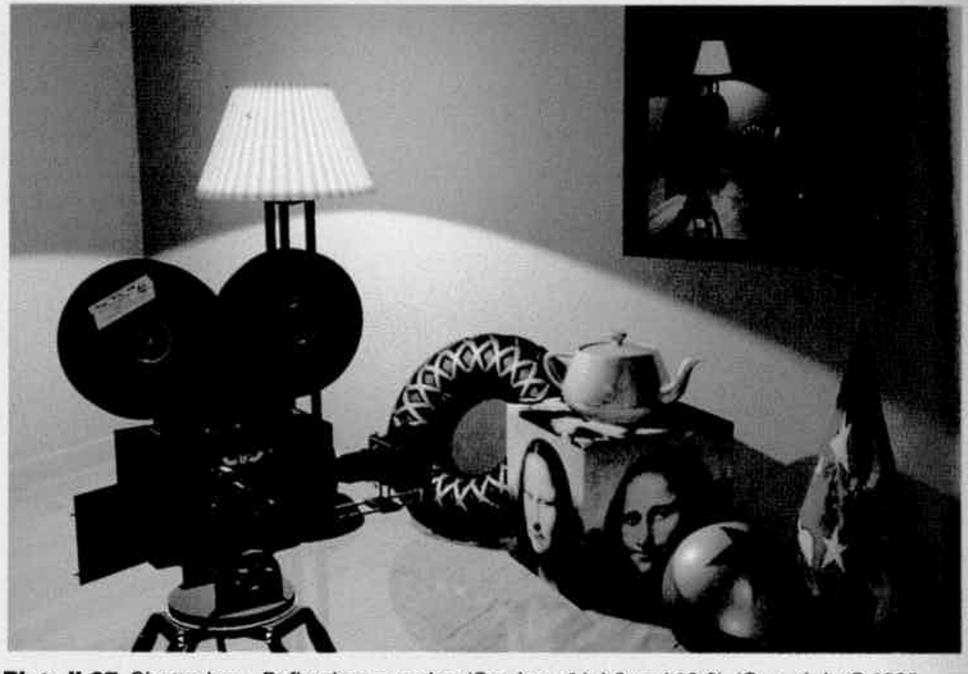


Plate II.37 Shutterbug. Reflection mapping (Sections 14.4.9 and 16.6). (Copyright © 1990, Pixar. Rendered by Thomas Williams and H.B. Siegel using Pixar's PhotoRealistic RenderMan™ software.)

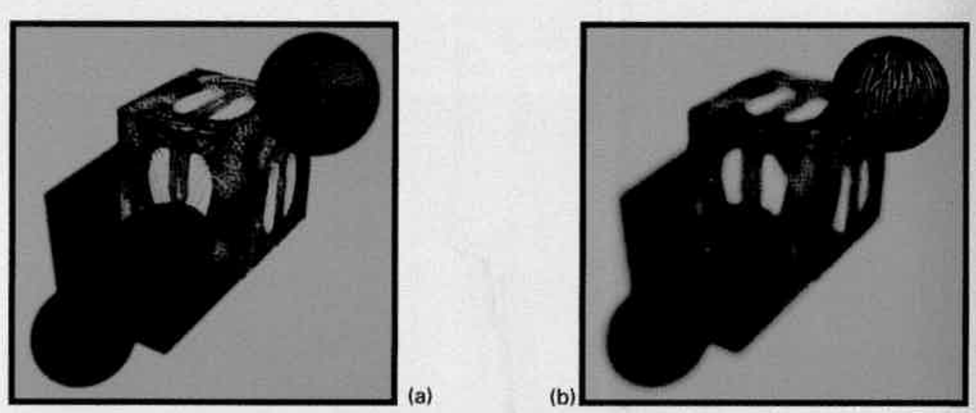


Plate II.38 Depth of field, implemented by postprocessing (Sections 14.4.10 and 16.10). (a) Focused at cube (550 mm), f/11 aperture. (b) Focused at sphere (290 mm), f/11 aperture. (Courtesy of Michael Potmesil and Indranil Chakravarty, RPI.)

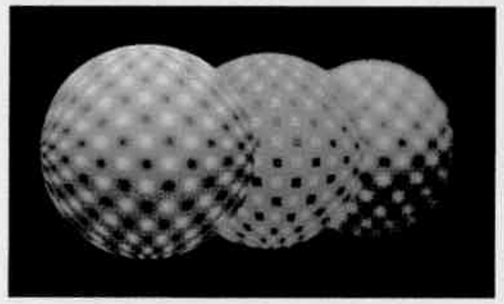


Plate II.39 Depth of field, implemented by distributed ray tracing (Sections 14.4.10 and 16.12.4). (By Robert Cook, Thomas Porter, and Loren Carpenter. Copyright © Pixar 1984. All rights reserved.)

just as in real life, so the viewer's eyes focus differently on different objects, depending on each object's proximity. Methods for producing and viewing stereo images are examined in more detail in Section 18.11.5; the mathematics of stereo projection is described in Exercise 6.27.

14.8 IMPROVED DISPLAYS

In addition to improvements in the software used to design and render objects, improvements in the displays themselves have heightened the illusion of reality. The history of computer graphics is in part that of a steady improvement in the visual quality achieved by display devices. Still, a modern monitor's color gamut and its dynamic intensity range are both a small subset of what we can see. We have a long way to go before the image on our display can equal the crispness and contrast of a well-printed professional photograph! Limited display resolution makes it impossible to reproduce extremely fine detail. Artifacts such as a visible phosphor pattern, glare from the screen, geometric distortion, and the stroboscopic effect of frame-rate flicker are ever-present reminders that we are viewing a display. The display's relatively small size, compared with our field of vision, also helps to remind us that the display is a window on a world, rather than a world itself.

14.9 INTERACTING WITH OUR OTHER SENSES

Perhaps the final step toward realism is the integration of realistic imagery with information presented to our other senses. Computer graphics has a long history of programs that rely on a variety of input devices to allow user interaction. Flight simulators are a current example of the coupling of graphics with realistic engine sounds and motion, all offered in a mocked-up cockpit to create an entire environment. The head-worn simulator of Color Plate I.16 monitors head motion, making possible another important 3D depth cue called *head-motion parallax*: when the user moves her head from side to side, perhaps to try to see more of a partially hidden object, the view changes as it would in real life. Other active work on head-mounted displays centers on the exploration of *virtual worlds*, such as the insides of molecules or of buildings that have not yet been constructed [CHUN89].

Many current arcade games feature a car or plane that the player rides, moving in time to a simulation that includes synthesized or digitized images, sound, and force feedback, as shown in Color Plate I.7. This use of additional output and input modalities points the way to systems of the future that will provide complete immersion of all the senses, including hearing, touch, taste, and smell.

14.10 ALIASING AND ANTIALIASING

In Section 3.17, we introduced the problem of aliasing and discussed some basic techniques for generating antialiased 2D primitives. Here we examine aliasing in more detail so that we can understand when and why it occurs, laying the groundwork for incorporating antialiasing into the visible-surface and shading algorithms covered in the following chapters. Additional material may be found in [CROW77b; CROW81]; an excellent set of examples is included in [BLIN89a; BLIN89b].

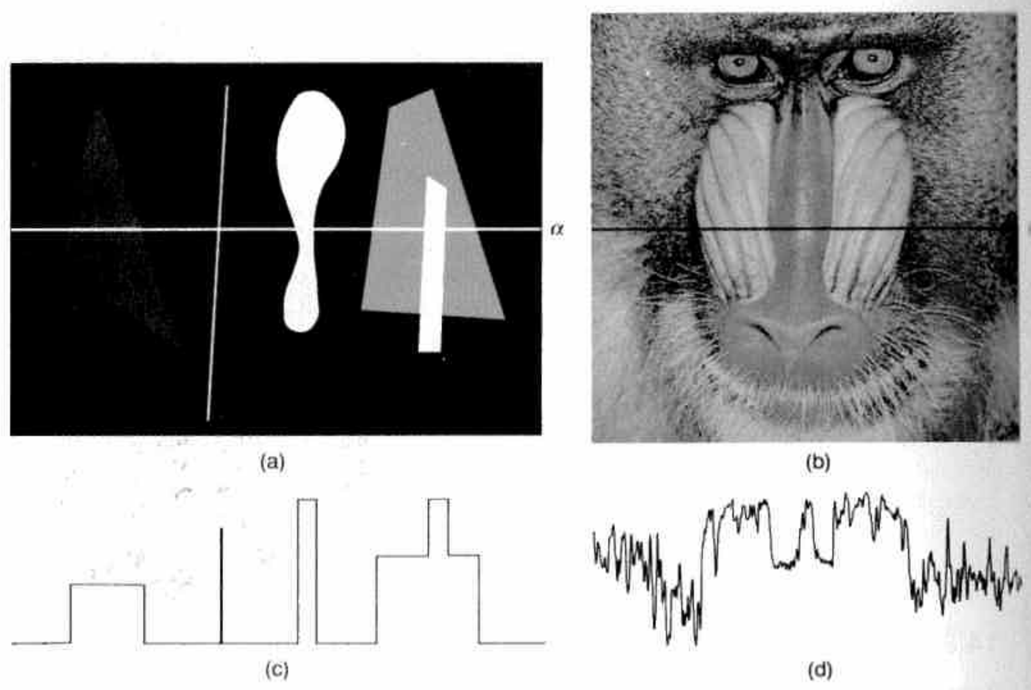


Fig. 14.8 Image. (a) Graphical primitives. (b) Mandrill. (c) Intensity plot of scan line α in (a). (d) Intensity plot of scan line α in (b). (Part d is courtesy of George Wolberg, Columbia University.)

To understand aliasing, we have to introduce some basic concepts from the field of signal processing. We start with the concept of a *signal*, which is a function that conveys information. Signals are often thought of as functions of time, but can equally well be functions of other variables. Since we can think of images as intensity variations over space, we will refer to signals in the *spatial domain* (as functions of spatial coordinates), rather than in the *temporal domain* (as functions of time). Although images are 2D functions of two independent spatial variables (x and y), for convenience our examples will often use the 1D case of a single spatial variable x. This case can be thought of as an infinitesimally thin slice through the image, representing intensity along a single horizontal line. Figure 14.8(a) and (b) show 2D signals, and parts (c) and (d) of the figure show plots of the intensity along the horizontal line α.

Signals can be classified by whether or not they have values at all points in the spatial domain. A *continuous signal*[1] is defined at a continuum of positions in space; a *discrete signal* is defined at a set of discrete points in space. Before scan conversion, the projection of our 3D objects onto the view plane may be treated as a continuous 2D signal whose value

[1]Not to be confused with the definition of continuity in calculus.

at each infinitesimal point in the plane indicates the intensity at that point. In contrast, the array of pixel values in the graphics system's frame buffer is a discrete 2D signal whose value is defined only at the positions in the array. Our rendering algorithms must determine the intensities of the finite number of pixels in the array so that they best represent the continuous 2D signal defined by the projection. The precise meaning of "best represent" is not at all obvious, however. We shall discuss this problem further.

A continuous signal may contain arbitrarily fine detail in the form of very rapid (high-frequency) variations in its value as its continuous parameter is changed. Since a discrete signal can change value only at discrete points, it clearly has a maximum rate of variation. Therefore, it should be clear that converting a continuous signal to a finite array of values may result in a loss of information. Our goal is to ensure that as little information as possible is lost, so that the resulting pixel array can be used to display a picture that looks as much as possible like the original signal would look if we were able to display it directly. The process of selecting a finite set of values from a signal is known as *sampling*, and the selected values are called *samples*. Once we have selected these samples, we must then display them using a process, known as *reconstruction*, that attempts to recreate the original continuous signal from the samples. The array of pixels in the frame buffer is reconstructed by the graphics system's display hardware, which converts these discrete intensity values to continuous, analog voltages that are applied to the CRT's electron gun (see Chapter 4). An idealized version of this pipeline is shown in Fig. 14.9. Signal-processing theory [GONZ87] establishes the minimum frequency at which samples must be selected from a given signal to reconstruct an exact copy of the signal, and specifies how to perform the reconstruction process. As we show later, however, this minimum sampling frequency will be infinite for many kinds of signals in which we are interested, so perfect reconstruction will often be impossible. Furthermore, as described in Section 14.10.5, the reconstruction method typically used by the display hardware differs from the approach prescribed by theory. Therefore, even properly sampled signals will not be reconstructed perfectly.

14.10.1 Point Sampling

The most straightforward way to select each pixel's value is known as *point sampling*. In point sampling, we select one point for each pixel, evaluate the original signal at this point, and assign its value to the pixel. The points that we select are typically arranged in a regular grid, as shown in Fig. 14.10. Unlike the scan-conversion algorithms of Chapter 3, projected vertices are not constrained to lie on integer grid points. Because the signal's values at a finite set of points are sampled, however, important features of the signal may be missed. For example, objects A and C in Fig. 14.10 are represented by the samples, whereas objects B and D are not. To make matters worse, if the viewing specification changes slightly or if the objects move, objects may pop in or out of visibility. What if we sample at a higher rate? The more samples we collect from the signal, the more we know about it. For example, we can see easily that, by increasing sufficiently the number of samples taken horizontally and vertically in Fig. 14.10, we can make sure that no object is missed in that particular picture. This is a necessary, but *not* a sufficient condition for adequate sampling. Nevertheless, sampling at a higher rate, we can generate images with more pixels representing each portion of the picture. We can also generate an image with

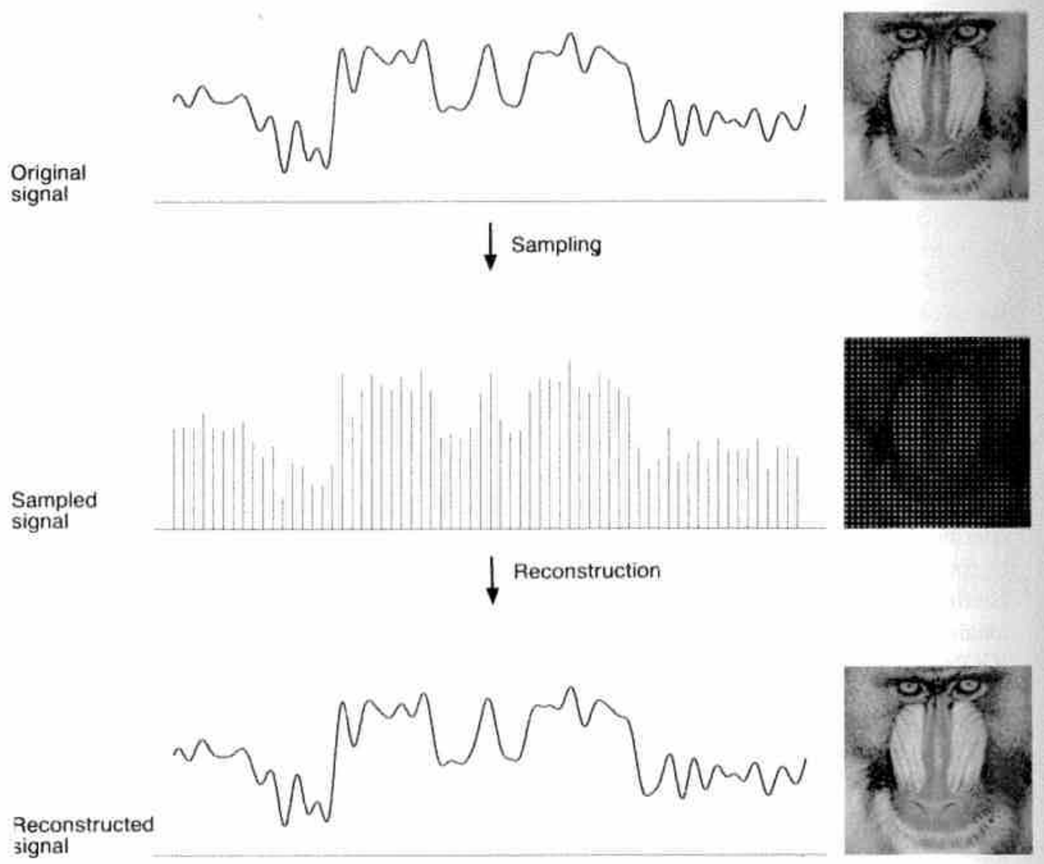


Fig. 14.9 The original signal is sampled, and the samples are used to reconstruct the signal. (Sampled 2D image is an approximation, since point samples have no area.) (Courtesy of George Wolberg, Columbia University.)

fewer pixels, by combining several adjacent samples (e.g., by averaging) to determine the value of each pixel of the smaller image. This means that all the features that would be present in the larger image at least contribute to the smaller one.

The approach of taking more than one sample for each pixel and combining them is known as *supersampling*. It actually corresponds to reconstructing the signal and resampling the reconstructed signal. For reasons described later, sampling the reconstructed signal is often better than sampling the original signal. This technique is popular in computer graphics precisely because it is so easy and often achieves good results, despite the obvious increase in computation. But, how many samples are enough? How do we know that there are no features that our samples are missing? Merely testing whether every object's projection is sampled is not sufficient. The projection may have a complex shape or variations in shading intensity that the samples do not reflect. We would like some way to guarantee that the samples we take are spaced close enough to reconstruct the original

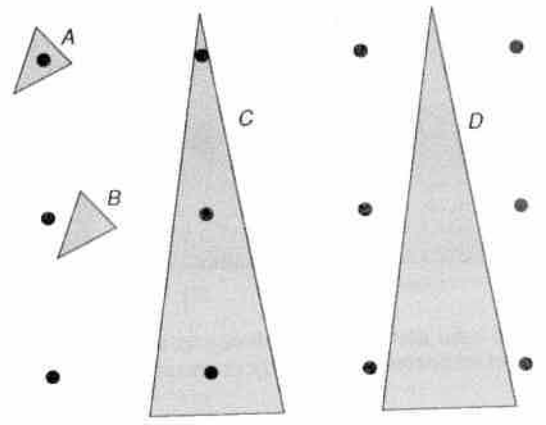


Fig. 14.10 Point-sampling problems. Samples are shown as black dots (●). Objects *A* and *C* are sampled, but corresponding objects *B* and *D* are not.

signal. As we shall see, sampling theory tells us that, on the basis of a particular signal's properties, we can compute a minimum sampling rate that will be adequate. Unfortunately, the rate turns out to be infinite for certain kinds of signals, including the signal shown in Fig. 14.10! We shall explain the reason for this in more detail later; for now, we can see that taking a finite number of samples cannot guarantee to capture the exact x coordinate at which the intensity jumps from one value to another in the figure. Furthermore, even if we find a finite sampling rate at which all of the current objects are sampled, we can always imagine adding just one more object positioned between samples that will be missed entirely.

14.10.2 Area Sampling

The problem of objects "falling between" samples and being missed suggests another approach: integrating the signal over a square centered about each grid point, dividing by the square's area, and using this average intensity as that of the pixel. This technique, called *unweighted area sampling*, was introduced in Chapter 3. The array of nonoverlapping squares is typically thought of as representing the pixels. Each object's projection, no matter how small, contributes to those pixels that contain it, in strict proportion to the amount of each pixel's area it covers, and without regard to the location of that area in the pixel, as shown by the equal weighting function of Fig. 14.11(a). No objects are missed, as may happen with point sampling. The definition of the definite integral requires evaluating a function at many points of an interval, and then taking the limit as the number of points increases. Thus, integrating amounts to a kind of infinite sampling process.

Unweighted area sampling has drawbacks caused by this evenhandedness with which objects are treated. Consider a small black object wholly contained inside of one of the pixels and surrounded by a white background, as in Fig. 14.11(b). This small object may move freely inside the pixel, and for each position the value computed for the pixel (shown as the pixel's shade) remains the same. As soon as the object crosses over into an adjoining pixel, however, the values of the original pixel and the adjoining pixel are both affected.

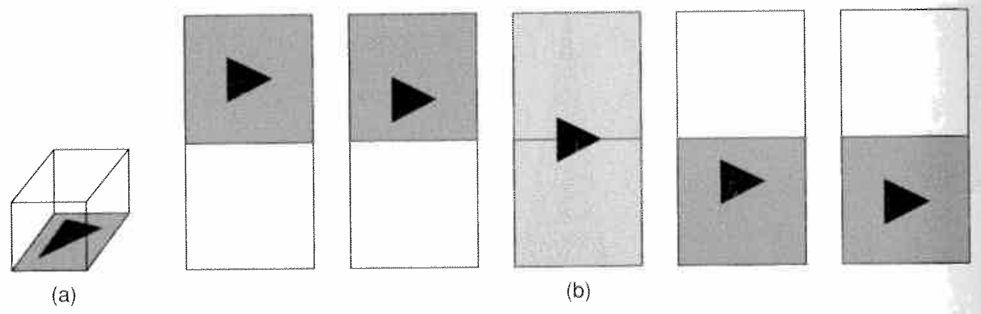


Fig. 14.11 Unweighted area sampling. (a) All points in the pixel are weighted equally. (b) Changes in computed intensities as an object moves between pixels.

Thus, the object causes the image to change only when it crosses pixel boundaries. As the object moves farther from the center of one pixel and closer to the center of another, however, we would like this change to be represented in the image. In other words, we would like the object's contribution to the pixel's intensity to be *weighted* by its distance from the pixel's center: the farther away it is, the less it should contribute.

In Chapter 3, we noted that *weighted area sampling* allows us to assign different weights to different parts of the pixel, and we suggested that the weighting functions of adjacent pixels should overlap. To see why the overlap is needed, we consider a weighting function consisting of an upright pyramid erected over a single pixel, as shown in Fig. 14.12(a). Under this weighting, as desired, an object contributes less to a pixel as it moves away from the pixel's center. But a drawback of unweighted area sampling still remains: An object contributes to only the single pixel that contains it. Consider a subpixel-sized black object moving over a white background from the center of one pixel to the center of an adjacent pixel, as shown in Fig. 14.12(b). As the object moves away from the center of the first pixel, its contribution to the first pixel decreases as it nears its edge. It begins to contribute to the pixel it enters only after it has crossed its border, and reaches its maximum contribution when it reaches the center of the new pixel. Thus, even though the black object has constant intensity, the first pixel increases in intensity before the second pixel decreases in intensity.

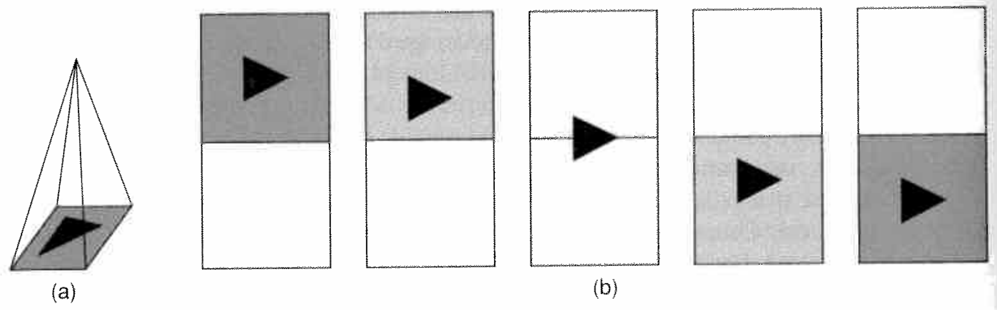


Fig. 14.12 Weighted area sampling. (a) Points in the pixel are weighted differently. (b) Changes in computed intensities as an object moves between pixels.

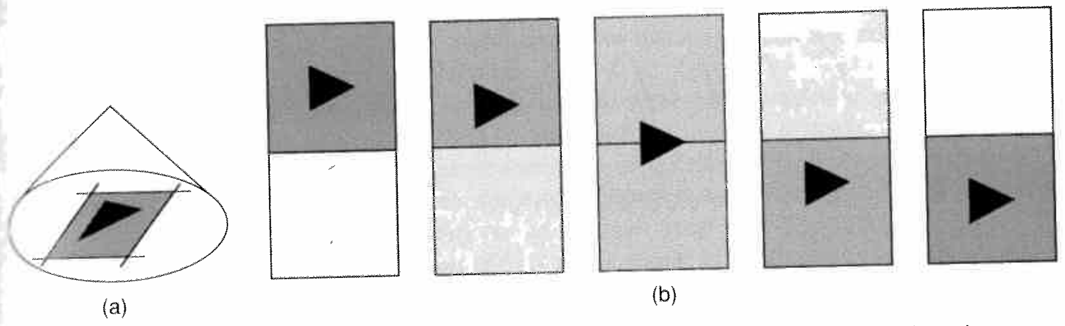


Fig. 14.13 Weighted area sampling with overlap. (a) Typical weighting function. (b) Changes in computed intensities as an object moves between pixels.

The net effect is that the display changes in intensity depending on the object's position, a change that gives rise to flickering as the object moves across the screen. It is clear that, to correct this problem, we must allow our weighting functions to overlap, so that a point on an object can simultaneously influence more than one pixel, as shown in Fig. 14.13. This figure also uses a radially symmetric weighting function. Here, it is appropriate to turn to sampling theory to discover the underlying reasons for increasing the weighting function's size, and to find out exactly what we need to do to sample and reconstruct a signal.

14.10.3 Sampling Theory

Sampling theory provides an elegant mathematical framework to describe the relationship between a continuous signal and its samples. So far, we have considered signals in the *spatial domain*; that is, we have represented each of them as a plot of amplitude against spatial position. A signal may also be considered in the *frequency domain*; that is, we may represent it as a sum of sine waves, possibly offset from each other (the offset is called *phase shift*), and having different frequencies and amplitudes. Each sine wave represents a component of the signal's *frequency spectrum*. We sum these components in the spatial domain by summing their values at each point in space.

Periodic signals, such as those shown in Fig. 14.14, can each be represented as the sum of phase-shifted sine waves whose frequencies are integral multiples (*harmonics*) of the signal's *fundamental* frequency. But what of nonperiodic signals such as images? Since an image is of finite size, we can define its signal to have a value of zero outside the area of the image. Such a signal, which is nonzero over a finite domain, and, more generally, any signal $f(x)$ that tapers off sufficiently fast (faster than $1/x$ for large values of x) can also be represented as a sum of phase-shifted sine waves. Its frequency spectrum, however, will not consist of integer multiples of some fundamental frequency, but may contain any frequency at all. The original signal cannot be represented as a sum of countably many sine waves, but instead must be represented by an integral over a continuum of frequencies. It is often the case, however, that an image (perhaps padded with surrounding zeros) is treated as one cycle of a periodic signal. This was done in Fig. 14.14(b), which shows the first ten components of Fig. 14.8(d). Each signal in the spatial domain has one representation in the frequency domain, and vice versa. As we shall see later, using two representations for a

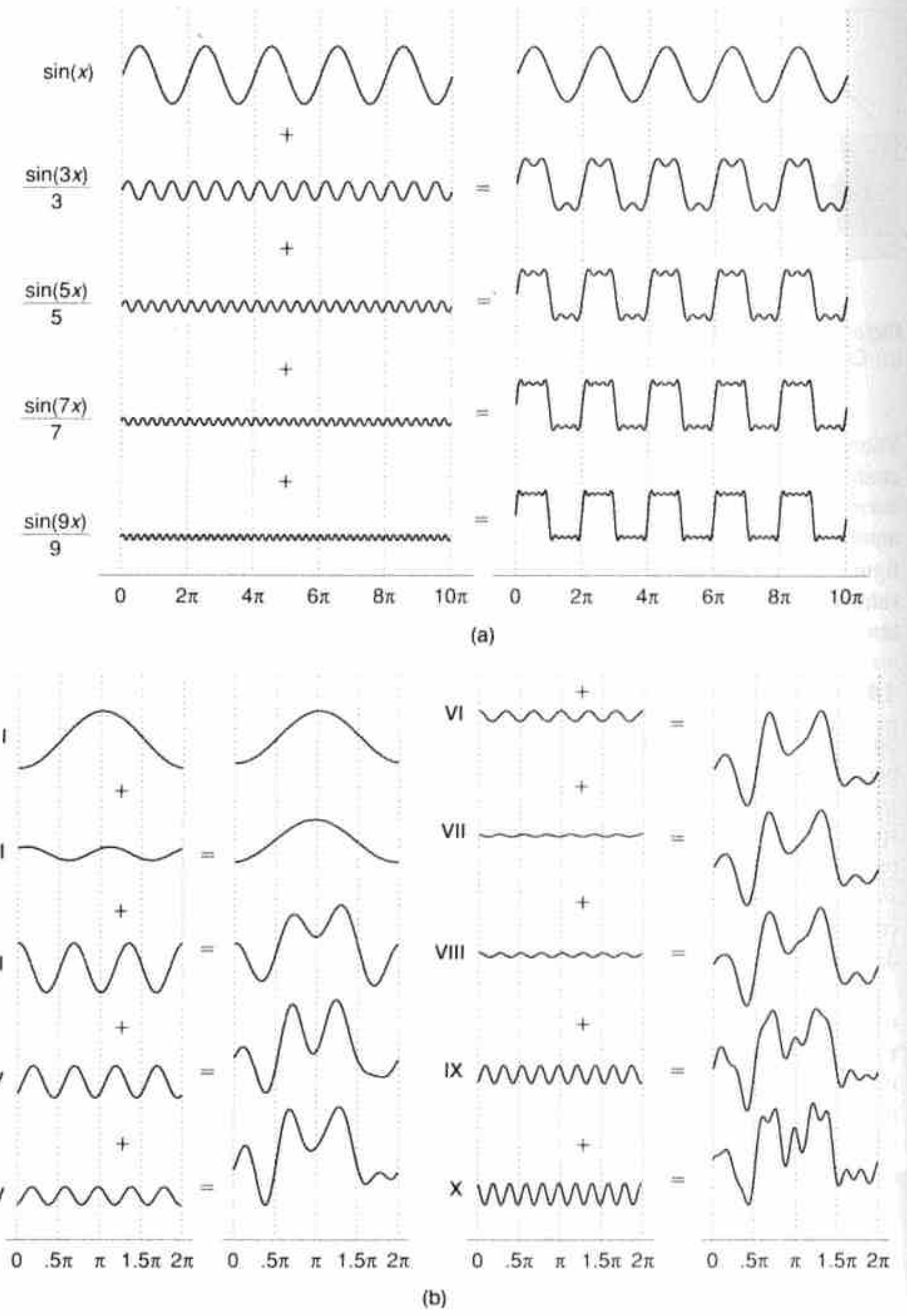


Fig. 14.14 A signal in the spatial domain is the sum of phase-shifted sines. Each component is shown with its effect on the signal shown at its right. (a) Approximation of a square wave. (b) Approximation of Fig. 14.8(d). (Courtesy of George Wolberg, Columbia University.)

signal is advantageous, because some useful operations that are difficult to carry out in one domain are relatively easy to do in the other.

Determining which sine waves must be used to represent a particular signal is the central topic of *Fourier analysis* [GONZ87]. Starting from an original signal, $f(x)$, we can generate a different function, the *Fourier transform* of f, called $F(u)$, whose argument u represents frequency. The value $F(u)$, for each frequency u, tells how much (i.e., the amplitude) of the frequency u appears in the original signal $f(x)$. The function $F(u)$ is therefore also called the *representation of f* (or of *the signal*) *in the frequency domain; $f(x)$* itself is called the representation of the signal in the spatial domain. The Fourier transform of a continuous, integrable signal $f(x)$ from the spatial domain to the frequency domain is defined by

$$F(u) = \int_{-\infty}^{+\infty} f(x)[\cos 2\pi ux - i\sin 2\pi ux]dx, \tag{14.1}$$

where $i = \sqrt{-1}$ and u represents the frequency of a sine and cosine pair. (Note that this applies only to functions that taper off sufficiently fast.) Recall that the cosine is just the sine, phase shifted by $\pi/2$. Together they can be used to determine the amplitude and phase shift of their frequency's component. For each u, the value of $F(u)$ is therefore a complex number. This is a clever way of encoding the phase shift and amplitude of the frequency u component of the signal: The value $F(u)$ may be written as $R(u) + iI(u)$, where $R(u)$ and $I(u)$ are the real and imaginary parts, respectively. The amplitude (or magnitude) of $F(u)$ is defined by

$$|F(u)| = \sqrt{R^2(u) + I^2(u)}, \tag{14.2}$$

and the phase shift (also known as the *phase angle*) is given by

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]. \tag{14.3}$$

In turn, an integrable signal $F(u)$ may be transformed from the frequency domain to the spatial domain by the *inverse Fourier transform*

$$f(x) = \int_{-\infty}^{+\infty} F(u)[\cos 2\pi ux + i\sin 2\pi ux]du. \tag{14.4}$$

The Fourier transform of a signal is often plotted as magnitude against frequency, ignoring phase angle. Figure 14.15 shows representations of several signals in both domains. In the spatial domain, we label the abscissa with numbered pixel centers; in the frequency domain, we label the abscissa with cycles per pixel (or more precisely, cycles per interval between pixel centers). In each case, the spike at $u = 0$ represents the DC (direct current) component of the spectrum. Substituting $\cos 0 = 1$ and $\sin 0 = 0$ in Eq. (14.1) reveals that this corresponds to integrating $f(x)$. If .5 were subtracted from each value of $f(x)$ in Fig. 14.15 (a) or (b), the magnitude of the signal's DC component would be 0.

Most of the figures in this chapter that show signals and their Fourier transforms were actually computed using discrete versions of Eqs. (14.1) and (14.4) that operate on signals represented by N regularly spaced samples. The *discrete Fourier transform* is

$$F(u) = \sum_{0 \le x \le N-1} f(x)[\cos(2\pi ux/N) - i\sin(2\pi ux/N)],\ 0 \le u \le N-1, \tag{14.5}$$

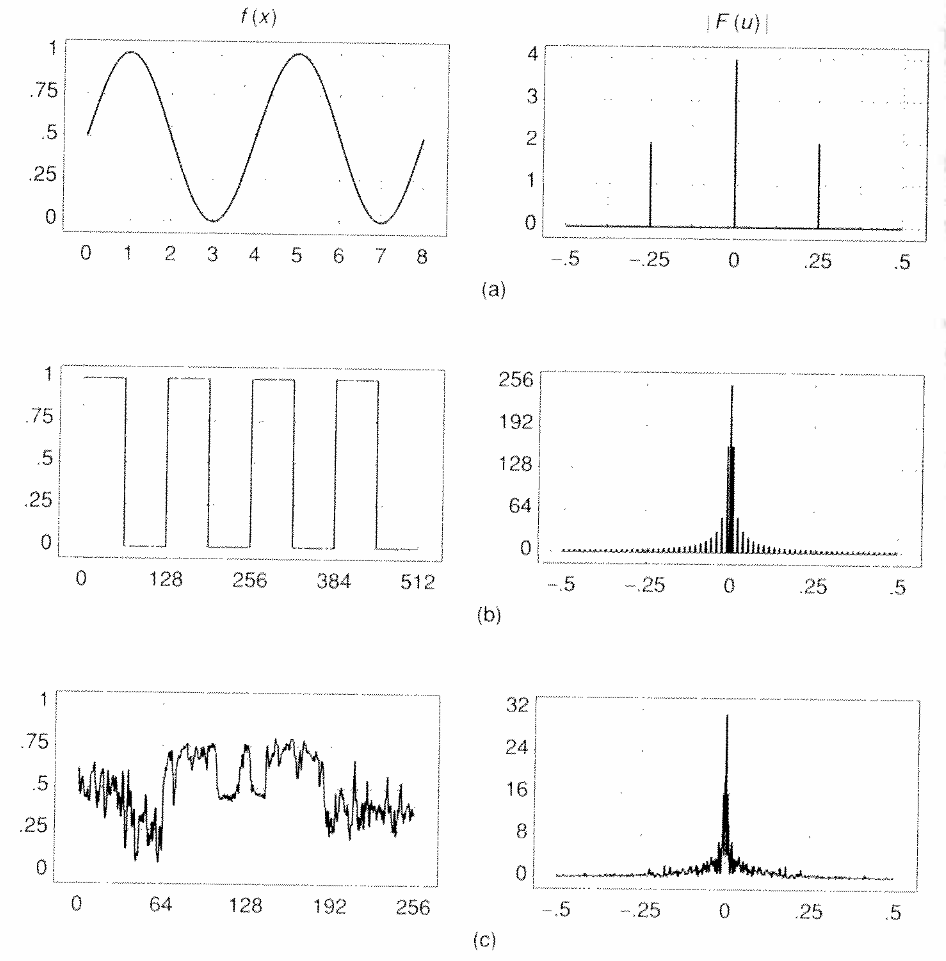


Fig. 14.15 Signals in the spatial and frequency domains. (a) Sine. (b) Square Wave. (c) Mandrill. The DC value in the frequency domain is truncated to make the other values legible and should be 129. (Courtesy of George Wolberg, Columbia Univeristy.)

and the *inverse discrete Fourier transform* is

$$f(x) = \frac{1}{N} \sum_{0 \le u \le N-1} F(u)[\cos(2\pi ux/N) + i\sin(2\pi ux/N)],\ 0 \le x \le N-1. \quad (14.6)$$

By choosing a sufficiently high sampling rate, a good approximation to the behavior of the continuous Fourier transform is obtained for most signals. (The discrete Fourier transform may also be computed more efficiently than Eqs. (14.5) and (14.6) would imply, by using a clever reformulation known as the *fast Fourier transform* [BRIG74].) The discrete Fourier transform always yields a finite spectrum. Note that, if a signal is symmetric about the origin, then $I(u) = 0$. This is true because the contribution of each sine term on one side of the origin is canceled by its equal and opposite contribution on the other side. In this case,

following [BLIN89a], we will plot the signed function $R(u)$, instead of the magnitude $|F(u)|$.

Sampling theory tells us that a signal can be properly reconstructed from its samples if the original signal is sampled at a frequency that is greater than twice f_h, the highest-frequency component in its spectrum. This lower bound on the sampling rate is known as the *Nyquist rate*. Although we do not give the formal proof of the adequacy of sampling above the Nyquist rate, we can provide an informal justification. Consider one cycle of a signal whose highest-frequency component is at frequency f_h. This component is a sine wave with f_h maxima and f_h minima, as shown in Fig. 14.16. Therefore, at least $2f_h$ samples are required to capture the overall shape of the signal's highest-frequency component. Note that exactly $2f_h$ samples is, in fact, a special case that succeeds only if the samples are taken precisely at the maxima and minima (Fig. 14.16a). If they are taken anywhere else, then the amplitude will not be represented correctly (Fig. 14.16b) and may even be determined to be zero if the samples are taken at the zero crossings (Fig. 14.16c). If we sample below the Nyquist rate, the samples we obtain may be identical to what would have been obtained from sampling a lower-frequency signal, as demonstrated in Fig. 14.17. This phenomenon of high frequencies masquerading as low frequencies in the reconstructed signal is known as *aliasing*: The high-frequency components appear as though they were actually lower-frequency components. Another example of aliasing is demonstrated in Fig. 14.18. Figure 14.18(a) shows an image and a plot of its intensity across a horizontal line, representing a set of intensity fluctuations that increase in spatial frequency from left to right. The image in Fig. 14.18(b) was created by selecting every 8th pixel from each line of Fig. 14.18(a) and replicating it eight times. It shows aliasing as the bands increase in spatial frequency.

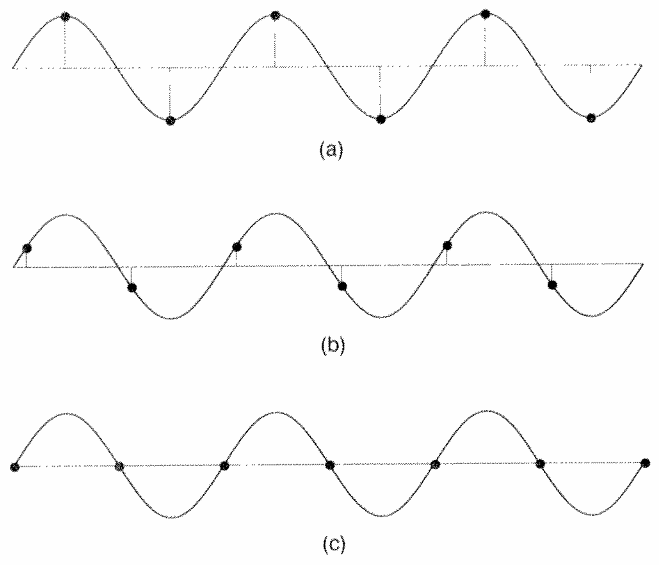


Fig. 14.16 Sampling at the Nyquist rate (a) at peaks, (b) between peaks, (c) at zero crossings. (Courtesy of George Wolberg, Columbia University.)

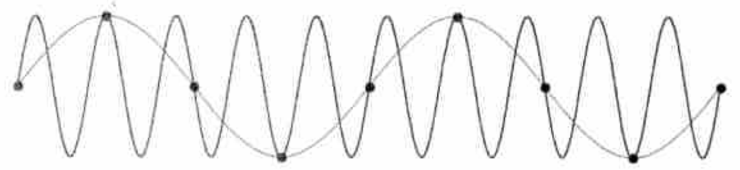


Fig. 14.17 Sampling below the Nyquist rate. (Courtesy of George Wolberg, Columbia University.)

A signal's shape is determined by its frequency spectrum. The sharper and more angular a waveform is, the richer it is in high-frequency components; signals with discontinuities have an infinite frequency spectrum. Figure 14.10 reveals the sharp edges of the objects' projections that our algorithms attempt to represent. This signal has an infinite frequency spectrum, since the image intensity changes discontinuously at object boundaries. Therefore, the signal cannot be represented properly with a finite number of samples. Computer graphics images thus exhibit two major kinds of aliasing. First, "jaggies" along edges are caused by discontinuities at the projected edges of objects: a point sample either does or does not lie in an object's projection. Even the presence of a single such edge in an environment's projection means that the projection has an infinite frequency spectrum. The frequency spectrum tapers off quite rapidly, however, like those of Fig. 14.15(b) and (c). Second, textures and objects seen in perspective may cause arbitrarily many discontinuities and fluctuations in the environment's projection, making it possible for objects whose projections are too small and too close together to be alternately missed and sampled, as in the right hand side of Fig. 14.18(b). The high frequency components representing the frequency at which these projections cross a scan line may have high amplitude (e.g., alternating black and white checkerboard squares). This often affects picture quality more seriously than jaggies do.

14.10.4 Filtering

There is a partial solution to the problems discussed in the previous section. If we could create a new signal by removing the offending high frequencies from the original signal,

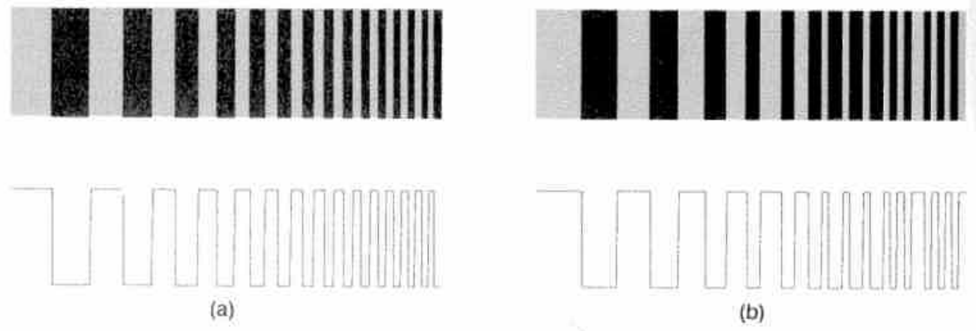


Fig. 14.18 Aliasing. (a) Image and intensity plot of a scan line. (b) Sampled image and intensity plot of a scan line. (Courtesy of George Wolberg, Columbia University.)

then the new signal could be reconstructed properly from a finite number of samples. The more high frequencies we remove, the lower the sampling frequency needed, but the less the signal resembles the original. This process is known as *bandwidth limiting* or *band limiting* the signal. It is also known as *low-pass filtering*, since filtering a signal changes its frequency spectrum; in this case, high frequencies are filtered out and only low frequencies are allowed to pass. Low-pass filtering causes blurring in the spatial domain, since fine visual detail is captured in the high frequencies that are attenuated by low-pass filtering, as shown in Fig. 14.19. We shall revise the pipeline of Fig. 14.9 to include an optional filter, as shown in Fig. 14.20.

A perfect low-pass filter completely suppresses all frequency components above some specified cut-off point, and lets those below the cut-off point pass untouched. We can easily do this filtering in the frequency domain by multiplying the signal's spectrum by a *pulse function*, as shown in Fig. 14.21. We can multiply two signals by taking their product at each point along the paired signals. The pulse function

$$S(u) = \begin{cases} 1, & \text{when } -k \le u \le k, \\ 0, & \text{elsewhere.} \end{cases} \qquad (14.7)$$

cuts off all components of frequency higher than k. Therefore, if we were to low-pass filter the signal so as to remove all variation, we would be left with only its DC value.

So far, it would seem that a recipe for low-pass filtering a signal in the spatial domain would involve transforming the signal into the frequency domain, multiplying it by an appropriate pulse function, and then transforming the product back into the spatial domain. Some important relationships between signals in the two domains, however, make this procedure unnecessary. It can be shown that multiplying two Fourier transforms in the frequency domain corresponds exactly to performing an operation called *convolution* on their inverse Fourier transforms in the spatial domain. The *convolution* of two signals $f(x)$

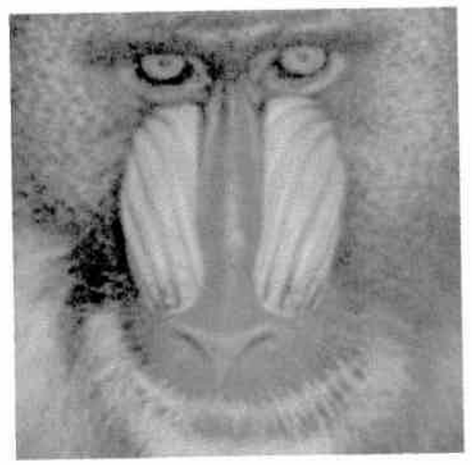


Fig. 14.19 Figure 14.8(b) after low-pass filtering. (Courtesy of George Wolberg, Columbia University.)

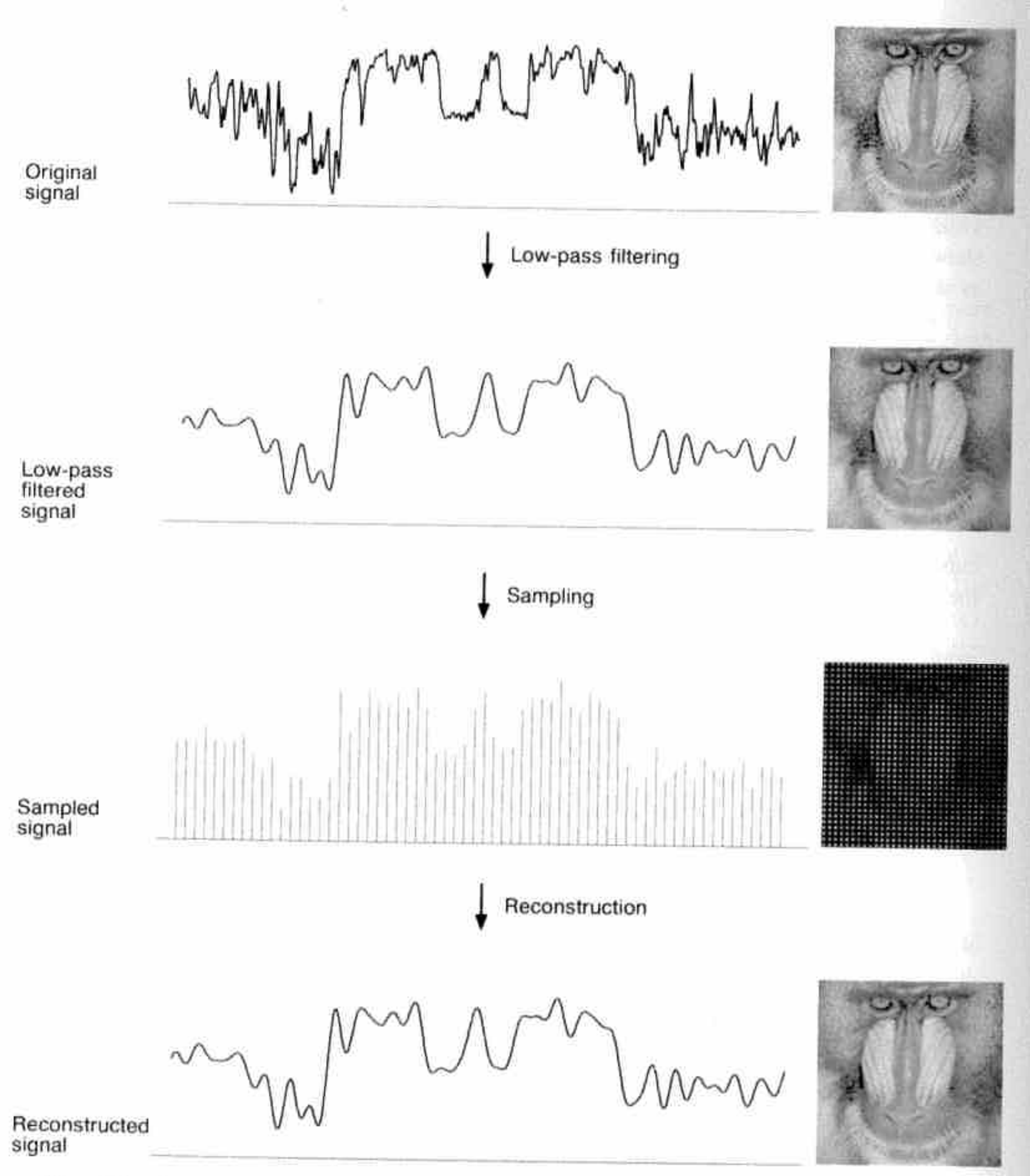


Fig. 14.20 The sampling pipeline with filtering. (Courtesy of George Wolberg, Columbia University.)

and $g(x)$, written as $f(x) * g(x)$, is a new signal $h(x)$ defined as follows. The value of $h(x)$ at each point is the integral of the product of $f(x)$ with the filter function $g(x)$ flipped about its vertical axis and shifted such that its origin is at that point. This corresponds to taking a weighted average of the neighborhood around each point of the signal $f(x)$—weighted by a flipped copy of filter $g(x)$ positioned at the point—and using it for the value of $h(x)$ at the

point. The size of the neighborhood is determined by the size of the domain over which the filter is nonzero. This is known as the filter's *support*, and a filter that is nonzero over a finite domain is said to have *finite support*. We use τ as a dummy variable of integration when defining the convolution. Thus,

$$h(x) = f(x) * g(x) \overset{\Delta}{=} \int_{-\infty}^{+\infty} f(\tau)g(x - \tau)d\tau. \tag{14.8}$$

Conversely, convolving two Fourier transforms in the frequency domain corresponds exactly to multiplying their inverse Fourier transforms in the spatial domain. The filter function is often called the *convolution kernel* or *filter kernel*.

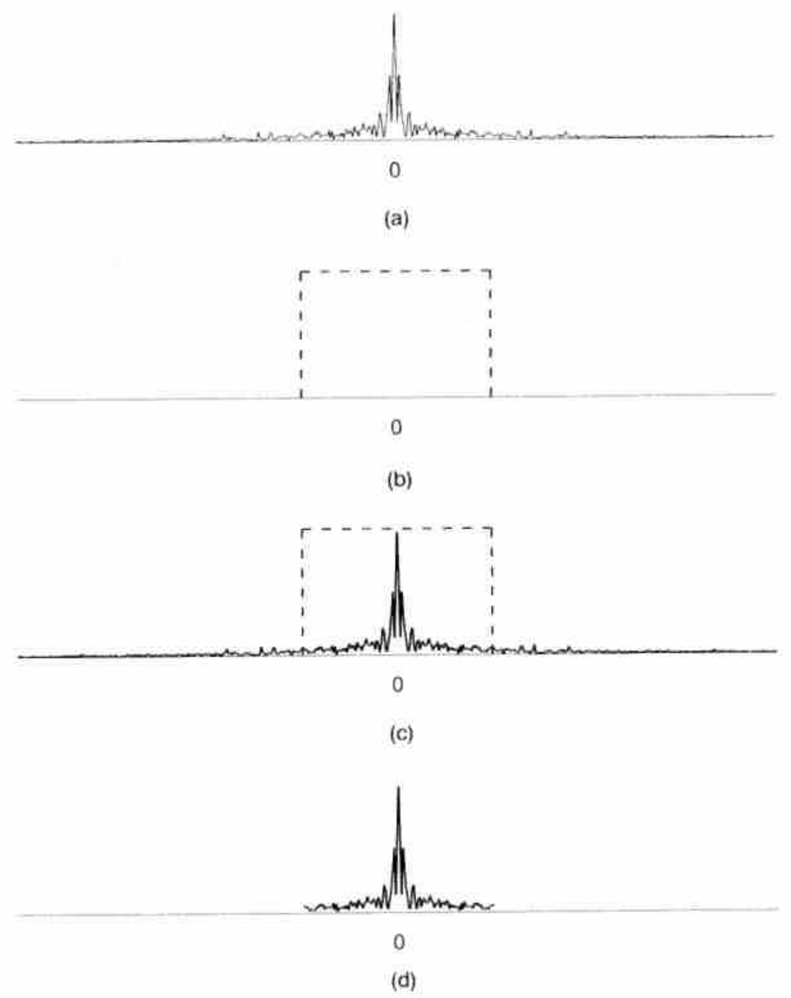


Fig. 14.21 Low-pass filtering in the frequency domain. (a) Original spectrum. (b) Low-pass filter. (c) Spectrum with filter. (d) Filtered spectrum. (Courtesy of George Wolberg, Columbia University.)

Convolution can be illustrated graphically. We will convolve the function $f(x) = 1, 0 \leq x \leq 1$, with the filter kernel $g(x) = c, 0 \leq x \leq 1$, shown in Figs. 14.22(a) and (b). By using functions of τ, we can vary x to move the filter relative to the signal being filtered. To create the function $g(x - \tau)$, we first flip $g(\tau)$ about the origin to yield $g(-\tau)$, and then offset it by x to form $g(x - \tau)$, as depicted in Figs. 14.22(c) and (d). The integral, with respect to τ, of the product $f(\tau)g(x - \tau)$, which is the area of the shaded portions of the figures, is 0 for $-\infty \leq x < 0$, xc for $0 \leq x \leq 1$ (Fig. 14.22e), $(2 - x)c$ for $1 \leq x \leq 2$ (Fig. 14.22f), and 0 for $2 < x \leq \infty$. The convolution $f(x) * g(x)$ is illustrated in Fig. 14.22(g). Note how convolution with this kernel smooths the discontinuities of $f(x)$ while it widens the area over which $f(x)$ is nonzero.

Multiplying by a pulse function in the frequency domain has the same effect as convolving with the signal that corresponds to the pulse in the spatial domain. This signal is known as the *sinc* function, which is defined as $\sin(\pi x)/\pi x$. Figure 14.23 shows the sinc function and an example of the result of convolving it with another signal. Convolving with a sinc function therefore low-pass filters the signal. How do we choose the height and width of the sinc used in Fig. 14.23(c)? As shown in Fig. 14.24, there is a relationship (that we do not prove) between the height and width of the perfect low-pass filter in the spatial and frequency domains. In Fig. 14.24(a), if W is the cutoff frequency and A is the amplitude, then it must be the case that $A/2W = 1$ for all frequencies up to the cutoff frequency to be passed unattenuated. Therefore, $A = 2W$. Both the amplitude and width of the sinc in Fig. 14.24(a) vary with W. When $W = .5$ cycles per pixel (the highest frequency that can be represented when sampling once per pixel), $A = 1$ and the sinc has zero crossings at pixel centers. As the cutoff frequency W is made lower or higher, the sinc becomes shorter and broader, or taller and narrower, respectively. This makes sense because we would like the integral of a filter in the spatial domain to be 1, a necessary restriction if the filter is to maintain the gray level (DC value) of the image, neither brightening nor dimming it. (We

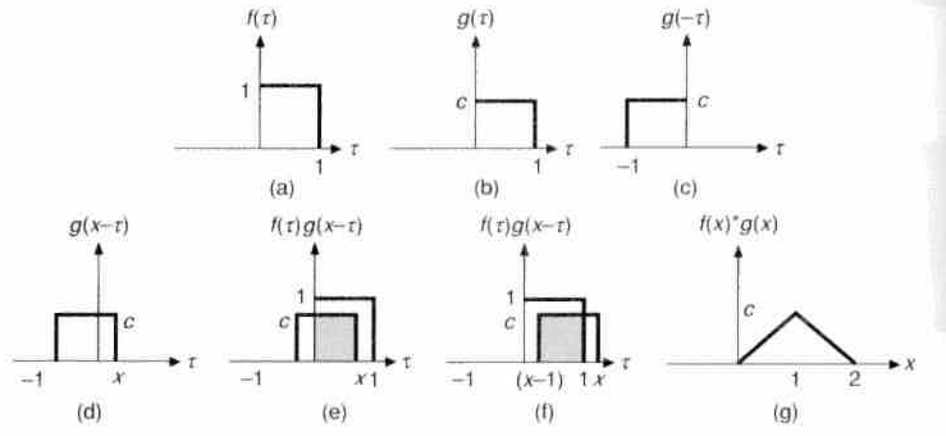


Fig. 14.22 Graphical convolution. (a) Function $f(\tau) = 1, 0 \leq \tau \leq 1$. (b) Filter kernel $g(\tau) = c, 0 \leq \tau \leq 1$. (c) $g(-\tau)$. (d) $g(x - \tau)$. (e) $\int_{-\infty}^{+\infty} f(\tau)g(x - \tau)d\tau = xc, 0 \leq x \leq 1$. (f) $\int_{-\infty}^{+\infty} f(\tau)g(x - \tau)d\tau = (2 - x)c, 1 \leq x \leq 2$. (g) $f(x) * g(x)$.

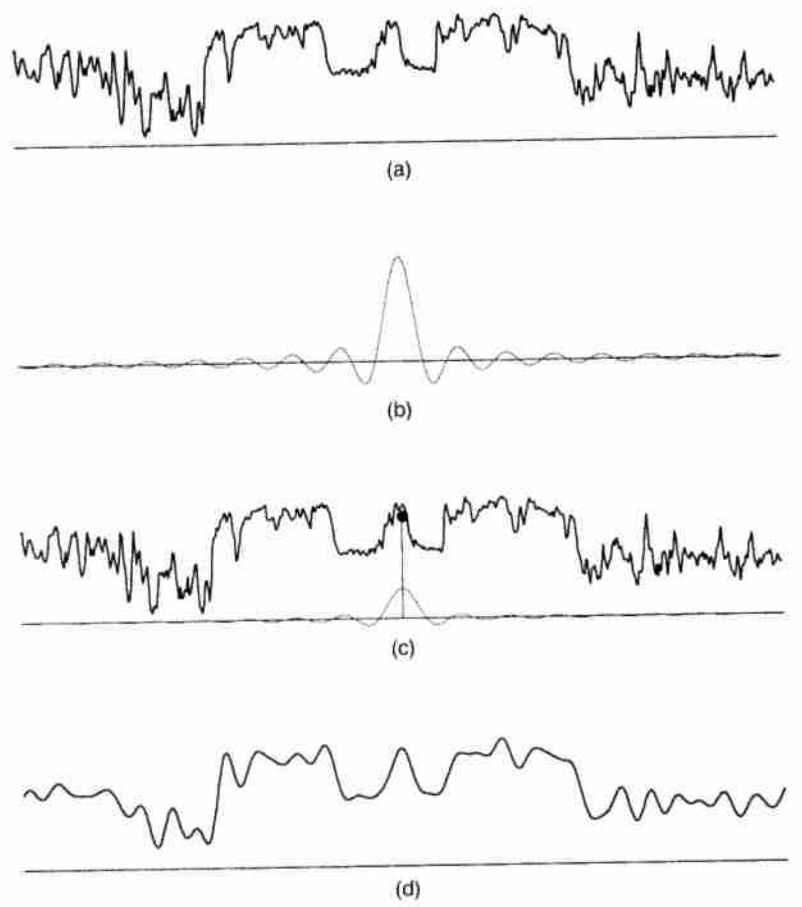


Fig. 14.23 Low-pass filtering in the spatial domain. (a) Original signal. (b) Sinc filter. (c) Signal with filter, with value of filtered signal shown as a black dot (●) at filter's origin. (d) Filtered signal. (Courtesy of George Wolberg, Columbia University.)

can see that this is true by considering the convolution of a filter with a signal that has the same value c at each point.)

The sinc function has the unfortunate property that it is nonzero on points arbitrarily far from the origin (i.e., it has *infinite support* since it is infinitely wide). If we truncate the sinc function, by multiplying it by a pulse function, we can restrict the support, as shown in Fig. 14.24(b). This is a special case of a *windowed* sinc function that has been restricted to a finite window. We might reason that we are throwing away only those parts of the filter where the value is very small anyhow, so it should not influence the result too much. Unfortunately, the truncated version of the filter has a Fourier transform that suffers from *ringing* (also called the *Gibbs phenomenon*): A truncated sinc in the spatial domain no longer corresponds to a pure pulse function in the frequency domain, but instead corresponds to a pulse function with ripples near the cutoff frequency, as shown in Fig. 14.24(b). This causes some frequency components to pass that should be suppressed, and both attenuates and amplifies others around the cutoff point; the domain over which this effect occurs decreases in size as a greater portion of the sinc signal is used, but the amplitude of the ringing does not decrease as long as the sinc is truncated. The approximation to a square wave in Fig. 14.14(a) exhibits ringing in the spatial domain, which appears as little intensity "ripples" at discontinuities. A truncated sinc is obtained by multiplying the sinc by a pulse function. An alternative is to use a windowed sinc function that has been multiplied by a shape that, unlike the pulse, is not discontinuous, which allows the sinc to fall off smoothly. Blinn [BLIN89b] describes the derivation of one such filter.

One final problem is that the sinc, along with windowed filters derived from it, has parts that dip below zero, known as *negative lobes*. When a signal is convolved with a filter that has negative lobes, the resulting signal may itself dip below zero. If the signal represents intensity values, these values correspond to unrealizable negative intensities, and must therefore ultimately be clamped to zero.

Although windowed sinc functions are useful, they are relatively expensive because the window must be fairly wide; thus, a variety of other functions is often used instead. Filters

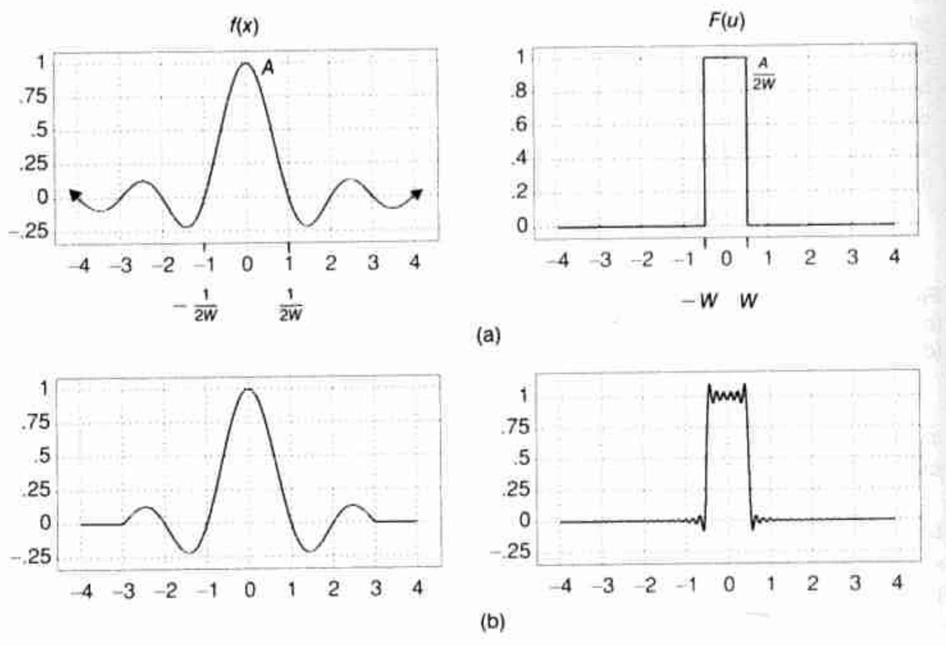


Fig. 14.24 (a) Sinc in spatial domain corresponds to pulse in frequency domain. (b) Truncated sinc in spatial domain corresponds to ringing pulse in frequency domain. (Courtesy of George Wolberg, Columbia University.)

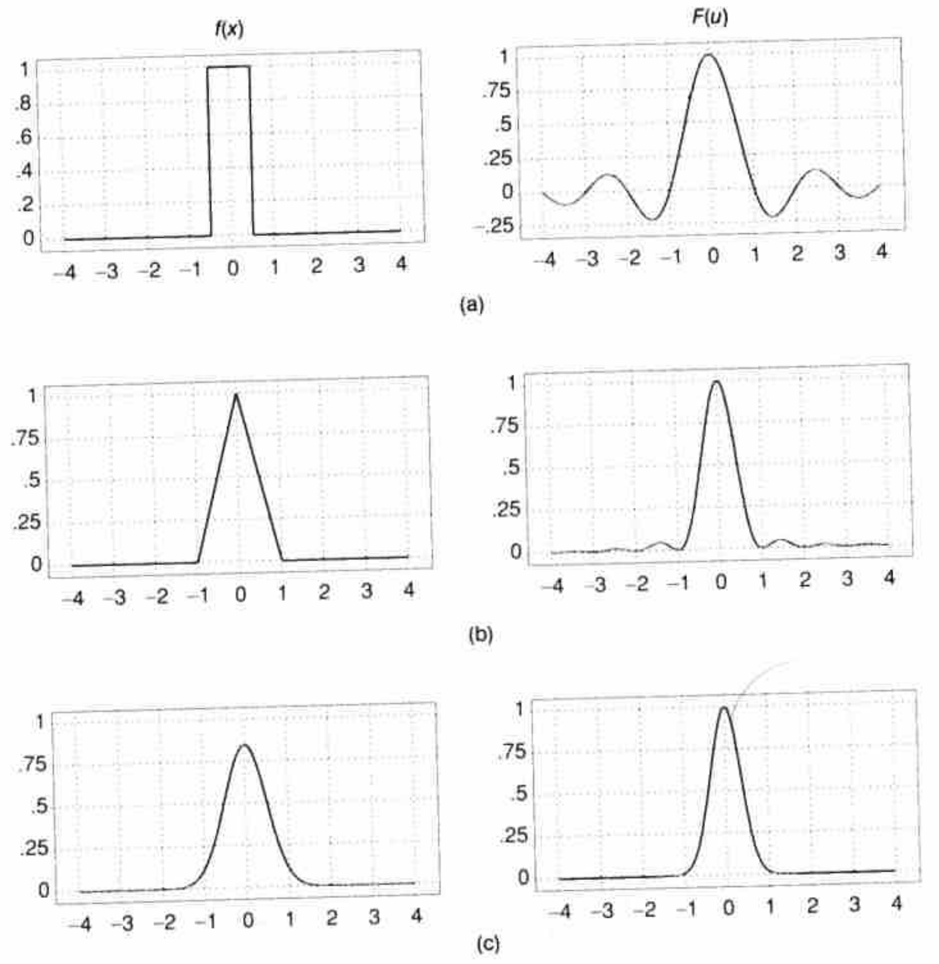


Fig. 14.25 Filters in spatial and frequency domains. (a) Pulse—sinc. (b) Triangle—sinc^2. (c) Gaussian—Gaussian. (Courtesy of George Wolberg, Columbia University.)

with finite support are known as finite impulse-response (FIR) filters, in contrast to the untruncated sinc filter, which is an infinite impulse-response (IIR) filter. Figure 14.25 shows several popular filters in both spatial and frequency domains.

We have now reduced the sampling problem to one of convolving the signal with a suitable filter and then sampling the filtered signal. Notice, however, that if our only use of the filtered signal is to sample it, then the work done filtering the signal anywhere but at the sample points is wasted. If we know in advance exactly where the samples will be taken, we need only to evaluate the convolution integral (Eq. 14.8) at each sample point to determine the sample's value. This is precisely how we perform the weighting operation in using area sampling to determine the intensity of each pixel. The weighting distribution constructed over each pixel's center is a filter. The pulse function with which we convolve the signal in performing unweighted area sampling is often called a *box filter*, because of its appearance. Just as the pulse function in the frequency domain corresponds to the sinc function in the spatial domain, the pulse function in the spatial domain (the box filter's 1D equivalent) corresponds to the sinc function in the frequency domain (Fig. 14.25a). This correspondence underscores how badly a box filter or pulse filter approximates a perfect low-pass filter. Multiplying with a sinc in the frequency domain not only fails to cut off sharply, but passes infinitely high frequencies. Furthermore, the pulse filter attenuates frequencies that are within the desired range, since its Fourier transform—the sinc function—begins to trail off before the ideal low-pass filter. Therefore, it also blurs the image excessively.

14.10.5 Reconstruction

At this point, let us assume that we have sampled the signal $f(x)$ at a frequency f_s to obtain the sampled signal, which we call $\hat{f}(x)$. Sampling theory shows that the frequency spectrum of $\hat{f}(x)$ looks like that of $f(x)$, replicated at multiples of f_s. To see that this relationship holds, we note that sampling a signal corresponds to multiplying it in the spatial domain by a *comb* function, so named because of its appearance, as shown in Fig. 14.26(a). The comb

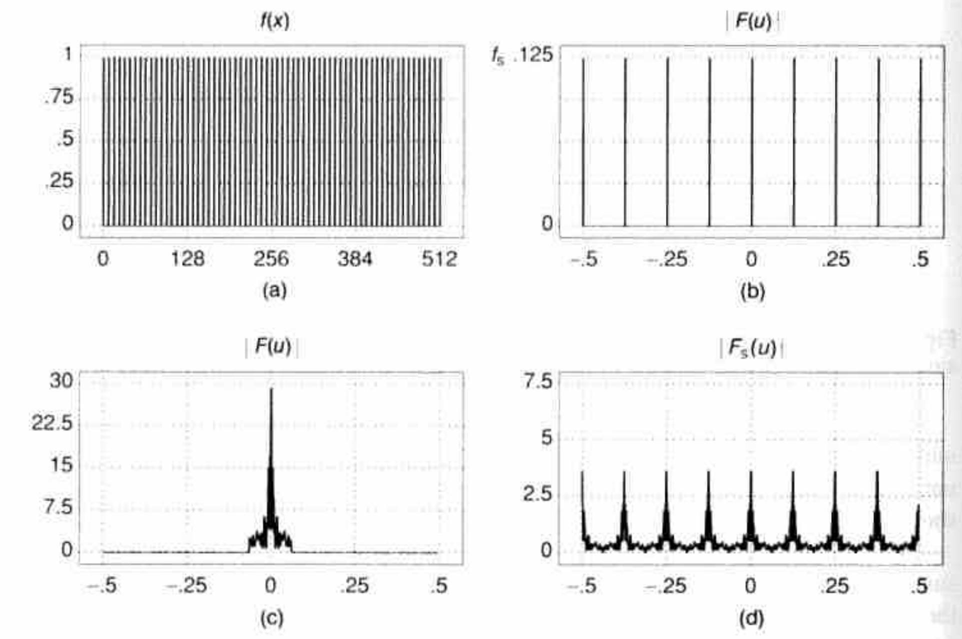


Fig. 14.26 (a) Comb function and (b) its Fourier transform. Convolving the comb's Fourier transform with (c) a signal's Fourier transform in the frequency domain yields (d) the replicated spectrum of the sampled signal. (Courtesy of George Wolberg, Columbia University.)

function has a value of 0 everywhere, except at regular intervals, corresponding to the sample points, where its value is 1. The (discrete) Fourier transform of a comb turns out to be just another comb with teeth at multiples of f_s (Fig. 14.26b). The height of the teeth in the comb's Fourier transform is f_s in cycles/pixel. Since multiplication in the spatial domain corresponds to convolution in the frequency domain, we obtain the Fourier transform of the sampled signal by convolving the Fourier transforms of the comb function and the original signal (Fig. 14.26c). By inspection, the result is the replicated spectrum (Fig. 14.26d). Try performing graphical convolution with the comb to verify this, but note that $F(u)$, not $|F(u)|$ must actually be used. A sufficiently high f_s yields spectra that are replicated far apart from each other. In the limiting case, as f_s approaches infinity, a single spectrum results.

Recall that *reconstruction* is recreation of the original signal from its samples. The result of sampling a signal (Fig. 14.27a) at a finite sampling frequency is a signal with an infinite frequency spectrum (Fig. 14.27b). If once again we deal with the signal in the frequency domain, the familiar operation of multiplying a signal by a pulse function can be used to eliminate these replicated spectra (Fig. 14.27c), leaving only a single copy of the original spectrum (Fig. 14.27d). Thus, we can reconstruct the signal from its samples by multiplying the Fourier transform of the samples by a pulse function in the frequency domain or by convolving the samples with a sinc with $A = 1$ in the spatial domain.

To make the Fourier transforms of signals and filters easier to see in Figs. 14.27–29, we have taken several liberties:

- The DC value of the Fourier transform in part (a) of each figure has been truncated. This corresponds to a signal in the spatial domain with the same shape as shown, but with a negative DC offset. (Such a signal cannot be displayed as an image without further processing, because it contains negative intensity values.)
- Filters in the frequency domain have not been drawn with the correct magnitude. Their heights should be 1 in Fig. 14.27 and 2 in Figs. 14.28–29 to restore the single copy of the spectrum to its original magnitude.

Figure 14.27(e) and (f) show the result of reconstructing the samples with a triangle filter (also known as a Bartlett filter). Convolving with this filter is equivalent to linearly interpolating the samples.

If the sampling frequency is too low, the replicated copies of the frequency spectra overlap, as in Fig. 14.28. In this case, the reconstruction process will fail to remove those parts of the replicated spectra that overlapped the original signal's spectrum. High-frequency components from the replicated spectra are mixed in with low-frequency components from the original spectrum, and therefore are treated like low frequencies during the reconstruction process. Note how an inadequate sampling rate causes aliasing by making a higher frequency appear identical to a lower one before and after reconstruction. There are two ways to resolve this problem. We may choose to sample at a high enough frequency, an approach that is sufficient only if the signal does not have an infinite spectrum. Alternatively, we may filter the signal before sampling to remove all components above $f_s/2$, as shown in Fig. 14.29.

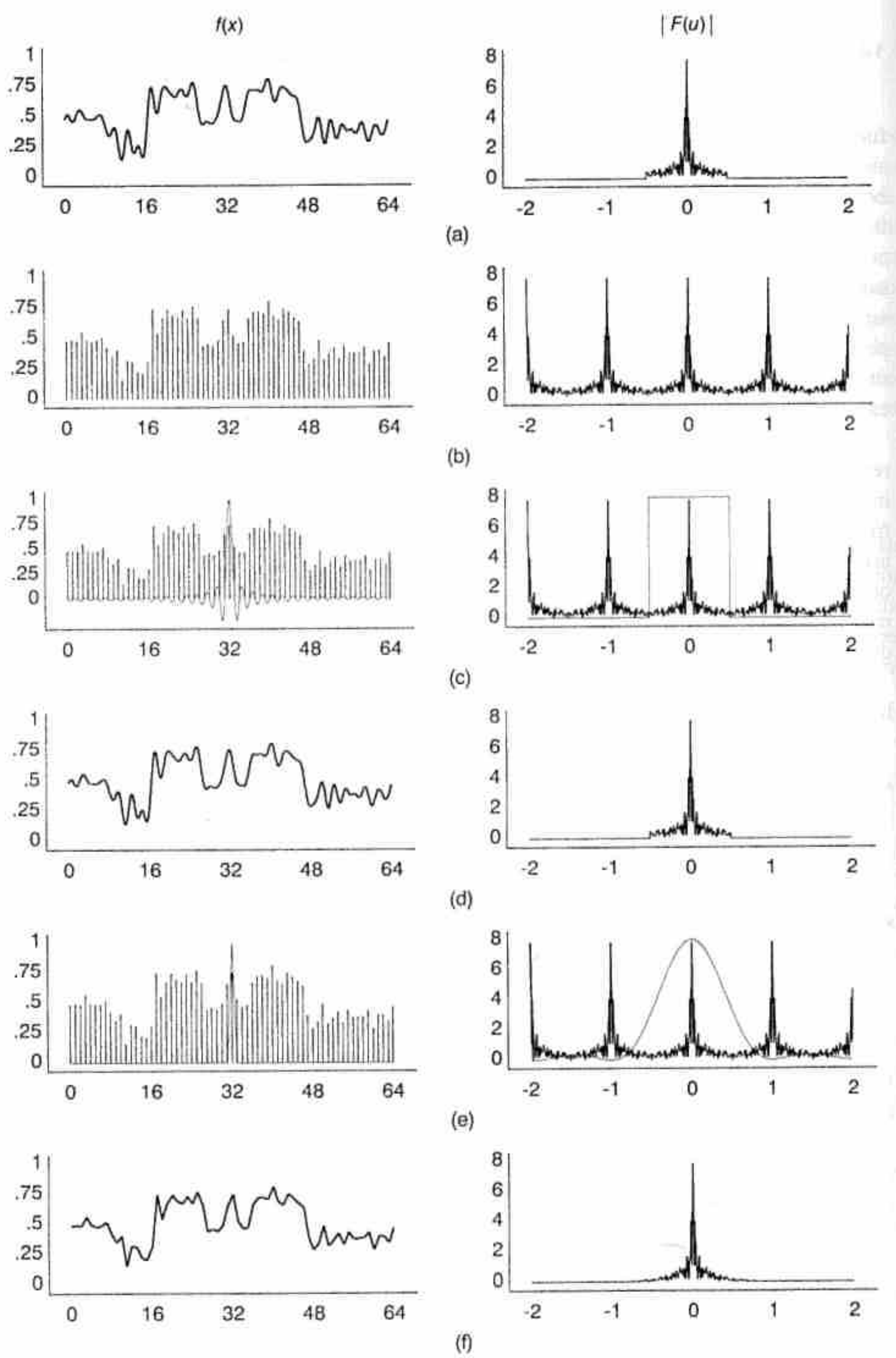


Fig. 14.27 Sampling and reconstruction: Adequate sampling rate. (a) Original signal. (b) Sampled signal. (c) Sampled signal ready to be reconstructed with sinc. (d) Signal reconstructed with sinc. (e) Sampled signal ready to be reconstructed with triangle. (f) Signal reconstructed with triangle. (Courtesy of George Wolberg, Columbia University.)

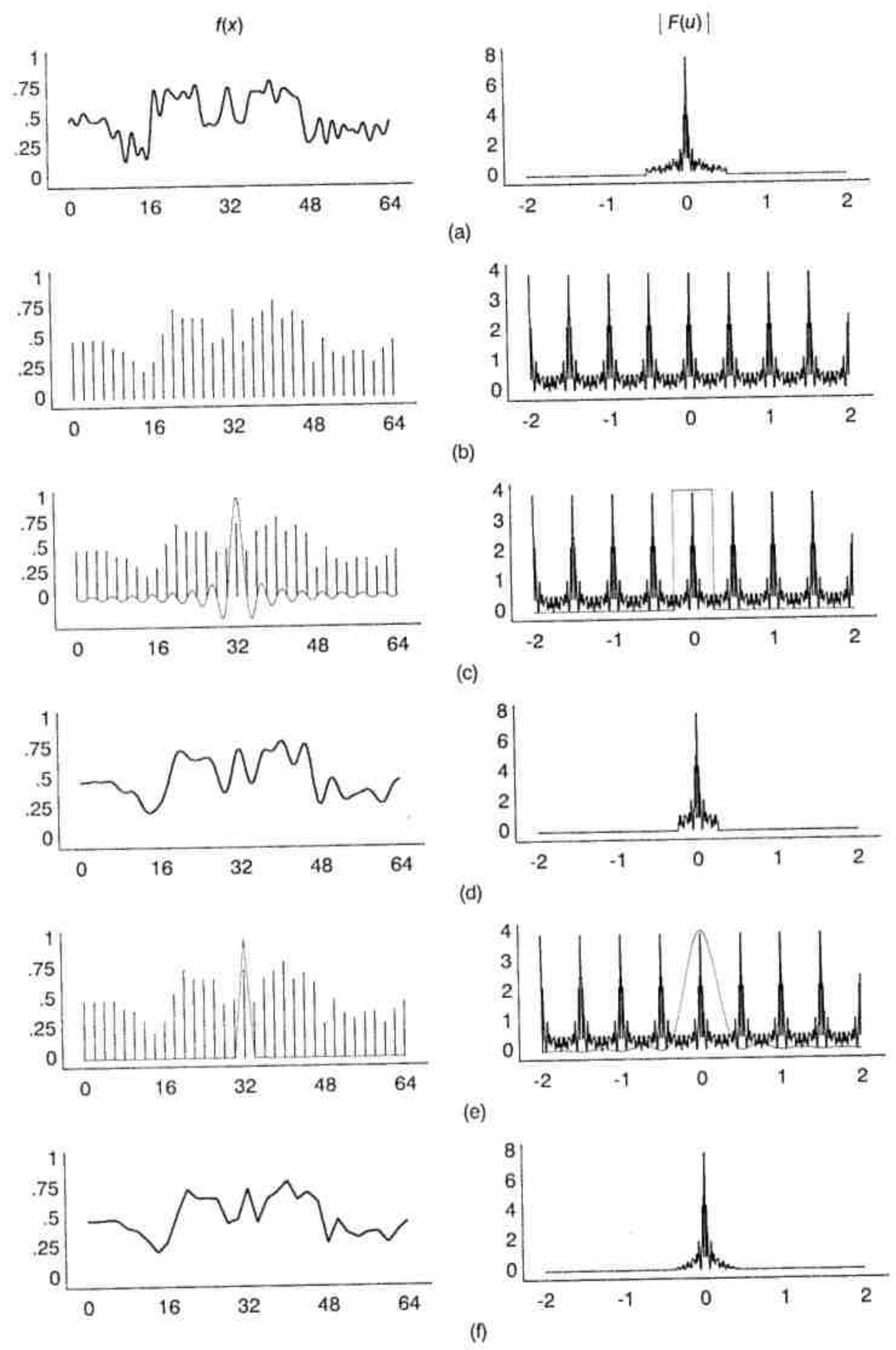


Fig. 14.28 Sampling and reconstruction: Inadequate sampling rate. (a) Original signal. (b) Sampled signal. (c) Sampled signal ready to be reconstructed with sinc. (d) Signal reconstructed with sinc. (e) Sampled signal ready to be reconstructed with triangle. (f) Signal reconstructed with triangle. (Courtesy of George Wolberg, Columbia University.)

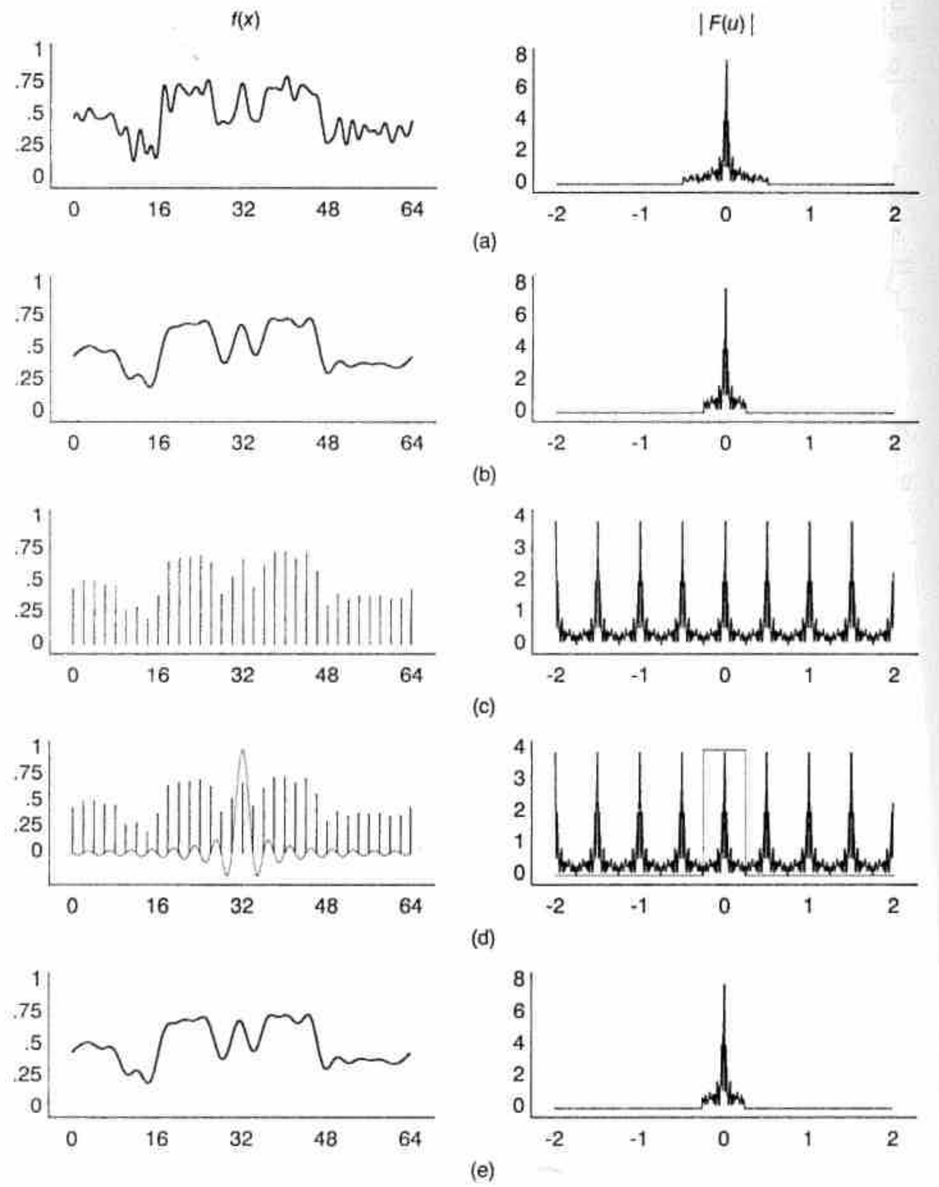


Fig. 14.29 Filtering, sampling, and reconstruction: Sampling rate is adequate after filtering. (a) Original signal. (b) Low-pass filtered signal. (c) Sampled signal. (d) Sampled signal ready to be reconstructed with sinc. (e) Signal reconstructed with sinc. (Courtesy of George Wolberg, Columbia University.)

What happens if reconstruction is done by convolving with some signal other than a sinc? Samples in the frame buffer are translated into a continuous video signal, by a process known as *sample and hold*: for the signal to be reconstructed, the value of each successive sample is simply held for the duration of a pixel. This process corresponds to convolving the samples with a 1-pixel-wide box filter, as shown in Fig. 14.30, and gives rise to our common conception of a pixel as one of a set of square boxes tiling the display. The resulting signal has sharp transitions between pixels, corresponding to high-frequency components that are not represented by the samples. This effect is often known as *rastering*. Although the video hardware nominally samples and holds each pixel's intensity, the circuitry that generates the analog voltages applied to the CRT and the CRT itself are generally not fast enough to produce discontinuous jumps in intensity between pixels. The Gaussian distribution of the CRT spot also reduces this problem. Thus, the sampled signal is reconstructed by the equivalent of convolution with a box filter, followed by convolution with a Gaussian. Rastering is especially easy to see, however, when pixel-replicating zoom is used in raster CRT displays, increasing the amount of screen space allocated to an individual pixel. Rastering is also more evident in printer, digital film recorder, and LCD

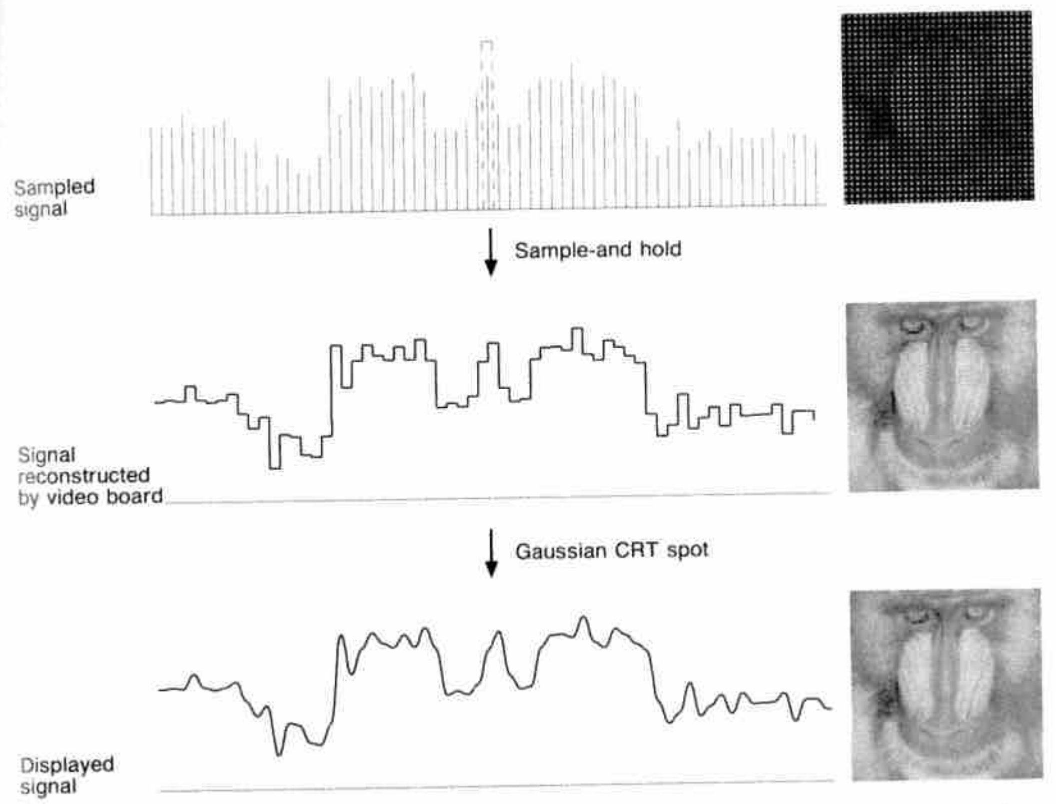


Fig. 14.30 Reconstruction by sample and hold and Gaussian CRT spot. (Courtesy of George Wolberg, Columbia University.)

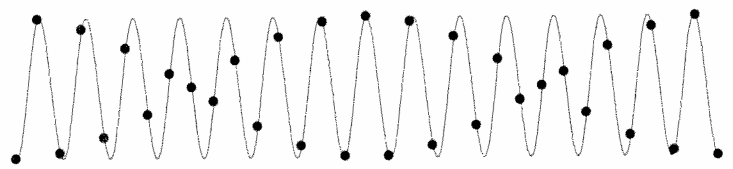


Fig. 14.31 A signal sampled at slightly over the Nyquist rate. (Courtesy of George Wolberg, Columbia University.)

technologies, in which pixel-to-pixel transitions are much sharper and produce relatively hard-edged square pixels of constant intensity.

We noted earlier that a signal must be sampled at a frequency greater than $2f_h$ to make perfect reconstruction possible. If the filter used to reconstruct the samples is not an untruncated sinc, as is always the case when displaying an image, then the sampling frequency must be even higher! Consider, for example, a sampling frequency slightly greater than $2f_h$. The resulting samples trace out the original signal modulated by (multiplied by) a low-frequency sine wave, as shown in Fig. 14.31. The low-frequency amplitude modulation remains, compounded by rastering, if the signal is reconstructed with a 1-pixel-wide box filter. If convolution is performed with an untruncated sinc, however, the original signal is recovered. The inevitable use of nonideal filters before and after sampling therefore mandates higher sampling rates. Mitchell and Netravali [MITC88] discuss some of the problems involved in doing a good job of reconstruction.

14.10.6 Antialiasing in Practice

We have seen that image synthesis involves sampling and reconstruction, noting that there is little that we can do (in software) about the reconstruction approach employed in hardware. Rendering algorithms that perform antialiasing use either point sampling or an analytic approach, such as area sampling. In either case, a single value must ultimately be determined for each pixel. Catmull's algorithm, discussed in Section 15.7.3, is an example of an analytic (and expensive) approach using unweighted area sampling. It corresponds to filtering at object precision before calculating the value of each pixel's sample. Filtering before sampling is often called *prefiltering*. When supersampling is used, the samples are combined according to a filter weighting in a discrete version of the continuous convolution and sampling that we discussed earlier. The filter is represented by an array of values. As shown in Fig. 14.32, the filter array is positioned over the array of supersampled values and the sum of the products of values in corresponding positions determines a single sample taken at the center of the filter. The filter array is then moved to the position at which the next sample will be taken, with the number of samples corresponding to the pixel resolution of the filtered image being created. This approach is often called *postfiltering*, since filtering is performed after point sampling. It actually corresponds to reconstructing the signal from its samples only at selected points in space. These reconstructed values are then used as new samples. Supersampling thus performs a discrete approximation to weighted area sampling.

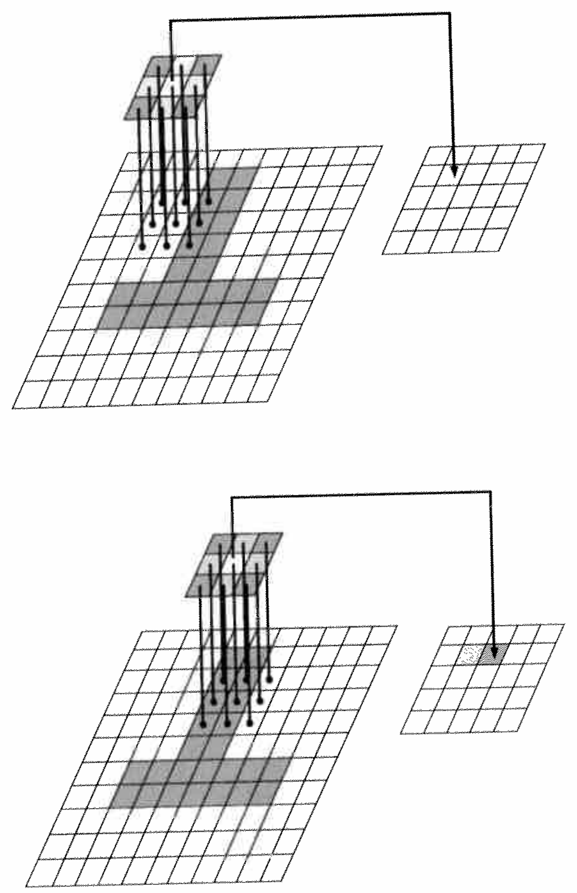
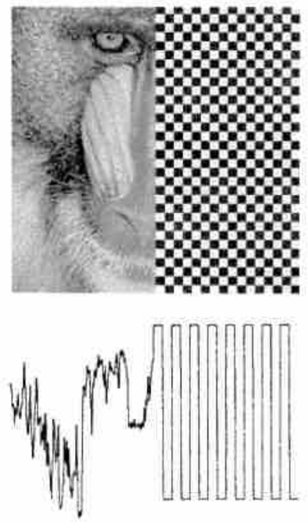


Fig. 14.32 Digital filtering. Filter is used to combine samples to create a new sample.

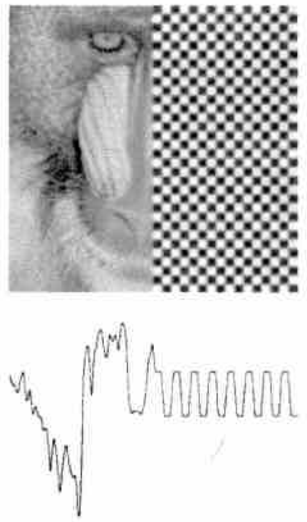
Although it is computationally attractive to use a 1-pixel-wide box filter that averages all subpixel samples, better filters can produce better results, as demonstrated in Fig. 14.33. Note that, no matter what filter is used to postfilter the samples, damage caused by an inadequate initial sampling rate will not be repaired. A rule of thumb is that supersampling four times in each of x and y often will be satisfactory [WHIT85]. This works because the high frequencies in most graphics images are caused by discontinuities at edges, which have a Fourier transform that tapers off rapidly (like the Fourier transform of a pulse—the sinc). In contrast, images with textures and distant objects viewed in perspective have a Fourier transform that is richer in high frequencies and that may be arbitrarily difficult to filter.

Although it is easy to increase the sampling rate, this approach is limited in its usefulness by corresponding increases in both processing time and storage. A number of variations on point sampling have been implemented to address these issues without sacrificing the conceptually simple mechanism of point sampling itself. In *adaptive*



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

(a)

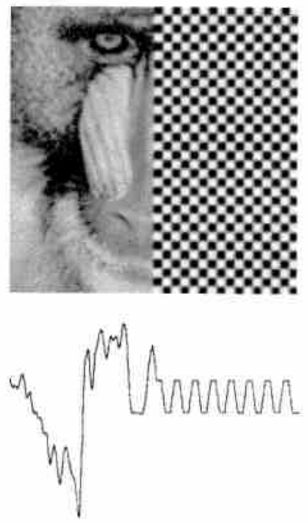


1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

(c)

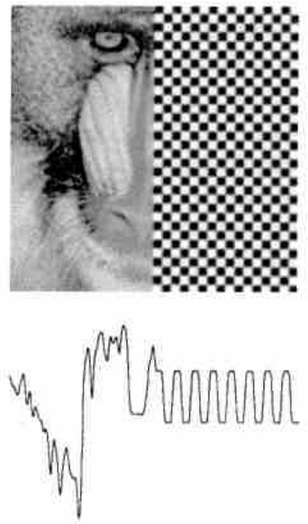
Fig. 14.33 *(Cont'd.)*

supersampling, an example of which is discussed in Section 15.10.4, the sampling rate is varied across the image, with additional samples taken when the system determines that they are needed. *Stochastic supersampling*, discussed in Section 16.12.4, places samples at stochastically determined positions, rather than in a regular grid. This approach produces



1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

(b)



1	4	8	10	8	4	1
4	12	25	29	25	12	4
8	25	49	58	49	25	8
10	29	58	67	58	29	10
8	25	49	58	49	25	8
4	12	25	29	25	12	4
1	4	8	10	8	4	1

(d)

Fig. 14.33 Filtered images with intensity plot of middle scan line and filter kernel. (a) Original image. (b) Box filter. (c) Bartlett filter. (d) Gaussian filter. Images are 512 × 512, and filters are 7 × 7. Middle scan line is at bottom of a checkerboard row. Because 2D filter covers light and dark squares above and below scan line, amplitude of filtered checkerboard signal along middle scan line is greatly diminished. (Courtesy of George Wolberg, Columbia University.)

aliasing in the form of noise, which our visual system finds less irritating than the clearly defined frequency components of regular aliasing. These two approaches can be combined, allowing the determination of where to place new samples to be based on the statistical properties of those that have already been obtained.

When the original source signal is itself a sampled image, postfiltering followed by resampling may be used to create a new image that has been scaled, rotated, or distorted in a variety of ways. These *image transformations* are discussed in Chapter 17.

14.11 SUMMARY

In this chapter, we provided a high-level introduction to the techniques used to produce realistic images. We then examined the causes of and cures for aliasing. In the following chapters, we discuss in detail how these techniques can be implemented. There are five key questions that you should bear in mind when you read about the algorithms presented in later chapters:

1. *Is the algorithm general or special purpose?* Some techniques work best only in specific circumstances; others are designed to be more general. For example, some algorithms assume that all objects are convex polyhedra and derive part of their speed and relative simplicity from this assumption.
2. *Can antialiasing be incorporated?* Some algorithms may not accommodate antialiasing as easily as others do.
3. *What is the algorithm's space–time performance?* How is the algorithm affected by factors such as the size or complexity of the database, or the resolution at which the picture is rendered?
4. *How convincing are the effects generated?* For example, is refraction modeled correctly, does it look right only in certain special cases, or is it not modeled at all? Can additional effects, such as shadows or specular reflection, be added? How convincing will they be? Sacrificing the accuracy with which an effect is rendered may make possible significant improvements in a program's space or time requirements.
5. *Is the algorithm appropriate, given the purpose for which the picture is created?* The philosophy behind many of the pictures in the following chapters can be summed up by the credo, "If it looks good, do it!" This directive can be interpreted two ways. A simple or fast algorithm may be used if it produces attractive effects, even if no justification can be found in the laws of physics. On the other hand, a shockingly expensive algorithm may be used if it is the only known way to render certain effects.

EXERCISES

14.1 Suppose you had a graphics system that could draw any of the color plates referenced in this chapter in real time. Consider several application areas with which you are (or would like to be) familiar. For each area, list those effects that would be most useful, and those that would be least useful.

14.2 Show that you cannot infer the direction of rotation from orthographic projections of a monochrome, rotating, wireframe cube. Explain how additional techniques can help to make the direction of rotation clear without changing the projection.

14.3 Consider the pulse function $f(x) = 1$ for $-1 \le x \le 1$, and $f(x) = 0$ elsewhere. Show that the Fourier transform of $f(x)$ is a multiple of the sinc function. Hint: The Fourier transform of $f(x)$ can be

computed as

$$F(u) = \int_{-1}^{+1} f(x)[\cos 2\pi ux - i\sin 2\pi ux]dx,$$

because the regions where $f(x) = 0$ contribute nothing to the integral, and $f(x) = 1$ in the remaining region. (Apply the inverse Fourier transform to your answer. You should get the original function.)

14.4 Prove that reconstructing a signal with a triangle filter of width 2 corresponds to linearly interpolating its samples. What happens if the filter is wider?

14.5 Write a program that allows you to convolve an image with a filter kernel. Use different filter kernels to create images of the same size from original images with 2, 4, and 8 times the number of pixels in x or y as the new images. You can obtain original images by saving the frame buffer generated by the graphics packages that you have been using. Do your filtered images look better than original images of the same resolution? Does your experience corroborate the rule of thumb mentioned in Section 14.10.6?