J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*
Copyright © 1999 John Wiley & Sons, Inc.

IMAGE MANIPULATION

This article addresses the problem of applying geometric transformations to digital images. The principle ideas behind this form of image manipulation include sampling theory, reconstruction, and antialiasing. We will review these topics and provide codes and examples to illustrate their use in image manipulation.

Sampling theory is central to the study of sampled-data systems, for example, digital image transformations. It lays a firm mathematical foundation for the analysis of sampled signals, offering invaluable insight into the problems and solutions of sampling. It does so by providing an elegant mathematical formulation describing the relationship between a continuous signal and its samples. We use it to resolve the problems of image reconstruction and aliasing. Reconstruction is an interpolation procedure applied to the sampled data. It permits us to evaluate the discrete signal at any desired position, not just the integer lattice upon which the sampled signal is given. This is useful when implementing geometric transformations, or warps, on the image. Aliasing refers to the presence of unreproducibly high frequencies in the image and the resulting artifacts that arise upon undersampling.

Together with defining theoretical limits on the continuous reconstruction of discrete input, sampling theory yields the guidelines for numerically measuring the quality of various proposed filtering techniques. This proves most useful in formally describing reconstruction, aliasing, and the filtering necessary to combat the artifacts that may appear at the output.

In order to better motivate the importance of sampling theory and filtering, we demonstrate its role with the following examples. A checkerboard texture is shown projected onto an oblique planar surface in Fig. 1. The image exhibits two forms of artifacts: jagged edges and moire patterns. Jagged edges are prominent toward the bottom of the image, where the input checkerboard undergoes magnification. It reflects poor reconstruction of the underlying signal. The moire patterns, on the other hand, are noticeable at the top, where minification (compression) forces many input pixels to occupy fewer output pixels. This artifact is due to aliasing, a symptom of undersampling.

Figure 1(a) was generated by projecting the center of each output pixel into the checkerboard and sampling (reading) the value of the nearest input pixel. This point sampling method performs poorly, as is evident by the objectionable results of Fig. 1(a). This conclusion is reached by sampling theory as well. Its role here is to precisely quantify this phenomena and to prescribe a solution. Figure 1(b) shows the same mapping with improved results. This time, the necessary steps were taken to preclude artifacts. In particular, a superior reconstruction algorithm was used for interpolation to suppress the jagged edges, and antialiasing filtering was carried out to combat the symptoms of undersampling that gave rise to the moire patterns.

Sampling Theory

Both reconstruction and antialiasing share the two-fold problem addressed by sampling theory:

(1) Given a continuous input signal $g(x)$ and its sampled counterpart $g_s(x)$, are the samples of $g_s(x)$ sufficient to exactly describe $g(x)$?
(2) If so, how can $g(x)$ be reconstructed from $g_s(x)$?

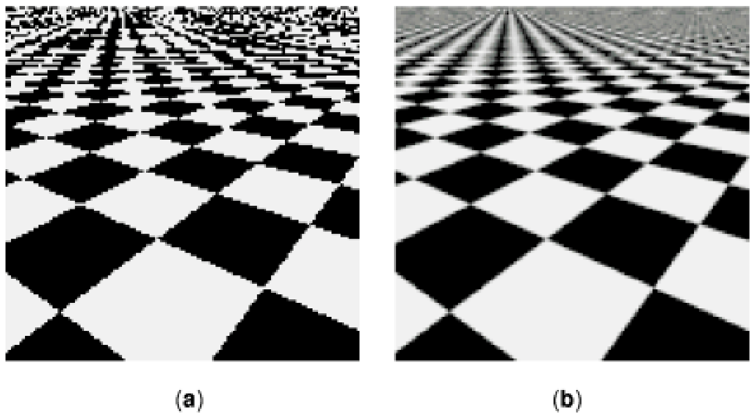

Fig. 1. Oblique checkerboard: (a) unfiltered, (b) filtered.

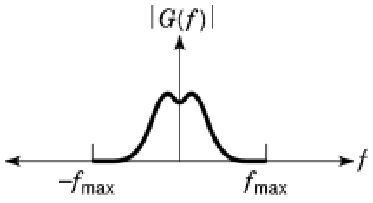

Fig. 2. Spectrum $G(f)$.

The solution lies in the frequency domain, whereby spectral analysis is used to examine the spectrum of the sampled data.

The conclusions derived from examining the reconstruction problem will prove to be directly useful for resampling and indicative of the filtering necessary for antialiasing. Sampling theory thereby provides an elegant mathematical framework in which to assess the quality of reconstruction, establish theoretical limits, and predict when it is not possible.

Sampling. Consider a 1-D signal $g(x)$ and its spectrum $G(f)$, as determined by the Fourier transform.

$$G(f) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi fx}\, dx \qquad (1)$$

Note that x represents spatial position, and f denotes spatial frequency.

The magnitude spectrum of a signal is shown in Fig. 2. It shows the frequency content of the signal with a high concentration of energy in the low-frequency range, tapering off toward the higher frequencies. Since there are no frequency components beyond $f_{\max}$, the signal is said to be bandlimited to frequency $f_{\max}$.

The continuous output $g(x)$ is then digitized by an ideal impulse sampler, the comb function, to get the sampled signal $g_s(x)$. The ideal 1-D sampler is given as

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nT_s) \qquad (2)$$

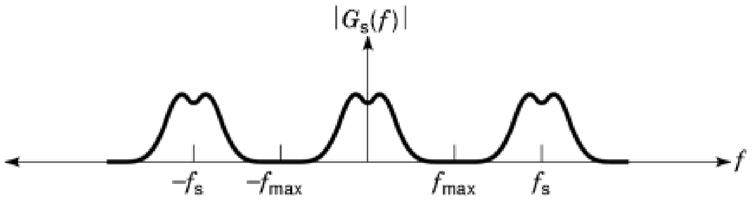

Fig. 3. Spectrum $G_s(f)$.

where δ is the familiar impulse function, and T_s is the sampling period. The running index n is used with δ to define the impulse train of the comb function. We now have

$$g_s(x) = g(x)s(x) \tag{3}$$

Taking the Fourier transform of $g_s(x)$ yields

$$G_s(f) = G(f)^* S(f) \tag{4}$$

$$= G(f) * \left[\sum_{n=-\infty}^{n=\infty} f_s \delta(f - nf_s) \right] \tag{5}$$

$$= f_s \sum_{n=-\infty}^{n=\infty} G(f - nf_s) \tag{6}$$

where f_s is the sampling frequency, and $*$ denotes convolution. The above equations make use of the following well-known properties of Fourier transforms:

(1) Multiplication in the spatial domain corresponds to convolution in the frequency domain. Therefore, Eq. (3) gives rise to a convolution in Eq. (4).
(2) The Fourier transform of an impulse train is itself an impulse train, giving us Eq. (5).
(3) The spectrum of a signal sampled with frequency f_s ($T_s = 1/f_s$) yields the original spectrum replicated in the frequency domain with period f_s in Eq. (6).

This last property has important consequences. It yields spectrum $G_s(f)$ which, in response to a sampling period $T_s = 1/f_s$, is periodic in frequency with period f_s. This is depicted in Fig. 3. Notice then, that a small sampling period is equivalent to a high sampling frequency yielding spectra replicated far apart from each other. In the limiting case when the sampling period approaches zero ($T_s \to 0, f_s \to \infty$), only a single spectrum appears—a result consistent with the continuous case.

Reconstruction

The above result reveals that the sampling operation has left the original input spectrum intact, merely replicating it periodically in the frequency domain with a spacing of f_s. This allows us to rewrite $G_s(f)$ as a sum of two terms, the low frequency (baseband) and high frequency components. The baseband spectrum is exactly $G(f)$, and the high frequency components, $G_{\text{high}}(f)$, consist of the remaining replicated versions of $G(f)$ that constitute harmonic versions of the sampled image.

$$G_s(f) = G(f) + G_{\text{high}}(f) \tag{7}$$

Exact signal reconstruction from sampled data requires us to discard the replicated spectra $G_{\text{high}}(f)$, leaving only $G(f)$, the spectrum of the signal we seek to recover. This is a crucial observation in the study of sampled-data systems.

Reconstruction Conditions. The only provision for exact reconstruction is that $G(f)$ be undistorted due to overlap with $G_{\text{high}}(f)$. Two conditions must hold for this to be true:

(1) The signal must be bandlimited. This avoids spectra with infinite extent that are impossible to replicate without overlap.
(2) The sampling frequency f_s must be greater than twice the maximum frequency $f_{\max}$, present in the signal. This minimum sampling frequency, known as the Nyquist rate, is the minimum distance between the spectra copies, each with bandwidth $f_{\max}$.

The first condition merely ensures that a sufficiently large sampling frequency exists that can be used to separate replicated spectra from each other. Since all imaging systems impose a bandlimiting filter in the form of a point spread function, this condition is always satisfied for images captured through an optical system. Note that this does not apply to synthetic images, for example, computer-generated imagery.

The second condition proves to be the most revealing statement about reconstruction. It answers the problem regarding the sufficiency of the data samples to exactly reconstruct the continuous input signal. It states that exact reconstruction is possible only when $f_s > f_{\text{Nyquist}}$, where $f_{\text{Nyquist}} = 2\,f_{\max}$. Collectively, these two conclusions about reconstruction form the central message of sampling theory, as pioneered by Claude Shannon in his landmark papers on the subject (1,2).

Ideal Low-Pass Filter. We now turn to the second central problem: Given that it is theoretically possible to perform reconstruction, how may it be done? The answer lies with our earlier observation that sampling merely replicates the spectrum of the input signal, generating $G_{\text{high}}(f)$ in addition to $G(f)$. Therefore, the act of reconstruction requires us to completely suppress $G_{\text{high}}(f)$. This is done by multiplying $G_s(f)$ with $H(f)$, given as

$$H(f) = \begin{cases} 1 & |f| < f_{\max} \\ 0 & |f| \geq f_{\max} \end{cases} \tag{8}$$

$H(f)$ is known as an ideal low-pass filter and is depicted in Fig. 4, where it is shown suppressing all frequency components above $f_{\max}$. This serves to discard the replicated spectra $G_{\text{high}}(f)$. It is ideal in the sense that the $f_{\max}$ cut-off frequency is strictly enforced as the transition point between the transmission and complete suppression of frequency components.

Sinc Function. In the spatial domain, the ideal low-pass filter is derived by computing the inverse Fourier transform of $H(f)$. This yields the sinc function shown in Fig. 5. It is defined as

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \tag{9}$$

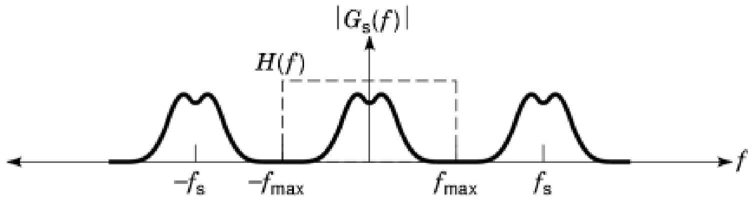

Fig. 4. Ideal low-pass filter $H(f)$

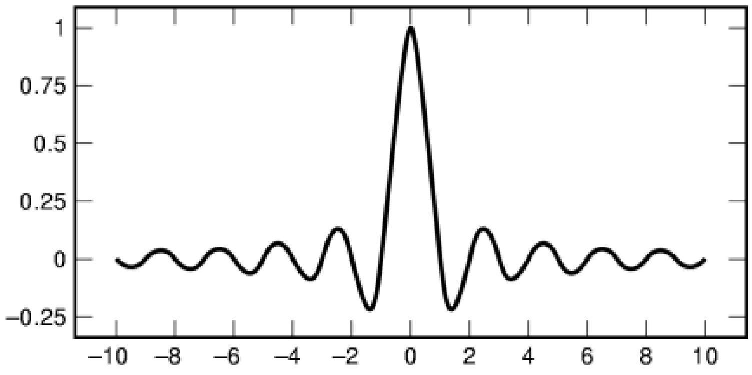

Fig. 5. The sinc function.

The reader should note the reciprocal relationship between the height and width of the ideal low-pass filter in the spatial and frequency domains. Let A denote the amplitude of the sinc function, and let its zero crossings be positioned at integer multiples of $\frac{1}{2}W$. The spectrum of this sinc function is a rectangular pulse of height $A/2W$ and width $2W$, with frequencies ranging from $-W$ to W. In our example above, $A = 1$ and $W = f_{\max}, = .5$ cycles/pixel. This value for W is derived from the fact that digital images must not have more than one half cycle per pixel in order to conform to the Nyquist rate.

The sinc function is one instance of a large class of functions known as cardinal splines, which are interpolating functions defined to pass through zero at all but one data sample, where they have a value of one. This allows them to compute a continuous function that passes through the uniformly-spaced data samples.

Since multiplication in the frequency domain is identical to convolution in the spatial domain, sinc(x) represents the convolution kernel used to evaluate any point x on the continuous input curve g given only the sampled data g_s:

$$g(x) = \mathrm{sinc}(x) * g_s(x) = \int_{-\infty}^{\infty} \mathrm{sinc}(\lambda) g_s(x-\lambda)\, d\lambda \qquad (10)$$

Equation (10) highlights an important impediment to the practical use of the ideal low-pass filter. The filter requires an infinite number of neighboring samples (i.e., an infinite filter support) in order to precisely compute the output points. This is, of course, impossible owing to the finite number of data samples available. However, truncating the sinc function allows for approximate solutions to be computed at the expense of undesirable ringing, that is, ripple effects. These artifacts, known as the Gibbs phenomenon, are the overshoots and undershoots caused by reconstructing a signal with truncated frequency terms. The two rows in Fig. 6 show that truncation in one domain leads to ringing in the other domain. This indicates that a truncated sinc

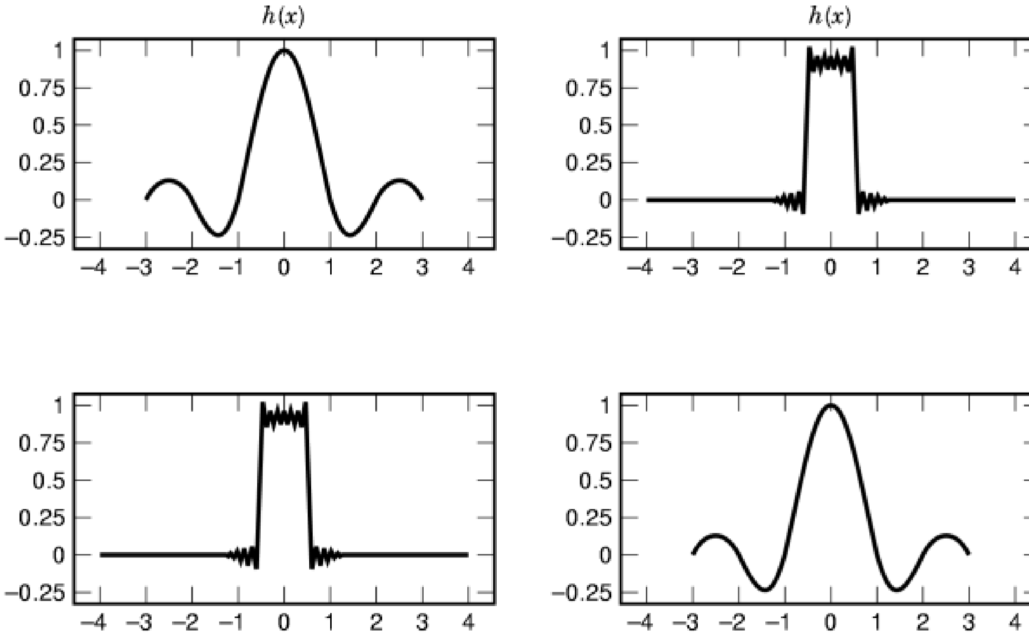

Fig. 6. Truncation in one domain causes ringing in the other domain.

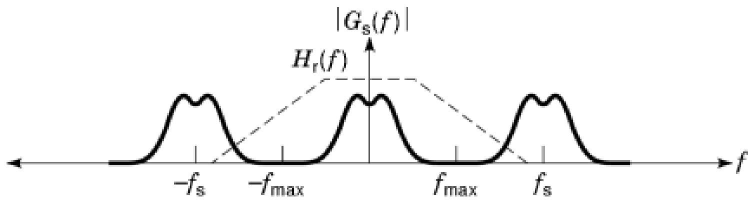

Fig. 7. Nonideal reconstruction.

function is actually a poor reconstruction filter because its spectrum has infinite extent and thereby fails to bandlimit the input.

In response to these difficulties, a number of approximating algorithms have been derived, offering a trade-off between precision and computational expense. These methods permit local solutions that require the convolution kernel to extend only over a small neighborhood. The drawback, however, is that the frequency response of the filter has some undesirable properties. In particular, frequencies below $f_{\max}$, are tampered, and high frequencies beyond $f_{\max}$, are not fully suppressed. Thus, nonideal reconstruction does not permit us to exactly recover the continuous underlying signal without artifacts.

Nonideal Reconstruction. The process of nonideal reconstruction is depicted in Fig. 7, which indicates that the input signal satisfies the two conditions necessary for exact reconstruction. First, the signal is bandlimited since the replicated copies in the spectrum are each finite in extent. Second, the sampling frequency exceeds the Nyquist rate since the copies do not overlap. However, this is where our ideal scenario ends. Instead of using an ideal low-pass filter to retain only the baseband spectrum components, a nonideal reconstruction filter is shown in the figure.

The filter response $H_r(f)$ deviates from the ideal response $H(f)$ shown in Fig. 4. In particular, $H_r(f)$ does not discard all frequencies beyond $f_{\max}$. Furthermore, that same filter is shown to attenuate some frequencies that should have remained intact. This brings us to the problem of assessing the quality of a filter.

The accuracy of a reconstruction filter can be evaluated by analyzing its frequency domain characteristics. Of particular importance is the filter response in the passband and stopband. In this problem, the passband consists of all frequencies below $f_{\max}$. The stopband contains all higher frequencies arising from the sampling process.

An ideal reconstruction filter, as described earlier, will completely suppress the stopband while leaving the passband intact. Recall that the stopband contains the offending high frequencies that, if allowed to remain, would prevent us from performing exact reconstruction. As a result, the sinc filter was devised to meet these goals and serve as the ideal reconstruction filter. Its kernel in the frequency domain applies unity gain to transmit the passband and zero gain to suppress the stopband.

The breakdown of the frequency domain into passband and stopband isolates two problems that can arise due to nonideal reconstruction filters. The first problem deals with the effects of imperfect filtering on the passband. Failure to impose unity gain on all frequencies in the passband will result in some combination of image smoothing and image sharpening. Smoothing, or blurring, will result when the frequency gains near the cut-off frequency start falling off. Image sharpening results when the high frequency gains are allowed to exceed unity. This follows from the direct correspondence of visual detail to spatial frequency. Furthermore, amplifying the high passband frequencies yields a sharper transition between the passband and stopband, a property shared by the sinc function.

The second problem addresses nonideal filtering on the stopband. If the stopband is allowed to persist, high frequencies will exist that will contribute to aliasing (described later). Failure to fully suppress the stopband in a condition known as frequency leakage. This allows the offending frequencies to fold over into the passband range. These distortions tend to be more serious since they are visually perceived more readily.

In the spatial domain, nonideal reconstruction is achieved by centering a finite-width kernel at the position in the data at which the underlying function is to be evaluated, that is, reconstructed. This is an interpolation problem which, for equally spaced data, can be expressed as

$$f(x) = \sum_{k=0}^{K-1} f(x_k)h(x - x_k) \tag{11}$$

where h is the reconstruction kernel that weighs K data samples at x_k. Equation (11) formulates interpolation as a convolution operation. In practice, h is nearly always a symmetric kernel; that is, $h(-x) = h(x)$. We shall assume this to be true in the discussion that follows.

The computation of one interpolated point is illustrated in Fig. 8. The kernel is centered at x, the location of the point to be interpolated. The value of that point is equal to the sum of the values of the discrete input scaled by the corresponding values of the reconstruction kernel. This follows directly from the definition of convolution.

Reconstruction Kernels

The numerical accuracy and computational cost of reconstruction is directly tied to the convolution kernel used for low-pass filtering. As a result, filter kernels are the target of design and analysis in the creation and evaluation of reconstruction algorithms. They are subject to conditions influencing the tradeoff between accuracy and efficiency. This section reviews several common nonideal reconstruction filter kernels in the order of their complexity: box filter, triangle filter, cubic convolution, and windowed sinc functions.

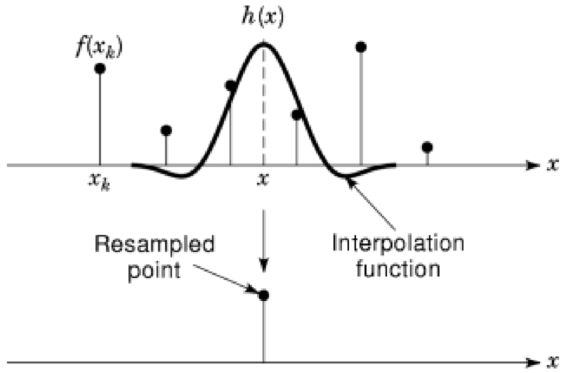

Fig. 8. Interpolation of a single point.

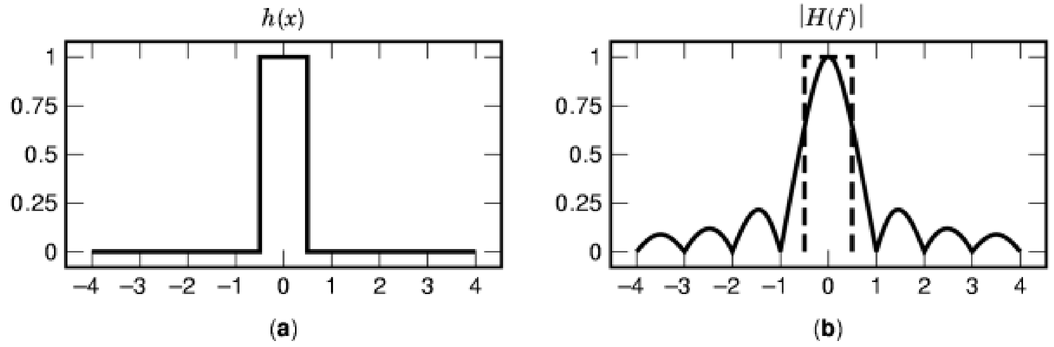

Fig. 9. Box filter: (a) kernel, (b) Fourier transform.

Box Filter. The box filter kernel is defined as

$$h(x) = \begin{cases} 1 & -.5 < x \leq .5 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Various other names are used to denote this simple kernel, including the sample-and-hold function and Fourier window. The kernel and its Fourier transform are shown in Fig. 9.

Convolution in the spatial domain with the rectangle function h is equivalent in the frequency domain to multiplication with a sinc function. Due to the prominent side lobes and infinite extent, a sinc function makes a poor low-pass filter. Consequently, this filter kernel has a poor frequency domain response relative to that of the ideal low-pass filter. The ideal filter, drawn as a dashed rectangle, is characterized by unity gain in the passband and zero gain in the stopband. This permits all low frequencies (below the cut-off frequency) to pass and all higher frequencies to be suppressed.

Triangle Filter. The triangle filter kernel is defined as

$$h(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & 1 \leq |x| \end{cases} \tag{13}$$

Kernel h is also referred to as a tent filter, roof function, Chateau function, or Bartlett window.

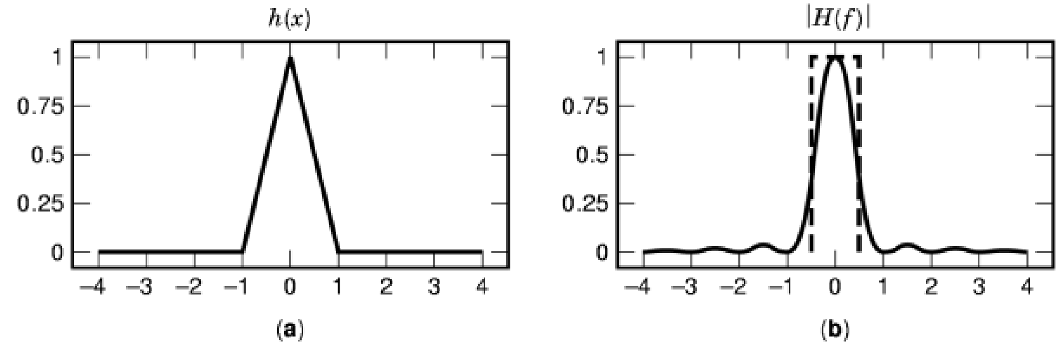

Fig. 10. Triangle filter: (a) kernel, (b) Fourier transform.

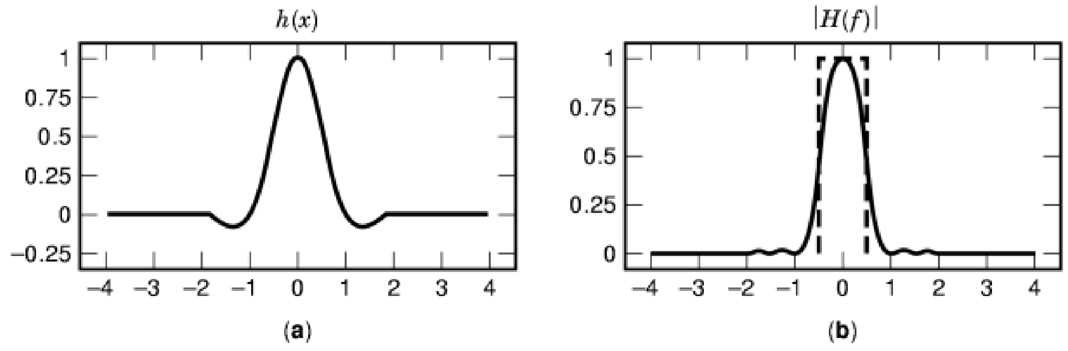

Fig. 11. Cubic convolution: (a) kernel ($a = -.5$), (b) Fourier transform.

This kernel corresponds to a reasonably good low-pass filter in the frequency domain. As shown in Fig. 10, its response is superior to that of the box filter. In particular, the side lobes are far less prominent, indicating improved performance in the stopband. Nevertheless, a significant amount of spurious high-frequency components continue to leak into the passband, contributing to some aliasing. In addition, the passband is moderately attenuated, resulting in image smoothing.

Cubic Convolution. The cubic convolution kernel is a third-degree approximation to the sinc function. It is symmetric, space-invariant, and composed of piecewise cubic polynomials:

$$h(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{14}$$

where $-3 < a < 0$ is used to make h resemble the sinc function.

Of all the choices for a, the value -1 is preferable if visually enhanced results are desired. That is, the image is sharpened, making visual detail perceived more readily. However, the results are not mathematically precise, where precision is measured by the order of the Taylor series. To maximize this order, the value $a = -.5$ is preferable. A cubic convolution kernel with $a = -.5$ and its spectrum are shown in Fig. 11.

Windowed Sinc Function. Sampling theory establishes that the sinc function is the ideal interpolation kernel. Although this interpolation filter is exact, it is not practical since it is an infinite impulse response (*IIR*)

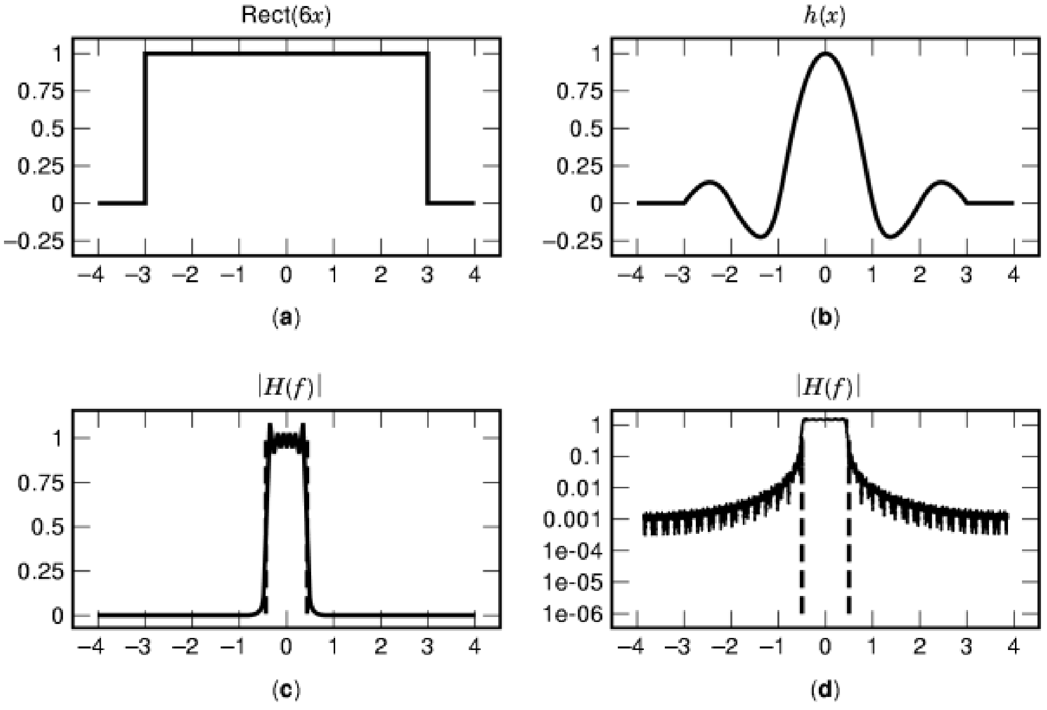

Fig. 12. (a) Rectangular window, (b) Windowed sinc, (c) Spectrum, (d) Log plot.

filter defined by a slowly converging infinite sum. Nevertheless, it is perfectly reasonable to consider the effects of using a truncated, and therefore finite, sinc function as the interpolation kernel.

The results of this operation are predicted by sampling theory, which demonstrates that truncation in one domain leads to ringing in the other domain. This is due to the fact that truncating a signal is equivalent to multiplying it with a rectangle function Rect(x), defined as the box filter of Eq. (12). Since multiplication in one domain is convolution in the other, truncation amounts to convolving the signal's spectrum with a sinc function, the transform pair of Rect(x). Since the stopband is no longer eliminated, but rather attenuated by a ringing filter (i.e., a sinc), the input is not bandlimited, and aliasing artifacts are introduced. The most typical problems occur at step edges, where the Gibbs phenomena becomes noticeable in the form of undershoots, overshoots, and ringing in the vicinity of edges.

The Rect function above served as a window, or kernel, that weighs the input signal. In Fig. 12(a), we see the Rect window extended over three pixels on each side of its center, that is, Rect($6x$) is plotted. The corresponding windowed sinc function $h(x)$ is shown in Fig. 12(b). This is simply the product of the sinc function with the window function, i.e., sinc(x)Rect($6x$). Its spectrum, shown in Fig. 12(c), is nearly an ideal low-pass filter. Although it has a fairly sharp transition from the passband to the stopband, it is plagued by ringing. In order to more clearly see the values in the spectrum, we use a logarithmic scale for the vertical axis of the spectrum in Fig. 12(d). The next few figures will be illustrated by using this same four-part format.

Ringing can be mitigated by using a different windowing function exhibiting smoother fall-off than the rectangle. The resulting windowed sinc function can yield better results. However, since slow fall-off requires larger windows, the computation remains costly.

Aside from the rectangular window mentioned above, the most frequently used window functions are Hann, Hamming, Blackman, and Kaiser. These filters identify a quantity known as the ripple ratio, defined as

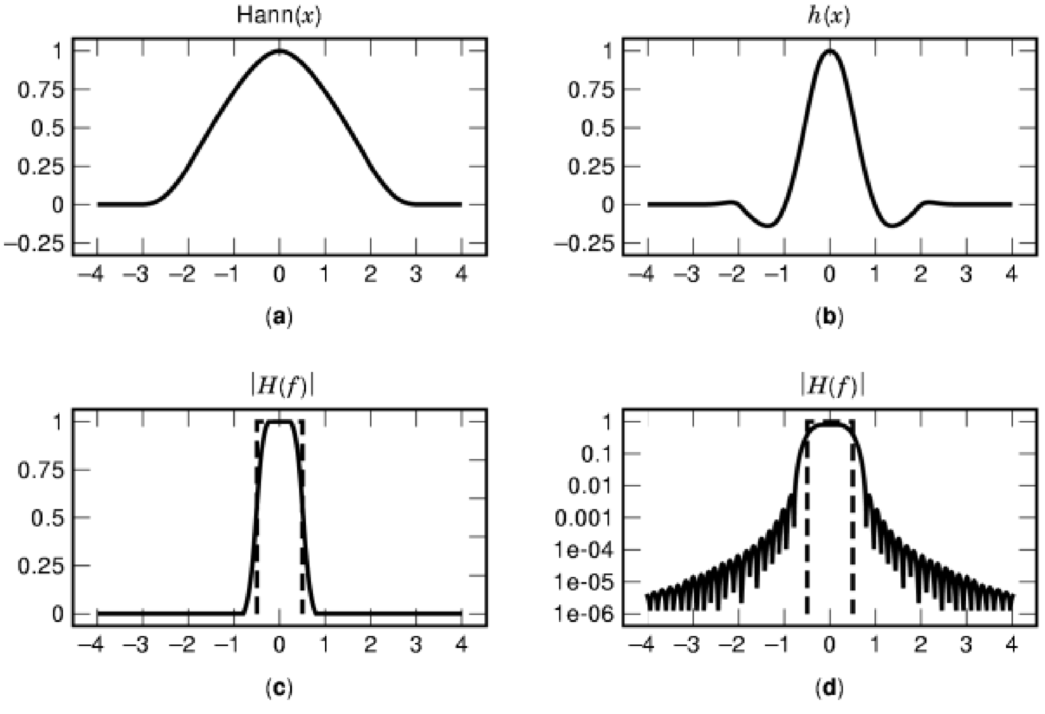

Fig. 13. (a) Hann window, (b) Windowed sinc, (c) Spectrum, (d) Log plot.

the ratio of the maximum side-lobe amplitude to the main-lobe amplitude. Good filters will have small ripple ratios to achieve effective attenuation in the stopband. A tradeoff exists, however, between ripple ratio and main-lobe width. Therefore, as the ripple ratio is decreased, the main-lobe width is increased. This is consistent with the reciprocal relationship between the spatial and frequency domains; that is, narrow bandwidths correspond to wide spatial functions.

In general, though, each of these smooth window functions is defined over a small finite extent. This is tantamount to multiplying the smooth window with a rectangle function. While this is better than the Rect function alone, there will inevitably be some form of aliasing. Nevertheless, the window functions described below offer a good compromise between ringing and blurring.

Hann and Hamming Windows. The Hann and Hamming windows are defined as

$$\text{Hann/Hamming}(x) = \begin{cases} \alpha + (1-\alpha)\cos\dfrac{2\pi x}{N-1} & |x| < \dfrac{N-1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where N is the number of samples in the windowing function. The two windowing functions differ in their choice of α. In the Hann window, $\alpha = 0.5$, and in the Hamming window $\alpha = 0.54$. Since they both amount to a scaled and shifted cosine function, they are also known as the raised cosine window. The Hann window is illustrated in Fig. 13. Notice that the passband is only slightly attenuated, but the stopband continues to retain high frequency components in the stopband, albeit less than that of Rect(x).

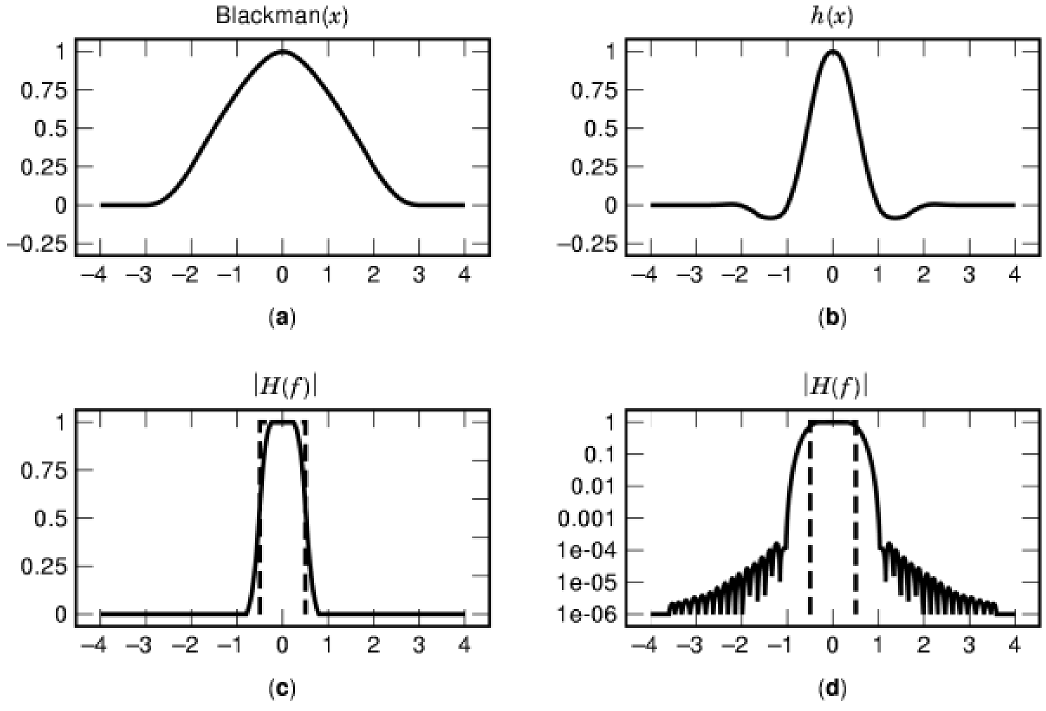

Fig. 14. (a) Blackman window, (b) Windowed sinc, (c) Spectrum, (d) Log plot.

Blackman Window. The Blackman window is similar to the Hann and Hamming windows. It is defined as

$$\text{Blackman}(x) = \begin{cases} 0.42 + 0.5\cos\dfrac{2\pi x}{N-1} + 0.08\cos\dfrac{4\pi x}{N-1} & |x| < \dfrac{N-1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The purpose of the additional cosine term is to further reduce the ripple ratio. This window function is shown in Fig. 14.

Kaiser Window. The Kaiser window is defined as

$$\text{Kaiser}(x) = \begin{cases} \dfrac{I_0(\beta)}{I_0(\alpha)} & |x| < \dfrac{N-1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where I_0, is the zeroth-order Bessel function of the first kind, α is a free parameter, and

$$\beta = \alpha \left[1 - \left(\frac{2x}{N-1} \right)^2 \right]^{1/2} \tag{18}$$

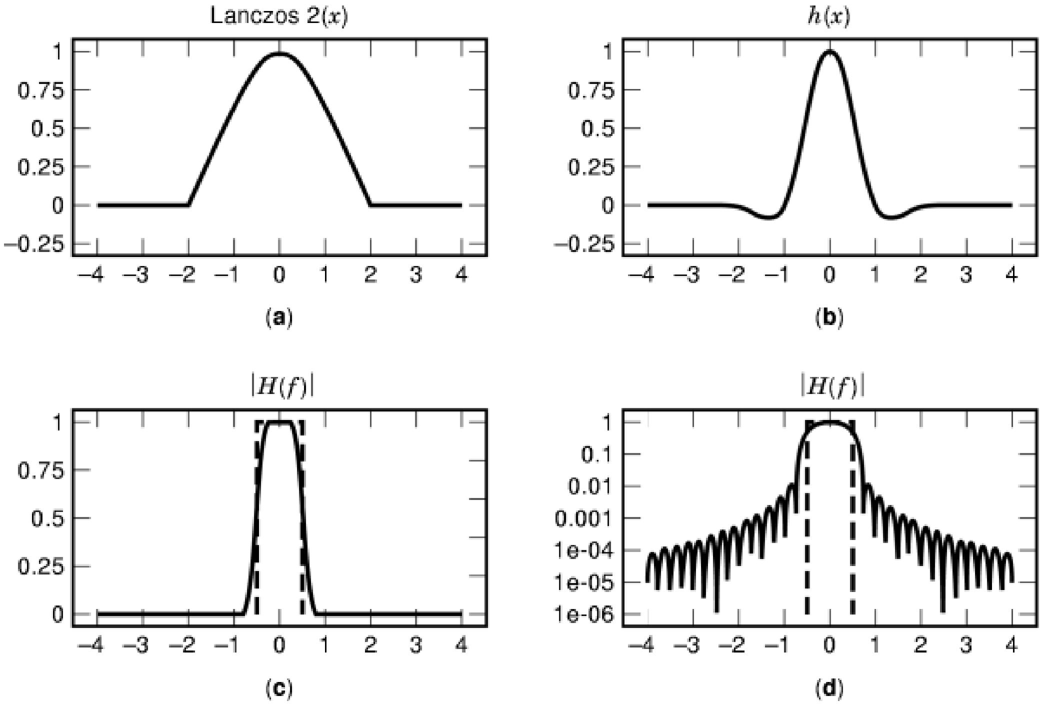

Fig. 15. (a) Lanczos2 window, (b) Windowed sinc, (c) Spectrum, (d) Log plot.

The Kaiser window leaves the filter designer much flexibility in controlling the ripple ratio by adjusting the parameter α. As α is incremented, the level of sophistication of the window function grows as well. Therefore, the rectangular window corresponds to a Kaiser window with $\alpha = 0$, while more sophisticated windows, such as the Hamming window, correspond to $\alpha = 5$.

Lanczos Window. The Lanczos window is based on the sinc function rather than cosines, as used in the previous methods. The two-lobed Lanczos window function is defined as

$$\text{Lanczos2}(x) = \begin{cases} \dfrac{\sin(\pi x/2)}{\pi x/2} & 0 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases} \qquad (19)$$

The Lanczos2 window function, shown in Fig. 15, is the central lobe of a sinc function. It is wide enough to extend over two lobes of the ideal low-pass filter, that is, a second sinc function. This formulation can be generalized to an N-lobed window function by replacing the value 2 in Eq. (19) with the value N. Larger N results in superior frequency response.

Aliasing

If the two reconstruction conditions outlined earlier are not met, sampling theory predicts that exact reconstruction is not possible. This phenomenon, known as aliasing, occurs when signals are not bandlimited or when they are undersampled; that is, $f_s \leq f_{\text{Nyquist}}$. In either case, there will be unavoidable overlapping of

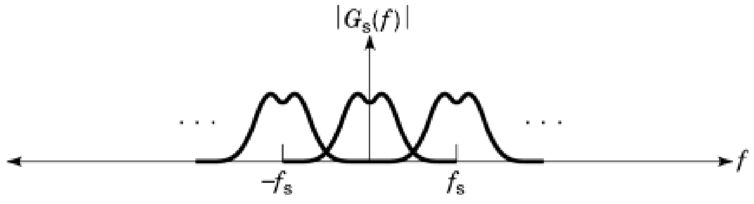

Fig. 16. Overlapping spectral components give rise to aliasing.

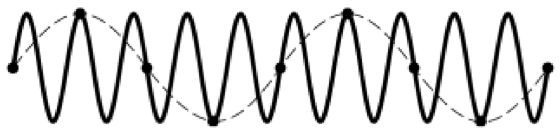

Fig. 17. Aliasing artifacts due to undersampling.

spectral components, as in Fig. 16. Notice that the irreproducible high frequencies fold over into the low frequency range. As a result, frequencies originally beyond $f_{\max}$ will, upon reconstruction, appear in the form of much lower frequencies. Unlike the spurious high frequencies retained by nonideal reconstruction filters, the spectral components passed due to undersampling are more serious since they actually corrupt the components in the original signal.

Aliasing refers to the higher frequencies becoming aliased and indistinguishable from the lower frequency components in the signal if the sampling rate falls below the Nyquist frequency. In other words, undersampling causes high-frequency components to appear as spurious low frequencies. This is depicted in Fig. 17, where a high-frequency signal appears as a low frequency signal after sampling it too sparsely. In digital images, the Nyquist rate is determined by the highest frequency that can be displayed: one cycle every two pixels. Therefore, any attempt to display higher frequencies will produce similar artifacts.

There is sometimes a misconception in the computer graphics literature that jagged (staircased) edges are always a symptom of aliasing. This is only partially true. Technically, jagged edges arise from high frequencies introduced by inadequate reconstruction. Since these high frequencies are not corrupting the low-frequency components, no aliasing is actually taking place. The confusion lies in that the suggested remedy of increasing the sampling rate is also used to eliminate aliasing. Of course, the benefit of increasing the sampling rate is that the replicated spectra are now spaced farther apart from each other. This relaxes the accuracy constraints for reconstruction filters to perform ideally in the stopband where they must suppress all components beyond some specified cut-off frequency. In this manner, the same nonideal filters will produce less objectionable output.

It is important to note that a signal may be densely sampled (far above the Nyquist rate) and continue to appear jagged if a zero-order reconstruction filter is used. Box filters used for pixel replication in real-time hardware zooms are a common example of poor reconstruction filters. In this case, the signal is clearly not aliased but rather poorly reconstructed. The distinction between reconstruction and aliasing artifacts becomes clear when we notice that the appearance of jagged edges is improved by blurring. For example, it is not uncommon to step back from an image exhibiting excessive blockiness in order to see it more clearly. This is a defocusing operation that attenuates the high frequencies admitted through nonideal reconstruction. On the other hand, once a signal is truly undersampled, there is no postprocessing possible to improve its condition. After all, applying an ideal low-pass (reconstruction) filter to a spectrum whose components are already overlapping will only blur the result, not rectify it.

Antialiasing

The filtering necessary to combat aliasing is known as antialiasing. In order to determine corrective action, we must directly address the two conditions necessary for exact signal reconstruction. The first solution calls for low-pass filtering before sampling. This method, known as prefiltering, bandlimits the signal to levels below $f_{\max}$, thereby eliminating the offending high frequencies. Notice that the frequency at which the signal is to be sampled imposes limits on the allowable bandwidth. This is often necessary when the output sampling grid must be fixed to the resolution of an output device, for example, screen resolution. Therefore, aliasing is often a problem that is confronted when a signal is forced to conform to an inadequate resolution due to physical constraints. As a result, it is necessary to bandlimit, or narrow, the input spectrum to conform to the allotted bandwidth as determined by the sampling frequency.

The second solution is to point sample at a higher frequency. In doing so, the replicated spectra are spaced farther apart, thereby separating the overlapping spectra tails. This approach theoretically implies sampling at a resolution determined by the highest frequencies present in the signal. Since a surface viewed obliquely can give rise to arbitrarily high frequencies, this method may require extremely high resolution. Whereas the first solution adjusts the bandwidth to accommodate the fixed sampling rate, f_s, the second solution adjusts f_s to accommodate the original bandwidth. Antialiasing by sampling at the highest frequency is clearly superior in terms of image quality. That is, of course, operating under different assumptions regarding the possibility of varying f_s. In practice, antialiasing is performed through a combination of these two approaches. That is, the sampling frequency is increased so as to reduce the amount of bandlimiting to a minimum.

Point Sampling. The naive approach for generating an output image is to perform point sampling, where each output pixel is a single sample of the input image taken independently of its neighbors (Fig. 18). It is clear that information is lost between the samples, and that aliasing artifacts may surface if the sampling density is not sufficiently high to characterize the input. This problem is rooted in the fact that intermediate intervals between samples, which should have some influence on the output, are skipped entirely.

The Star image is a convenient example that overwhelms most resampling filters due to the infinitely high frequencies found toward the center. Nevertheless, the extent of the artifacts are related to the quality of the filter and the actual spatial transformation. Figure 19 shows two examples of the moire effects that can appear when a signal is undersampled using point sampling. In Fig. 19(a), one out of every two pixels in the Star image was discarded to reduce its dimension. In Fig. 19(b), the artifacts of undersampling are more pronounced as only one out of every four pixels are retained. In order to see the small images more clearly, they are magnified using cubic spline reconstruction. Clearly, these examples show that point sampling behaves poorly in high frequency regions.

Aliasing can be reduced by point sampling at a higher resolution. This raises the Nyquist limit, accounting for signals with higher bandwidths. Generally, though, the display resolution places a limit on the highest frequency that can be displayed, and thus limits the Nyquist rate to one cycle every two pixels. Any attempt to display higher frequencies will produce aliasing artifacts such as moire patterns and jagged edges. Consequently, antialiasing algorithms have been derived to bandlimit the input before resampling onto the output grid.

Area Sampling. The basic flaw in point sampling is that a discrete pixel actually represents an area, not a point. In this manner, each output pixel should be considered a window looking onto the input image. Rather than sampling a point, we must instead apply a low-pass filter (*LPF*) upon the projected area in order to properly reflect the information content being mapped onto the output pixel. This approach, depicted in Fig. 20, is called area sampling, and the projected area is known as the preimage. The low-pass filter comprises the prefiltering stage. It serves to defeat aliasing by bandlimiting the input image prior to resampling it onto the

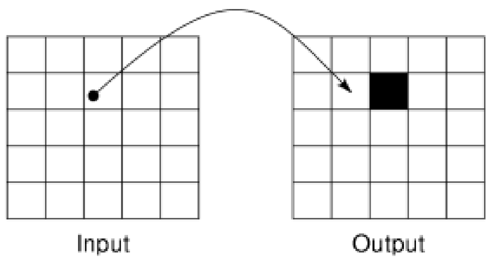

Fig. 18. Point sampling.

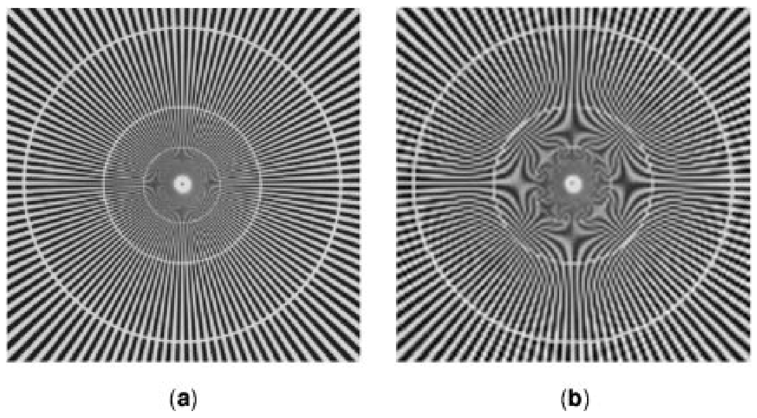

Fig. 19. Aliasing due to point sampling: (a) $\frac{1}{2}$ and (b) $\frac{1}{4}$ scale.

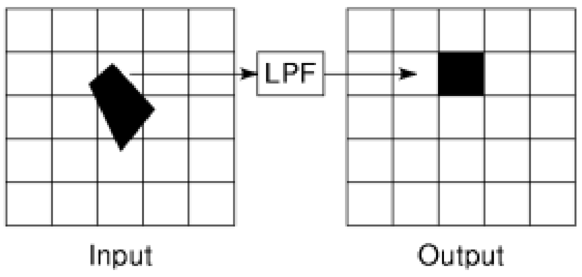

Fig. 20. Area sampling.

output grid. In the general case, prefiltering is defined by the convolution integral

$$g(x, y) = \int \int f(u, v)h(x - u, y - v)\, du\, dv \qquad (20)$$

where f is the input image, g is the output image, h is the filter kernel, and the integration is applied to all [u, v] points in the preimage.

Images produced by area sampling are demonstrably superior to those produced by point sampling. Figure 21 shows the Star image subjected to the same downsampling transformation as that in Fig. 19. Area sampling was implemented by applying a box filter (i.e., unweighted averaging) to the Star image before point sampling.

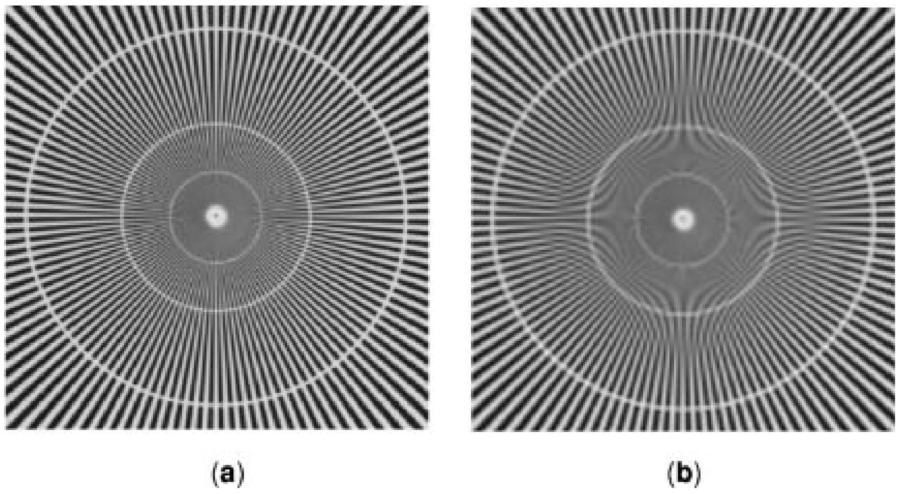

Fig. 21. Aliasing due to area sampling: (a) $\frac{1}{2}$ and (b) $\frac{1}{4}$ scale.

Notice that antialiasing through area sampling has traded moire patterns for some blurring. Although there is no substitute to high-resolution imagery, filtering can make lower resolution less objectionable by attenuating aliasing artifacts.

Supersampling. The process of using more than one regularly spaced sample per pixel is known as supersampling. Each output pixel value is evaluated by computing a weighted average of the samples taken from their respective preimages. For example, if the supersampling grid is three times denser than the output grid (i.e., there are nine grid points per pixel area), each output pixel will be an average of the nine samples taken from its projection in the input image. If, say, three samples hit a green object, and the remaining six samples hit a blue object, the composite color in the output pixel will be one-third green and two-thirds blue, assuming a box filter is used.

Supersampling reduces aliasing by bandlimiting the input signal. The purpose of the high-resolution supersampling grid is to refine the estimate of the preimages seen by the output pixels. The samples then enter the prefiltering stage, consisting of a low-pass filter. This permits the input to be resampled onto the (relatively) low-resolution output grid without any offending high frequencies introducing aliasing artifacts. In Fig. 22, we see an output pixel subdivided into nine subpixel samples which each undergo inverse mapping, sampling the input at nine positions. Those nine values then pass through a low-pass filter to be averaged into a single output value.

Supersampling was used to achieve antialiasing in Fig. 1 for pixels near the horizon. There are two problems, however, associated with straightforward supersampling. The first problem is that the newly designated high frequency of the prefiltered image continues to be fixed. Therefore, there will always be sufficiently higher frequencies that will alias. The second problem is cost. In our example, supersampling will take nine times longer than point sampling. Although there is a clear need for the additional computation, the dense placement of samples can be optimized. Adaptive supersampling is introduced to address these drawbacks.

Adaptive Supersampling. In adaptive supersampling, the samples are distributed more densely in areas of high intensity variance. In this manner, supersamples are collected only in regions that warrant their use. Early work in adaptive supersampling for computer graphics is described in (3). The strategy is to subdivide areas between previous samples when an edge, or some other high frequency pattern, is present.

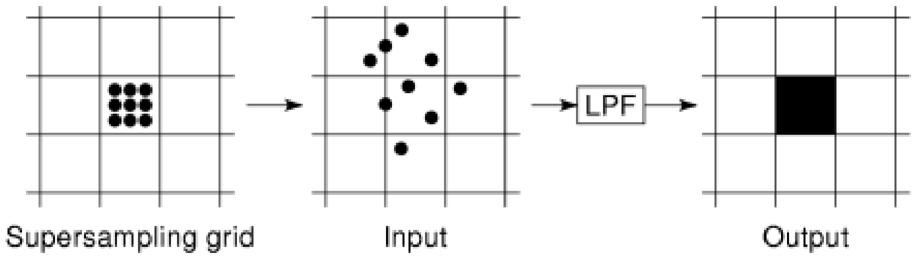

Fig. 22. Supersampling.

Two approaches to adaptive supersampling have been described in the literature. The first approach allows sampling density to vary as a function of local image variance (4,5). A second approach introduces two levels of sampling densities: a regular pattern for most areas and a higher-density pattern for regions demonstrating high frequencies. The regular pattern simply consists of one sample per output pixel. The high density pattern involves local supersampling at a rate of 4 to 16 samples per pixel. Typically, these rates are adequate for suppressing aliasing artifacts.

A strategy is required to determine where supersampling is necessary. In (6), the author describes a method in which the image is divided into small square supersampling cells, each containing eight or nine of the low-density samples. The entire cell is supersampled if its samples exhibit excessive variation. In (4), the variance of the samples are used to indicate high frequency. It is well-known, however, that variance is a poor measure of visual perception of local variation. Another alternative is to use contrast, which more closely models the nonlinear response of the human eye to rapid fluctuations in light intensities (7). Contrast is given as

$$C = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}} \qquad (21)$$

Adaptive sampling reduces the number of samples required for a given image quality. The problem with this technique, however, is that the variance measurement is itself based on point samples, and so this method can fail as well. This is particularly true for subpixel objects that do not cross pixel boundaries. Nevertheless, adaptive sampling presents a far more reliable and cost-effective alternative to supersampling.

Prefiltering

Area sampling can be accelerated if constraints on the filter shape are imposed. Pyramids and preintegrated tables are introduced to approximate the convolution integral with a constant number of accesses. This compares favorably against direct convolution which requires a large number of samples that grow proportionately to the preimage area. As we shall see, though, the filter area will be limited to squares or rectangles, and the kernel will consist of a box filter. Subsequent advances have extended their use to more general cases with only marginal increases in cost.

Pyramids. Pyramids are multi-resolution data structures commonly used in image processing and computer vision. They are generated by successively bandlimiting and subsampling the original image to form a hierarchy of images at ever decreasing resolutions. The original image serves as the base of the pyramid, and its coarsest version resides at the apex. Thus, in a lower resolution version of the input, each pixel represents the average of some number of pixels in the higher resolution version.

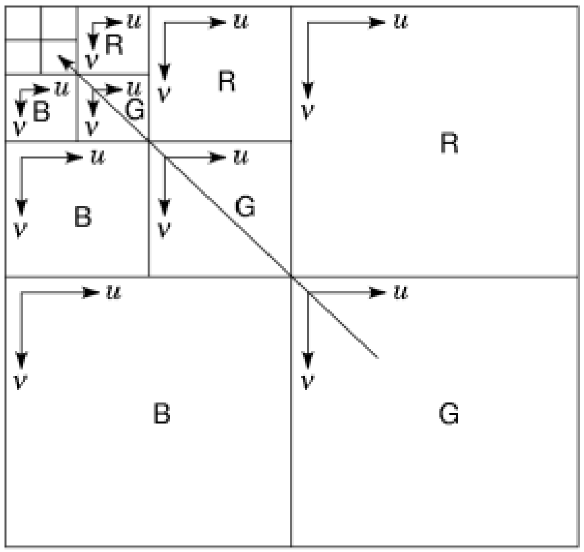

Fig. 23. Mip Map memory organization.

The resolution of successive levels typically differs by a power of two. This means that successively coarser versions each have one quarter of the total number of pixels as their adjacent predecessors. The memory cost of this organization is modest: $1 + \frac{1}{4} + 1/16 + \ldots = 4/3$ times that needed for the original input. This requires only 33% more memory.

To filter a preimage, one of the pyramid levels is selected based on the size of its bounding square box. That level is then point sampled and assigned to the respective output pixel. The primary benefit of this approach is that the cost of the filter is constant, requiring the same number of pixel accesses independent of the filter size. This performance gain is the result of the filtering that took place while creating the pyramid. Furthermore, if preimage areas are adequately approximated by squares, the direct convolution methods amount to point sampling a pyramid. This approach was first applied to texture mapping in (8) and described in (9).

There are several problems with the use of pyramids. First, the appropriate pyramid level must be selected. A coarse level may yield excessive blur, while the adjacent finer level may be responsible for aliasing due to insufficient bandlimiting. Second, preimages are constrained to be squares. This proves to be a crude approximation for elongated preimages. For example, when a surface is viewed obliquely, the texture may be compressed along one dimension. Using the largest bounding square will include the contributions of many extraneous samples and result in excessive blur. These two issues were addressed in (10) and (11), respectively, along with extensions proposed by other researchers.

Williams (10) proposed a pyramid organization called *mip map* to store color images at multiple resolutions in a convenient memory organization. The acronym mip stands for *multum in parvo,* a Latin phrase meaning *many things in a small place.* The scheme supports trilinear interpolation, where both intra- and inter-level interpolation can be computed using three normalized coordinates: u, v, and d. Both u and v are spatial coordinates used to access points within a pyramid level. The d coordinate is used to index and interpolate between different levels of the pyramid. This is depicted in Fig. 23.

The quadrants touching the east and south borders contain the original red, green, and blue (RGB) components of the color image. The remaining upper-left quadrant contains all the lower resolution copies of the original. The memory organization depicted in Fig. 23 clearly supports the earlier claim that memory cost is 4/3 times that required for the original input. Each level is shown indexed by the $[u, v, d]$ coordinate system, where d is shown slicing through the pyramid levels. Since corresponding points in different pyramid levels have indices which are related by some power of two, simple binary shifts can be used to access these points across the multiresolution copies. This is a particularly attractive feature for hardware implementation.

The primary difference between mip maps and ordinary pyramids is the trilinear interpolation scheme possible with the $[u, v, d]$ coordinate system. By allowing a continuum of points to be accessed, mip maps are referred to as pyramidal parametric data structures. In Williams' implementation, a box filter was used to create the mip maps, and a triangle filter was used to perform intra- and inter-level interpolation. The value of d must be chosen to balance the tradeoff between aliasing and blurring. Heckbert suggests

$$d^2 = \mathrm{MAX}\left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2, \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right) \qquad (22)$$

where d is proportional to the span of the preimage area, and the partial derivatives can be computed from the surface projection (12).

Summed-Area Tables. An alternative to pyramidal filtering was proposed by Crow in (11). It extends the filtering possible in pyramids by allowing rectangular areas, oriented parallel to the coordinate axes, to be filtered in constant time. The central data structure is a preintegrated buffer of intensities, known as the summed-area table. This table is generated by computing a running total of the input intensities as the image is scanned along successive scanlines. For every position P in the table, we compute the sum of intensities of pixels contained in the rectangle between the origin and P. The sum of all intensities in any rectangular area of the input may easily be recovered by computing a sum and two differences of values taken from the table. For example, consider the rectangles R_0, R_1, R_2, and R shown in Fig. 24. The sum of intensities in rectangle R can be computed by considering the sum at $[x1, y1]$, and discarding the sums of rectangles R_0, R_1, and R_2. This corresponds to removing all areas lying below and to the left of R. The resulting area is rectangle R, and its sum S is given as

$$S = T[x1, y1] - T[x1, y0] - T[x0, y1] + T[x0, y0] \qquad (23)$$

where $T[x, y]$ is the value in the summed-area table indexed by coordinate pair $[x, y]$.

Since $T[x1, y0]$ and $T[x0, y1]$ both contain R_0, the sum of R_0 was subtracted twice in Eq. (23). As a result, $T[x0, y0]$ was added back to restore the sum. Once S is determined, it is divided by the area of the rectangle. This gives the average intensity over the rectangle, a process equivalent to filtering with a Fourier window (box filtering).

There are two problems with the use of summed-area tables. First, the filter area is restricted to rectangles. This is addressed in (13), where an adaptive, iterative technique is proposed for obtaining arbitrary filter areas by removing extraneous regions from the rectangular bounding box. Second, the summed-area table is restricted to box filtering. This, of course, is attributed to the use of unweighted averages that keeps the algorithm simple. In (14) and (15), the summed-area table is generalized to support more sophisticated filtering by repeated integration.

It is shown that by repeatedly integrating the summed-area table n times, it is possible to convolve an orthogonally oriented rectangular region with an nth-order box filter (B-spline). The output value is computed by using $(n + 1)^2$ weighted samples from the preintegrated table. Since this result is independent of the size

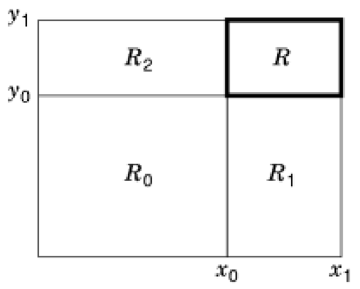

Fig. 24. Summed-area table calculation.

of the rectangular region, this method offers a great reduction in computation over that of direct convolution. Perlin (14) called this a selective image filter because it allows each sample to be blurred by different amounts.

Repeated integration has rather high memory costs relative to pyramids. This is due to the number of bits necessary to retain accuracy in the large summations. Nevertheless, it allows us to filter rectangular or elliptical regions, rather than just squares as in pyramid techniques. Since pyramid and summed-area tables both require a setup time, they are best suited for input that is intended to be used repeatedly, that is, stationary background scenes or texture maps. In this manner, the initialization overhead can be amortized over each use.

Example: Image Scaling

In this section, we demonstrate the role of reconstruction and antialiasing in image scaling. The resampling process will be explained in one dimension rather than two, since resampling is carried out in each axis independently. For example, the horizontal scanlines are first processed, yielding an intermediate image which then undergoes a second pass of interpolation in the vertical direction. The result is independent of the order: processing the vertical lines before the horizontal lines gives the same results.

A skeleton of a C program that resizes an image in two passes is given below. The input image is assumed to have *INheight* rows and *INwidth* columns. The first pass visits each row and resamples them to have width *OUTwidth*. The second pass visits each column of the newly formed intermediate image and resamples them to have height *OUTheight*

```
INwidth   = input  image width  (pixels/row);
INheight  = input  image height (rows/image);
OUTwidth  = output image width  (pixels/row);
OUTheight = output image height (rows/image);
filter    = convolution kernel to use to filter image;
offset    = inter-pixel offset (stride);

allocate an intermediate image of size OUTwidth by INheight;

offset = 1;
for(y=0; y<INheight; y++) {        /* process rows */
     src = pointer to row y of input image;
     dst = pointer to row y of intermediate image;
     resample1D(src, dst, INwidth, OUTwidth, filter, offset);
}

offset = OUTwidth;
for(x=0; x<w; x++) {          /* process columns */
     src = pointer to column x of intermediate image;
     dst = pointer to column x of output image;
     resample1D(src, dst, INheight, OUTheight, filter, offset);
}
```

Function resample1*D* is the workhorse of the resizing operation. The inner workings of this function will be described later. In addition to the input and output points and dimensions, resample1*D* must be passed *filter*, an integer code specifying which convolution kernel to apply. In order to operate on both rows and columns, the parameter *offset* is given to denote the distance between successive pixels in the scanline. Horizontal scanlines (rows) have *offset* $= 1$, and vertical scanlines (columns) have *offset* $= OUTwidth$.

There are two operations which resample1*D* must be able to handle: magnification and minification. As mentioned earlier, these two operations are closely related. They both require us to project each output sample into the input, center a kernel, and convolve. The only difference between magnification and minification is the shape of the kernel. The magnification kernel is fixed at $h(x)$, whereas the minification kernel is $ah(ax)$, for $a < 1$. The width of the kernel for minification is due to the need for a low-pass filter to perform antialiasing. That filter now has a narrower response than that of the interpolation function. Consequently, we exploit the following well known Fourier transform pair:

$$h(ax) \longleftrightarrow \frac{1}{a}H\left(\frac{f}{a}\right) \tag{24}$$

This equation expresses the reciprocal relationship between the spatial and frequency domains. Notice that multiplying the spatial axis by a factor of a results in dividing the frequency axis and the spectrum values by that same factor. Since we want the spectrum values to be left intact, we use $ah(ax)$ as the convolution kernel for blurring, where $a > 1$. This implies that the shape of the kernel changes as a function of scale factor when we are downsampling the input. This was not the case for magnification.

A straightforward method to perform 1-D resampling is given below. It details the inner workings of the resample 1-*D* function outlined earlier. In addition, a few interpolation functions are provided. More such functions can easily be added by the user.

```
#define PI           3.14159265358979311160E0
#define SGN(A)       ((A) > 0 ? 1 : ((A) < 0 ? -1 : 0 ))
#define FLOOR(A)     ((int) (A))
#define CEILING(A)   ((A)==FLOOR(A) ? FLOOR(A) : SGN(A)+FLOOR(A))
#define CLAMP(A,L,H) ((A)<=(L) ? (L) : (A)<=(H) ? (A) : (H))

resample1D(IN, OUT, INlen, OUTlen, filtertype, offset)
unsigned char *IN, *OUT;
int INlen, OUTlen, filtertype, offset;
{
    int i;
    int left, right;     /* kernel extent in input    */
    int pixel;           /* input pixel value   */
    double r, x;         /* input (r) , output (x)    */
    double scale;        /* resampling scale factor */
    double fwidth;       /* filter width (support)    */
    double fscale;       /* filter amplitude scale    */
    double weight;       /* kernel weight        */
    double acc;          /* convolution accumulator */

    scale = (double) OUTlen / INlen;

    switch(filtertype) {
    case 0: filter = boxFilter;      /* box filter (nearest neighbor)    */
            fwidth = .5;
            break;
    case 1:  filter = triFilter;     /* triangle filter (linear intrp) */
            fwidth = 1;
            break;
    case 2:  filter = cubicConv;     /* cubic convolution filter     */
            fwidth = 2;
            break;
    case 3:  filter = lanczos3;   /* Lanczos3 windowed sinc function */
            fwidth = 3;
            break;
    case 4:  filter = hann4; /* Hann windowed sinc function */
            fwidth = 4;             /* 8-point kernel */
            break;
        }

        if(scale < 1.0) {           /* minification: h(x)->h(x*scale)*scale */
            fwidth = fwidth / scale;       /* broaden filter */
            fscale = scale;          /* lower amplitude */
        } else      fscale = 1.0;
        /* project each output pixel to input, center kernel, and convolve */
        for(x=0; x<OUTlen; x++) {
            /* map output x to input u: inverse mapping */
            r = x / scale;

            /* left and right extent of kernel centered at u */
            if(r - fwidth < 0) {
                left = FLOOR    (r - fwidth);
            else left = CEILING(r - fwidth);
            right = FLOOR(r + fwidth);

            /* reset acc for collecting convolution products */
            acc = 0;

            /* weigh input pixels around u with kernel */
            for(i=left; i <= right; i++) {
                pixel  = IN[ CLAMP(i, 0, INlen-1)*offset];
                weight = (*filter) ((r - i) * fscale);
                acc   += (pixel * weight);
            }

            /* assign weighted accumulator to OUT */
            OUT[x*offset] = acc * fscale;
          }
      }
```

```
/* ---------------------------------------------------------------------------
* boxFilter:
*
* Box (nearest neighbor) filter.
*/
double
boxFilter(t)
double t;
{
    if((t > -.5) && (t <= .5)) return(1.0);
    return(0.0);
}
/* ---------------------------------------------------------------------------
* triFilter:
*
* Triangle filter (used for linear interpolation).
*/
double
triFilter(t)
double t;
{
    if(t < 0) t = -t;
    if(t < 1.0) return(1.0 - t);
return(0.0);
}

/* ---------------------------------------------------------------------------
* cubicConv:
*
* Cubic convolution filter.
*/
double
cubicConv(t)
double t;
{
    double A, t2, t3;

    if(t < 0) t = -t;
    t2 = t * t;
    t3 = t2 * t;

    A = -1.0; /* user-specified free parameter */
    if(t < 1.0) return((A+2)*t3 - (A+3)*t2 + 1);
    if(t < 2.0) return(A*(t3 - 5*t2 + 8*t - 4));
    return(0.0);
}
/* --------------------------------------------------------------------------
* sinc:
*
* Sinc function.
*/
double
sinc(t)
double t;
{
    t *= PI;
    if(t != 0) return(sin(t) / t);
    return(1.0);
}
```

```
/* -------------------------------------------------------------------------
* lanczos3:
*
* Lanczos3 filter.
*/
double
lanczos3(t)
double t;
{
    if(t < 0) t = -t;
    if(t < 3.0) return(sinc(t) * sinc(t / 3.0));
    return(0.0);
}

/* -------------------------------------------------------------------------
* hann:
*
* Hann windowed sinc function. Assume N (width) = 4.
*/
double
hann4(t)
double t;
{
    int N = 4; /* fixed filter width */

    if(t < 0) t = -t;
    if(t < N) return(sinc(t) * (.5 + .5*cos(PI*t / N)));
 return(0.0);
}
```

There are several points worth mentioning about this code. First, the filter width *fwidth* of each of the supported kernels is initialized for use in interpolation (for magnification). We then check to see if the scale factor scale is less than one to rescale *fwidth* accordingly. Furthermore, we set *fscale,* the filter amplitude scale factor to 1 for interpolation, or *scale* for minification. We then visit each of *OUTlen* output pixels and project them back into the input where we center the filter kernel. The kernel overlaps a range of input pixels from *left* to *right*. All pixels in this range are multiplied by a corresponding kernel value. The products are added in an accumulator *acc* and assigned to the output buffer.

Note that the *CLAMP* macro is necessary to prevent us from attempting to access a pixel beyond the extent of the input buffer. By clamping to either end, we are effectively replicating the border pixel for use with a filter kernel that extends beyond the image.

In order to accommodate the processing of rows and columns, the variable *offset* is introduced to specify the inter-pixel distance. When processing rows, *offset* = 1. When processing columns, *offset* is set to the width of a row.

This code can accommodate a polynomial transformation by making a simple change to the evaluation of r. Rather than computing $u = x/scale$, we may let u be expressed by a polynomial. The method of forward differences is recommended to simplify the computation of polynomials (16).

The code given above suffers from three limitations, all dealing with efficiency.

(1) A division operation is used to compute the inverse projection. Since we are dealing with a linear mapping function, the new position at which to center the kernel may be computed incrementally. That is, there is a constant offset between each projected output sample. Accordingly, *left* and *right* should be computed incrementally as well.

(2) The set of kernel weights used in processing the first scanline apply equally to all the remaining scanlines as well. There should be no need to recompute them each time. This matter is addressed in the code supplied by (17).
(3) The kernel weights are evaluated by calling the appropriate filter function with the normalized distance from the center. This involves a lot of run-time overhead, particularly for the more sophisticated kernels that require the evaluation of a sin function, division, and several multiplies.

Additional sophisticated algorithms to deal with these issues are given in (16).

Research Issues and Summary

The computer graphics literature is replete with new and innovative work addressing the demands of sampling, reconstruction, and antialiasing. Nonuniform sampling has become important in computer graphics because it facilitates variable sampling density, and it allows us to trade structured aliasing for noise. Recent work in adaptive sampling and nonuniform reconstruction is discussed in (18). Excellent surveys in nonuniform reconstruction, which is also known as scattered data interpolation, can be found in (19), and (20). These problems are also of direct consequence to image compression. The ability to determine a unique minimal set of samples to completely represent a signal within some specified error tolerance remains an active area of research. The solution must be closely coupled with a nonuniform reconstruction method. Although traditional reconstruction methods are well-understood within the framework described in this article, the analysis of nonuniform sampling and reconstruction remains challenging.

We now summarize the basic principles of sampling theory, reconstruction, and antialiasing that were presented in this article. We have shown that a continuous signal may be reconstructed from its samples if the signal is bandlimited and the sampling frequency exceeds the Nyquist rate. These are the two necessary conditions for image reconstruction to be possible. Since sampling can be shown to replicate a signal's spectrum across the frequency domain, ideal low-pass filtering was introduced as a means of retaining the original spectrum while discarding its copies. Unfortunately, the ideal low-pass filter in the spatial domain is an infinitely wide sinc function. Since this is difficult to work with, nonideal reconstruction filters are introduced to approximate the reconstructed output. These filters are nonideal in the sense that they do not completely attenuate the spectra copies. Furthermore, they contribute to some blurring of the original spectrum. In general, poor reconstruction leads to artifacts such as jagged edges.

Aliasing refers to the phenomenon that occurs when a signal is undersampled. This happens if the reconstruction conditions mentioned above are violated. In order to resolve this problem, one of two actions may be taken. Either the signal can be bandlimited to a range that complies with the sampling frequency, or the sampling frequency can be increased. In practice, some combination of both options are taken, leaving some relatively unobjectionable aliasing in the output.

Examples of the concepts discussed thus are concisely depicted in Figs. 25 through 27. They attempt to illustrate the effects of sampling and low-pass filtering on the quality of the reconstructed signal and its spectrum. The first row of Fig. 25 shows a signal and its spectra, bandlimited to .5 cycle/pixel. For pedagogical purposes, we treat this signal as if it were continuous. In actuality, though, it is really a 256-sample horizontal cross-section taken from a digital image. Since each pixel has 4 samples contributing to it, there is a maximum of two cycles per pixel. The horizontal axes of the spectra account for this fact.

The second row shows the effect of sampling the signal. Since $f_s = 1$ sample/pixel, there are four copies of the baseband spectrum in the range shown. Each copy is scaled by $f_s = 1$, leaving the magnitudes intact. In the third row, the 64 samples are shown convolved with a sinc function in the spatial domain. This corresponds to a rectangular pulse in the frequency domain. Since the sinc function is used here for image reconstruction, it

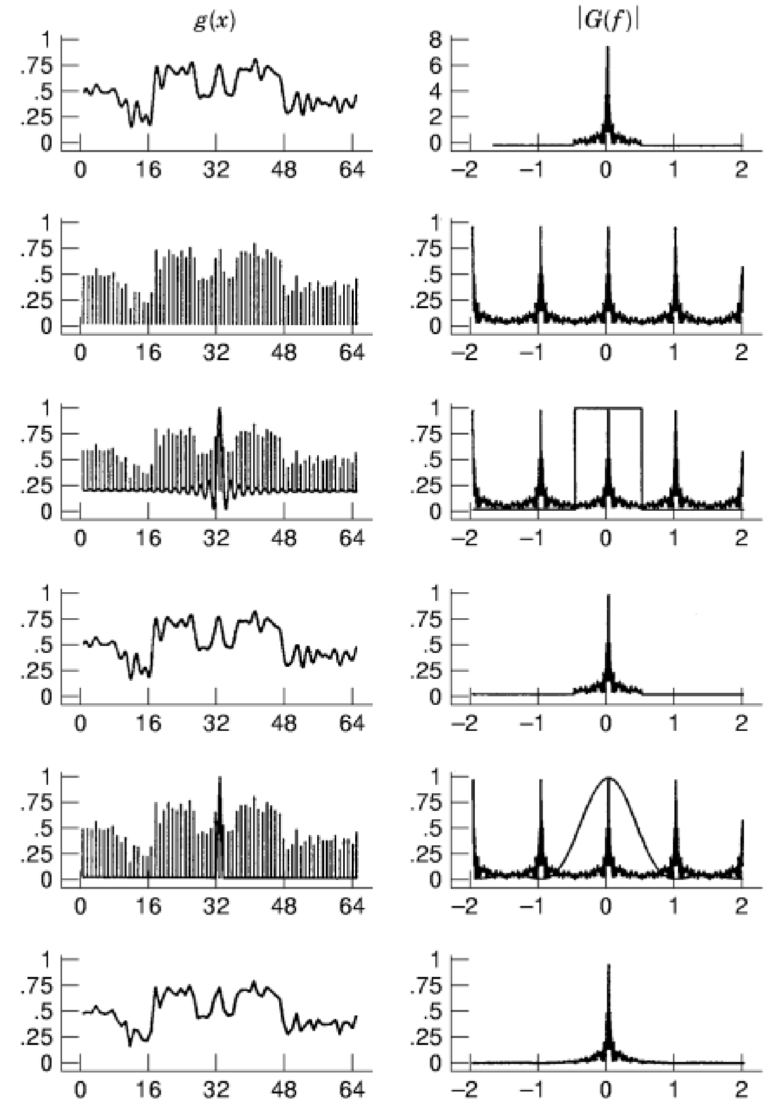

Fig. 25. Sampling and reconstruction (with an adequate sampling rate).

must have an amplitude of unity value in order to interpolate the data. This forces the height of the rectangular pulse in the frequency domain to vary in response to f_s.

A few comments on the reciprocal relationship between the spatial and frequency domains are in order here, particularly as they apply to the ideal low-pass filter. We again refer to the variables A and W as the sinc amplitude and bandwidth. As a sinc function is made broader, the value $\frac{1}{2}$ W is made to change since W is decreasing to accommodate zero crossings at larger intervals. Accordingly, broader sinc functions cause more blurring, and their spectra reflect this by reducing the cut-off frequency to some smaller W. Conversely, narrower sinc functions cause less blurring, and W takes on some larger value. In either case, the amplitude of the sinc function or its spectrum will change. That is, we can fix the amplitude of the sinc function so that

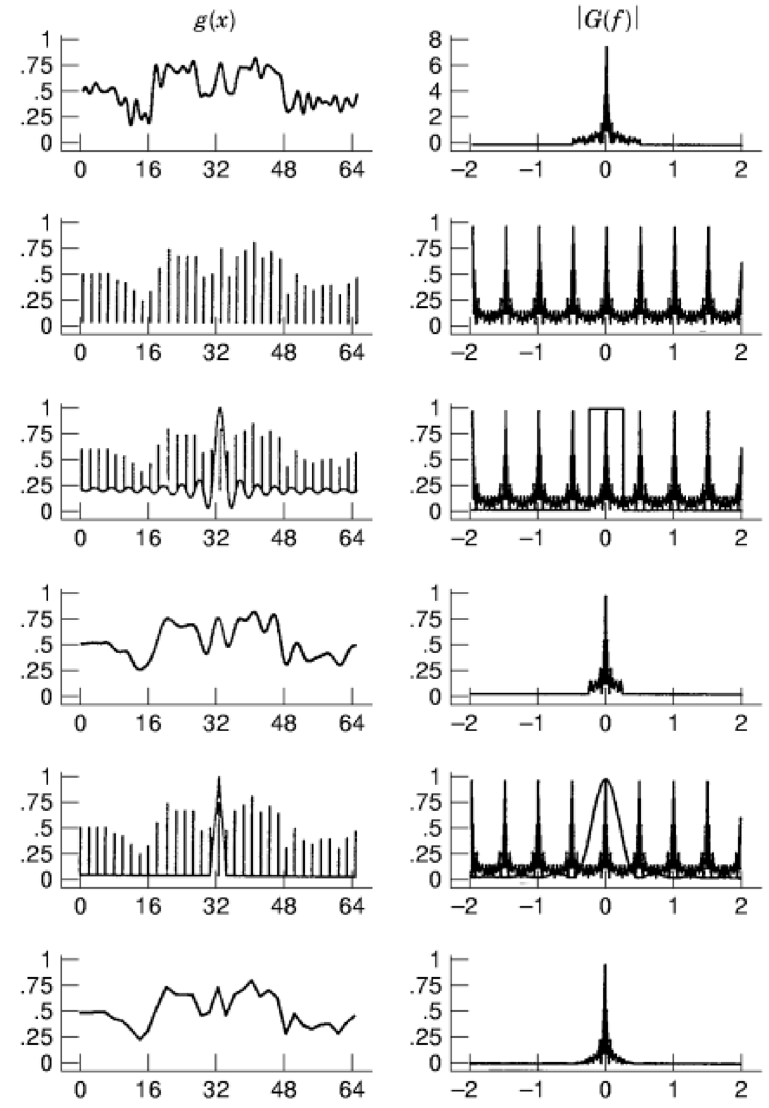

Fig. 26. Sampling and reconstruction (with an inadequate sampling rate).

only the rectangular pulse of the spectrum changes height $A/2W$ as W varies. Alternatively, we can fix $A/2W$ to remain constant as W changes, forcing us to vary A. The choice depends on the application.

When the sinc function is used to interpolate data, it is necessary to fix A to 1. Therefore, as the sampling density changes, the positions of the zero crossings shift, causing W to vary. This makes the amplitude of the spectrum's rectangular pulse change. On the other hand, if the sinc function is applied to bandlimit, not interpolate, the input signal, then it is important to fix $A/2W$ to 1 so that the passband frequencies remain intact. Since W is once again varying, A must change proportionately to keep $A/2W$ constant. Therefore, this application of the ideal low-pass filter requires the amplitude of the sinc function to be responsive to W.

In the examples presented below, our objective is to interpolate (reconstruct) the input, and so $A = 1$, regardless of the sampling density. Consequently, the height of the spectrum of the reconstruction filter changes.

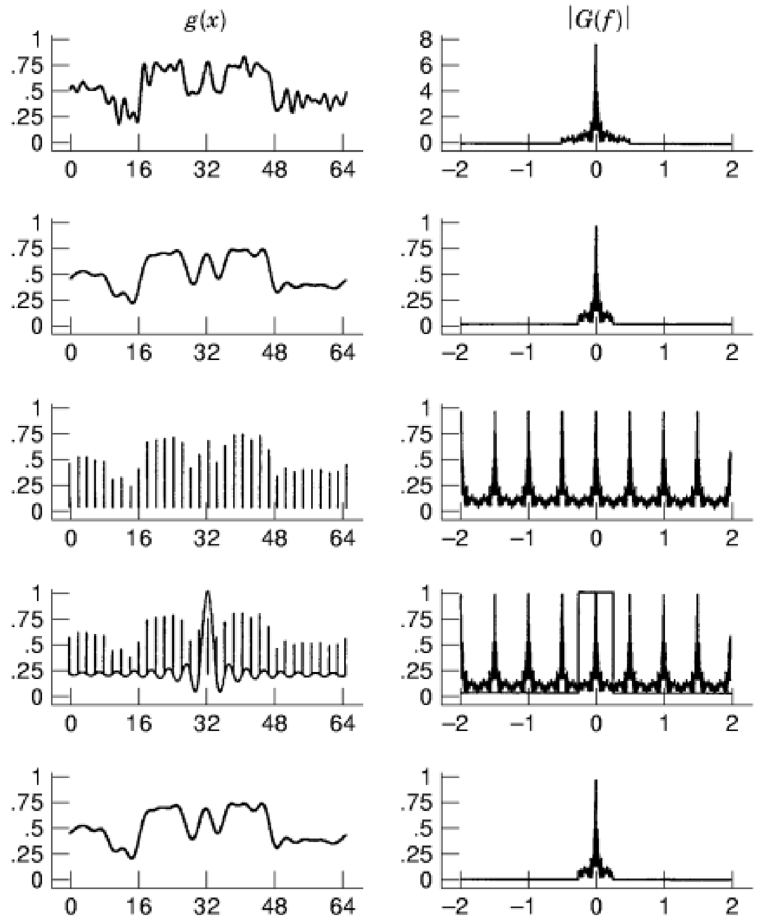

Fig. 27. Antialiasing filtering, sampling, and reconstruction stages.

To make the Fourier transforms of the filters easier to see, we have not drawn the frequency response of the reconstruction filters to scale. Therefore, the rectangular pulse function in the third row of Fig. 25 actually has height $A/2W = 1$. The fourth row of the figure shows the result after applying the ideal low-pass filter. As sampling theory predicts, the output is identical to the original signal. The last two rows of the figure illustrate the consequences of nonideal reconstruction filtering. Instead of using a sinc function, a triangle function corresponding to linear interpolation was applied. In the frequency domain, this corresponds to the square of the sinc function. Not surprisingly, the spectrum of the reconstructed signal suffers in both the passband and the stopband.

The identical sequence of filtering operations is performed in Fig. 26. In this figure, though, the sampling rate has been lowered to f_s, = .5, meaning that only one sample is collected for every two output pixels. Consequently, the replicated spectra are multiplied by .5, leaving the magnitudes at 4. Unfortunately, this sampling rate causes the replicated spectra to overlap. This, in turn, gives rise to aliasing, as depicted in the fourth row of the figure. Applying the triangle function to perform linear interpolation also yields poor results.

In order to combat these artifacts, the input signal must be bandlimited to accommodate the low sampling rate. This is shown in the second row of Fig. 27 where we see that all frequencies beyond $W = .25$ are truncated. This causes the input signal to be blurred. In this manner, we have traded aliasing for blurring, a far less objectionable artifact. Sampling this function no longer causes the replicated copies to overlap. Convolving with an ideal low-pass filter now properly isolates the bandlimited spectrum.

Glossary

Adaptive supersampling = Supersampling with samples distributed more densely in areas of high intensity variance.

Aliasing = Artifacts due to undersampling a signal. This condition prevents the signal from being reconstructed from its samples.

Antialiasing = The filtering necessary to combat aliasing. This generally requires bandlimiting the input before sampling to remove the offending high frequencies that will fold over in the frequency spectrum.

Area sampling = An antialiasing method that treats a pixel as an area, not a point. After projecting the pixel to the input, all samples in the preimage are averaged to compute a representative sample.

Bandlimit = The act of truncating all frequency components beyond some specified frequency. Useful for antialiasing, where offending high frequencies must be removed to prevent aliasing.

Frequency leakage = A condition in which the stopband is allowed to persist, permitting it to fold over into the passband range.

Gibbs phenomenon = Overshoots and undershoots caused by reconstructing a signal with truncated frequency components.

Nyquist rate = The minimum sampling frequency. It is twice the maximum signal frequency.

Passband = Consists of all frequencies that must be retained by the applied filter.

Point sampling = Each output pixel is a single sample of the input image. This approach generally leads to aliasing because a pixel is treated as a point, not an area.

Prefilter = The low-pass filter (blurring) applied to achieve antialiasing by bandlimiting the input image prior to resampling it onto the output grid.

Preimage = The projected area of an output pixel as it is mapped to the input image.

Pyramid = Multi-resolution data structures generated by successively bandlimiting and subsampling the original image to form a hierarchy of images at ever decreasing resolutions. Useful for accelerating antialiasing. Filtering limited to square regions and unweighted averaging.

Reconstruction = The act of recovering a continuous signal from its samples. Interpolation.

Stopband = Consists of all frequencies that must be suppressed by the applied filter.

Summed-area table = Preintegrated buffer of intensities generated by computed a running total of the input intensities as the image is scanned along successive scanlines. Useful for accelerating antialiasing. Filtering limited to rectangular regions and unweighted averaging.

Supersampling = An antialiasing method that collects more than one regularly-spaced sample per pixel.

BIBLIOGRAPHY

1. C. E. Shannon A Mathematical Theory of Communication, *Bell Syst. Tech. J.*, **27**: 379–423, July 1948; **27**: 623–656, October 1948.
2. C. E. Shannon Communication in the Presence of Noise, *Proc. Inst. Radio Eng.*, **37** (1): 10–21, 1949.
3. T. Whitted An Improved Illumination Model for Shaded Display, *Comm. ACM*, **23** (6): 343–349, 1980.
4. M. Lee R. A. Redner S. P. Uselton Statistically Optimized Sampling for Distributed Ray Tracing, *Comput. Graphics*, SIGGRAPH ’85 Proc., **19** (3): 1985, pp. 61–67.
5. S. J. Kajiya The Rendering Equation, *Comput. Graphics*, SIGGRAPH ’86 Proc., **20** (4): 1986, pp. 143–150.
6. D. P. Mitchell Generating Antialiased Images at Low Sampling Densities, *Comput. Graphics*, SIGGRAPH ’87 Proc., **21** (4): 1987, pp. 65–72.

7. T. Caelli *Visual Perception: Theory and Practice*, Oxford: Pergamon, 1981.
8. E. Catmull *A Subdivision Algorithm for Computer Display of Curved Surfaces*, Ph.D. Thesis, Dept. Comput. Sci., University of Utah, 1974.
9. W. Dungan, Jr. A. Stenger G. Sutty Texture Tile Considerations for Raster Graphics, *Comput. Graphics*, SIGGRAPH '78 Proc., **12** (3): 1978, pp. 130–134.
10. L. Williams Pyramidal Parametrics, *Comput. Graphics*, SIGGRAPH '83 Proc., **17** (3): 1983, pp. 1–11.
11. F. C. Crow Summed-Area Tables for Texture Mapping, *Comput. Graphics*, SIGGRAPH '84 Proc., **18** (3): 1984, pp. 207–212.
12. P. Heckbert *Texture Mapping Polygons in Perspective*, Tech. Memo No. 13, New York: NYIT Computer Graphics Lab, 1983.
13. A. Glassner Adaptive Precision in Texture Mapping, SIGGRAPH '86 Proc., **20** (4): 1986, pp. 297–306.
14. K. Perlin Course Notes, *SIGGRAPH '85 State of the Art in Image Synthesis Seminar Notes*, 1985.
15. P. Heckbert Filtering by Repeated Integration, *Comput. Graphics*, SIGGRAPH '86 Proc., **20** (4): 1986, pp. 315–321.
16. G. Wolberg *Digital Image Warping*, Los Alamitos, CA: IEEE Computer Society Press, 1990.
17. D. Schumacher General Filtered Image Rescaling, in David Kirk (ed.), *Graphics Gems III*, New York: Academic Press, 1992.
18. A. Glassner *Principles of Digital Image Synthesis*, San Francisco: Morgan Kaufmann, 1995.
19. R. Franke G. M. Nielson Scattered Data Interpolation and Applications: A Tutorial and Survey, in H. Hagen and D. Roller (eds.), *Geometric Modelling: Methods and Their Application*, Berlin: Springer-Verlag, 1991, pp. 131–160.
20. J. Hoschek D. Lasser *Computer Aided Geometric Design*, Wellesley, MA: AK Peters, 1993.
21. K. Castleman *Digital Image Processing*, Upper Saddle River, NJ: Prentice-Hall, 1996.
22. R. C. Gonzalez R. Woods *Digital Image Processing*, Reading, MA: Addison-Wesley, 1992.
23. A. K. Jain *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
24. W. K. Pratt *Digital Image Processing*, 2nd ed., New York: Wiley, 1991.
25. J. C. Russ *The Image Processing Handbook*, Boca Raton, FL: CRC Press, 1992.

READING LIST

The material contained in this chapter was drawn from (16). Additional image processing texts that offer a comprehensive treatment of sampling, reconstruction, and antialiasing include (18), (21), (22), (23), (24), and (25).

Advances in the field are reported in several journals, including *IEEE Transactions on Image Processing, IEEE Transactions on Signal Processing, IEEE Transactions on Acoustics, Speech*, and *Signal Processing*, and *Graphical Models and Image Processing.* Related work in computer graphics is also reported in *Computer Graphics* (ACM SIGGRAPH Proceedings), *IEEE Computer Graphics and Applications*, and *IEEE Transactions on Visualization and Computer Graphics.*

GEORGE WOLBERG
City College of New York/CUNY