered as a different target object.

Figure 3. *Cross correlation with letters in different fonts. The letters in the first set (repeated in set 6 and 9) are matched one by one in images b–f as discussed in the text.*



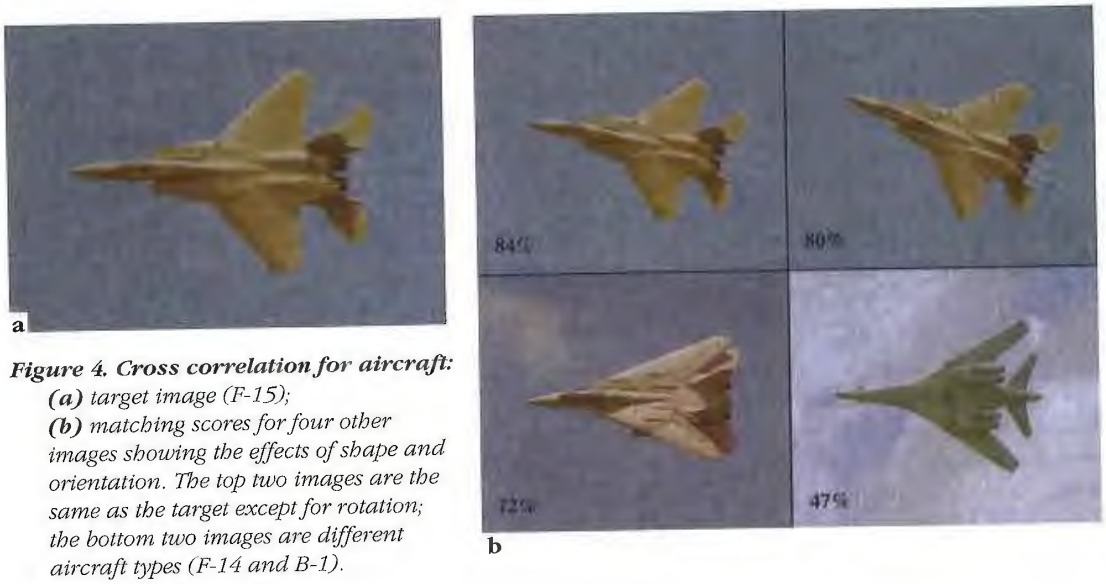


Figure 4. Cross correlation for aircraft: ***(a)*** *target image (F-15);* ***(b)*** *matching scores for four other images showing the effects of shape and orientation. The top two images are the same as the target except for rotation; the bottom two images are different aircraft types (F-14 and B-1).*

The simplest situation, but one that satisfies a very large percentage of practical applications, uses the feature-specific measurement parameters introduced in the previous chapter. These methods are called parametric descriptions. **Figure 5** shows a simple example, in which the features can be grouped into two classes based on their shape. Several of the shape descriptors introduced before can be used here; the example shows the formfactor (4π area/perimeter^2). A distribution of formfactor values for the features in the image shows two well separated populations. Setting a limit between the two groups can separate them into the respective classes. In the figure, the "round" features have been identified with a red mark.

Note that other measurement parameters such as area (**Figure 5c**) do not distinguish between the groups. Finding a single parameter that can be successfully used to separate classes is not usually possible, and when it is, selecting the best one from the many possible candidates by trial and error can be a time-consuming process.

In most cases, a combination of parameters is needed for successful classification, and statistical methods are used to find the best ones. As a simple thought experiment (from Castleman, 1996), consider a system to sort fruit, where the target classes are apples, lemons, grapefruit, and cherries. Apples and cherries can be distinguished on the basis of size (e.g., diameter), as can lemons and grapefruit. But apples have a range of sizes that overlap lemons and grapefruit, so a second parameter is needed to distinguish them, for instance the average hue, distinguishing between red and yellow (at least, this will work if green apples are excluded from the process). This can best be shown as a two-dimensional plot of parameter space, as shown in **Figure 6**.

Note that the various classes are still distinct in this example, so that drawing "decision lines" between them is straightforward once the range of values for the different classes has been established. This is most typically done by measuring actual samples. Rather than just measuring "typical" specimens, it would be most efficient in this case to intentionally select examples of each fruit that are considered to be extremes (largest, smallest, reddest, yellowest) to map out the boundaries.

The goals in deciding upon the measurement parameters to use are:

1. The ability to discriminate the various classes, preferably with no overlap at all but at least with minimum overlap (handling of situations in which there is overlap is described in the following paragraphs)

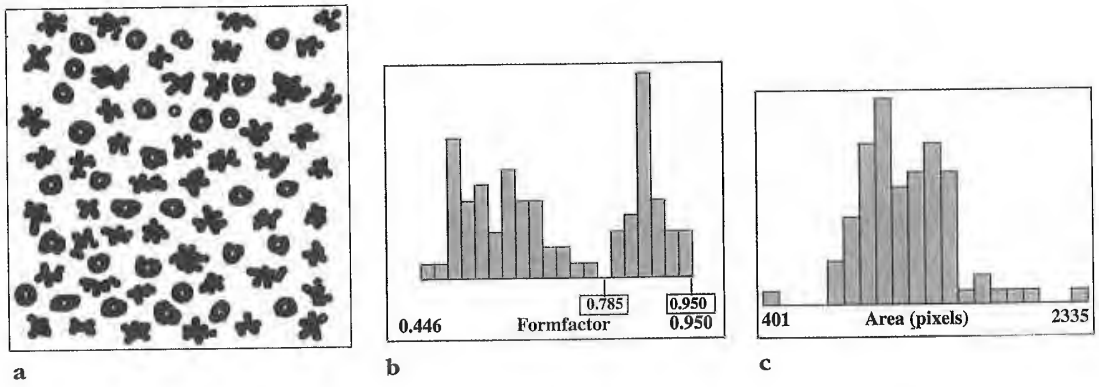
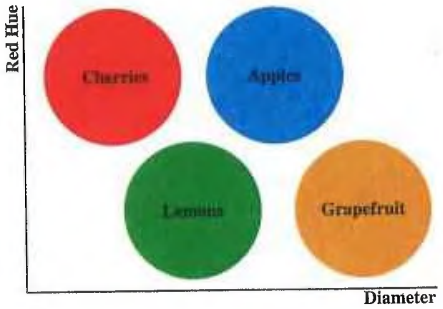


Figure 5. *Some hand-drawn features, with their histograms for formfactor* ***(b)*** *and area* ***(c)****. The formfactor histogram shows two separate classes; the features with values greater than 0.785 are marked with red dots and are visually "rounder" than the others. The area histogram shows only a single class and does not distinguish the two types of features.*

Figure 6. *Classes for fruit sorting discussed in the text.*



2. To have the smallest practical number of parameters, which simplifies the training process in which extremes in all combinations of parameters are desired
3. To be independent of each other, so that each one measures a very different characteristic of the objects
4. To be reliable, so that the parameter values will not vary over time or due to uncontrolled variables, and they can be measured consistently if system hardware is modified or replaced

Sometimes, these goals conflict with one another. For example, adding more parameters often provides less overlap of classes, but it may complicate the training process. Finding multiple parameters that are truly independent can be difficult. In the case of the apples, one parameter is a measure of size and the other of color, so we may expect them to be independent. But shape parameters, which are often very important for classification, may be quite interrelated to each other and to the size parameters from which they are calculated. It usually requires a statistical procedure to determine which function best in any particular situation (an example is shown in the following).

Reliability is difficult to predict when a classification scheme is initially established. By definition, changes in object appearance due to unexpected variations cannot be excluded. For instance, if color is being used to distinguish fabricated components and the pigmentation used in the ink being applied is changed by the supplier, or incandescent lights are replaced by fluorescent ones, or a different camera is substituted for one that has failed, the RGB values and whatever hue or other information is derived from them will change. It may or may not be practical to introduce a calibration step that will adjust the new values to match the old. The alternative is to remeasure a training population using the new ink, lights or camera, and reestablish class limits.

Three situations are encountered in general classification. Each situation will be described in this chapter:

1. *Imposed criteria (the classical expert system)* — A human expert supplies rules. Software may optimize the order in which they are applied and search for relevant rules, but does not derive them
2. *Supervised classification* — This is a training set of examples that are supposed to be prototypes of the range of classes and class variations are presented to the system, which then develops a strategy for identifying them. The number of classes and examples identified for each class are specified by a human.
3. *Unsupervised classification* — The system is presented with a set of examples as described previously, but not told which class each belongs to, or perhaps even the number of classes. The system attempts to define the class regions in order to maximize the similarity of objects within each cluster and the differences between groups.

There are other situations in which classification of images is not based on the statistics of individual feature measurement parameters, but on characteristics of the entire scene. The most often cited example of this type of discrimination uses the Brodatz (Brodatz, 1966) texture images representing various kinds of natural and man-made surfaces. A few are shown in **Figure 7**. The Fourier transform power spectra of these images (**Figure 8**) reveal characteristic frequencies and orientations, or ratios of amplitudes at specific frequencies, which can be used to distinguish them. This same logic can be implemented in hardware using a set of filter banks to select the various frequencies, and performing identification in real time.

Figure 7. A few of the Brodatz textures:
(a) *bark;* ***(b)*** *wool cloth;*
(c) *wood grain;* ***(d)*** *bricks;*
(e) *straw;* ***(f)*** *herringbone cloth;*
(g) *sand;* ***(h)*** *bubbles;*
(i) *pigskin.*

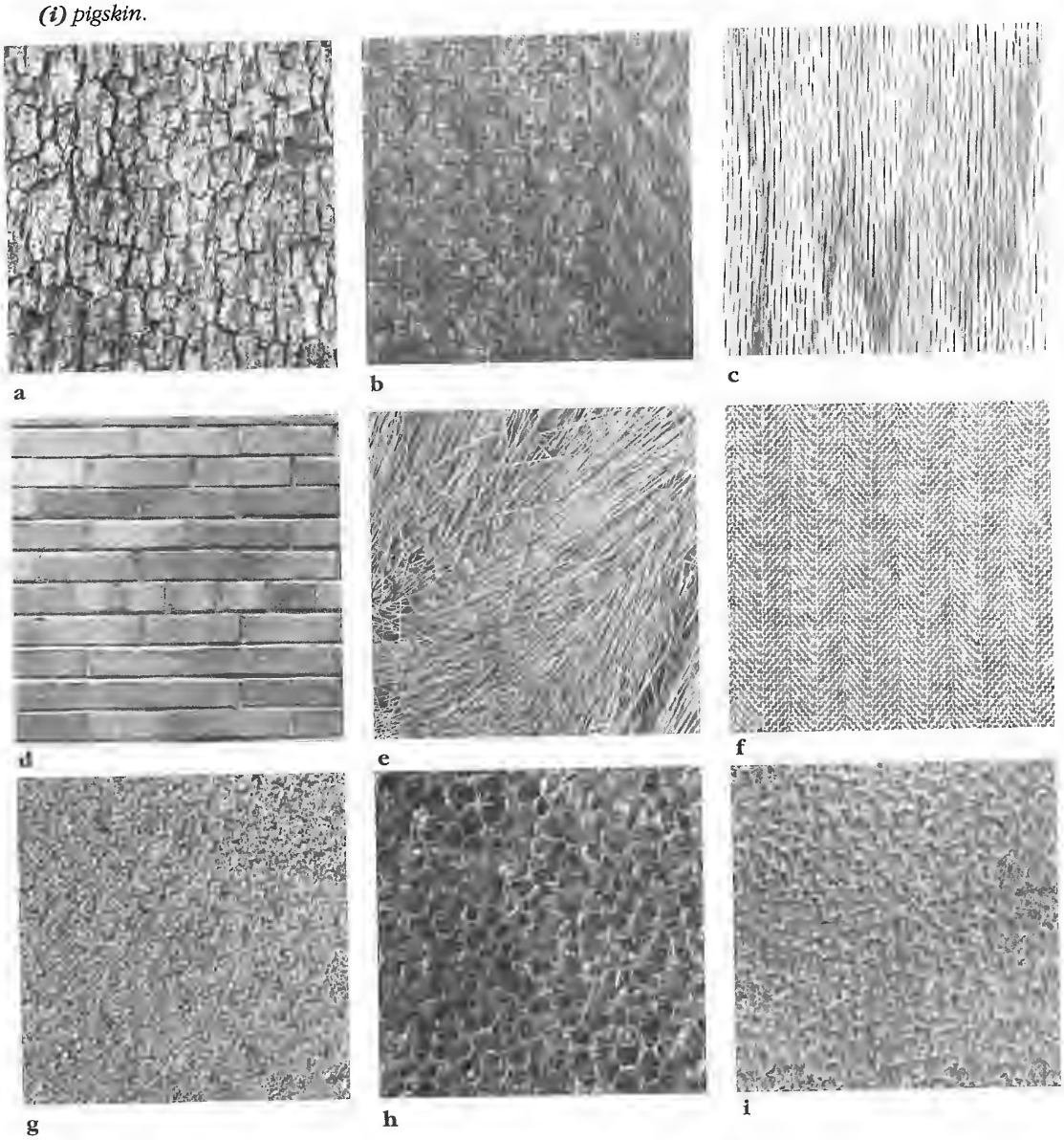
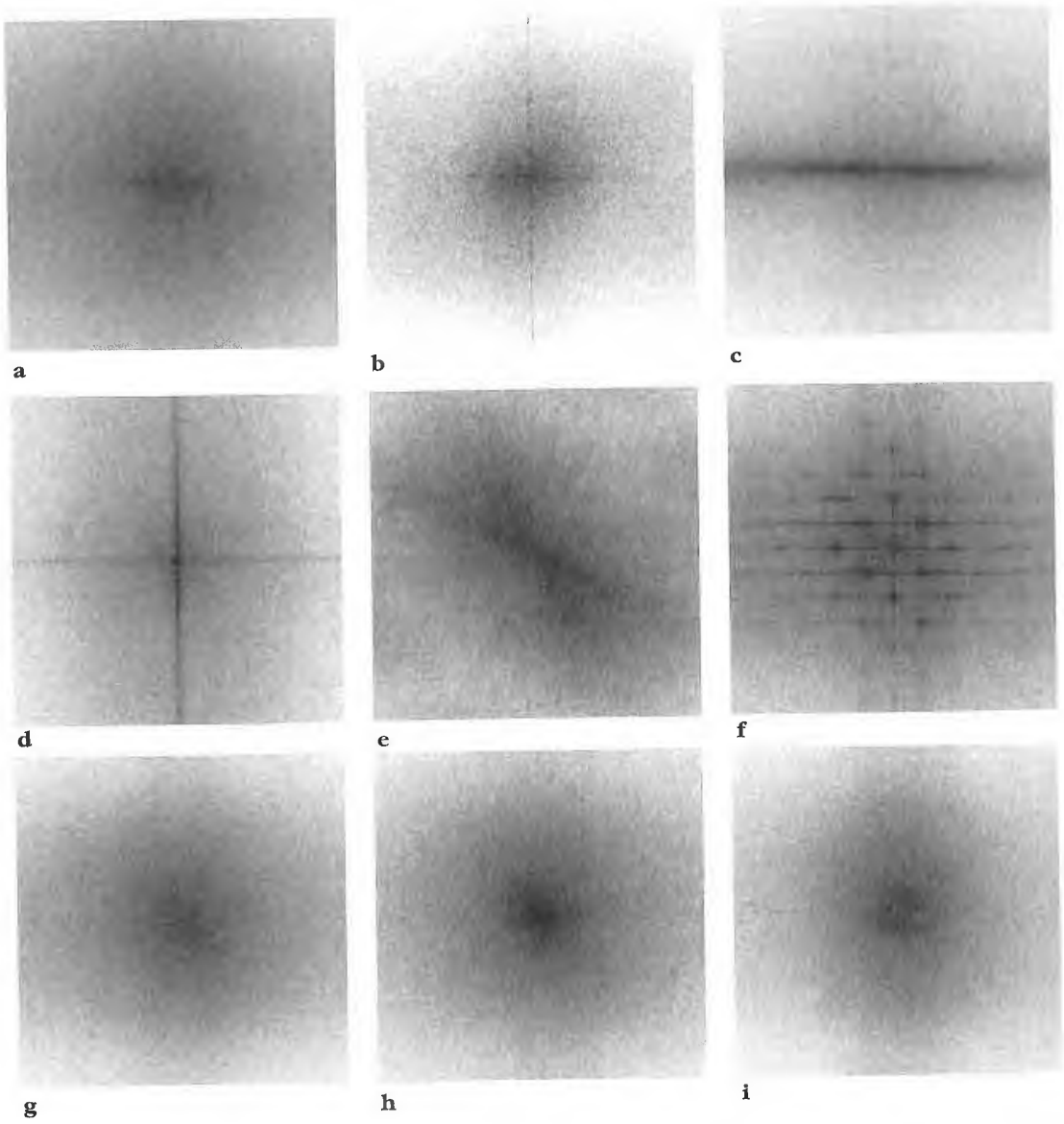


Figure 8. *The Fourier transform power spectra for each of the images in* ***Figure 7.***



The principal use of this technology seems to be in classification of remote sensing (satellite and aerial photographs). When combined with multiband spectral intensity ratios as discussed below, the textures can be used to identify various types of crops, rocks, and other land use patterns. Although they apply to the entire scene rather than a single feature, the various numeric values, such as ratios of amplitudes at various frequencies and orientations, are used in the same way as feature-specific measurements in the classification processes described in this chapter.

Decision points

In many practical cases, the classes are not as completely distinct as the example shown in **Figure 5**. Commonly, when training populations are measured, the histograms of parameter values may

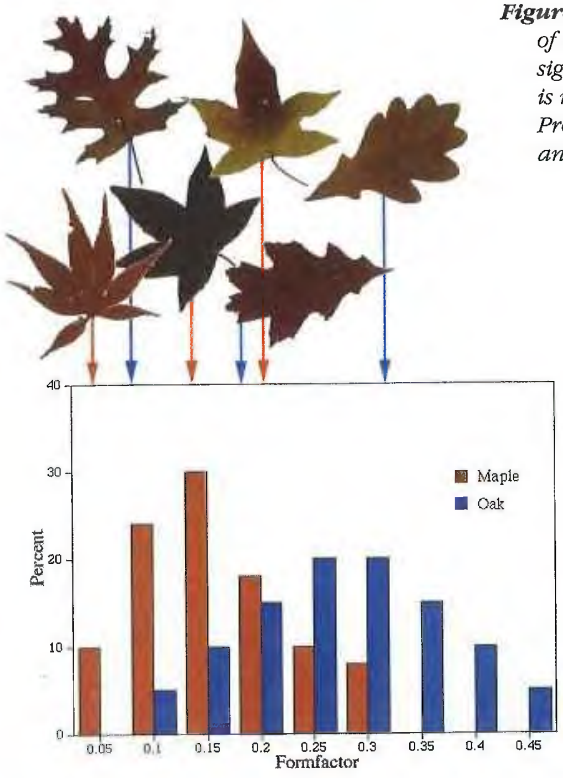


Figure 9. *Values of formfactor measured on a variety of maple and oak leaves. The histograms are significantly overlapped, indicating that formfactor is not the best parameter to use for identification. Probably a method using the skeleton end points and external branches would be more successful.*

overlap as shown in **Figure 9**. Such overlaps usually indicate the need to find other parameters that offer better discrimination, but it is not always possible to avoid some degree of overlap, and in that case it is necessary to establish a decision threshold that produces an acceptable probability of error.

As shown in **Figure 10**, the decision threshold combines with the probability distribution function for the measurement parameter to establish the expected error rate. The area of the tail of the distribution function (as a percentage of the total) gives the probability of identifying a member of one population as being in the other class. Sometimes the decision threshold is set to make the two errors (wrongly identifying A as B, or B as A) equal, which minimizes the total error.

In other cases, economic factors must be taken into account. For example, consider the case of two different "O" rings used in engines, one in a tractor engine and the other in a helicopter engine, which are distinguished by an imaging operation that measures thickness. The economic cost of occasionally shipping a part intended for a helicopter to be used in a tractor is probably very small. The part will likely function there, in which case the only extra cost is perhaps the use of a more costly polymer or the loss of the more valuable part. But shipping a tractor part for use in a helicopter would produce much greater liability if the engine failed, and probably a greater likelihood of failure in a more demanding environment. In such a case it would be desirable to set the error rate for mistakenly identifying a tractor part as a helicopter one to something very low (one in 1 million or less) and accepting a much higher rate (perhaps one in 1000) for the opposite error.

Similar considerations arise in biomedical imaging. There is much concern about automated diagnosis, such as the identification using image analysis of cancer cells in Pap smears, or lesions in mammograms (Karssemeijer, 1998), because of liability issues. The most widely accepted course of action has been to set the decision thresholds for identifying suspicious features very

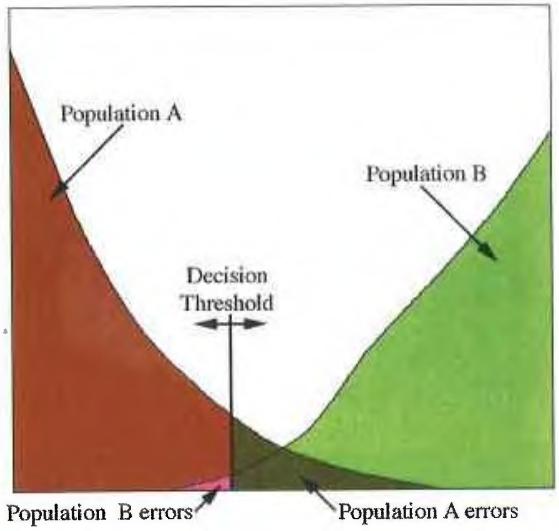


Figure 10. *Setting a decision point when distributions overlap, as discussed in the text. The probability of misclassifying a feature is measured by the area in the tail of each distribution past the decision threshold.*

low, accepting many false positives which can subsequently be screened again by an experienced human, but hopefully minimizing false negatives in which disease might be missed.

Multidimensional classification

In the examples shown earlier, the histogram or probability distribution function has been plotted as a function of a single measurement parameter. In the example of **Figure 6**, two parameters (size and color) were used. In such cases, it is often possible to reduce the problem to a single derived parameter by fitting a new axis line through the data, usually called either a linear discriminant line or a context line. As shown in **Figure 11**, a distribution of data points measured on a training population and plotted on two parameter axes may have values that overlap in both individual parameters, but be much better separated when projected onto another derived axis which is a linear combination of the two. This line can be fit by linear regression through the points, or by using principal components analysis.

Curved lines can be used in principle, but in practice it is difficult to perform robust nonlinear regression unless the form of the functional dependence is known ***a priori***, and in that case it is usually better to transform the measurement variables beforehand. For instance, instead of area, it might be better to use equivalent diameter (which varies as the square root of area) to obtain a plot for which a linear regression line provides a better fit, as illustrated in **Figure 12**.

Once the new axis has been fit to the data points, a histogram of values projected along the new derived parameter (shown in **Figure 11**) can be used for classification just as discussed previously for a directly measured parameter. In fact, the formfactor used previously is actually itself a combination of area and perimeter measurements, so this process is simply one more step toward obtaining a derived parameter that most directly distinguishes the classes of interest.

Using a decision point determined along the best-fit line between two populations generates a decision line (or in higher dimension spaces when more than two parameters are used, a decision plane) that separates the two populations. This works well when the populations are reasonably compact and equiaxed, but, as shown in **Figure 13**, does not separate populations that have irregular shapes. Techniques that can locate an optimum decision plane (the Bayes classifier) for such cases are much more computationally intensive. It is also possible to distinguish multiple populations by piecewise-linear decision boundaries, or by quadratic or higher power boundaries.

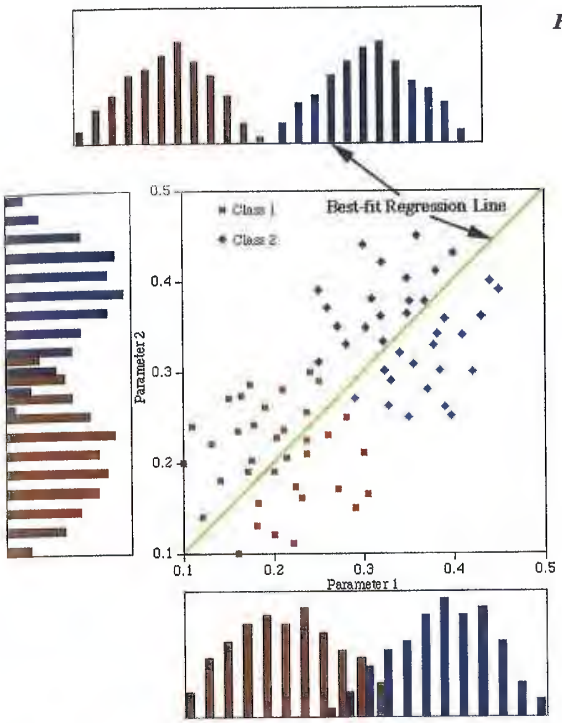


Figure 11. *Two populations of points that overlap on both parameter axes, with a linear discriminant or context line along which they are best separated.*

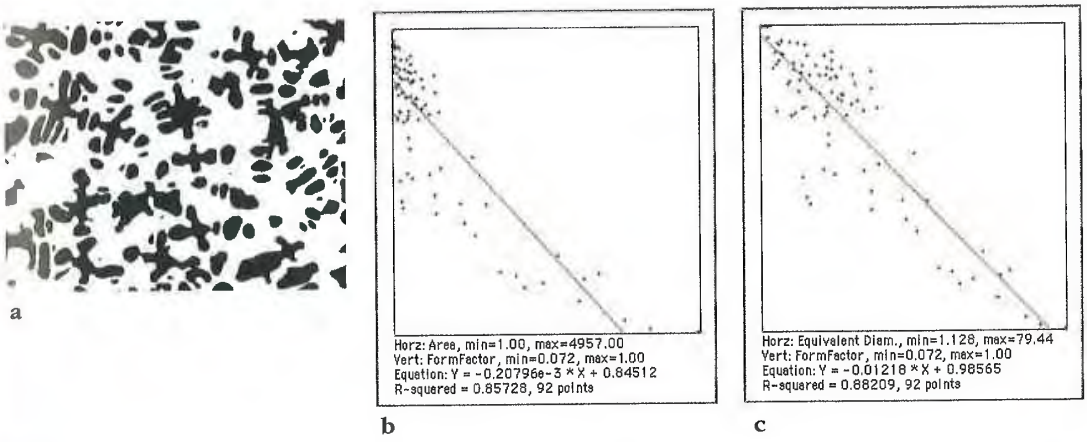


Figure 12. *Features from an image of dendrites in a metal alloy, showing regression plots for formfactor vs. area* ***(b)*** *and formfactor vs. equivalent diameter* ***(c)****.*

Fortunately, in most such cases the addition of additional parameters separates the various population classes well enough that simple linear methods can be used.

Population classes like those shown in **Figure 13b**, which are elongated and oriented at an angle to the axes defined by the measurement parameters, have a some correlation between the parameters.

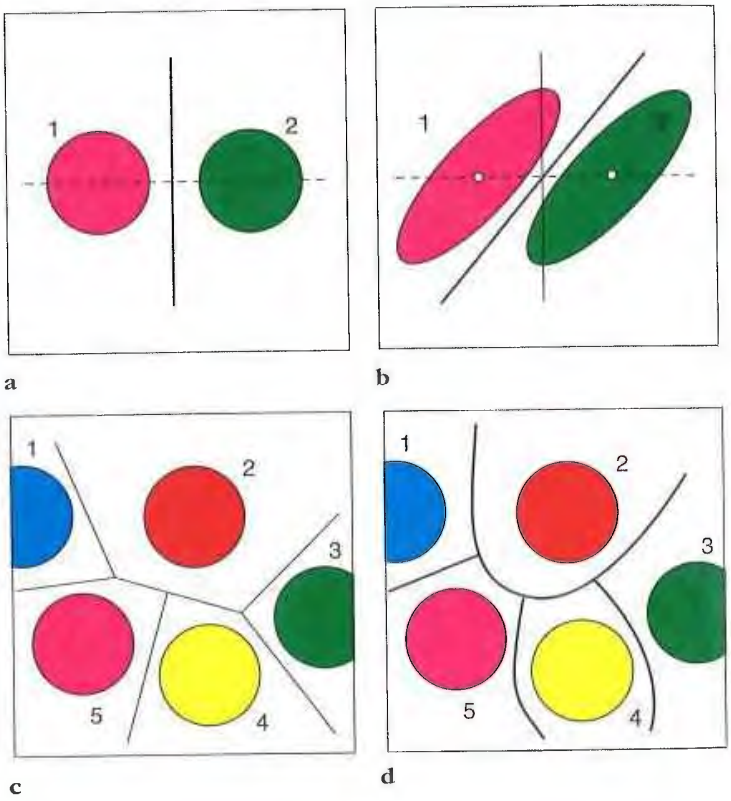


Figure 13. Classification examples in two dimensions: ***(a)*** *separable classes showing the decision boundary perpendicular to the linear discriminant or regression line;* ***(b)*** *classes that are not separable by the boundary in figure* ***a****, with an optimal Bayes classifier (blue line) that does separate them;* ***(c)*** *multiple classes separated by piecewise linear decision boundaries;* ***(d)*** *multiple classes separated by quadratic decision boundaries.*

This creates problems for classification, as noted above, so it is desirable to make the population area more equiaxed, and preferably circular. The covariance of the data set is defined as

$$c(i,j) = \frac{\sum_{k=1}^{n}\left(x_{k,i} - \mu_i\right)\cdot\left(x_{k,j} - \mu_j\right)}{n-1} \tag{2}$$

where x is the measurement value, i and j identify the two parameters, μ are the mean values for each parameter, and k runs through the n features in the population. The covariance can vary between $+\sigma_i\sigma_j$ and $-\sigma_i\sigma_j$ where σ are the standard deviation values for each parameter. A value of zero for $c(i,j)$ indicates no correlation and the minimum or maximum value indicates perfect correlation as shown in **Figure 14**.

The previous examples showed only two measured parameters, but in general there may be N axes for a high-dimensionality space. In that case, all of the covariances $c(i,j)$ for the various parameters measured can be collected into a covariance matrix **C**. This matrix can be used to transform an irregular population cluster to an equiaxed, circular one. For each measurement vector x (defined by all of the n measurement parameters), the quantity r calculated as

$$r^2 = (x-\mu)'C^{-1}(x-\mu) \tag{3}$$

is called the Mahalanobis distance from the point representing the feature measurement parameters to the mean of the population μ. This is a generalization of the usual concept of distance,

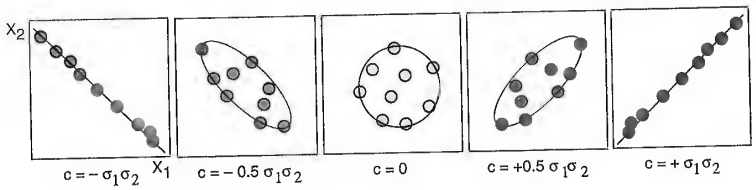


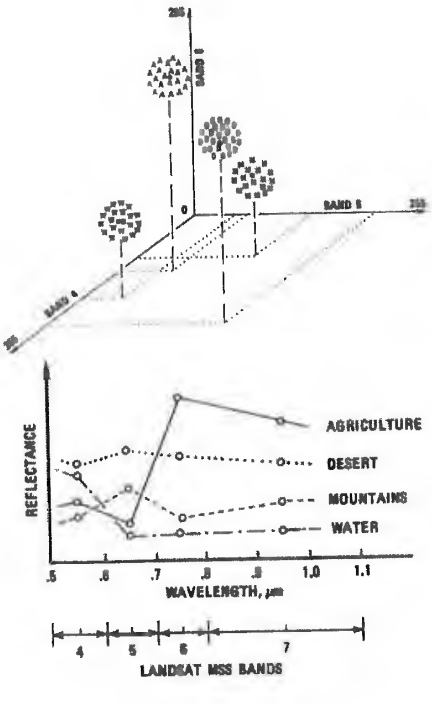
Figure 14. *The value of covariance can vary between perfect positive and perfect negative correlation, as discussed in the text. A zero value represents uncorrelated values.*

appropriate for measurements in which the axes have different meanings and scales. In the case of an equiaxed population cluster (zero covariance), the Mahalanobis distance is the same as the usual Euclidean distance.

It is helpful in measuring distances along the various axes to compensate for the different scales of the various measurement parameters by normalizing the distance measurements. This is done by dividing each one by an appropriate scaling constant, ideally the standard deviation of the parameter values. This is called the "standardized Euclidean distance."

Of course, the classification methods can be generalized to more than two dimensions (it just becomes harder to illustrate with simple graphics). **Figure 15** shows an example from remote imaging, in which the intensities of reflected light detected in different Landsat Thematic Mapper wavelength bands are used to classify terrain types (Grasselli, 1969; Sabins, 1986). Patterns of spectral

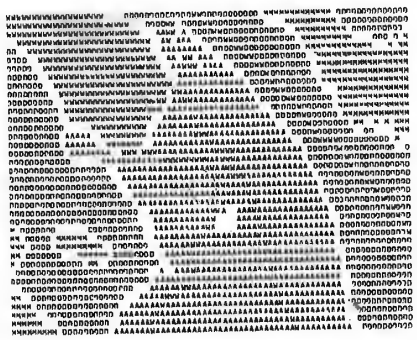
Figure 15. Land use classification using Thematic Mapper images:
(a) *one image showing reflectivity in a single wavelength range;*
(b) *reflectance vs. wavelength plots and cluster diagram for reflectivity values from different terrain types;*
(c) *classification of land use using these data.*



b



a



c

intensity are associated with particular types of vegetation, minerals, etc. A supervised training program is used in which an operator marks locations and the image analysis system plots the point in terms of the measured intensities in each wavelength band. The clusters of points (which are shown schematically) then mark the various classes, which are used to label the images pixel by pixel.

There is no reason to expect the clusters of points corresponding to each class to form a regular spherical shape in the N-dimensional parameter space, of course, and examples below will show alternative ways of establishing these regions. One of the simplest methods is to establish maximum and minimum limit values for each parameter. This corresponds to a set of rectangular boxes in parameter space (**Figure 16**) that define each class. Training such a system is simplified because it is not necessary to find a representative population, but rather to find or predict extreme values.

Another method for establishing region boundaries is to measure a representative training population for each class, and to characterize the distribution of values by the mean and standard deviation for each parameter. Using the mean values as the coordinates of the center, and a multiple of each parameter's standard deviation as the lengths of the axes, this generates ellipsoids as shown in **Figure 17**. The surfaces of these ellipsoids may be used as absolute class boundaries, but it is more useful to measure the distance of each new feature's parameter coordinates from the various classes in terms of the standard deviation.

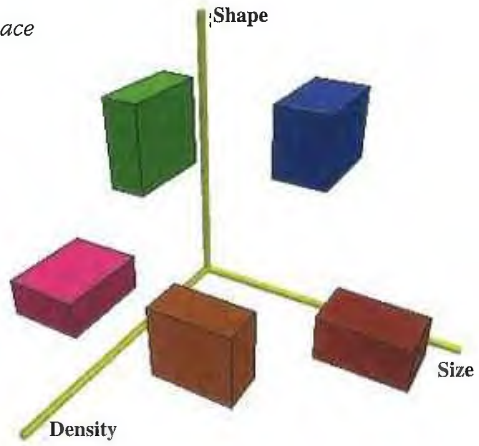


Figure 16. *Range limits for class regions in parameter space (illustrating three independent parameters)*

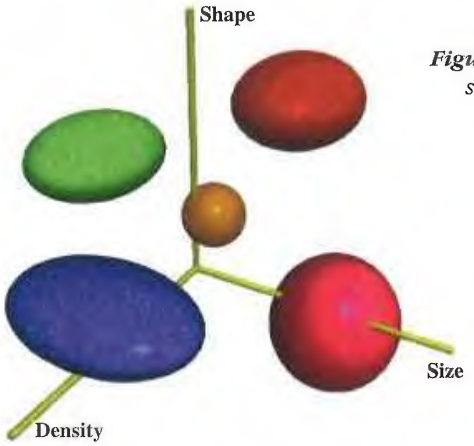


Figure 17. *Ellipsoidal class regions calculated from the standard deviation of measured data.*

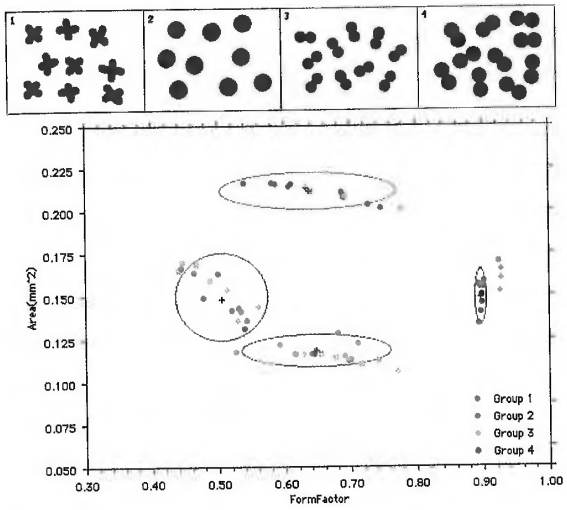


Figure 18. *Two-dimensional example of statistical limits. Note that some points lie outside the 2-sigma boundaries. Solid points indicate the training populations shown at the top; shaded points represent identification of additional features. Points are identified with the nearest cluster measured as discussed in the text.*

Figure 18 shows a two-dimensional example for simplicity. The training populations are shown at the top, along with the plots of the points and the two-sigma ellipses for each class. Note that for the training population and for additional features measured subsequently, some of the points lie outside the two-sigma limit, as would be expected. Note also that some of the classes are much larger (more variation in the parameter values) than others, and that the ellipses may be elongated in some cases because one parameter varies more than another. The standard deviation, shown as the length of each ellipse axis, becomes a scale with which to interpret the rather confusing distances of parameter space. Using the standard deviation normalizes the various axis units as discussed above. The concept of a "distance" in parameter space is not well defined and may be distorted in different directions or regions of the plot. Unfortunately, this is often the case when shape factors are used for identification since they are often not all independent (so the axes are not orthogonal) and are certainly not linear (a change in aspect ratio from 1.1 to 1.5 is far different from a change from 4.1 to 4.5, for instance).

If a new feature is measured, and its parameter coordinates do not place it inside one of the class boundaries, it may still be identified as being "most like" one class by measuring the distance in parameter space from the feature coordinates to the center of each class (the mean values), and scaling this number by dividing by the standard deviation. Because of the different cluster sizes, this means that a point could actually be closer to one class on the plot but more "like" a different one. In the example, a point close to the blue class might be fewer standard deviations away from the magenta one, and would be identified accordingly.

Learning systems

It can be very difficult to collect a truly representative set of features with which to establish training populations that produce classes that can be applied without alteration to many new features and situations. In particular, the extreme-valued features that lie near the end of the distributions are usually rare. Furthermore, an approach using mean and standard deviation rather than the actual shape of the distribution may oversimplify the measurements, which in many image analysis situations do not produce Gaussian distributions.

Figure 11, in the previous section, illustrated the ability to estimate the probability of classification error from the shape of the distribution histogram. Using the actual histogram thus extends traditional "hard" classification in which a feature is assigned to one class or excluded from it, to the

"fuzzy" logic situation in which shades of probability are admitted. Fuzzy logic is an important extension of classical logic in which the various classification rules in the knowledge base contain probabilities. An example would be "If ***A*** can fly, there is a 70% chance that it is a bird; if ***A*** cannot fly, there is a 5% chance that it is a bird," which accommodates butterflies, which fly but are not birds, and penguins, which are birds but do not fly. In our case, the parameter histogram gives a direct way to measure the conditional probabilities.

Fuzzy logic has been applied to expert systems and neural nets (both discussed below as engines for assigning classifications based on multiple rules). Typically the result is to slow the convergence process because it is necessary to evaluate all paths and find the most likely, not simply to find one path through the rules. In a formal sense, fuzzy logic is equivalent to Bayesian logic with leveled classic sets. For comprehensive information on fuzzy logic, refer to (Zadeh, 1965; Negoita and Ralescu, 1975, 1987; Sanchez and Zadeh, 1987; Zimmerman, 1987).

As an illustration of a learning system that uses histograms to establish decision points, and works in multi-parameter space using linear discriminant or context line methods, populations of nine kinds of natural nuts were used (**Figure 19**). Some of these represented more than one species (e.g., one class included both white and red oak acorns, both with and without their caps, and another included several kinds of pecans) so that histograms were generally not Gaussian and sometimes were multimodal (**Figure 20**). Images from a training population of about 20 of each variety of nut were captured with a monochrome video camera (this work was done in the 1980s, so much of the hardware was primitive by current standards), and the features measured to obtain a total of 40 parameters for each feature.

Some parameters such as location and orientation were discarded based on human expectation that they were not meaningful. Others were discarded by subsequent statistical analysis, leaving a net of 17 that were actually used, so discrimination was performed in a 17-dimensional space. Not all of the dimensions are truly independent (orthogonal), however, because many of the shape parameters are based on size measures used in various combinations. Linear discriminant lines were



Figure 19. *A few of the nuts used in the learning example discussed in the text.*

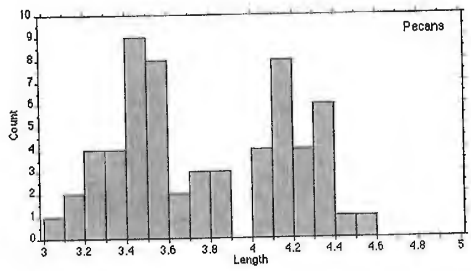


Figure 20. *A bimodal length distribution for two species of pecans.*

constructed in this space between each pair of nut classes, using stepwise regression (Draper and Smith, 1981). Parameters were added to or removed from the equation of the context line based on the F-value (an arbitrary cutoff value of 4.0 was used), eliminating variables that did not improve the ability to distinguish particular population groups.

This elimination helps toward the goal of using independent parameters, as well as simplifying the total problem and reducing the number of training examples required. Many of the shape parameters, for example, use some of the same size information. For example, the convexity and form-factor defined in Chapter 8 both use the perimeter. If one of them shows a high correlation with category discrimination, there is a good chance that others that incorporate perimeter will also show a correlation. Stepwise regression will select the one that has the highest correlation and predictive power, and discard the others.

The resulting context lines used as few as two or as many as eight terms. Some parameters were used in many cases (e.g., average brightness), while others appeared in only one or two equations but were then often the most important single parameter (e.g., convex area was highly important in distinguishing pistachios from almonds). On the average, each of the 17 parameters was used 11 times in the total of 55 pairwise context lines, and each context line equation included an average of 5 parameters. **Figure 21** shows several examples. The coefficients in the equations for the linear discriminant or context lines are derived by regression, and represent the angles between the line and each of the parameter axes in N-space.

Discrimination of hickory nuts from almonds, and acorns from peanuts, is easy because the distributions are entirely separate, even though the distributions are somewhat irregular in shape. The acorn–pecan distribution (**Figure 22**) shows a slight overlap, which was the case in about 16% of the pairwise discriminant plots. Using the logic shown earlier in **Figure 11**, the decision point was located to make the probability of misclassification errors equal for both classes. In the acorn–pecan case, this amounted to about 1% and arose when a large acorn was viewed from the side without its end cap, as shown in **Figure 23**. The most likely error was confusion of an acorn with a filbert.

Once the discriminant functions (the equations of the context lines) have been established and distributions of derived parameter values along those lines set up, the system can go to work

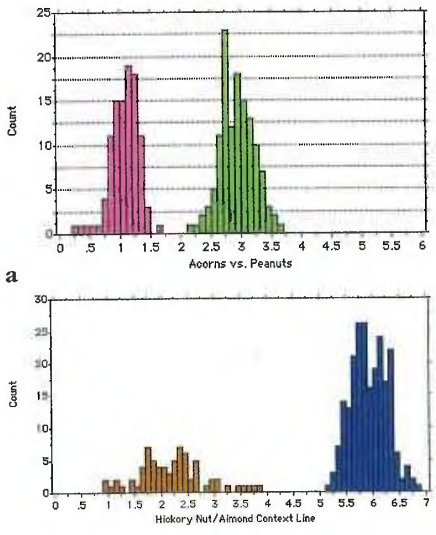


Figure 21. *Context lines and frequency distributions for hickory nuts–almonds and acorns–peanuts, examples of entirely separated classes. The regression equations are:*

Acorns (type 1) vs. Peanuts (type 3) = 3.928 + 0.773 · Aspect Ratio - 5.007 · Fractal Dimension + 0.068 · Brightness

Hickory Nuts (type 2) vs. Almonds (type 6) = –2.266 – 2.372 · Breadth + 2.184 · Width + 0.226 · Brightness +0.318 · Contrast +2.374 · Texture

Figure 22. *Context line and frequency distribution for acorns-pecans, showing overlap. The regression equation is:*

Acorns (type 1) vs. Pecans (type 4) = 1.457 + 6.141 · Formfactor - 2.153 · Convex Perimeter + 1.768 · Convex Area + 2.926 · Length - 1.575 · Breadth - 1.131 · Extent

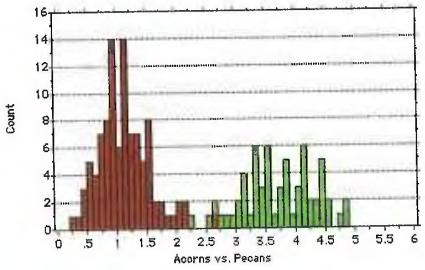




Figure 23. *View of several large acorns without their end caps and pecans, showing a situation in which confusion between the two classes may arise.*

identifying nuts. The decision points along each context line position planes (perpendicular to the lines) in the high-dimensionality space which form polyhedra around each class. The problem is that with a training population of only 20 nuts from each class, the positions of those decision points are imperfect.

As the system examines more images (eventually a total of about 1000 nuts), the derived parameter values (positions along each context line) are calculated and the histograms are updated. This does not require any additional computer memory, since the histograms have already been established. Identification of nut types proceeds on the basis of the stored decision points. But whenever a value is added to the end of a histogram (within the last 5% of its area), the program asked the operator to confirm the identification and then reestablished the decision threshold using the then current histogram. In this way the system learned from the additional specimens to which it has been exposed, which gradually increased the number of observations near the important limits of each distribution.

Figure 24 shows a table with the final results. A total of 3.7% of the nuts were misidentified, but most of these errors occurred very early in the learning process; there was only one error in the last 200 nuts. The system started out with an approximate idea of the limits of each class, and refined these limits with more information. It should be noted that many of the parameters used for each identification were not familiar to the human operators, and almost certainly did not represent the logic they used to identify the nuts. But the system quickly became as proficient as the humans.

In other examples, an automated system based on this same logic has surpassed the humans who trained it (Russ and Rovner, 1987). In the study of archaeological sites in the American southwest and much of central America, a subject of considerable interest is the process of domestication of corn, *Zea mays*. It is widely believed, although not proven, that corn is a domesticated offshoot of the wild grass teosinte, of which there are many varieties. Little remains of corn in an archaeological context that can be dated, but it happens that corn, like all grasses (and many other plants), produces small silica bodies called opal phytoliths in and between the cells on the stalk, leaves, and

		Identified Type								
Actual Type		1	2	3	4	5	6	7	8	9
Acorns	1	91	0	0	1	0	0	0	10	0
Hickory Nuts	2	0	70	0	1	0	0	1	0	0
Peanuts	3	0	0	126	0	0	1	0	0	0
Pecans	4	2	1	0	54	0	0	3	0	0
Pistachios	5	0	0	0	0	126	0	0	0	0
Almonds	6	0	0	0	0	0	224	0	0	0
Brazil Nuts	7	0	3	0	6	0	0	115	0	0
Filberts	8	8	0	0	0	0	0	0	67	0
Walnuts	9	0	0	0	0	0	0	0	0	107

Figure 24. *"Confusion matrix" showing errors in nut identification.*

other parts of the plant. They are generally a few micrometers in size, and act to stiffen the plant tissue. They are also selectively produced at any site of injury (including cropping by animals).

From an anthropological point of view, phytoliths are of interest for two reasons. First, they survive in the soil for long periods of time and can be recovered from layers that are datable and show other signs of human habitation. Second, they have shapes that are distinctive. Research over several decades has shown that phytoliths can be used taxonomically to identify species of grasses, including corn and its precursors (Twiss et al., 1969; Rovner, 1971; Pearsall, 1978; Piperno, 1984). Most of this identification has been carried out by humans, who have accumulated photographs (mostly using the scanning electron microscope since the particles are only a few micrometers in size and have significant three-dimensional structure so that they cannot be studied satisfactorily with the light microscope) and built catalogs from which matching comparison to unknowns can be performed. The success rate for skilled observers in blind tests is generally better than 95%. This has been at the cost of a very substantial amount of effort, however, and the knowledge and experience are not readily transferred to other researchers. The human eye does not deal too well with the need to characterize variation, and the individual phytoliths vary widely in shape. Such a situation is ripe for the use of computer image analysis methods.

Slides were prepared with phytoliths extracted from a training suite of five known species of maize and teosinte and SEM images analyzed to determine several size and shape parameters, which were used as described above to establish discriminant classes. On the original set of objects, this produced a better than 99% correct identification. After applying the results to a total of 300 objects from 5 additional species of plants (three maize and two teosinte), the results had improved to better than 99.6% accuracy in distinguishing the two classes. Furthermore, the system can examine hundreds of phytoliths (enough to produce statistically significant results) from more species of corn or other grasses in one month than have been done by hand and eye in the last 20 years. The same system has recently been used for classification of squash seeds (Rovner, personal communication).

kNN and cluster analysis

Other methods are used to establish limits for classes in parameter space. The two most widely used, *k*-nearest neighbor (*k*NN) and cluster analysis, share the drawback that unlike the histogram and linear discriminant method described earlier, it is necessary to save the actual coordinates for each identified feature, which means that storage requirements grow continually if the system is to continue learning. The other difficulty with these methods is that the time required to make an identification rises with the number of stored values, even with efficient algorithms to sort or prune the data set and work with only a reasonable number of candidate points. On the other hand, these methods do not presuppose any particular shape for the class regions, such as the rectangular prism, ellipsoid, or polyhedral methods described previously. It is even possible to handle such situations as shown in **Figure 25**, in which one class is nonconvex, disjoint, or is largely or completely surrounded by another.



Figure 25. *Examples of nonconvex, disjoint, or surrounded classes (indicated by different colors) that are difficult to represent by simple geometric shapes.*

The *k*-nearest neighbor classification works by searching the database of previously identified features for those which are "most like" the current measurement, as shown in **Figure 26**. This amounts to finding the distance between the coordinates of the current feature's parameters, and those for other features. The distance is generally calculated in a simple Pythagorean sense (square root of the sum of squares of differences), but this overlooks the important fact that the different axes in parameter space have very different units and metrics. Usually the distance along each axis is simply expressed as a fraction of the total range of values for that parameter, but there is no real justification for such an assumption. It would be better to use some normalizing scale such as the standard deviation of values within each class, if this is available (the same logic with Mahalanobis distance as used above for the ellipsoid class limits).

In the simplest form of *k*-neighbor comparison, *k* is one, and the search is simply for the one single prior measurement that is most similar, and which is then assumed to identify the class of the new feature. For features near the boundary between classes, single-nearest-neighbor matching proves to be quite noisy and produces irregular boundaries that do not promote robust identification, as shown in **Figure 27**. Using the identification of the majority of five, nine, or even more nearest-neighbors produces a smoother boundary between classes, but is still highly sensitive to the relative size of the populations. If one population has many more members than the other, it effectively shrinks the class limits for the minor group by making it more likely that the majority of matches will be with members of the more numerous population.

The methods described so far presume that some training population in which the features belong to known classes can be used to construct the class limits. As noted before, this is known as "supervised

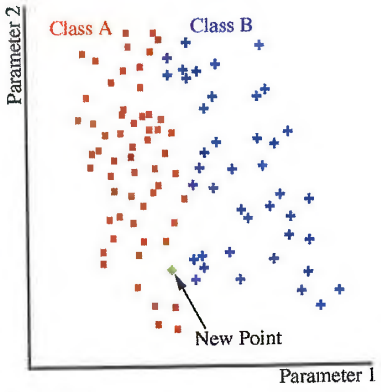


Figure 26. *Three of the five nearest-neighbors to the new point lie in Class B, so it is assigned to that class.*

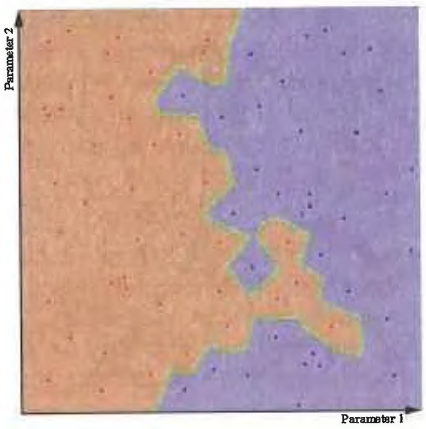


Figure 27. *The irregular boundary that forms between two classes (whose individual members have parameter values indicated by the colored points) using single-nearest-neighbor classification.*

learning" in which the operator identifies the actual class or population to which each feature belongs, and the software uses that information along with whatever measurement parameters are available to determine an identification strategy. There are some situations in which it is not known *a priori* what the classes are, or perhaps even how many of them are present. In these cases, it is still useful to plot each feature's measured parameters as values in a parameter space, and then look for clusters of points. Cluster analysis with non-supervised training is a rich topic that goes beyond the scope of this text (see, for example, Pentland, 1986; James, 1988; Bow, 1992; Fukunaga, 1990).

One typical way that cluster analysis proceeds uses the distance between points to identify the members of a cluster. For example, start with the two points closest together. Assume these represent a class, and add to the class other nearby points until the distance from the nearest point already assigned to the cluster exceeds some (arbitrary) limit, which may be calculated from the size of the growing class region. The method can obviously be generalized to use more than a single nearest-neighbor. Once a given class has stopped growing, begin again with the remaining points, and continue until all have been assigned. There are many refinements of this approach, which allow for assigned class regions to be merged or split.

A somewhat complementary method for finding clusters uses all of the points simultaneously. It starts by constructing a minimal spanning tree for all of the points present (an intensive amount of computation for large data sets). The links in this tree can be weighted either by their Euclidean or Mahalanobis length, or by the number of matches present in a table of k-nearest-neighbors for each point. In other words, if the same points are each listed in each other's nearest neighbor lists, they are probably in the same cluster. The spanning tree is then pruned by cutting the links with the greatest length or the smallest number of matches, to leave the identified clusters. **Figure 28** shows an example of a sparse cluster of points, the minimal spanning tree, and the clusters that result from pruning (after Bow, 1992).

Another, somewhat different approach to find clusters in data, particularly when there is only one significant measured variable, is the dendrogram. **Figure 29** shows an example. The features are ranked into order based on the value of the measured variable, such as formfactor. Then the features that have the smallest value differences are connected (numbers 7 and 21 in the example, followed by 4 and 23, 27 and 14, 13 and 16, etc.) by horizontal lines positioned according to the difference value. Groups of features are usually linked based on the smallest difference between any two members of the group, although it is also possible to use the difference between the mean or median values in each group. Eventually all of the features and groups are connected into a single tree.

If multiple measurement parameters (e.g., formfactor, diameter, and density) are used, the difference between features is calculated as the distance between the vector representation of the

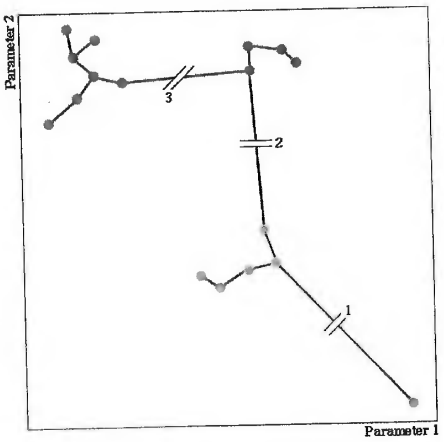


Figure 28. *Parameter values plotted as points, and the minimal spanning tree connecting them. Cutting the branches with the greatest length separates the clusters (indicated by different colors).*

multiple values in N-dimensional space. This typically presents a problem because the units of the various parameters are quite different, and the lengths of the various axes are not compatible so there is no natural way to scale them in proportion. This can sometimes be handled by normalizing each measured parameter based on (for example) the total range or standard deviation of the measured parameters, but such a method is inherently arbitrary. Without a full characterization of the statistical distributions of each parameter, it is not possible to calculate a true Mahalanobis distance for the parameter space.

Once the dendrogram has been constructed, it can be used to determine any number of clusters by selecting a different threshold value for the difference. For example, in **Figure 29** a value of 0.6 would yield two clusters while a value of 0.4 would yield four clusters and a value of 0.3 would yield 5 clusters. By definition, in the unsupervised clustering situation, we do not know *a priori* how many classes are present. Therefore, the decision requires human judgment.

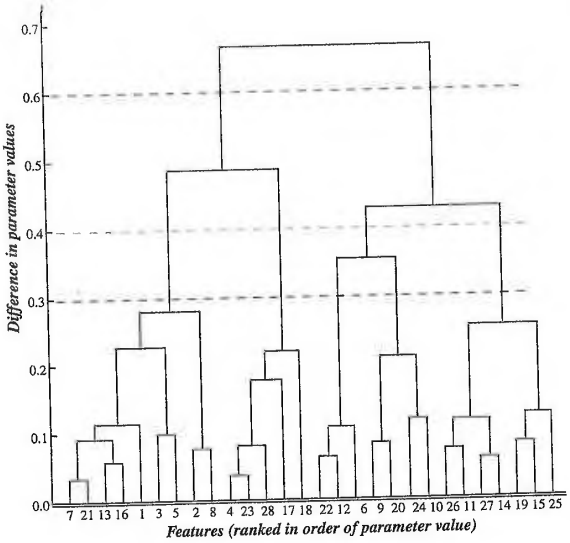


Figure 29. *Example dendrogram as discussed in the text. The red, green, and blue lines indicate the effect of selecting different threshold values for the difference between feature measurements on the number of classes.*

Expert systems

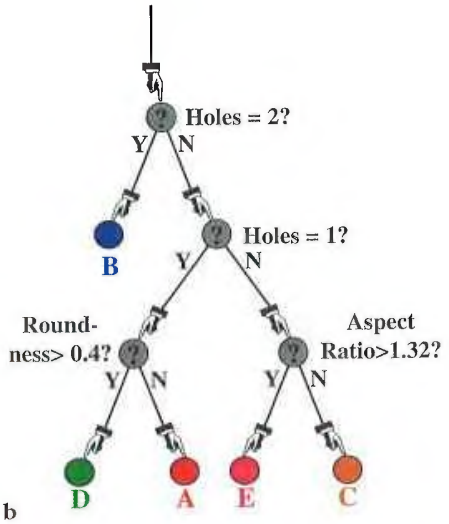
After clusters have been identified by one of these methods, the class limits must still be constructed by one of the methods shown earlier. This could be a geometric shape (rectangular prism, ellipsoid, polyhedron) or a *k*NN boundary. Now we will consider how to apply these class boundaries as a set of rules for the identification of subsequent features.

The simplest and most common method is the traditional expert system. A set of rules (such as the limits — either hard or fuzzy — for the class boundaries) is created which must be applied to each new set of feature measurements. Returning to the case of the letters A through E, **Figure 30** shows a very simple expert system with four rules, requiring measurement of only three parameters. In the most efficient implementation, only those parameters required would be measured, so for the letter B (which is identified by the number of holes in the shape) the roundness and aspect ratio parameters, which are not needed, would not be determined at all. When they are used, the cutoff values of the parameters for each letter type were determined experimentally, by measuring characters from a few different fonts. In situations such as this, it becomes important to design the system so that the least "expensive" parameters are determined first, to reduce the time needed for the overall identification. Counting holes requires less computation than measuring dimensions needed for the shape factors. Also notice that, unlike the template matching and cross correlation examples shown earlier, this method is insensitive to position, size or orientation of the letters, and tolerates several different fonts.

Simple classification systems such as this are sometimes called decision trees or production rules, consisting of an ordered set of IF/THEN relationships. Classic expert systems separate the data base (rules) from the inference engine used to find a solution (Black, 1986; Winstanley, 1987; Rolston, 1988; Slatter, 1987; Walters, 1988). One of the drawbacks to such systems is that the addition



a



b

Figure 30. *Expert system to classify the letters A through E. The colors shown in*
(a) *were assigned using the logic in*
(b) *Note that the shape factors used as parameters do not depend on size, orientation or position, and that several different fonts (with and without serifs) are present.*

of another class (e.g., the letter F) does not simply graft on another step, but may completely reshuffle the order in which the rules should be applied, or even eliminate some rules altogether and replace them with others. The optimum decision tree is not necessarily the one with the fewest branches. "Knowledge shaping" procedures such as including the cost of obtaining each input value can greatly improve overall efficiency and determine an optimum search and solution tree (Cockett, 1987).

Most real expert systems have far more rules than this one and the order in which they are to be applied is not necessarily obvious. The complexity of decision paths increases rapidly with a large number of rules. General purpose expert systems try to find a path between the various input values and a conclusion in one of two ways. If, as shown in **Figure 31**, the number of observations is less than the number of possible conclusions, forward chaining that starts at the left and explores pathways through various rules toward a conclusion would be used. In other cases, backward chaining that works from conclusions back toward observations may be more efficient. In the very schematic diagram shown in the figure, there are 6 initial observations (measurements), 21 rules, and 12 possible conclusions; real systems are several orders of magnitude more complex. Some of the rules shown have more than two possible outcomes; in some systems, each possible outcome would represent a separate rule.

Normal forward searching for a successful path through this network would start at some point on the left, follow a path to a rule, and follow the outcome until a contradiction was met (a rule could not be satisfied). The system would then backtrack to the preceding node and try a different path, until a conclusion was reached. This approach does not test all possible paths from observations to conclusions. Heuristics to control the order in which possible paths are tested are very important. In cases of realistic complexity, it is not possible to test all consequences of applying one rule, and so simplifications such as depth-first or breadth-first strategies are employed. Pruning, or rejecting a path before working through it to exhaustion, is also used. (You may use as a rough analogy the search for a move in a chess program. Some initial moves are rejected immediately, while others are searched to various depths to evaluate the various responses possible and the outcomes). Reverse searching works in the same way, starting from possible conclusions and searching for paths that lead to observations. Some search "engines" combine both forward and reverse searching methods.

When fuzzy logic is used, the various rules contain probabilities. In that case, it is ideally necessary to construct a total probability by combining the values for nodes along each path, to select the most likely path and hence the most probable result. The heuristics that control search order are less important, but the total computation load can be much greater.

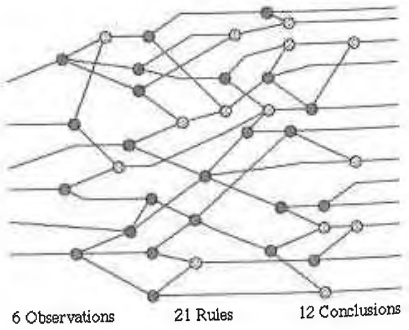


Figure 31. *Schematic diagram for an expert system.*

Neural nets

Searching the multiple pathways through an expert system, especially one utilizing fuzzy logic probabilities, is an inherently parallel problem. An attractive way to solve it uses neural networks, an implementation of a simplified and idealized model of the functioning of biological processes. Each element in a neural net is a threshold logic unit, illustrated in **Figure 32**. It is analogous to a simplified model of a functioning neuron, as shown in the classic McCulloch–Pitts model (McCulloch and Pitts, 1943; Rosenblatt, 1958; Lettvin et al., 1959; Minsky and Papert, 1969; Grossberg, 1988). Multiple inputs (which in the image analysis case could be either measured parameter values, or pixel intensities) are weighted and summed, and the total compared to a threshold value. If the threshold is exceeded, the neuron "fires." In a biological system, it sends a train of pulses as inputs to other neurons; in a neural network it transmits a calculated output value (a probability) to another threshold logic unit.

To understand the functioning of these simple units, it is worthwhile to consider a classical description of a theoretical construct called the "grandmother cell." Imagine a cell that has been trained to recognize your grandmother, and patiently examines every image that is transmitted from your eyes searching for clues. Earlier logic units, some of them in the eye and some in the visual cortex, find low level structures. So the presence of white hair, blue eyes, and the spacing of eyes, nose, mouth, cheekbones, etc., are all inputs to the grandmother cell. It has been shown that human vision is very efficient at locating facial features (even babies do it, immediately after birth), and that it is the ratios of spacings between these features that are important clues to recognition of individuals.

Other clues might include the presence of a familiar dress, jewelry, or glasses. Some of these are more important than others (have larger weighting), and some factors may simply be missing (e.g., if your view of the person prevents seeing the color of her eyes). There may also be some factors with large negative weights, such as the presence of a bushy red mustache or a height over

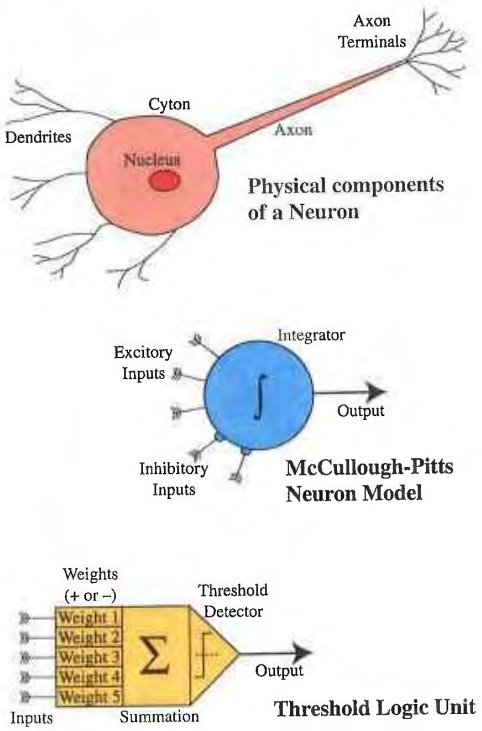


Figure 32. *A threshold logic unit and an idealized neuron.*

six feet tall. When all of the inputs are summed, if the total is large enough, the cell alerts your conscious brain that grandmother is present. Such a system accounts for the occurrence of false positives (thinking that you see someone familiar, who on close examination turns out not to be the expected person). The result will not always be correct (we may miss grandma in some situations, or mistake someone else for her occasionally), but it is very efficient and works quickly and well enough.

Connecting together a network of these simple devices (perceptrons) in multiple layers, with some low level decisions passing information to higher levels, produces a system capable of making decisions based on diverse types of inputs. The classic topology of a neural network is shown in **Figure 33**. Each artificial neuron calculates like the perceptron, producing an output that is a weighted combination (not necessarily linear) of its input values. These outputs are routed through at least one hidden layer before reaching the output layer. The output layer produces outputs that are discrete, selecting one (or at most a few) values. Training the neural net adjusts the weights in the neurons to produce the desired output patterns for the various training patterns (Hebbs, 1949). This is typically accomplished by training the system with examples. When a correct conclusion is reached, the weights of the inputs from the neurons that contributed to the decision are increased, and vice versa. This feedback system eventually results in pathways that correspond to descriptions of the objects being recognized. Various methods for adjusting the weights are in use, some working forward and some backwards through the network (Rumelhart et al., 1986).

In principle, the weights in the various units describe the shape of each class space in a high dimensional space (corresponding to the number of inputs). In practice, it is very difficult if not impossible to interpret the weights, particularly in the inner layers of the system. Dealing with overlapping classes (the possibility that the same set of inputs may lead to more than one output) is exactly analogous to the situation shown earlier in **Figure 11**. Many systems cannot explain the reasons for the answers they deliver; indeed, it is difficult to find just where the "knowledge" resides in the system (like our own brains, it seems to be distributed widely throughout the network). It is difficult to assign probabilities or confidence limits to the result, but neural net systems usually fail gracefully. Unlike classic expert systems, the solution time for a neural system decreases as more information is made available. Their principal shortcoming is the need for unusual computer architectures for efficient realization. Although the neural network is in theory a parallel device, many actual implementations use traditional serial computers to perform all of the calculations, simulating a parallel structure, and in this case the speed advantage is lost.

The weights in a neural network typically take a long time to stabilize, which is another way to say that it takes a lot of training to produce a network that gives correct answers most of the time. The same requirements for finding representative training populations exist as for the statistical classification methods discussed previously. The quality of the training is usually more important than how the solution is implemented. The only real difference between a neural network and one of the statistical methods shown above is the intricacy with which parameter space can be dissected

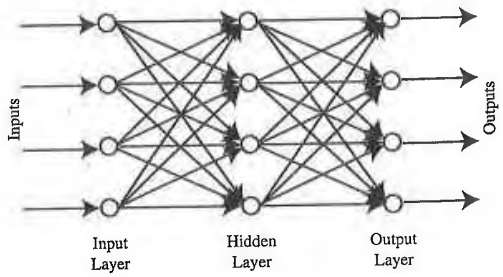


Figure 33. *Schematic diagram of a minimal neural network with one hidden layer.*