CHAPTER

25

Special Imaging Techniques

This chapter presents four specific aspects of image processing. First, ways to characterize the *spatial resolution* are discussed. This describes the minimum size an object must be to be seen in an image. Second, the *signal-to-noise ratio* is examined, explaining how faint an object can be and still be detected. Third, *morphological* techniques are introduced. These are nonlinear operations used to manipulate binary images (where each pixel is either black or white). Fourth, the remarkable technique of *computed tomography* is described. This has revolutionized medical diagnosis by providing detailed images of the interior of the human body.

Spatial Resolution

Suppose we want to compare two imaging systems, with the goal of determining which has the best spatial resolution. In other words, we want to know which system can detect the smallest object. To simplify things, we would like the answer to be a *single number* for each system. This allows a direct comparison upon which to base design decisions. Unfortunately, a single parameter is not always sufficient to characterize all the subtle aspects of imaging. This is complicated by the fact that spatial resolution is limited by two distinct but interrelated effects: *sample spacing* and *sampling aperture size*. This section contains two main topics: (1) how a single parameter can best be used to characterize spatial resolution, and (2) the relationship between sample spacing and sampling aperture size.

Figure 25-1a shows profiles from three circularly symmetric PSFs: the pillbox, the Gaussian, and the exponential. These are representative of the PSFs commonly found in imaging systems. As described in the last chapter, the pillbox can result from an improperly focused lens system. Likewise, the Gaussian is formed when random errors are combined, such as viewing stars through a turbulent atmosphere. An exponential PSF is generated when electrons or x-rays strike a phosphor layer and are converted into

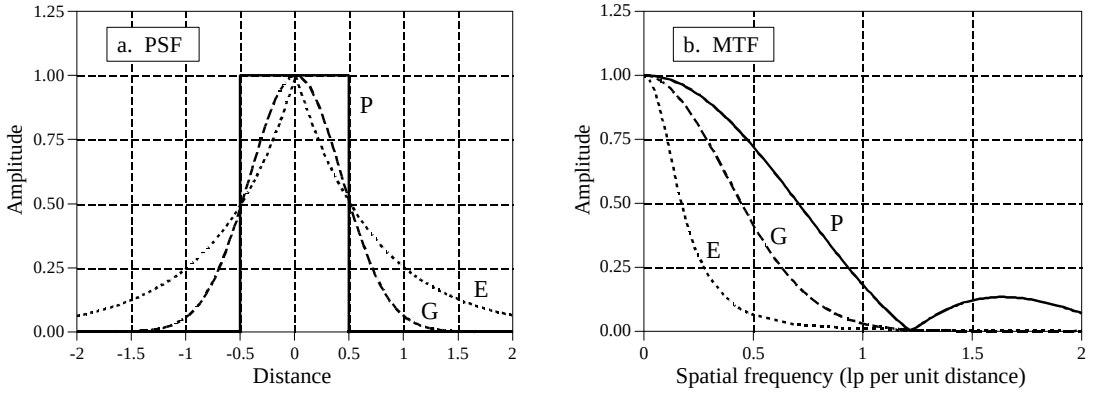


FIGURE 25-1
FWHM versus MTF. Figure (a) shows profiles of three PSFs commonly found in imaging systems: (P) pillbox, (G) Gaussian, and (E) exponential. Each of these has a FWHM of one unit. The corresponding MTFs are shown in (b). Unfortunately, similar values of FWHM do not correspond to similar MTF curves.

light. This is used in radiation detectors, night vision light amplifiers, and CRT displays. The exact shape of these three PSFs is not important for this discussion, only that they broadly represent the PSFs seen in real world applications.

The PSF contains complete information about the spatial resolution. To express the spatial resolution by a single number, we can ignore the *shape* of the PSF and simply measure its *width*. The most common way to specify this is by the Full-Width-at-Half-Maximum (FWHM) value. For example, all the PSFs in (a) have an FWHM of 1 unit.

Unfortunately, this method has two significant drawbacks. First, it does not match other measures of spatial resolution, including the subjective judgement of observers viewing the images. Second, it is usually very difficult to directly measure the PSF. Imagine feeding an impulse into an imaging system; that is, taking an image of a very small white dot on a black background. By definition, the acquired image will be the PSF of the system. The problem is, the measured PSF will only contain a few pixels, and its contrast will be low. Unless you are very careful, random noise will swamp the measurement. For instance, imagine that the impulse image is a 512×512 array of all zeros except for a single pixel having a value of 255. Now compare this to a normal image where all of the 512×512 pixels have an average value of about 128. In loose terms, the signal in the impulse image is about 100,000 times weaker than a normal image. No wonder the signal-to-noise ratio will be bad; there's hardly any signal!

A basic theme throughout this book is that signals should be understood in the domain where the information is encoded. For instance, audio signals should be dealt with in the frequency domain, while image signals should be handled in the spatial domain. In spite of this, one way to measure image resolution is by looking at the *frequency response*. This goes against the fundamental

philosophy of this book; however, it is a common method and you need to become familiar with it.

Taking the two-dimensional Fourier transform of the PSF provides the two-dimensional frequency response. If the PSF is circularly symmetric, its frequency response will also be circularly symmetric. In this case, complete information about the frequency response is contained in its profile. That is, after calculating the frequency domain via the FFT method, columns 0 to $N/2$ in row 0 are all that is needed. In imaging jargon, this display of the frequency response is called the **Modulation Transfer Function (MTF)**. Figure 25-1b shows the MTFs for the three PSFs in (a). In cases where the PSF is not circularly symmetric, the entire two-dimensional frequency response contains information. However, it is usually sufficient to know the MTF curves in the vertical and horizontal directions (i.e., columns 0 to $N/2$ in row 0, and rows 0 to $N/2$ in column 0). Take note: this procedure of extracting a row or column from the two-dimensional frequency spectrum is *not* equivalent to taking the one-dimensional FFT of the profiles shown in (a). We will come back to this issue shortly. As shown in Fig. 25-1, similar values of FWHM do not correspond to similar MTF curves.

Figure 25-2 shows a **line pair gauge**, a device used to measure image resolution via the MTF. Line pair gauges come in different forms depending on the particular application. For example, the black and white pattern shown in this figure could be directly used to test video cameras. For an x-ray imaging system, the ribs might be made from lead, with an x-ray transparent material between. The key feature is that the black and white lines have a closer spacing toward one end. When an image is taken of a line pair gauge, the lines at the closely spaced end will be blurred together, while at the other end they will be distinct. Somewhere in the middle the lines will be just barely separable. An observer looks at the image, identifies this location, and reads the corresponding resolution on the calibrated scale.

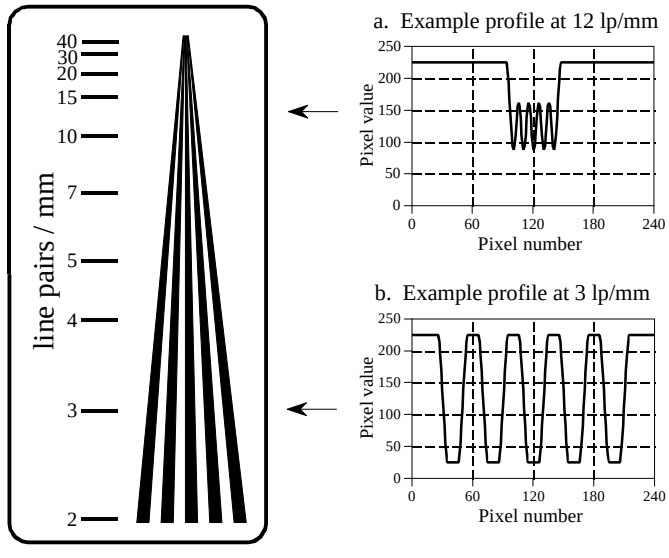


FIGURE 25-2
Line pair gauge. The line pair gauge is a tool used to measure the resolution of imaging systems. A series of black and white ribs move together, creating a continuum of spatial frequencies. The resolution of a system is taken as the frequency where the eye can no longer distinguish the individual ribs. This example line pair gauge is shown several times larger than the calibrated scale indicates.

The *way* that the ribs blur together is important in understanding the limitations of this measurement. Imagine acquiring an image of the line pair gauge in Fig. 25-2. Figures (a) and (b) show examples of the profiles at low and high spatial frequencies. At the low frequency, shown in (b), the curve is flat on the top and bottom, but the edges are blurred, At the higher spatial frequency, (a), the *amplitude* of the modulation has been reduced. This is exactly what the MTF curve in Fig. 25-1b describes: higher spatial frequencies are reduced in amplitude. The individual ribs will be distinguishable in the image as long as the amplitude is greater than about 3% to 10% of the original height. This is related to the eye's ability to distinguish the low contrast difference between the peaks and valleys in the presence of image noise.

A strong advantage of the line pair gauge measurement is that it is simple and fast. The strongest disadvantage is that it relies on the human eye, and therefore has a certain subjective component. Even if the entire MTF curve is measured, the most common way to express the system resolution is to quote the frequency where the MTF is reduced to either 3%, 5% or 10%. Unfortunately, you will not always be told which of these values is being used; product data sheets frequently use vague terms such as "limiting resolution." Since manufacturers like their specifications to be as good as possible (regardless of what the device actually does), be safe and interpret these ambiguous terms to mean 3% on the MTF curve.

A subtle point to notice is that the MTF is defined in terms of *sine* waves, while the line pair gauge uses *square* waves. That is, the ribs are uniformly dark regions separated by uniformly light regions. This is done for manufacturing convenience; it is very difficult to make lines that have a sinusoidally varying darkness. What are the consequences of using a square wave to measure the MTF? At high spatial frequencies, all frequency components but the fundamental of the square wave have been removed. This causes the modulation to appear sinusoidal, such as is shown in Fig. 25-2a. At low frequencies, such as shown in Fig. 25-2b, the wave appears square. The fundamental sine wave contained in a square wave has an amplitude of $4/\pi = 1.27$ times the amplitude of the square wave (see Table 13-10). The result: the line pair gauge provides a slight overestimate of the true resolution of the system, by starting with an effective amplitude of more than pure black to pure white. Interesting, but almost always ignored.

Since square waves and sine waves are used interchangeably to measure the MTF, a special terminology has arisen. Instead of the word "cycle," those in imaging use the term **line pair** (a dark line next to a light line). For example, a spatial frequency would be referred to as *25 line pairs per millimeter*, instead of *25 cycles per millimeter*.

The width of the PSF doesn't track well with human perception and is difficult to measure. The MTF methods are in the *wrong domain* for understanding how resolution affects the encoded information. Is there a more favorable alternative? The answer is yes, the **line spread function (LSF)** and the **edge response**. As shown in Fig. 25-3, the line spread

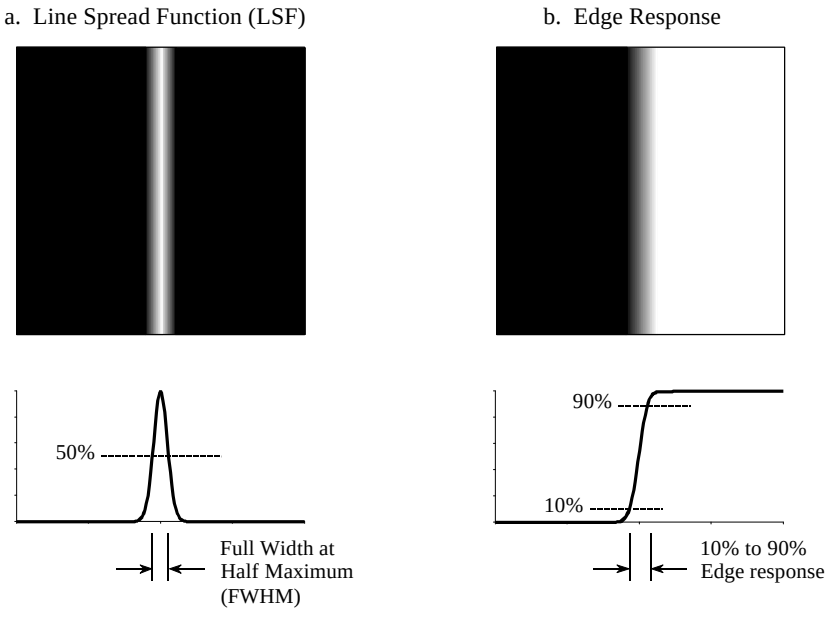


FIGURE 25-3
Line spread function and edge response. The line spread function (LSF) is the derivative of the edge response. The width of the LSF is usually expressed as the Full-Width-at-Half-Maximum (FWHM). The width of the edge response is usually quoted by the 10% to 90% distance.

function is the response of the system to a thin line across the image. Similarly, the edge response is how the system responds to a sharp straight discontinuity (an edge). Since a line is the derivative (or first difference) of an edge, the LSF is the derivative (or first difference) of the edge response. The single parameter measurement used here is the distance required for the edge response to rise from 10% to 90%.

There are many advantages to using the edge response for measuring resolution. First, the measurement is in the same form as the image information is encoded. In fact, the main reason for wanting to know the resolution of a system is to understand how the edges in an image are *blurred*. The second advantage is that the edge response is simple to measure because edges are easy to generate in images. If needed, the LSF can easily be found by taking the first difference of the edge response.

The third advantage is that all common edges responses have a similar shape, even though they may originate from drastically different PSFs. This is shown in Fig. 25-4a, where the edge responses of the pillbox, Gaussian, and exponential PSFs are displayed. Since the shapes are similar, the 10%-90% distance is an excellent single parameter measure of resolution. The fourth advantage is that the MTF can be directly found by taking the one-dimensional FFT of the LSF (unlike the PSF to MTF calculation that must use a two-dimensional Fourier transform). Figure 25-4b shows the MTFs corresponding to the edge responses of (a). In other words, the curves in (a) are converted into the curves in (b) by taking the first difference (to find the LSF), and then taking the FFT.

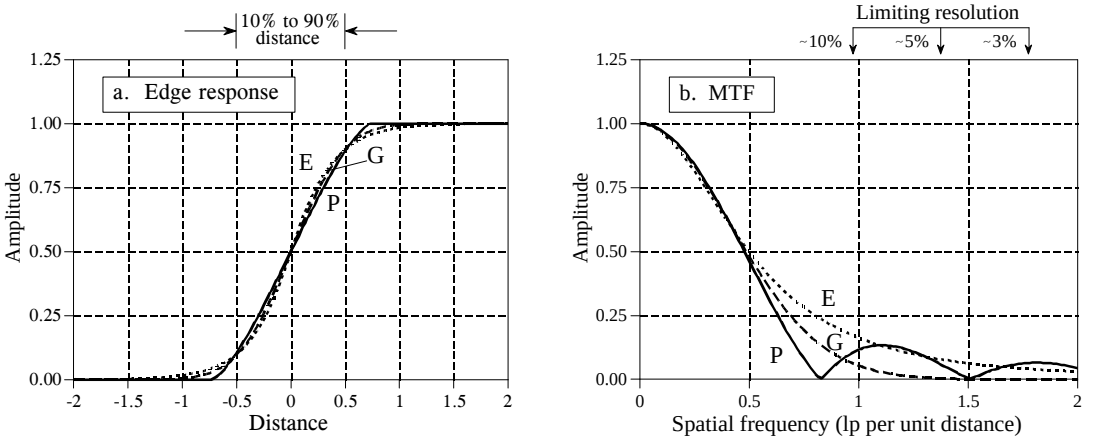


FIGURE 25-4
Edge response and MTF. Figure (a) shows the edge responses of three PSFs: (P) pillbox, (G) Gaussian, and (E) exponential. Each edge response has a 10% to 90% rise distance of 1 unit. Figure (b) shows the corresponding MTF curves, which are similar above the 10% level. *Limiting resolution* is a vague term indicating the frequency where the MTF has an amplitude of 3% to 10%.

The fifth advantage is that similar edge responses have similar MTF curves, as shown in Figs. 25-4 (a) and (b). This allows us to easily convert between the two measurements. In particular, a system that has a 10%-90% edge response of *x* distance, has a limiting resolution (10% contrast) of about 1 line pair per *x* distance. The units of the "distance" will depend on the type of system being dealt with. For example, consider three different imaging systems that have 10%-90% edge responses of 0.05 mm, 0.2 milliradian and 3.3 pixels. The 10% contrast level on the corresponding MTF curves will occur at about: 20 lp/mm, 5 lp/milliradian and 0.33 lp/pixel, respectively.

Figure 25-5 illustrates the mathematical relationship between the PSF and the LSF. Figure (a) shows a pillbox PSF, a circular area of value 1, displayed as white, surrounded by a region of all zeros, displayed as gray. A profile of the PSF (i.e., the pixel values along a line drawn across the center of the image) will be a rectangular pulse. Figure (b) shows the corresponding LSF. As shown, the LSF is mathematically equal to the *integrated profile* of the PSF. This is found by sweeping across the image in some direction, as illustrated by the rays (arrows). Each value in the integrated profile is the *sum* of the pixel values along the corresponding ray.

In this example where the rays are vertical, each point in the integrated profile is found by adding all the pixel values in each column. This corresponds to the LSF of a line that is *vertical* in the image. The LSF of a line that is *horizontal* in the image is found by summing all of the pixel values in each *row*. For continuous images these concepts are the same, but the summations are replaced by integrals.

As shown in this example, the LSF can be directly calculated from the PSF. However, the PSF cannot always be calculated from the LSF. This is because the PSF contains information about the spatial resolution in *all directions*, while the LSF is limited to only one specific direction. A system

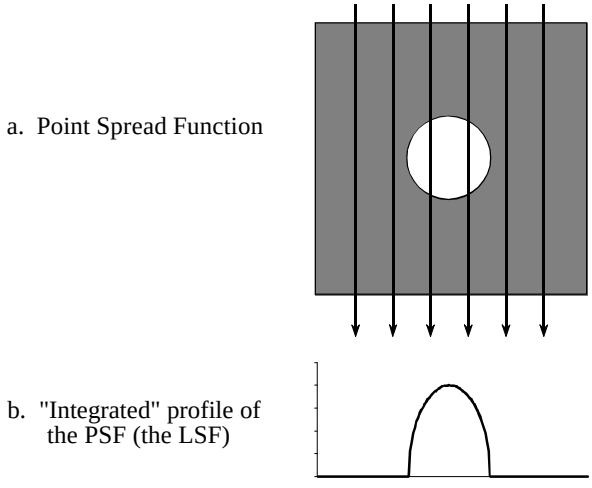


FIGURE 25-5
Relationship between the PSF and LSF. A pillbox PSF is shown in (a). Any row or column through the white center will be a rectangular pulse. Figure (b) shows the corresponding LSF, equivalent to an *integrated profile* of the PSF. That is, the LSF is found by sweeping across the image in some direction and adding (integrating) the pixel values along each ray. In the direction shown, this is done by adding all the pixels in each column.

has only one PSF, but an infinite number of LSFs, one for each angle. For example, imagine a system that has an oblong PSF. This makes the spatial resolution different in the vertical and horizontal directions, resulting in the LSF being different in these directions. Measuring the LSF at a single angle does not provide enough information to calculate the complete PSF except in the special instance where the PSF is circularly symmetric. Multiple LSF measurements at various angles make it possible to calculate a non-circular PSF; however, the mathematics is quite involved and usually not worth the effort. In fact, the problem of calculating the PSF from a number of LSF measurements is exactly the same problem faced in *computed tomography*, discussed later in this chapter.

As a practical matter, the LSF and the PSF are not dramatically different for most imaging systems, and it is very common to see one used as an approximation for the other. This is even more justifiable considering that there are two common cases where they are identical: the rectangular PSF has a rectangular LSF (with the same widths), and the Gaussian PSF has a Gaussian LSF (with the same standard deviations).

These concepts can be summarized into two skills: how to *evaluate* a resolution specification presented to you, and how to *measure* a resolution specification of your own. Suppose you come across an advertisement stating: "This system will resolve 40 line pairs per millimeter." You should interpret this to mean: "A sinusoid of 40 lp/mm will have its amplitude reduced to 3%-10% of its true value, and will be just barely visible in the image." You should also do the mental calculation that 40 lp/mm @ 10% contrast is equal to a 10%-90% edge response of 1/(40 lp/mm) = 0.025 mm. If the MTF specification is for a 3% contrast level, the edge response will be about 1.5 to 2 times wider.

When you measure the spatial resolution of an imaging system, the steps are carried out in reverse. Place a sharp edge in the image, and measure the

resulting edge response. The 10%-90% distance of this curve is the best single parameter measurement of the system's resolution. To keep your boss and the marketing people happy, take the first difference of the edge response to find the LSF, and then use the FFT to find the MTF.

Sample Spacing and Sampling Aperture

Figure 25-6 shows two extreme examples of sampling, which we will call a **perfect detector** and a **blurry detector**. Imagine (a) being the surface of an imaging detector, such as a CCD. Light striking anywhere inside one of the square pixels will contribute *only* to that pixel value, and no others. This is shown in the figure by the black sampling aperture exactly filling one of the square pixels. This is an optimal situation for an image detector, because *all* of the light is detected, and there is *no* overlap or crosstalk between adjacent pixels. In other words, the sampling aperture is exactly equal to the sample spacing.

The alternative example is portrayed in (e). The sampling aperture is considerably larger than the sample spacing, and it follows a Gaussian distribution. In other words, each pixel in the detector receives a contribution from light striking the detector in a region *around* the pixel. This should sound familiar, because it is the output side viewpoint of convolution. From the corresponding input side viewpoint, a narrow beam of light striking the detector would contribute to the value of several neighboring pixels, also according to the Gaussian distribution.

Now turn your attention to the edge responses of the two examples. The markers in each graph indicate the actual pixel values you would find in an image, while the connecting lines show the *underlying curve* that is being sampled. An important concept is that the shape of this underlying curve is determined *only* by the *sampling aperture*. This means that the resolution in the final image can be limited in two ways. First, the underlying curve may have poor resolution, resulting from the sampling aperture being too large. Second, the sample spacing may be too large, resulting in small details being lost between the samples. Two edge response curves are presented for each example, illustrating that the actual samples can fall anywhere along the underlying curve. In other words, the edge being imaged may be sitting exactly upon a pixel, or be straddling two pixels. Notice that the perfect detector has *zero* or *one* sample on the rising part of the edge. Likewise, the blurry detector has *three* to *four* samples on the rising part of the edge.

What is limiting the resolution in these two systems? The answer is provided by the *sampling theorem*. As discussed in Chapter 3, sampling captures all frequency components below one-half of the sampling rate, while higher frequencies are lost due to aliasing. Now look at the MTF curve in (h). The sampling aperture of the blurry detector has removed all frequencies greater than one-half the sampling rate; therefore, *nothing* is lost during sampling. This means that the resolution of this system is

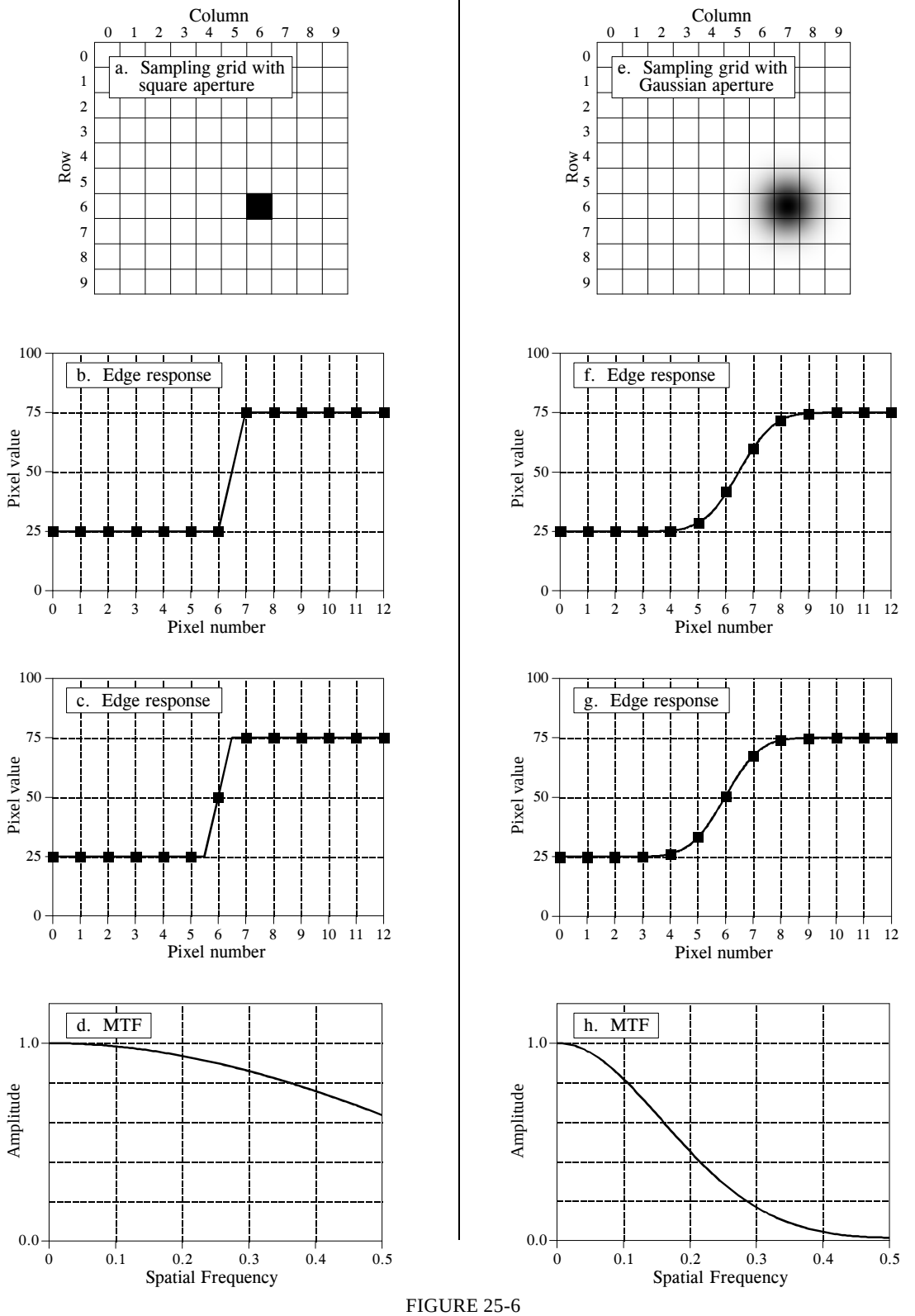

Example 1: Perfect detector
Example 2: Blurry detector
a. Sampling grid with square aperture
e. Sampling grid with Gaussian aperture
Column
Row
b. Edge response
f. Edge response
c. Edge response
g. Edge response
Pixel value
Pixel number
d. MTF
h. MTF
Amplitude
Spatial Frequency

FIGURE 25-6

completely limited by the sampling aperture, and not the sample spacing. Put another way, the sampling aperture has acted as an antialias filter, allowing lossless sampling to take place.

In comparison, the MTF curve in (d) shows that *both* processes are limiting the resolution of this system. The high-frequency fall-off of the MTF curve represents information lost due to the *sampling aperture*. Since the MTF curve has not dropped to zero before a frequency of 0.5, there is also information lost during sampling, a result of the finite *sample spacing*. Which is limiting the resolution more? It is difficult to answer this question with a number, since they degrade the image in different ways. Suffice it to say that the resolution in the perfect detector (example 1) is mostly limited by the sample spacing.

While these concepts may seem difficult, they reduce to a very simple rule for practical usage. Consider a system with some 10%-90% edge response distance, for example 1 mm. If the sample spacing is greater than 1 mm (there is less than one sample along the edge), the system will be limited by the *sample spacing*. If the sample spacing is less than 0.33 mm (there are more than 3 samples along the edge), the resolution will be limited by the *sampling aperture*. When a system has 1-3 samples per edge, it will be limited by both factors.

Signal-to-Noise Ratio

An object is visible in an image because it has a different brightness than its surroundings. That is, the contrast of the object (i.e., the signal) must overcome the image noise. This can be broken into two classes: limitations of the *eye*, and limitations of the *data*.

Figure 25-7 illustrates an experiment to measure the eye's ability to detect weak signals. Depending on the observation conditions, the human eye can detect a minimum contrast of 0.5% to 5%. In other words, humans can distinguish about 20 to 200 shades of gray between the blackest black and the whitest white. The exact number depends on a variety of factors, such

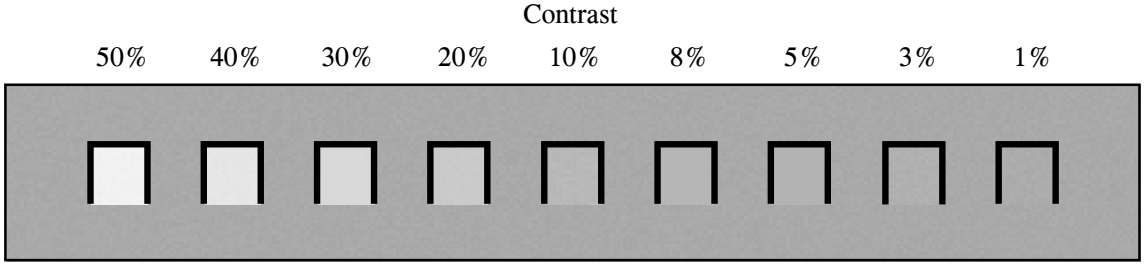


FIGURE 25-7
Contrast detection. The human eye can detect a minimum contrast of about 0.5 to 5%, depending on the observation conditions. 100% contrast is the difference between pure black and pure white.

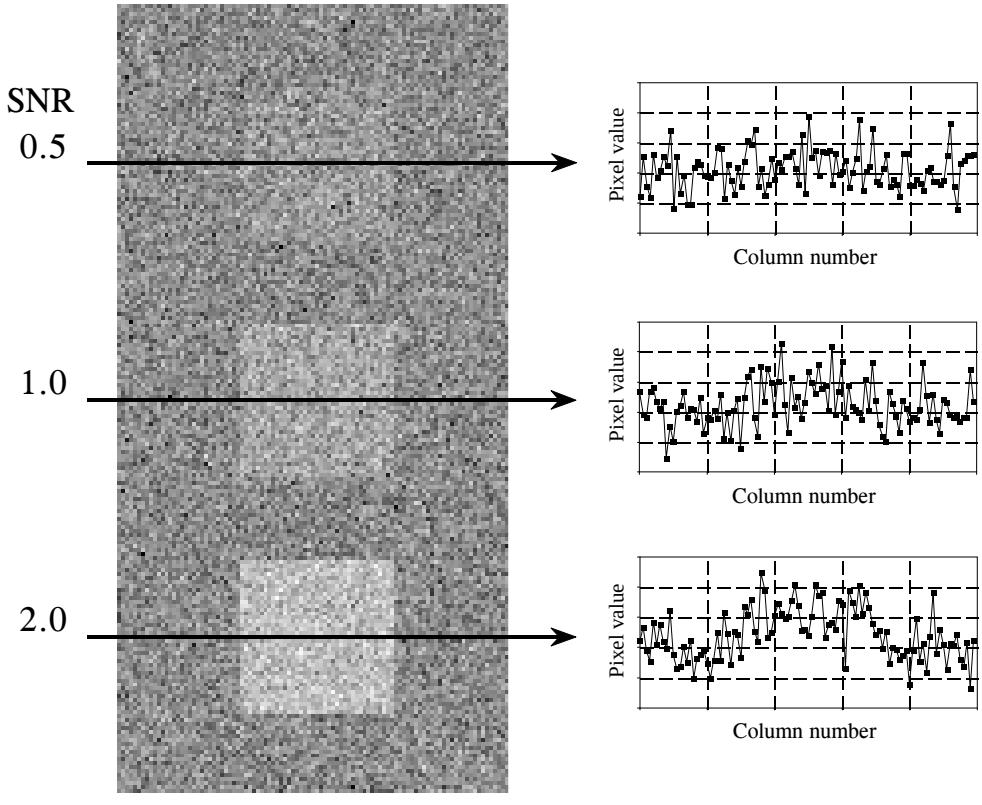


FIGURE 25-8
Minimum detectable SNR. An object is visible in an image only if its contrast is large enough to overcome the random image noise. In this example, the three squares have SNRs of 2.0, 1.0 and 0.5 (where the SNR is defined as the contrast of the object divided by the standard deviation of the noise).

as the brightness of the ambient lightning, the distance between the two regions being compared, and how the grayscale image is formed (video monitor, photograph, halftone, etc.).

The grayscale transform of Chapter 23 can be used to boost the contrast of a selected range of pixel values, providing a valuable tool in overcoming the limitations of the human eye. The contrast at one brightness level is increased, at the cost of reducing the contrast at another brightness level. However, this only works when the contrast of the object is not lost in random image noise. This is a more serious situation; the *signal* does not contain enough information to reveal the object, regardless of the performance of the eye.

Figure 25-8 shows an image with three squares having contrasts of 5%, 10%, and 20%. The background contains normally distributed random noise with a standard deviation of about 10% contrast. The SNR is defined as the contrast divided by the standard deviation of the noise, resulting in the three squares having SNRs of 0.5, 1.0 and 2.0. In general, trouble begins when the SNR falls below about 1.0.

The exact value for the minimum detectable SNR depends on the *size* of the object; the larger the object, the easier it is to detect. To understand this, imagine smoothing the image in Fig. 25-8 with a 3×3 square filter kernel. This leaves the contrast the same, but reduces the noise by a factor of three (i.e., the square root of the number of pixels in the kernel). Since the SNR is tripled, lower contrast objects can be seen. To see fainter objects, the filter kernel can be made even larger. For example, a 5×5 kernel improves the SNR by a factor of $\sqrt{25} = 5$. This strategy can be continued until the filter kernel is equal to the size of the object being detected. This means the ability to detect an object is proportional to the *square-root* of its *area*. If an object's diameter is doubled, it can be detected in twice as much noise.

Visual processing in the brain behaves in much the same way, smoothing the viewed image with various size filter kernels in an attempt to recognize low contrast objects. The three profiles in Fig. 25-8 illustrate just how good humans are at detecting objects in noisy environments. Even though the objects can hardly be identified in the profiles, they are obvious in the image. To really appreciate the capabilities of the human visual system, try writing algorithms that operate in this low SNR environment. You'll be humbled by what your brain can do, but your code can't!

Random image noise comes in two common forms. The first type, shown in Fig. 25-9a, has a constant amplitude. In other words, dark and light regions in the image are equally noisy. In comparison, (b) illustrates noise that *increases with the signal level*, resulting in the bright areas being more noisy than the dark ones. Both sources of noise are present in most images, but one or the other is usually dominant. For example, it is common for the noise to decrease as the signal level is decreased, until a plateau of constant amplitude noise is reached.

A common source of constant amplitude noise is the video *preamplifier*. All analog electronic circuits produce noise. However, it does the most harm where the signal being amplified is at its smallest, right at the CCD or other imaging sensor. Preamplifier noise originates from the random motion of electrons in the transistors. This makes the noise level depend on how the electronics are designed, but not on the level of the signal being amplified. For example, a typical CCD camera will have an SNR of about 300 to 1000 (40 to 60 dB), defined as the full scale signal level divided by the standard deviation of the constant amplitude noise.

Noise that increases with the signal level results when the image has been represented by a small number of individual particles. For example, this might be the *x-rays* passing through a patient, the *light photons* entering a camera, or the *electrons* in the well of a CCD. The mathematics governing these variations are called **counting statistics** or **Poisson statistics**. Suppose that the face of a CCD is uniformly illuminated such that an average of 10,000 electrons are generated in each well. By sheer chance, some wells will have more electrons, while some will have less. To be more exact, the number of electrons will be normally distributed with a mean of 10,000, with some standard deviation that describes how much variation there is from

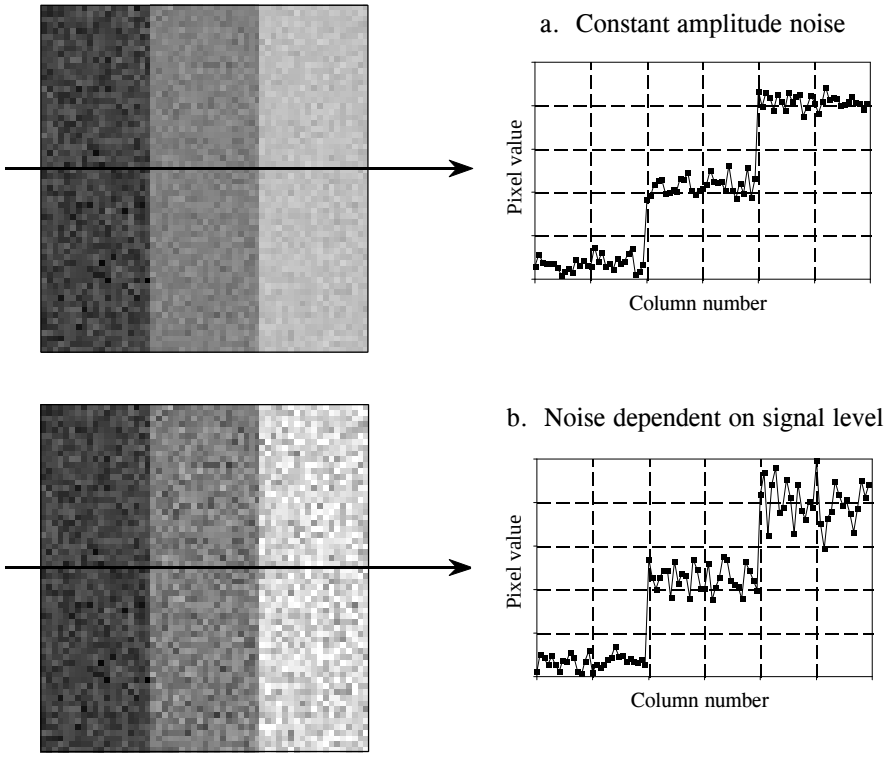


FIGURE 25-9
Image noise. Random noise in images takes two general forms. In (a), the amplitude of the noise remains constant as the signal level changes. This is typical of electronic noise. In (b), the amplitude of the noise increases as the square-root of the signal level. This type of noise originates from the detection of a small number of particles, such as light photons, electrons, or x-rays.

well-to-well. A key feature of Poisson statistics is that the standard deviation is equal to the square-root of the number of individual particles. That is, if there are N particles in each pixel, the mean is equal to N and the standard deviation is equal to $\sqrt{N}$. This makes the signal-to-noise ratio equal to $N/\sqrt{N}$, or simply, $\sqrt{N}$. In equation form:

EQUATION 25-1
Poisson statistics. In a Poisson distributed signal, the mean, μ, is the average number of individual particles, N. The standard deviation, σ, is equal to the square-root of the average number of individual particles. The signal-to-noise ratio (SNR) is the mean divided by the standard deviation.

$$\mu = N$$

$$\sigma = \sqrt{N}$$

$$SNR = \sqrt{N}$$

In the CCD example, the standard deviation is $\sqrt{10{,}000} = 100$. Likewise the signal-to-noise ratio is also $\sqrt{10{,}000} = 100$. If the average number of electrons per well is increased to one million, both the standard deviation and the SNR increase to 1,000. That is, the noise becomes larger as the signal becomes

larger, as shown in Fig. 25-9b. However, the signal is becoming larger *faster* than the noise, resulting in an overall improvement in the SNR. Don't be confused into thinking that a lower signal will provide less noise and therefore better information. Remember, your goal is *not* to reduce the noise, but to extract a signal *from* the noise. This makes the SNR the key parameter.

Many imaging systems operate by converting one particle type to another. For example, consider what happens in a medical x-ray imaging system. Within an x-ray tube, *electrons* strike a metal target, producing *x-rays*. After passing through the patient, the x-rays strike a vacuum tube detector known as an image intensifier. Here the x-rays are subsequently converted into *light photons*, then *electrons*, and then back to *light photons*. These light photons enter the camera where they are converted into *electrons* in the well of a CCD. In each of these intermediate forms, the image is represented by a finite number of particles, resulting in added noise as dictated by Eq. 25-1. The final SNR reflects the combined noise of *all* stages; however, one stage is usually dominant. This is the stage with the *worst* SNR because it has the *fewest* particles. This limiting stage is called the **quantum sink**.

In night vision systems, the quantum sink is the number of light photons that can be captured by the camera. The darker the night, the noisier the final image. Medical x-ray imaging is a similar example; the quantum sink is the number of x-rays striking the detector. Higher radiation levels provide less noisy images at the expense of more radiation to the patient.

When is the noise from Poisson statistics the primary noise in an image? It is dominant whenever the noise resulting from the quantum sink is greater than the other sources of noise in the system, such as from the electronics. For example, consider a typical CCD camera with an SNR of 300. That is, the noise from the CCD preamplifier is 1/300th of the full scale signal. An equivalent noise would be produced if the quantum sink of the system contains 90,000 particles per pixel. If the quantum sink has a smaller number of particles, Poisson noise will dominate the system. If the quantum sink has a larger number of particles, the preamplifier noise will be predominant. Accordingly, most CCD's are designed with a full well capacity of 100,000 to 1,000,000 electrons, minimizing the Poisson noise.

Morphological Image Processing

The identification of objects within an image can be a very difficult task. One way to simplify the problem is to change the grayscale image into a **binary image**, in which each pixel is restricted to a value of either 0 or 1. The techniques used on these binary images go by such names as: **blob analysis**, **connectivity analysis**, and **morphological image processing** (from the Greek word *morphe*, meaning shape or form). The foundation of morphological processing is in the mathematically rigorous field of *set theory*; however, this level of sophistication is seldom needed. Most morphological algorithms are simple logic operations and very *ad hoc*. In

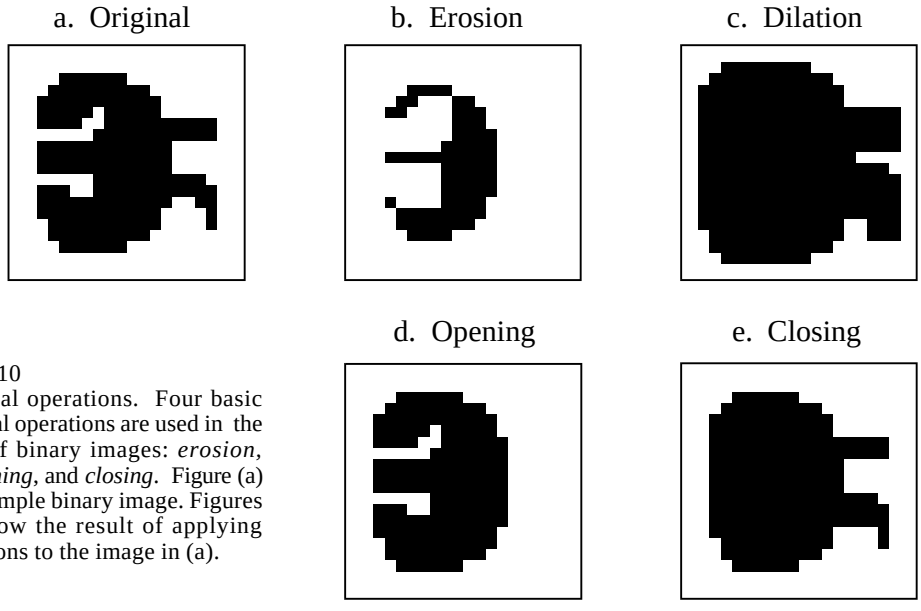


FIGURE 25-10
Morphological operations. Four basic morphological operations are used in the processing of binary images: *erosion*, *dilation*, *opening*, and *closing*. Figure (a) shows an example binary image. Figures (b) to (e) show the result of applying these operations to the image in (a).

other words, each application requires a custom solution developed by trial-and-error. This is usually more of an art than a science. A bag of tricks is used rather than standard algorithms and formal mathematical properties. Here are some examples.

Figure 25-10a shows an example binary image. This might represent an enemy tank in an infrared image, an asteroid in a space photograph, or a suspected tumor in a medical x-ray. Each pixel in the background is displayed as white, while each pixel in the object is displayed as black. Frequently, binary images are formed by thresholding a grayscale image; pixels with a value greater than a threshold are set to 1, while pixels with a value below the threshold are set to 0. It is common for the grayscale image to be processed with linear techniques before the thresholding. For instance, *illumination flattening* (described in Chapter 24) can often improve the quality of the initial binary image.

Figures (b) and (c) show how the image is changed by the two most common morphological operations, **erosion** and **dilation**. In erosion, every object pixel that is touching a background pixel is changed into a background pixel. In dilation, every background pixel that is touching an object pixel is changed into an object pixel. Erosion makes the objects smaller, and can break a single object into multiple objects. Dilation makes the objects larger, and can merge multiple objects into one.

As shown in (d), **opening** is defined as an erosion followed by a dilation. Figure (e) shows the opposite operation of **closing**, defined as a dilation followed by an erosion. As illustrated by these examples, *opening* removes small islands and thin filaments of *object pixels*. Likewise, *closing* removes

islands and thin filaments of *background pixels*. These techniques are useful for handling noisy images where some pixels have the wrong binary value. For instance, it might be known that an object cannot contain a "hole", or that the object's border must be smooth.

Figure 25-11 shows an example of morphological processing. Figure (a) is the binary image of a fingerprint. Algorithms have been developed to analyze these patterns, allowing individual fingerprints to be matched with those in a database. A common step in these algorithms is shown in (b), an operation called **skeletonization**. This simplifies the image by *removing* redundant pixels; that is, changing appropriate pixels from black to white. This results in each ridge being turned into a line only a single pixel wide.

Tables 25-1 and 25-2 show the skeletonization program. Even though the fingerprint image is binary, it is held in an array where each pixel can run from 0 to 255. A black pixel is denoted by 0, while a white pixel is denoted by 255. As shown in Table 25-1, the algorithm is composed of 6 iterations that gradually erode the ridges into a thin line. The number of iterations is chosen by trial and error. An alternative would be to stop when an iteration makes no changes.

During an iteration, each pixel in the image is evaluated for being *removable*; the pixel meets a set of criteria for being changed from black to white. Lines 200-240 loop through each pixel in the image, while the subroutine in Table 25-2 makes the evaluation. If the pixel under consideration is not removable, the subroutine does nothing. If the pixel is removable, the subroutine changes its value from 0 to 1. This indicates that the pixel is still black, but will be changed to white at the end of the iteration. After all the pixels have been evaluated, lines 260-300 change the value of the marked pixels from 1 to 255. This two-stage process results in the thick ridges being eroded equally from all directions, rather than a pattern based on how the rows and columns are scanned.

a. Original fingerprint

b. Skeletonized fingerprint

FIGURE 25-11
Binary skeletonization. The binary image of a fingerprint, (a), contains ridges that are many pixels wide. The skeletonized version, (b), contains ridges only a single pixel wide.

```
100 'SKELETONIZATION PROGRAM
110 'Object pixels have a value of 0 (displayed as black)
120 'Background pixels have a value of 255 (displayed as white)
130 '
140 DIM X%[149,149]              'X%[  ,  ] holds the image being processed
150 '
160 GOSUB XXXX                   'Mythical subroutine to load X%[ , ]
170 '
180 FOR ITER% = 0 TO 5           'Run through six iteration loops
190 '
200   FOR R% = 1 TO 148          'Loop through each pixel in the image.
210   FOR C% = 1 TO 148          'Subroutine 5000 (Table 25-2) indicates which
220     GOSUB 5000               'pixels can be changed from black to white,
230   NEXT C%                    'by marking the pixels with a value of 1.
240   NEXT R%
250 '
260   FOR R% = 0 TO 149          'Loop through each pixel in the image changing
270   FOR C% = 0 TO 149          'the marked pixels from black to white.
280     IF X%(R%,C%) = 1 THEN X%(R%,C%) = 255
290   NEXT C%
300   NEXT R%
310 '
320 NEXT ITER%
330 '
340 END
```

TABLE 25-1

The decision to remove a pixel is based on four rules, as contained in the subroutine shown in Table 25-2. *All* of these rules must be satisfied for a pixel to be changed from black to white. The first three rules are rather simple, while the fourth is quite complicated. As shown in Fig. 25-12a, a pixel at location [R,C] has eight neighbors. The four neighbors in the horizontal and vertical directions (labeled 2,4,6,8) are frequently called the **close neighbors**. The diagonal pixels (labeled 1,3,5,7) are correspondingly called the **distant neighbors**. The four rules are as follows:

Rule one: The pixel under consideration must presently be black. If the pixel is already white, no action needs to be taken.

Rule two: At least one of the pixel's close neighbors must be white. This insures that the erosion of the thick ridges takes place from the outside. In other words, if a pixel is black, and it is completely surrounded by black pixels, it is to be left alone on this iteration. Why use only the *close neighbors*, rather than *all* of the neighbors? The answer is simple: running the algorithm both ways shows that it works better. Remember, this is very common in morphological image processing; trial and error is used to find if one technique performs better than another.

Rule three: The pixel must have more than one black neighbor. If it has only one, it must be the end of a line, and therefore shouldn't be removed.

Rule four: A pixel cannot be removed if it results in its neighbors being *disconnected*. This is so each ridge is changed into a continuous line, not a group of interrupted segments. As shown by the examples in Fig. 25-12,

a. Pixel numbering

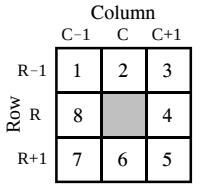
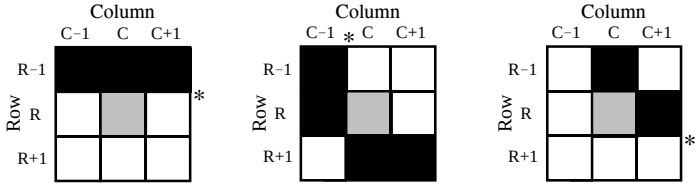
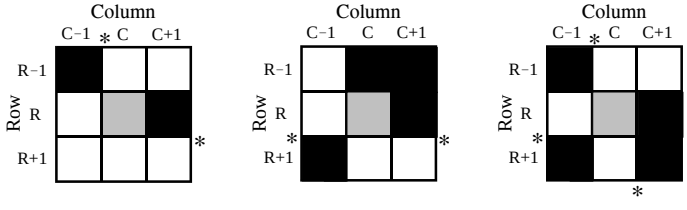


FIGURE 25-12
Neighboring pixels. A pixel at row and column [R,C] has eight neighbors, referred to by the numbers in (a). Figures (b) and (c) show examples where the neighboring pixels are *connected* and *unconnected*, respectively. This definition is used by rule number four of the skeletonization algorithm.

b. Connected neighbors



c. Unconnected neighbors



connected means that all of the black neighbors touch each other. Likewise, *unconnected* means that the black neighbors form two or more groups.

The algorithm for determining if the neighbors are connected or unconnected is based on counting the black-to-white transitions between adjacent neighboring pixels, in a *clockwise* direction. For example, if pixel 1 is black and pixel 2 is white, it is considered a black-to-white transition. Likewise, if pixel 2 is black and both pixel 3 and 4 are white, this is also a black-to-white transition. In total, there are eight locations where a black-to-white transition may occur. To illustrate this definition further, the examples in (b) and (c) have an asterisk placed by each black-to-white transition. The key to this algorithm is that there will be exactly *one* black-to-white transition if the neighbors are *connected*. More than one such transition indicates that the neighbors are *unconnected*.

As additional examples of binary image processing, consider the types of algorithms that might be useful after the fingerprint is skeletonized. A disadvantage of this particular skeletonization algorithm is that it leaves a considerable amount of *fuzz*, short offshoots that stick out from the sides of longer segments. There are several different approaches for eliminating these artifacts. For example, a program might loop through the image removing the pixel at the end of every line. These pixels are identified

```
5000 ' Subroutine to determine if the pixel at X%[R%,C%] can be removed.
5010 ' If all four of the rules are satisfied, then X%(R%,C%], is set to a value of 1,
5020 ' indicating it should be removed at the end of the iteration.
5030 '
5040 'RULE #1:  Do nothing if the pixel already white
5050 IF X%(R%,C%) = 255 THEN RETURN
5060 '
5070 '
5080 'RULE #2:  Do nothing if all of the close neighbors are black
5090 IF  X%[R% -1,C%  ] <> 255 AND X%[R%  ,C%+1] <> 255 AND
          X%[R%+1,C%  ] <> 255 AND X%[R%  ,C% -1] <> 255  THEN RETURN
5100 '
5110 '
5120 'RULE #3:  Do nothing if only a single neighbor pixel is black
5130 COUNT% = 0
5140 IF X%[R% -1,C% -1]      = 0 THEN COUNT% = COUNT% + 1
5150 IF X%[R% -1,C%    ]     = 0 THEN COUNT% = COUNT% + 1
5160 IF X%[R% -1,C%+1]  = 0 THEN COUNT% = COUNT% + 1
5170 IF X%[R%    ,C%+1]  = 0 THEN COUNT% = COUNT% + 1
5180 IF X%[R%+1,C%+1]  = 0 THEN COUNT% = COUNT% + 1
5190 IF X%[R%+1,C%    ]     = 0 THEN COUNT% = COUNT% + 1
5200 IF X%[R%+1,C% -1]      = 0 THEN COUNT% = COUNT% + 1
5210 IF X%[R%    ,C% -1]     = 0 THEN COUNT% = COUNT% + 1
5220 IF COUNT% = 1 THEN RETURN
5230 '
5240 '
5250 'RULE 4:  Do nothing if the neighbors are unconnected.
5260 'Determine this by counting the black-to-white transitions
5270 'while moving clockwise through the 8 neighboring pixels.
5280 COUNT% = 0
5290 IF X%[R% -1,C% -1] = 0 AND X%[R% -1,C%    ]          > 0  THEN COUNT% = COUNT% + 1
5300 IF X%[R% -1,C%    ] = 0 AND X%[R% -1,C%+1]          > 0  AND X%[R%  ,C%+1] > 0
          THEN COUNT% = COUNT% + 1
5310 IF X%[R% -1,C%+1]      = 0 AND X%[R%    ,C%+1]     > 0  THEN COUNT% = COUNT% + 1
5320 IF X%[R%    ,C%+1]     = 0 AND X%[R%+1,C%+1]      > 0  AND X%[R%+1,C%  ] > 0
          THEN COUNT% = COUNT% + 1
5330 IF X%[R%+1,C%+1]      = 0 AND X%[R%+1,C%    ]     > 0  THEN COUNT% = COUNT% + 1
5340 IF X%[R%+1,C%    ]     = 0 AND X%[R%+1,C% -1]     > 0  AND X%[R%  ,C%-1] > 0
          THEN COUNT% = COUNT% + 1
5350 IF X%[R%+1,C% -1]      = 0 AND X%[R%    ,C% -1]     > 0  THEN COUNT% = COUNT% + 1
5360 IF X%[R%    ,C% -1]     = 0 AND X%[R% -1,C% -1]     > 0  AND X%[R%-1,C%  ] > 0
          THEN COUNT% = COUNT% + 1
5370 IF COUNT% > 1 THEN RETURN
5380 '
5390 '
5400 'If all rules are satisfied, mark the pixel to be set to white at the end of the iteration
5410 X%(R%,C%) = 1
5420 '
5430 RETURN
```

TABLE 25-2

by having only one black neighbor. Do this several times and the fuzz is removed at the expense of making each of the correct lines shorter. A better method would loop through the image identifying *branch pixels* (pixels that have more than two neighbors). Starting with each branch pixel, count the number of pixels in each offshoot. If the number of pixels in an offshoot is less than some value (say, 5), declare it to be fuzz, and change the pixels in the branch from black to white.

Another algorithm might change the data from a *bitmap* to a *vector mapped* format. This involves creating a list of the ridges contained in the image and the pixels contained in each ridge. In the vector mapped form, each ridge in the fingerprint has an individual identity, as opposed to an image composed of many unrelated pixels. This can be accomplished by looping through the image looking for the endpoints of each line, the pixels that have only one black neighbor. Starting from the endpoint, each line is traced from pixel to connecting pixel. After the opposite end of the line is reached, all the traced pixels are declared to be a single *object*, and treated accordingly in future algorithms.

Computed Tomography

A basic problem in imaging with x-rays (or other penetrating radiation) is that a *two-dimensional image* is obtained of a *three-dimensional object*. This means that structures can *overlap* in the final image, even though they are completely separate in the object. This is particularly troublesome in medical diagnosis where there are many anatomic structures that can interfere with what the physician is trying to see. During the 1930's, this problem was attacked by moving the x-ray source and detector in a coordinated motion during image formation. From the geometry of this motion, a single *plane* within the patient remains in focus, while structures outside this plane become blurred. This is analogous to a camera being focused on an object at 5 feet, while objects at a distance of 1 and 50 feet are blurry. These related techniques based on motion blurring are now collectively called **classical tomography.** The word *tomography* means *"a picture of a plane."*

In spite of being well developed for more than 50 years, classical tomography is rarely used. This is because it has a significant limitation: the interfering objects are not *removed* from the image, only *blurred*. The resulting image quality is usually too poor to be of practical use. The long sought solution was a system that could create an image representing a 2D slice through a 3D object with *no* interference from other structures in the 3D object.

This problem was solved in the early 1970s with the introduction of a technique called **computed tomography (CT)**. CT revolutionized the medical x-ray field with its unprecedented ability to visualize the anatomic structure of the body. Figure 25-13 shows a typical medical CT image. Computed tomography was originally introduced to the marketplace under the names *Computed Axial Tomography* and *CAT scanner*. These terms are now frowned upon in the medical field, although you hear them used frequently by the general public.

Figure 25-14 illustrates a simple geometry for acquiring a CT slice through the center of the head. A narrow pencil beam of x-rays is passed from the x-ray source to the x-ray detector. This means that the measured value at the detector is related to the total amount of material placed *anywhere*

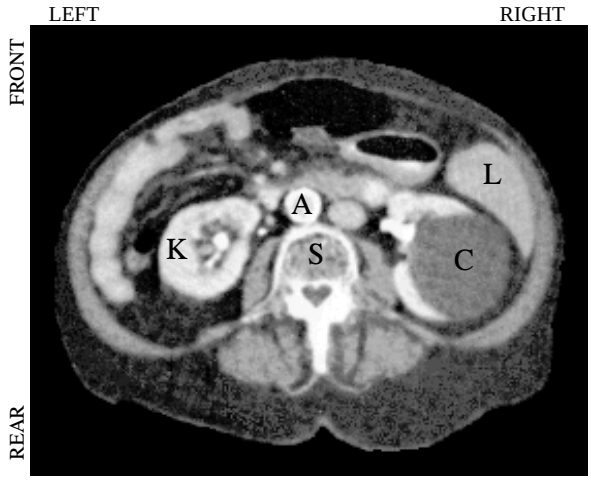


FIGURE 25-13
Computed tomography image. This CT slice is of a human abdomen, at the level of the navel. Many organs are visible, such as the (L) Liver, (K) Kidney, (A) Aorta, (S) Spine, and (C) Cyst covering the right kidney. CT can visualize internal anatomy far better than conventional medical x-rays.

along the beam's path. Materials such as bone and teeth block more of the x-rays, resulting in a lower signal compared to soft tissue and fat. As shown in the illustration, the source and detector assemblies are translated to acquire a **view** (CT jargon) at this particular angle. While this figure shows only a single view being acquired, a complete CT scan requires 300 to 1000 views taken at rotational increments of about 0.3° to 1.0°. This is accomplished by mounting the x-ray source and detector on a rotating gantry that surrounds the patient. A key feature of CT data acquisition is that x-rays pass *only* through the slice of the body being examined. This is unlike classical tomography where x-rays are passing through structures that you try to suppress in the final image. Computed tomography doesn't allow information from irrelevant locations to even enter the acquired data.

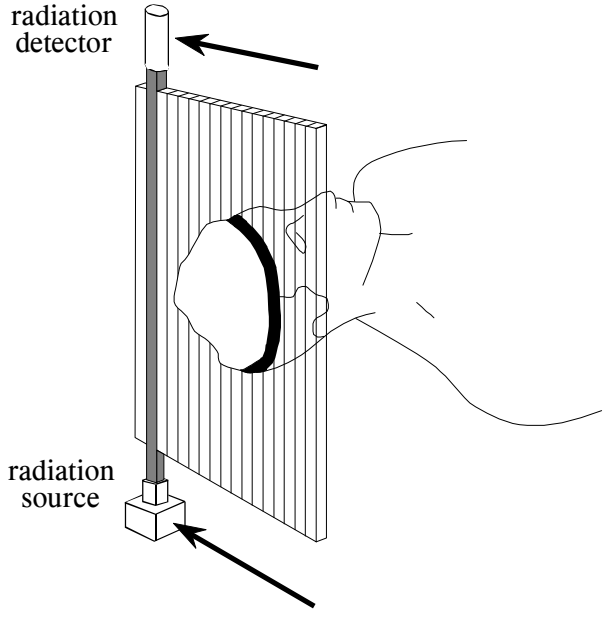


FIGURE 25-14
CT data acquisition. A simple CT system passes a narrow beam of x-rays through the body from source to detector. The source and detector are then translated to obtain a complete view. The remaining views are obtained by rotating the source and detector in about 1° increments, and repeating the translation process.

Several preprocessing steps are usually needed before the image reconstruction can take place. For instance, the logarithm must be taken of each x-ray measurement. This is because x-rays decrease in intensity exponentially as they pass through material. Taking the logarithm provides a signal that is linearly related to the characteristics of the material being measured. Other preprocessing steps are used to compensate for the use of *polychromatic* (more than one energy) x-rays, and *multielement* detectors (as opposed to the single element shown in Fig. 25-14). While these are a key step in the overall technique, they are not related to the reconstruction algorithms and we won't discuss them further.

Figure 25-15 illustrates the relationship between the measured views and the corresponding image. Each sample acquired in a CT system is equal to the sum of the image values along a ray pointing to that sample. For example, view 1 is found by adding all the pixels in each row. Likewise, view 3 is found by adding all the pixels in each column. The other views, such as view 2, sum the pixels along rays that are at an angle.

There are four main approaches to calculating the slice image given the set of its views. These are called **CT reconstruction algorithms**. The first method is totally impractical, but provides a better understanding of the problem. It is based on solving many **simultaneous linear equations**. One equation can be written for each measurement. That is, a particular sample in a particular profile is the sum of a particular group of pixels in the image. To calculate N^2 unknown variables (i.e., the image pixel values), there must be N^2 independent equations, and therefore N^2 measurements. Most CT scanners acquire about 50% more samples than rigidly required by this analysis. For example, to reconstruct a 512×512 image, a system might take 700 views with 600 samples in each view. By making the problem *overdetermined* in this manner, the final image has reduced noise and artifacts. The problem with this first method of CT reconstruction is computation time. Solving several hundred thousand simultaneous linear equations is an daunting task.

The second method of CT reconstruction uses **iterative** techniques to calculate the final image in small steps. There are several variations of this method: the Algebraic Reconstruction Technique (ART), Simultaneous Iterative Reconstruction Technique (SIRT), and Iterative Least Squares Technique (ILST). The difference between these methods is how the successive corrections are made: ray-by-ray, pixel-by-pixel, or simultaneously correcting the entire data set, respectively. As an example of these techniques, we will look at ART.

To start the ART algorithm, all the pixels in the image array are set to some arbitrary value. An iterative procedure is then used to gradually change the image array to correspond to the profiles. An iteration cycle consists of looping through each of the measured data points. For each measured value, the following question is asked: *how can the pixel values in the array be changed to make them consistent with this particular measurement*? In other words, the measured sample is compared with the

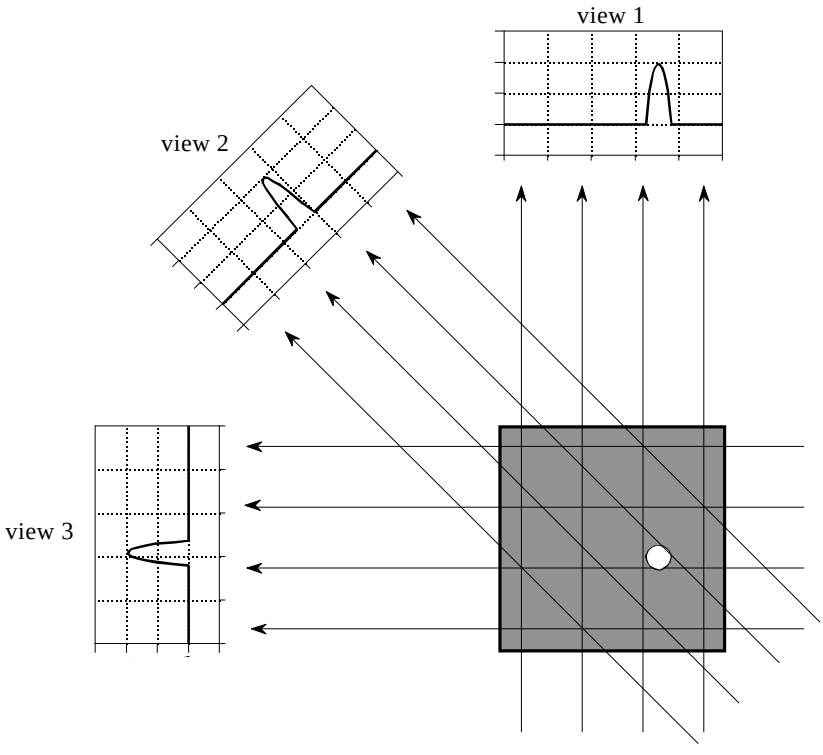


FIGURE 25-15
CT views. Computed tomography acquires a set of views and then reconstructs the corresponding image. Each sample in a view is equal to the sum of the image values along the ray that points to that sample. In this example, the image is a small pillbox surrounded by zeros. While only three views are shown here, a typical CT scan uses hundreds of views at slightly different angles.

sum of the image pixels along the ray pointing to the sample. If the ray sum is lower than the measured sample, all the pixels along the ray are increased in value. Likewise, if the ray sum is higher than the measured sample, all of the pixel values along the ray are decreased. After the first complete iteration cycle, there will still be an error between the ray sums and the measured values. This is because the changes made for any one measurement disrupts all the previous corrections made. The idea is that the errors become smaller with repeated iterations until the image converges to the proper solution.

Iterative techniques are generally slow, but they are useful when better algorithms are not available. In fact, ART was used in the first commercial medical CT scanner released in 1972, the EMI Mark I. We will revisit iterative techniques in the next chapter on neural networks. Development of the third and forth methods have almost entirely replaced iterative techniques in commercial CT products.

The last two reconstruction algorithms are based on formal mathematical solutions to the problem. These are elegant examples of DSP. The third method is called **filtered backprojection**. It is a modification of an older

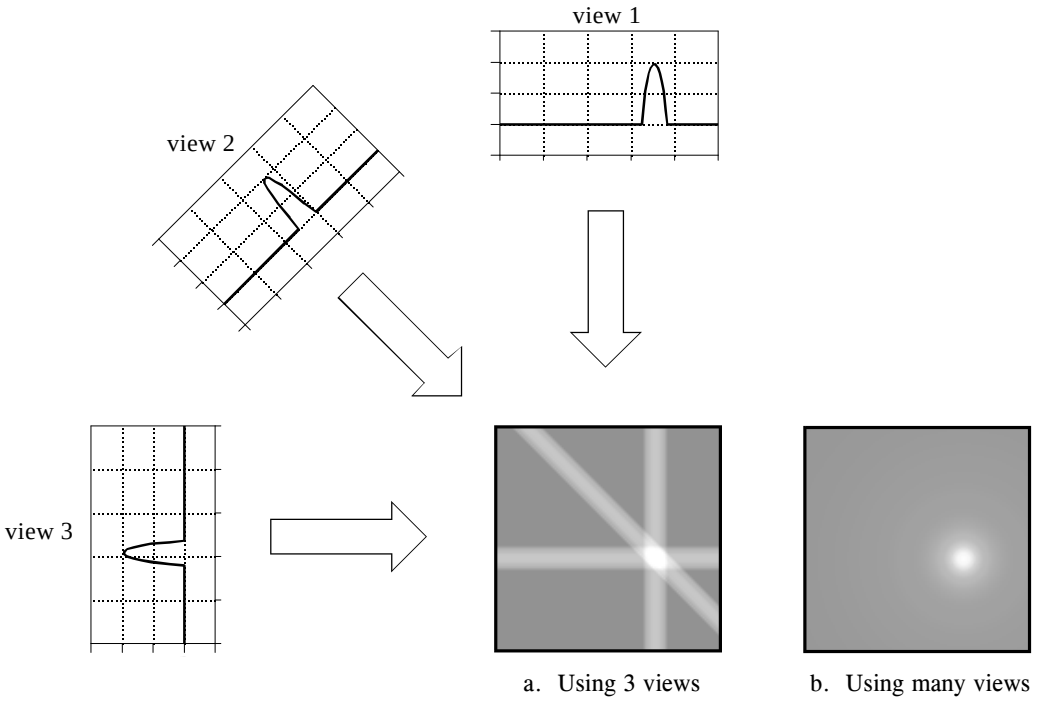


FIGURE 25-16
Backprojection. Backprojection reconstructs an image by taking each view and *smearing* it along the path it was originally acquired. The resulting image is a blurry version of the correct image.

technique, called **backprojection** or **simple backprojection.** Figure 25-16 shows that simple backprojection is a common sense approach, but very unsophisticated. An individual sample is backprojected by setting all the image pixels along the ray pointing to the sample to the same value. In less technical terms, a backprojection is formed by *smearing* each view back through the image in the direction it was originally acquired. The final backprojected image is then taken as the sum of all the backprojected views.

While backprojection is conceptually simple, it does not correctly solve the problem. As shown in (b), a backprojected image is very *blurry*. A single point in the *true* image is reconstructed as a circular region that decreases in intensity away from the center. In more formal terms, the point spread function of backprojection is circularly symmetric, and decreases as the reciprocal of its radius.

Filtered backprojection is a technique to correct the blurring encountered in simple backprojection. As illustrated in Fig. 25-17, each view is *filtered* before the backprojection to counteract the blurring PSF. That is, each of the one-dimensional views is convolved with a one-dimensional filter kernel to create a set of *filtered views*. These filtered views are then backprojected to provide the reconstructed image, a close approximation to the "correct" image. In fact, the image produced by filtered backprojection is *identical*

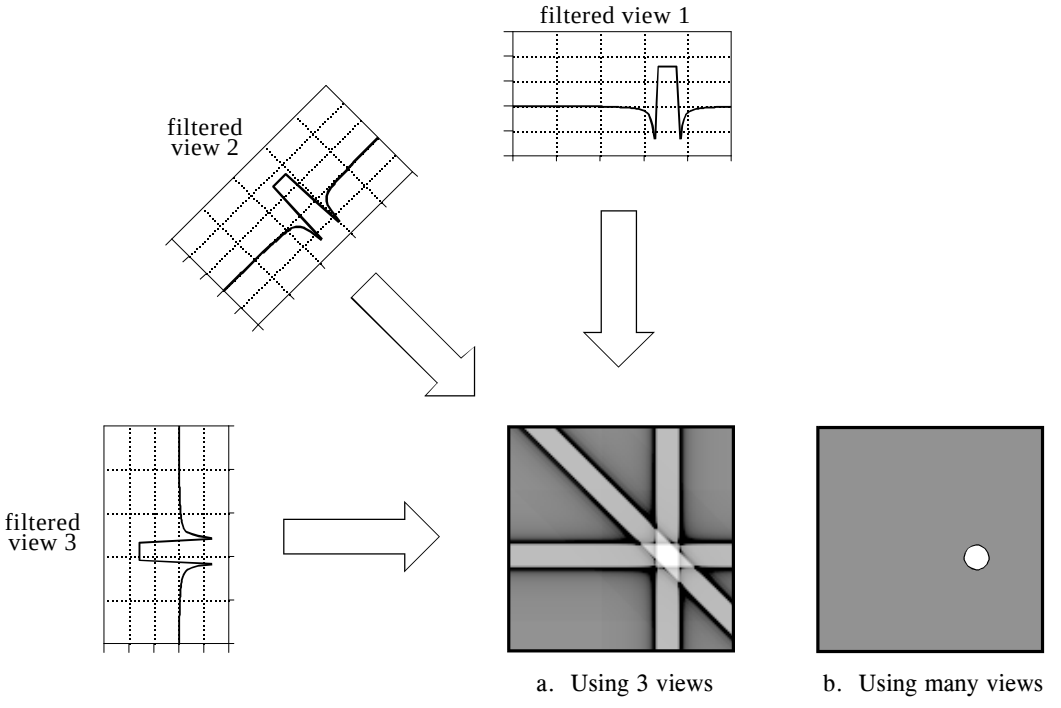


FIGURE 25-17
Filtered backprojection. Filtered backprojection reconstructs an image by filtering each view before backprojection. This removes the blurring seen in simple backprojection, and results in a mathematically exact reconstruction of the image. Filtered backprojection is the most commonly used algorithm for computed tomography systems.

to the "correct" image when there are an *infinite* number of views and an *infinite* number of points per view.

The filter kernel used in this technique will be discussed shortly. For now, notice how the profiles have been changed by the filter. The image in this example is a uniform white circle surrounded by a black background (a pillbox). Each of the acquired views has a flat background with a rounded region representing the white circle. Filtering changes the views in two significant ways. First, the top of the pulse is made flat, resulting in the final backprojection creating a *uniform* signal level within the circle. Second, negative spikes have been introduced at the sides of the pulse. When backprojected, these negative regions counteract the blur.

The fourth method is called **Fourier reconstruction**. In the spatial domain, CT reconstruction involves the relationship between a two-dimensional image and its set of one-dimensional views. By taking the two-dimensional Fourier transform of the image and the one-dimensional Fourier transform of each of its views, the problem can be examined in the frequency domain. As it turns out, the relationship between an image and its views is far simpler in the frequency domain than in the spatial domain. The frequency domain analysis

of this problem is a milestone in CT technology called the **Fourier slice theorem**.

Figure 25-18 shows how the problem looks in both the spatial and the frequency domains. In the spatial domain, each view is found by integrating the image along rays at a particular angle. In the frequency domain, the image spectrum is represented in this illustration by a two-dimensional grid. The spectrum of each view (a one-dimensional signal) is represented by a dark line superimposed on the grid. As shown by the positioning of the lines on the grid, the Fourier slice theorem states that the spectrum of a view is identical to the values along a line (slice) through the image spectrum. For instance, the spectrum of view 1 is the same as the center column of the image spectrum, and the spectrum of view 3 is the same as the center row of the image spectrum. Notice that the spectrum of each view is positioned on the grid at the same angle that the view was originally acquired. All these frequency spectra include the negative frequencies and are displayed with zero frequency at the center.

Fourier reconstruction of a CT image requires three steps. First, the one-dimensional FFT is taken of each view. Second, these view spectra are used to calculate the two-dimensional frequency spectrum of the image, as outlined by the Fourier slice theorem. Since the view spectra are arranged *radially*, and the correct image spectrum is arranged *rectangularly*, an interpolation routine is needed to make the conversion. Third, the inverse FFT is taken of the image spectrum to obtain the reconstructed image.

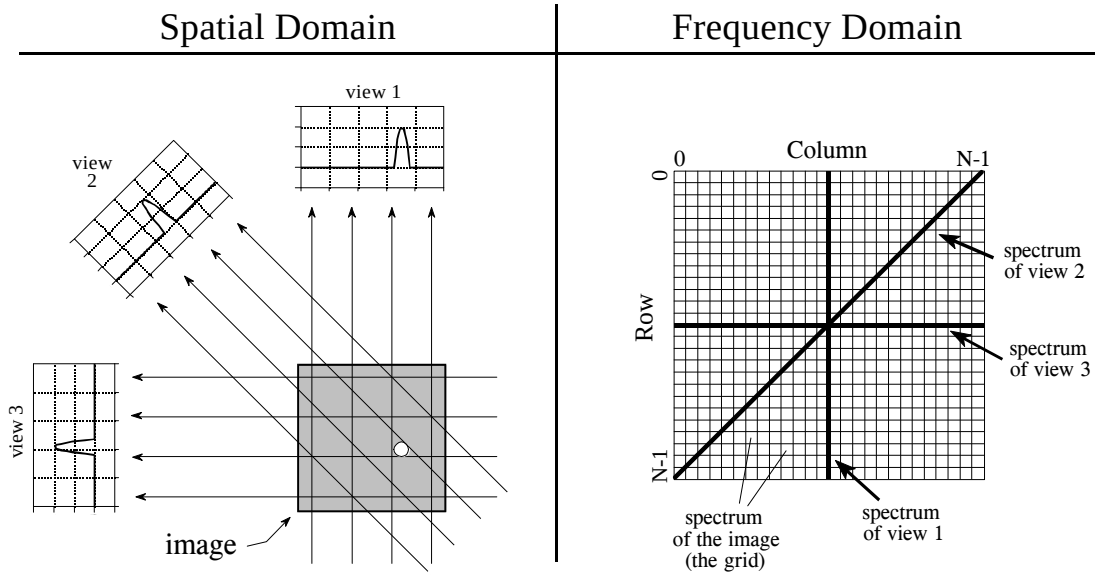


FIGURE 25-18
The Fourier Slice Theorem. The Fourier Slice Theorem describes the relationship between an image and its views in the frequency domain. In the spatial domain, each view is found by integrating the image along rays at a particular angle. In the frequency domain, the spectrum of each view is a one-dimensional "slice" of the two-dimensional image spectrum.

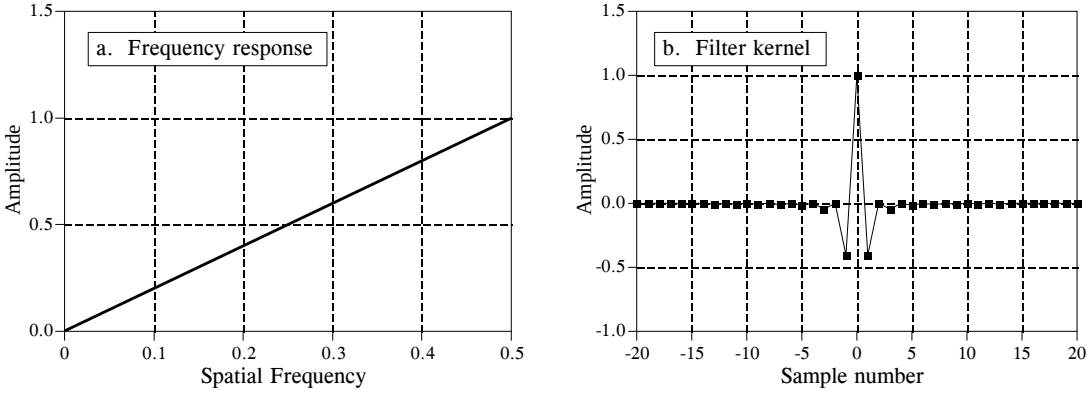


FIGURE 25-19
Backprojection filter. The frequency response of the backprojection filter is shown in (a), and the corresponding filter kernel is shown in (b). Equation 25-2 provides the values for the filter kernel.

This "radial to rectangular" conversion is also the key for understanding filtered backprojection. The radial arrangement is the spectrum of the backprojected image, while the rectangular grid is the spectrum of the correct image. If we compare one small region of the radial spectrum with the corresponding region of the rectangular grid, we find that the sample values are identical. However, they have a different *sample density*. The correct spectrum has uniformly spaced points throughout, as shown by the even spacing of the rectangular grid. In comparison, the backprojected spectrum has a higher sample density near the center because of its radial arrangement. In other words, the spokes of a wheel are closer together near the hub. This issue does not affect Fourier reconstruction because the interpolation is from the *values* of the nearest neighbors, not their *density*.

The filter in filtered backprojection cancels this unequal sample density. In particular, the frequency response of the filter must be the *inverse* of the sample density. Since the backprojected spectrum has a density of $1/f$, the appropriate filter has a frequency response of $H[f] = f$. This frequency response is shown in Fig. 25-19a. The filter kernel is then found by taking the inverse Fourier transform, as shown in (b). Mathematically, the filter kernel is given by:

EQUATION 25-2
The filter kernel for filtered backprojection. Figure 25-19b shows a graph of this kernel.

$$h[0] = 1$$

$$h[k] = 0 \qquad \text{for even values of } k$$

$$h[k] = \frac{4/\pi^2}{k^2} \qquad \text{for odd values of } k$$

Before leaving the topic of computed tomography, it should be mentioned that there are several similar imaging techniques in the medical field. All use extensive amounts of DSP. **Positron emission tomography (PET)** involves injecting the patient with a mildly radioactive compound that emits *positrons*. Immediately after emission, the positron annihilates with an electron, creating two gamma rays that exit the body in exactly opposite directions. Radiation detectors placed around the patient look for these back-to-back gamma rays, identifying the location of the *line* that the gamma rays traveled along. Since the point where the gamma rays were created must be somewhere along this line, a reconstruction algorithm similar to computed tomography can be used. This results in an image that looks similar to CT, except that *brightness* is related to the amount of the radioactive material present at each location. A unique advantage of PET is that the radioactive compounds can be attached to various substances used by the body in some manner, such as glucose. The reconstructed image is then related to the concentration of this biological substance. This allows the imaging of the body's *physiology* rather than simple *anatomy*. For example, images can be produced showing which portions of the human brain are involved in various mental tasks.

A more direct competitor to computed tomography is **magnetic resonance imaging (MRI)**, which is now found in most major hospitals. This technique was originally developed under the name **nuclear magnetic resonance (NMR)**. The name change was for public relations when local governments protested the use of anything *nuclear* in their communities. It was often an impossible task to educate the public that the term *nuclear* simply referred to the fact that all atoms contain a *nucleus*. An MRI scan is conducted by placing the patient in the center of a powerful magnet. Radio waves in conjunction with the magnetic field cause selected nuclei in the body to resonate, resulting in the emission of secondary radio waves. These secondary radio waves are digitized and form the data set used in the MRI reconstruction algorithms. The result is a set of images that appear very similar to computed tomography. The advantages of MRI are numerous: good soft tissue discrimination, flexible slice selection, and not using potentially dangerous x-ray radiation. On the negative side, MRI is a more expensive technique than CT, and poor for imaging bones and other hard tissues. CT and MRI will be the mainstays of medical imaging for many years to come.