7.1

Image Scanning, Sampling, and Interpolation

Jan P. Allebach
Purdue University

1 Image Capture

Image capture takes us from the continuous-parameter real world in which we live to the discrete-parameter, amplitude quantized domain of the digital devices that comprise an electronic imaging system. The process of converting from a continuous-parameter image to one that is discrete-parameter i.e., consists of an array of numbers, is referred to as *sampling*. The meaning of the term *scanning* is somewhat less precise. Its common usage refers to the notion of sequential acquisition of data through some type of electro-mechanical motion. It is also used to refer to the process of converting a two-dimensional signal to a signal that is one-dimensional. The process of quantizing an image that is continuous in amplitude to one that takes on values from a finite set is called *quantization*. Examples illustrating the effect of quantization may be found in Chapter 1.1.

1.1 Representations for the Sampled Image

Sampling a continuous-space image $g_c(x, y)$ yields a discrete-space image

$$g_d(m, n) = g_c(mX, nY), \tag{1}$$

where the subscripts c, d denote, respectively, continuous-space and discrete-space, and X is the spacing between sample points, also called the pitch. However, it is also convenient to represent the sampling process using the 2D Dirac delta function $\delta(x, y)$. In particular, we have from the sifting property of the delta function that multiplication of $g_c(x, y)$ by a delta function centered at the fixed point (x_0, y_0) followed by integration will yield the sample value $g_c(x_0, y_0)$, i.e.,

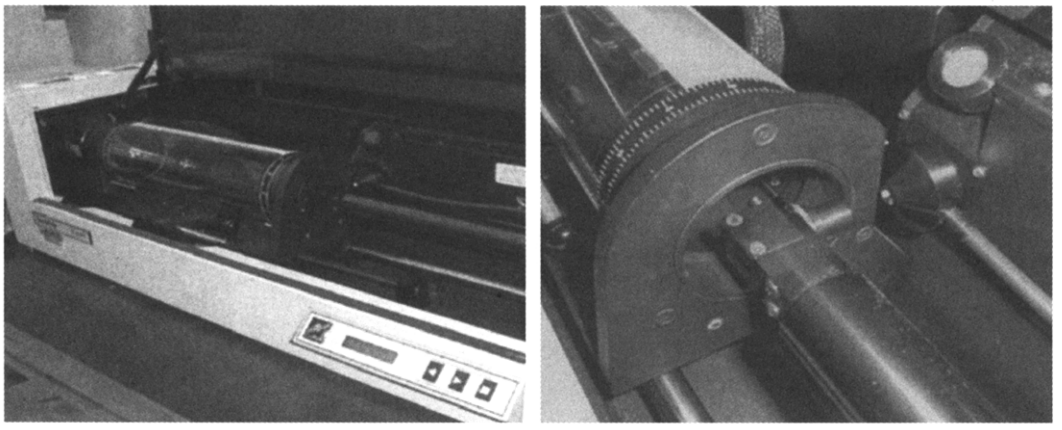
$$g_c(x_0, y_0) = \iint g_c(x, y)\delta(x - x_0, y - y_0)\, dxdy, \tag{2}$$

provided $g_c(x, y)$ is continuous at (x_0, y_0). It follows that

$$g_c(x, y)\delta(x - x_0, y - y_0) \equiv g_c(x_0, y_0)\delta(x - x_0, y - y_0), \tag{3}$$

that is, multiplication of an impulse centered at (x_0, y_0) by the continous-space image $g_c(x, y)$ is equivalent to multiplication of the impulse by the constant $g_c(x_0, y_0)$. It will also be useful to note from the sifting property that

$$g_c(x, y) * \delta(x - x_0, y - y_0) = g_c(x - x_0, y - y_0), \tag{4}$$



(a) (b)

FIGURE 1 High resolution drum scanner (a) scanner with cover open, and (b) closeup view showing screw-mounted "C" carriage with light source on inside arm, and detector optics on outside arm. (See color insert.)

that is, convolution of a continous-space function with an impulse located at (x_0, y_0) shifts the function to (x_0, y_0).

To get all the samples of the image, we define the comb function

$$\text{comb}_{X,Y}(x,y) = \sum_m \sum_n \delta(x - mX, y - nY). \tag{5}$$

Then we define the continuous-parameter sampled image, denoted with the subscript s, as

$$g_s(x,y) = g_c(x,y)\text{comb}_{X,Y}(x,y), \tag{6}$$

$$= \sum_m \sum_n g_d(m,n)\delta(x - mX, y - nY). \tag{7}$$

We see from (7) that the continuous- and discrete-space representations for the sampled image contain the same information about its sample values. In the sequel, we shall only use the subscripts c and d when necessary to provide additional clarity. In general, we can distinguish between functions that are continuous-space and those that are discrete-space on the basis of their arguments. We will usually denote continuous-space independent variables by (x, y) and discrete-space independent variables by (m, n).

1.2 Image Capture Technologies

There are two fundamental aspects of image capture. The first is the *raster* of points in two-or three-dimensional space where samples are taken. The second is the effect of the system *aperture,* which causes the data samples to consist of an average of the image or scene within a neighborhood of the nominal sampling point.

Devices for image capture may be divided into two classes, according to the mechanism by which the samples are acquired. The first class utilizes a *flying spot* mechanism for data acquisition. Examples of such mechanisms include an electron beam, as is used in an analog video camera, the electromechanical scan resulting from rotation of a drum and movement of a screw, as can be found in graphic arts drum scanners (see Fig. 1), diffractive optical beam formation, as is used in supermarket point-of-sale scanners, and phased array beam formation, as is used with radar. With all these systems, the spot trajectory and read times determine the sampling raster, whereas the aperture effects are governed by the shape of the illuminating and read spots and the dwell time, i.e., the time interval during which the read spot output signal is averaged to form a sample value. Although none of the examples cited above operate in this manner, flying spot scanners can also function in a passive mode. In this case, there is no write spot; and the read spot detects radiation emanating naturally from the scene. Air- and space-borne systems for remote sensing of the earth's surface are examples of passively scanning systems.

The second class of image capture devices utilizes a *focal plane mosaic,* which consists of an array of detector sites. The scene is imaged onto the surface of the array; and each detector integrates the radiation gathered from the active area of its surface. This gives rise to the aperture effect. The spatial arrangement of the detectors determines the sampling raster. Focal plane array technologies include charge coupled devices (CCDs), charge injection devices (CIDs), and CMOS devices. These technologies are widely used in digital still and video cameras. Some systems comprise a hybrid of the flying spot and focal plane mosaic architectures. The flatbed scanner which uses a mechanical means to move a one-dimensional array of detectors across the surface of the document being scanned is a good example.

1.3 General Model for the Image Capture Process

Despite the diversity of technologies and architectures for, image capture devices, it is possible to cast the sampling process for all of these systems within a common framework. We will illustrate this fact for two examples. The first example is that of a flying spot scanner. Since this device acquires data *time-sequentially*, i.e., one sample at a time, we can represent the scanned signal as a function of the single time parameter t. Accordingly, the operation of this device is described by

$$s(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_i[\xi - x_s(t),\ \eta - y_s(t)]\, p_r[\xi - x_s(t),\ \eta - y_s(t)]\, g(\xi,\eta)\, d\xi d\eta. \tag{8}$$

Here $s(t)$ is the continuous-time signal generated at the detector output, prior to A/D conversion; $p_i(x, y)$ and $p_r(x, y)$ are the illuminating and read spots, respectively; $[x_s(t), y_s(t)]$ is the trajectory of these spots across the image as a function of time; and $g(x, y)$ is the image to be sampled. What this equation shows is that at any time t, the detector output is given by an integral over the entire image $g(x, y)$, weighted by the spatially varying intensity of the illuminating spot and the spatially varying sensitivity of the detector (read spot), which are both centered at the trajectory coordinates $[x_s(t), y_s(t)]$ at that time. The final step in the sampling process is to sample the detector output at an appropriate set of times, yielding $s_d(k) = s(t_k)$, where again the subscript d denotes the fact that $s_d(k)$ is a discrete-time signal, and t_k is the sequence of sampling times, which are not necessarily uniformly spaced.

The scanning trajectory $[x_s(t), y_s(t)]$ and the set of sampling times t_k combine to determine the set of spatial points (x_k, y_k) at which samples are acquired. We shall represent each such sampling point by a 2D Dirac delta function $\delta(x - x_k, y - y_k)$, and the entire set of sampling points by the sampling function

$$q(x, y) = \sum_k \delta(x - x_k, y - y_k). \tag{9}$$

Because the image is not time-varying, the order in which the samples are acquired is immaterial to the characteristics of the sampled 2D signal.

Since the illuminating and read spot functions have the same arguments, they may be combined as a single function $p(x, y) = p_i(-x, -y)p_r(-x, -y)$ which accounts for all aperture effects due to the flying spot scanning process. We have reflected the coordinates simply for mathematic convenience. In addition, the averaging effect of the aperture may be represented as a 2D convolution of the continuous-parameter image $g(x, y)$ with the aperture function; so the continous-parameter representation of the sampled image is thus given by

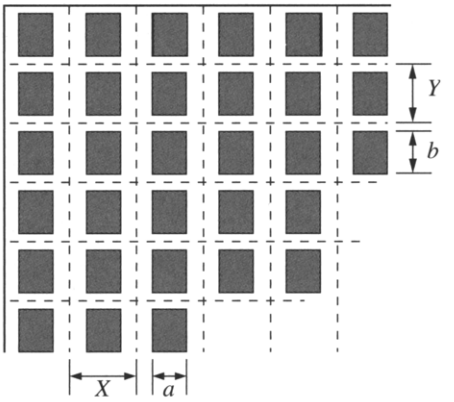


FIGURE 2 Focal plane array geometry.

$$g_s(x, y) = q(x, y)[p(x, y) * g(x, y)]. \tag{10}$$

With the appropriate choice of sampling times t_k for $s(t)$ in (8) and sampling points (x_k, y_k) for $q(x, y)$ in (10), these two representations for the sampled image are completely equivalent.

The second example that we wish to consider is that of a 2D focal plane array illustrated in Fig. 2. Here each sample is obtained by integrating over the active area of the corresponding detector; so we have

$$g_d(m, n) = \int_{mX-a/2}^{mX+a/2}\int_{nY+b/2}^{nY+b/2} g(\xi,\eta)d\xi d\eta, \tag{11}$$

where as before the spacing between sample points is $X \times Y$ and the size of the active area of each detector is $a \times b$. The averaging effect of the active area of the detector can again be accounted for by convolution with an appropriately chosen aperture function, in this case

$$p(x, y) = \text{rect}(x/a, y/b), \tag{12}$$

where $\text{rect}(x, y)$ is defined to be 1 if $|x| < 1/2$ and $|y| < 1/2$, and 0, otherwise. The sampling function is given by

$$q(x, y) = \text{comb}_{X,Y}(x, y). \tag{13}$$

With $p(x, y)$ given by (12) and $q(x, y)$ given by (13), (10) is completely equivalent to (11).

To summarize, the sampling process for a broad group of image capture devices may be modeled as a convolution with an appropriately chosen aperture function $p(x, y)$ followed by multiplication by an appropriate sampling function $q(x, y)$.

2 Fourier Analysis of Image Capture

Fourier analysis sheds a great deal of light on the effect of the sampling process. However, before we get to that, it will be helpful to first define the different spectral representations that we will be using.

2.1 Spectral Representations for Discrete and Continuous-Space Signals

For continuous-space images, the appropriate spectral representation is the continuous-space Fourier transform (CSFT). The forward and inverse versions of this transform are given respectively by

$$G_c(u,v) = \iint g_c(x,y)e^{-i2\pi(ux+vy)}dxdy. \tag{14}$$

$$g_c(x,y) = \iint G_c(u,v)e^{i2\pi(ux+vy)}dudv. \tag{15}$$

The units of frequency for (u, v) are cycles/unit distance. For discrete-space images, we use the discrete-space Fourier transform (DSFT) defined as

$$G_d(U,V) = \sum_m \sum_n g_d(m,n)e^{-i2\pi(Um+Vn)}. \tag{16}$$

$$g_d(m,n) = \int_{-0.5}^{0.5}\int_{-0.5}^{0.5} G_d(U,V)e^{i2\pi(Um+Vn)}dUdV. \tag{17}$$

The units of frequency for (U, V) are cycles/pixel. Again, we shall use the subscripts c and d only where needed for clarity.

In Section 1.1, we defined both continous-parameter and discrete-parameter representations for the sampled signal. To examine the spectral form of the continuous-parameter representation (6), we first note that

$$\mathrm{comb}_{X,Y}(x,y) \overset{CSFT}{\longleftrightarrow} \frac{1}{XY}\mathrm{comb}_{\frac{1}{X},\frac{1}{Y}}(u,v). \tag{18}$$

Then by the convolution theorem, we have that the CSFT of (6) is given by

$$G_s(u,v) = G_c(u,v) * \frac{1}{XY}\mathrm{comb}_{\frac{1}{X},\frac{1}{Y}}(u,v). \tag{19}$$

It follows directly from the definition of the comb function (5) and the convolution property of the impulse (4) that

$$G_s(u,v) = \frac{1}{XY}\sum_k \sum_l G_c(u-\frac{k}{X}, v-\frac{l}{Y}). \tag{20}$$

So sampling a continuous-space function on a lattice with interval (X, Y) causes the CSFT of that function to be replicated in the frequency domain on a lattice with interval $(1/X, 1/Y)$, and scaled overall by $1/XY$.

To relate this result to the DSFT of $g_d(m, n)$, we take the CSFT of (7) directly. Interchanging the summation over the terms in the comb function with the Fourier integral, and using the sifting property (2), we obtain

$$G_s(u,v) = \sum_m \sum_n g_d(m,n)e^{-i2\pi(umX+vnY)}. \tag{21}$$

Comparing this to (16), we see that (21) can be put in the form of the DSFT of $g_d(m, n)$ with an appropriate change of frequency variables. Thus

$$G_s(u,v) = G_d(u/\mathcal{U}, v/\mathcal{V}), \tag{22}$$

where $\mathcal{U}=1/X$ and $\mathcal{V}=1/Y$ are the sampling frequencies in the horizontal and vertical directions in units of cycles/unit distance. Thus we see that there is a simple and direct relation between the CSFT of the continuous-space representation of the sampled image and the DSFT of the discrete-space representation of that image combining (20) and (22), we obtain

$$G_d(U,V) = \mathcal{U}\mathcal{V}\sum_k \sum_l G_c((U-k)\mathcal{U}, (V-l)\mathcal{V}). \tag{23}$$

2.2 The General Image Capture Model Revisited

We are now ready to examine our general model for image capture from a frequency domain perspective. Taking the CSFT of (10) and using the convolution and product theorems, we obtain

$$G_s(u,v) = Q(u,v) * [P(u,v)G(u,v)]. \tag{24}$$

So we see that the spectrum of the sampled image is obtained by multiplying the spectrum of the continuous-space image by the CSFT $P(u, v)$ of the aperture function $p(x, y)$, and then convolving with the CSFT $Q(u, v)$ of the sampling function $q(x, y)$. Let us denote the effect of multiplication by the CSFT of the aperture with a tilde

$$\tilde{G}(u,v) = P(u,v)G(u,v). \tag{25}$$

For the special case where $q(x, y) = \mathrm{comb}_{X,Y}(x, y)$, we then have from (20) that

$$G_s(u,v) = \mathcal{U}\mathcal{V}\sum_k \sum_l \tilde{G}(u-k\mathcal{U}, v-l\mathcal{V}). \tag{26}$$

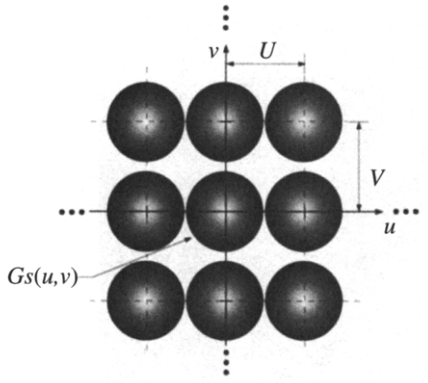


FIGURE 3 Spectrum of sampled image.

Figure 3 illustrates this result. Let us first assume that there is no aperture effect; so $P(u, v) \equiv 1$ and $\tilde{G}(u, v) \equiv G(u, v)$. We see that a sufficient condition for the spectral replications to not overlap is that

$$G(u, v) \neq 0 \text{ only if } |u| < \mathcal{U}/2 \text{ and } |v| < \mathcal{V}/2. \tag{27}$$

This is referred to as the Nyquist condition. Since $1/X = \mathcal{U}$ and $1/Y = \mathcal{V}$, this condition has the interpretation that we must sample at least twice per cycle of the highest horizontal and vertical frequencies found in the image. Provided the Nyquist condition is satisfied, we see that $G(u, v)$ may be recovered from $G_s(u, v)$ by multiplication with a scaled 2D rect function.

$$G(u, v) = \frac{1}{\mathcal{U}\mathcal{V}} \text{rect}(u/\mathcal{U}, v/\mathcal{V}) G_s(u, v). \tag{28}$$

Using the product theorem and the scaling property, we obtain in the spatial domain

$$g(x, y) = \text{sinc}(x/X, y/Y) * g_s(x, y), \tag{29}$$

which using (7) can be expressed as

$$g(x, y) = \sum_m \sum_n g(mX, nY) \text{sinc}(x - mX, y - nY), \tag{30}$$

where $\text{sinc}(x, y) \equiv \sin(\pi x) \sin(\pi y)$. This is the Whitaker-Kotelnikov-Shannon sampling expansion, which shows that we can reconstruct an appropriately bandlimited image by interpolating between samples with a sinc function. This result is commonly known as the 2D sampling theorem.

When the Nyquist condition is not satisfied, any frequency component in the continuous-parameter image that lies outside the region

$$\Omega_{\mathcal{U},\mathcal{V}}(u, v) = \{(u, v) : |u| < \mathcal{U}/2, |v| < \mathcal{V}/2\}, \tag{31}$$

will fold back into $\Omega_{\mathcal{U},\mathcal{V}}$, thus mimicking a lower frequency. Figure 4 illustrates this for the case of a simple 2D sinusoid. This phenomenon is known as *aliasing*. In images reconstructed from undersampled data, it manifests itself as moire patterns and staircasing or "jaggies" along straight edges.

Figure 5 illustrates the effect of undersampling a real image. At the top of Fig. 5a, we see a jagged edge along the crest of the dune. In addition, close inspection of the ripples in the sand reveals what appear to be fine lines oriented at 90° to the

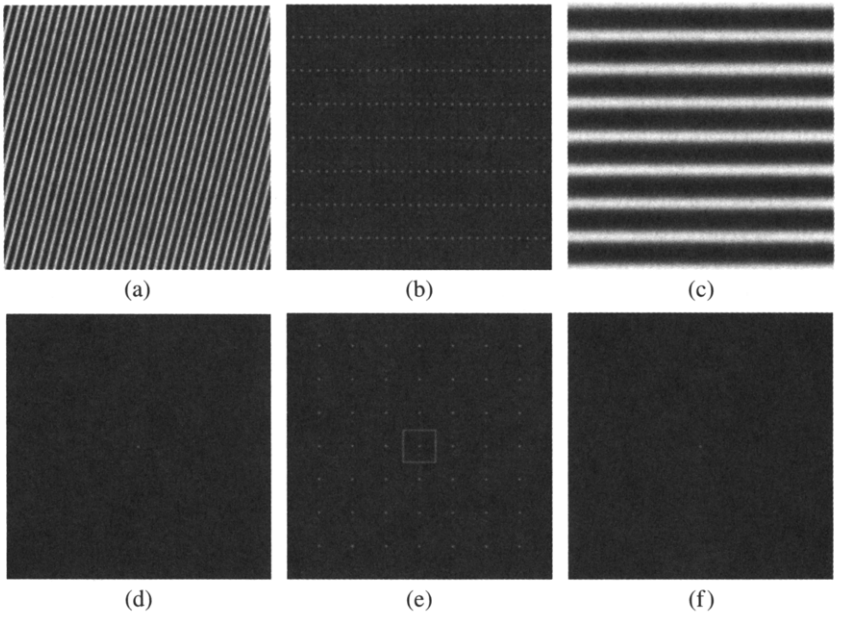
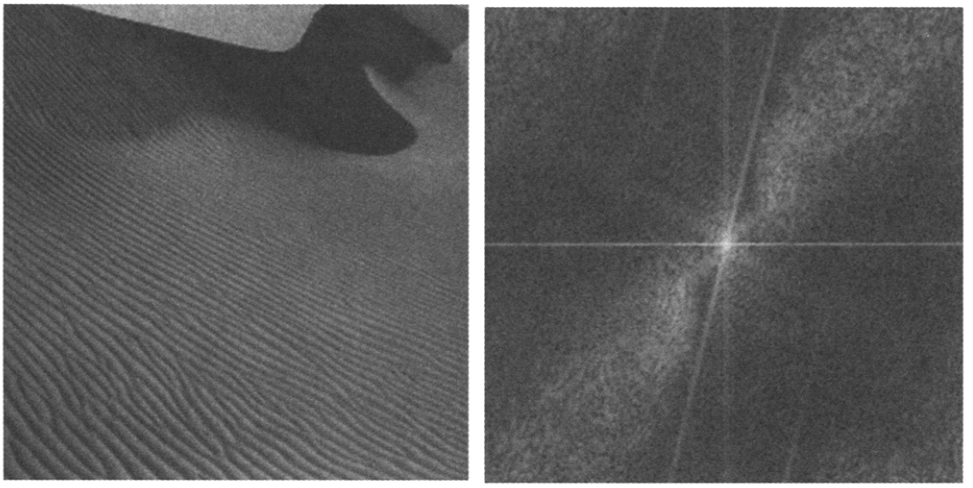


FIGURE 4 Effect of undersampling a 2D sinusoid: (a) original sinewave with DC offset to make it non-negative, (b) sampled sinewave (a), (c) reconstruction obtaining by bandlimiting (b) to Nyquist limit, (d)–(f) spectra of (a)–(c), respectively. Square in (e) indicates frequencies below Nyquist limit.



(a) (b)

FIGURE 5 Effect of undersampling an image of a sand dune: (a) undersampled image, and (b) the magnitude of its Fourier transform.

ripples. Both these artifacts are due to undersampling. In Fig. 5b, we see that the energy in the Fourier transform oriented along a fine line at about 75° from the positive U axis has folded back, creating short diagonal line segments in the second and fourth quadrants. This spectral component corresponds to the edge of the crest. In addition, the more diffuse cloud of energy oriented at 45° to the positive U axis has folded back, creating clouds in the upper left and lower right corners of the spectrum. This spectral component corresponds to the ripples in the sand.

The Nyquist condition may be stated more generally in a necessary and sufficient form: If and only if the support of $G(u, v)$ does not exceed an area of size $\mathcal{UV}$, $g(x, y)$ may be reconstructed from its samples taken on a rectangular lattice at interval $(1/\mathcal{U}, 1/\mathcal{V})$. The interpolating function will be the inverse CSFT of the indicator function for the support region, scaled by $1/\mathcal{UV}$.

Now let's consider the effect of the aperture indicated by (25). If the CSFT $P(u, v)$ of the aperture rolls off at frequencies outside $\Omega_{U,V}$, the aperture will attenuate any frequencies in the continuous-parameter image $g(x, y)$ outside $\Omega_{\mathcal{U},\mathcal{V}}$, thereby suppressing aliasing. This desirable effect is known as *prescan bandlimitation* or *antialiasing*. On the other hand, if $P(u, v)$ rolls off, i.e., $|P(u, v)| < 1$, for frequencies $(u, v) \in \Omega_{\mathcal{U},\mathcal{V}}$, then the aperture will have the undesired effect of attenuating frequencies in $g(x, y)$ that are not undersampled. Typically, it is the higher frequencies in the image that are attenuated in this manner, resulting in an image that looks soft or slightly blurred after reconstruction using (28) or (30). Provided $|P(u, v)| > 0$ for all frequencies $(u, v) \in \Omega_{\mathcal{U},\mathcal{V}}$, this effect may be compensated by replacing (28) with

$$G(u, v) = \frac{1}{UV} rect(u/\mathcal{U}, v/\mathcal{V}) [P(u, v)]^{-1} G_s(u, v). \quad (32)$$

Of course, at frequencies within $\Omega_{\mathcal{U},\mathcal{V}}$ where $|P(u, v)|$ is small, this reconstruction procedure will amplify any noise present in the sampled data.

2.3 Sampling with Nonrectangular Lattices

We saw in the preceding section that sampling an image on a rectangular lattice with interval $X \times Y$ causes replication of the image spectrum on a reciprocal lattice that is also rectangular, and which has interval $1/X \times 1/Y$. To prevent aliasing, these replications must be spaced far enough apart to prevent overlap. For an image bandlimited to a circular bandregion with highest frequency W, we must have $1/X > 2W$ and $1/Y > 2W$. The minimum sampling density is given by

$$d_R = \frac{1}{XY} = 4W^2 \text{ samples/unit area.} \quad (33)$$

Figure 3 shows a situation where the sampling in the vertical direction slightly exceeds the Nyquist rate. However, even if the sampling were at the Nyquist rate, the spectral replications would not completely cover the frequency domain. This suggests that it may be possible to use a different lattice that will more tightly pack the spectra in the frequency domain, resulting in a spreading of samples on the reciprocal lattice in the spatial domain, and hence a lower sampling density.

It is well known that the lattice which most tightly packs circles is hexagonal. Figure 6 shows the corresponding spatial lattice. Each sample point has 6 equidistant neighbors which are all separated from it by angles of 60°. To determine the reciprocal lattice for this sampling structure, we represent it using two interlaced rectangular lattices with the same period, as indicated in Fig. 6; so

$$q(x, y) = \text{comb}_{X,Y}(x, y) + \text{comb}_{X,Y}(x - X/2, y - Y/2), \quad (34)$$

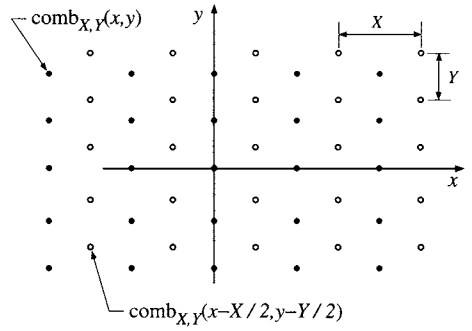


FIGURE 6 Hexagonal sampling lattice represented as superposition of two interleaved rectangular lattices.

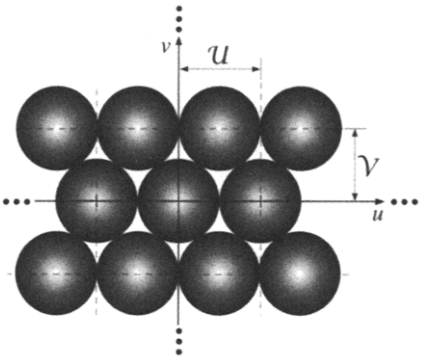


FIGURE 7 Spectrum of image sampled on a hexagonal lattice.

where $X=1/(2W)$ and $Y=1/(\sqrt{3}W)$. To determine the corresponding reciprocal lattice, we calculate the CSFT of (34), using the shifting property of the Fourier transform to yield

$$Q(u,v)=\frac{1}{XY}\mathrm{comb}_{\frac{1}{X},\frac{1}{Y}}(u,v)+\frac{1}{XY}\mathrm{comb}_{\frac{1}{X},\frac{1}{Y}}(u,v)e^{-i\pi(Xu+Yv)},$$

$$=\frac{1}{XY}\sum_k\sum_l\left(1+e^{-i\pi(m+n)}\right)\delta\left(u-\frac{k}{X},v-\frac{l}{Y}\right). \tag{35}$$

Because

$$\left(1+e^{-i\pi(m+n)}\right)=\begin{cases}2, & m+n \text{ even},\\ 0, & m+n \text{ odd},\end{cases} \tag{36}$$

the reciprocal lattice is also hexagonal. So the spectrum

$$G_s(u,v)=\mathcal{U}\mathcal{V}\sum_k\sum_l\left(1+e^{-i\pi(m+n)}\right)\tilde{G}(u-k\mathcal{U},v-l\mathcal{V}), \tag{37}$$

of the sampled image appears as shown in Fig. 7. Here, $\mathcal{U}=1/X$ and $\mathcal{V}=1/Y$, as before. Now the sampling density is

$$d_H=\frac{2}{XY}=4\sqrt{3}W^2, \tag{38}$$

The savings is $d_H/d_R=\sqrt{3}/2=0.866$, or 13.4%.

The hexagonal lattice is only one example of a non-rectangular lattice. Such lattices can be treated in a more general context of lattice theory. This framework is developed in Chapter 7.2.

3 Sampling Rate Conversion

In some instances, it is desirable to change the sampling rate of a digital image. This section addresses the procedures for doing this, and the effect of sampling rate changes.

3.1 Downsampling and Decimation

To decrease the sampling rate of a digital image $f(m,n)$ by integer factors of $C\times D$, we can *downsample* it using

$$g_{\downarrow}(m,n)=f(C_m,D_n). \tag{39}$$

So we simply discard all but every C-th sample in the m direction and all but every D-th sample in the n direction. To understand the effect of this operation, we derive an expression for the DSFT of $g_{\downarrow}(m,n)$ in terms of that of $f(m,n)$. By definition (16),

$$G_{\downarrow}(U,V)=\sum_m\sum_n f(Cm,Dn)e^{-i2\pi(Um+Vn)}. \tag{40}$$

With a change of indices of summation, we can write

$$G_{\downarrow}(U,V)=\sum_m\sum_n s_C(m)s_D(n)f(m,n)e^{-i2\pi(Um/C+Vn/D)}, \tag{41}$$

where

$$s_C(m)=\begin{cases}1, & m/C \text{ is an integer},\\ 0, & \text{else}.\end{cases} \tag{42}.$$

Since $s_C(m)$ is periodic with period $_C$, it can be expressed as a discrete Fourier series. Within one period, $sC(m)$ consists of a single impulse; so the Fourier coefficients all have value unity. Thus we can write

$$s_C(m)=\frac{1}{C}\sum_{k=0}^{C-1}e^{-i2\pi(mk/C)}. \tag{43}$$

Substituting this into (41), and interchanging orders of summation, we obtain

$$G_{\downarrow}(U,V)=\frac{1}{CD}\sum_{k=0}^{C-1}\sum_{l=0}^{D-1}\sum_m\sum_n f(m,n)e^{-i2\pi[(U-k)m/C+(V-l)n/D]}, \tag{44}$$

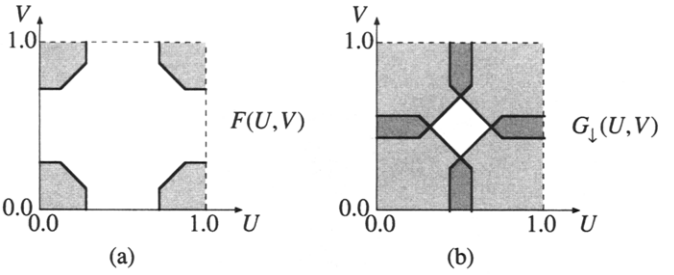


FIGURE 8 Effect of downsampling by factor 2×2 on DSFT of image, (a) before downsampling, (b) after downsampling.

which may be recognized as

$$G_{\downarrow}(U,V) = \frac{1}{CD}\sum_{k=0}^{C-1}\sum_{l=0}^{D-1} F((U-k)/C, (V-l)/D). \quad (45)$$

So we see that downsampling the image causes the DSFT to be expanded by a factor of C in the U direction and a factor of D in the V direction. This is a consequence of the fact that the image has contracted by these same factors in the spatial domain. The DSFT $G(U, V)$ is comprised of a summation of CD replications of the expanded DSFT $F(U/C, V/D)$ shifted by unit intervals in both the U and V directions. Figure 8 illustrates he overall result. Here the downsampling has resulted in overlap of the spectral replications of $F(U/C, V/D)$, thus resulting in aliasing.

The most important consequence of downsampling is the potential for additional aliasing, which will occur if $F(U, V) \neq 0$ for any $|U| \geq 1/(2C)$ or $|V| \geq 1/(2D)$. To prevent this, we can prefilter $f(m, n)$ prior to downsampling with a filter having frequency response

$$H(U,V) = CD\,\mathrm{rect}(CU, DV). \quad (46)$$

The impulse response corresponding to this filter is

$$h(m,n) = \mathrm{sinc}(m/C, n/D), \quad (47)$$

Because of the large negative sidelobes and slow roll-off of the filter, it can result in undesirable ringing at edges when it is truncated to finite extent. This is known as the *Gibb's phenomenon*. It can be avoided by tapering the filter with a window function. In practice, it is common to simply average the image samples within each $C \times D$ cell, or to use a Gaussian filter. The combination of a filter followed by a downsampler is called a *decimator*. Figure 9 shows the block diagram of such a system. Its net effect is

$$g(m,n) = \sum_k \sum_l f(k,l)h(Cm-k, Dn-l). \quad (48)$$

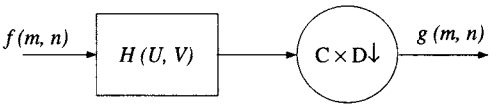


FIGURE 9 Decimator.

Here we have dropped the down-arrow subscript to denote the fact that we are not just downsampling, but rather are filtering first.

3.2 Upsampling and Interpolation

To understand how we increase the sampling rate of an image $f(m, n)$ by integer factors C in the m direction and D in the n direction, it is helpful to start with an *upsampler* which inserts $C-1$ zeros between each sample in the m direction and $D-1$ zeros between each sample in the n direction

$$g_{\uparrow}(m,n) = \begin{cases} f(m,n), & m/C \text{ and } n/D \text{ are integers,} \\ 0, & \text{else.} \end{cases} \quad (49).$$

We again seek an expression for the DSFT of $g_{\uparrow}(m, n)$ in terms of that of $f(m, n)$. Applying the definition of the DSFT, we can write

$$G_{\uparrow}(U,V) = \sum_m \sum_n s_C(m)s_D(n)f(m/C, n/D)e^{-i2\pi(Um+Un)}, \quad (50)$$

which after a change of variable becomes

$$G_{\uparrow}(U,V) = \sum_m \sum_n f(m,n)e^{-i2\pi(UCm+UDn)}, \quad (51)$$

$$= F(CU, DV). \quad (52)$$

Thus upsampling contracts the spectrum by a factor of C in the U direction and D in the V direction. There is no aliasing, because no information has been lost. The DSFT $G_{\uparrow}(U, V)$ is periodic with period $1/C \times 1/D$. To generate an image that is oversampled by a factor of $C \times D$, we need to filter out all but the baseband replication, again using the ideal low pass filter of (46). The combination of an upsampler followed by a filter is called an *interpolator*. Figure 10 shows the block diagram of such a system. Its net effect is given by

$$g(m,n) = \sum_k \sum_l f(k,l)h(m-Ck, n-Dl). \quad (53)$$

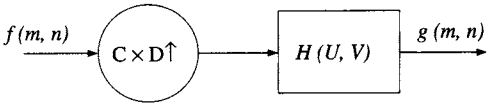


FIGURE 10 Interpolator.

For the special case of the ideal lowpass filter,

$$g(m,n) = \sum_k \sum_l f(k,l)\text{sinc}(m/C - k, n/D - l). \quad (54)$$

In the frequency domain, the interpolator is described by

$$G(U,V) = H(U,V)F(CU,DV). \quad (55)$$

For reasons similar to those discussed in the context of decimation as well as undesirably large computational requirements, the sinc filter is not widely used for image interpolation. In the following section, we examine some alternative approaches.

4 Image Interpolation

Both decimation and interpolation are fundamental image processing operations. As the resolution of desktop printers grows, interpolation is increasingly needed to scale images up to the resolution of the printer prior to *halftoning*, which is discussed in detail in Chapter 8.1. In addition to the quality of the interpolated image, the effort required to compute it is a very important consideration in these applications. We will use the theory developed in the preceding section as the basis for describing a variety of methods that can be used for image interpolation. We shall assume that the image is to be enlarged by integer factors $C \times D$, where for convenience, we assume that both C and D are even. We begin with several methods that can be modeled as an upsampler followed by a linear filter, as shown in Fig. 10.

4.1 Linear Filtering Approaches

At the lowest level of computational complexity, we have *pixel replication* also known as *nearest neighbor interpolation* or *zero-order interpolation*, which is widely used in many applications. In this case,

$$g(m,n) = f(\lfloor m/C \rceil, \lfloor n/D \rceil) \quad (56)$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. The corresponding filter in the interpolator structure shown in Fig. 10 is given by

$$h_0(m,n) = \begin{cases} 1/(CD), & -C/2 \le m < C/2, \ -D/2 \le n < D/2, \\ 0, & \text{else}, \end{cases} \quad (57)$$

where the subscript denotes the order of the interpolation. Pixel replication yields images that appear blocky, as shown in Fig. 11. Looking at (55) and the frequency response

$$H_0(U,V) = \frac{e^{i\pi(U+V)}}{CD}\left[\frac{\sin(\pi UC)}{\sin(\pi U)}\right]\left[\frac{\sin(\pi VD)}{\sin(\pi V)}\right] \quad (58)$$

of the filter, we see that this is a consequence of the fact that the filter does not effectively block the replications of $F(CU, DV)$ outside the region $\Omega_{1/C,1/D}(U,V)$, as shown in Fig. 12.

To obtain a smoother result, we can linearly interpolate between adjacent samples. The extension of this idea to 2D, called *bilinear interpolation*, is described by

$$\begin{aligned} g(m,n) =& \alpha\beta f(\lfloor m/C \rfloor, \lfloor n/D \rfloor) + (1-\alpha)\beta f(\lceil m/C \rceil, \lfloor n/D \rfloor) \\ &+ \alpha(1-\beta) f(\lfloor m/C \rfloor, \lceil n/D \rceil) + (1-\alpha)(1-\beta) f \\ &(\lceil m/C \rceil, \lceil n/D \rceil), \end{aligned} \quad (59)$$

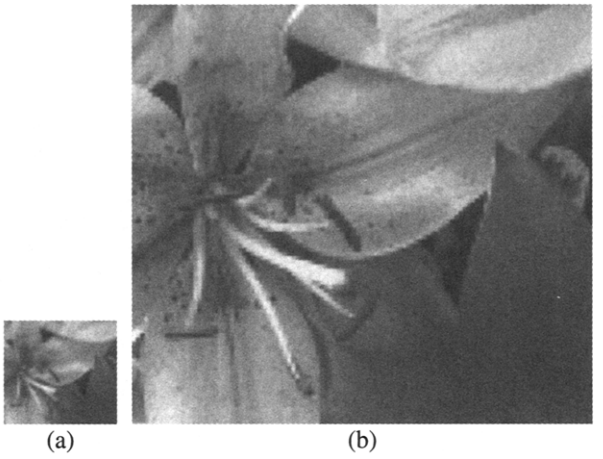


FIGURE 11 Interpolation by pixel replication: (a) original image, (b) image interpolated 4×. (See color insert.)

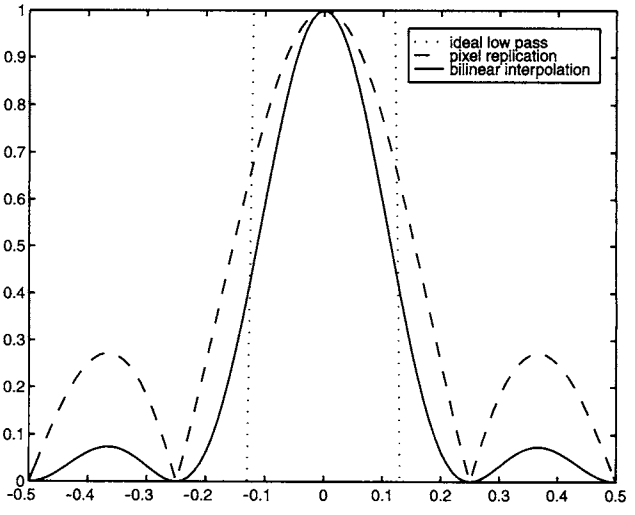


FIGURE 12 Magnitude of frequency response of linear filters for 4× interpolation.

FIGURE 13 Interpolation by 4× via bilinear interpolation. (See color insert.)

where $\alpha = \lceil m/C \rceil - m/C$ and $\beta = \lceil n/D \rceil - n/D$. In this case,

$$h_1(m,n) = \begin{cases} (1-|m/C|)(1-|n/D|), & |m| < C \text{ and } |n| < D \\ 0, & \text{else} \end{cases}$$

$$= h_0(m,n) * h_0(-m,-n) \tag{60}$$

It follows directly from (60) that

$$H_1(U,V) = |H_0(U,V)|^2, \tag{61}$$

which provides better suppression of the non-baseband replications of $F(U,V)$, as shown in Fig. 12. As can be seen in Fig. 13, bilinear interpolation yields an image that is free of the blockiness produced by pixel replication. However, the interpolated image has an overall appearance that is somewhat soft.

Both these strategies are examples of B-spline interpolation, which can be generalized to arbitrary order K. The corresponding frequency response is given by

$$H_K(U,V) = [H_0(U,V)]^K. \tag{62}$$

The choice $K=3$ is popular, since it yields a good tradeoff between smoothness of the interpolation and locality of dependence on the underlying data. For further discussion of image interpolation using splines, the reader is directed to [1] and [2]. The latter reference, in particular, discusses the design of an optimal prefilter for minimizing loss of information when splines are used for image reduction.

4.2 Model-based Approaches

In many applications, spline interpolation does not yield images that are sufficiently sharp. This problem can be traced to the way in which edges and textures are rendered. In recent years, there has been a great deal of interest in techniques for improving the quality of interpolated images by basing the interpolation on some type of image model [3–26]. These works use a variety of underlying interpolation methods: bilinear [7, 10, 12, 17, 22], cubic splines [6, 8, 17], directional filtering [3, 4, 5, 8], and least-squares fit to a model [7, 10]. To account for local image structure, these methods can be modified by changing the weights that are applied to the low-resolution image samples in the neighborhood of the desired high-resolution sample, or the values of the samples themselves. Another approach is to locally preprocess the low-resolution image with a distance warping function that moves more important image samples closer to the point of the desired high-resolution sample value [17, 25]. Some of the methods are based on statistical estimation [9, 15, 22]. A number of them use wavelets [11, 18, 19, 24]. Other frameworks include the theory of optimal recovery [26] and a binary decision tree [16]. Both these latter two methods rely on classification of local neighborhoods into one of a set of possible structures.

The methods cited here exploit a variety of local image features, such as the gradient [3, 25] or similar measure of local change [17], nature of contours or level sets [5, 23, 24], edge map [6, 12, 14], autccorrelation [22], or rate of decay of wavelet coefficients [18]. When estimation of model parameters across scales (from low resolution to high resolution) is required, it can be based solely on the assumed model structure [18, 22], or it can done within a training framework [16, 26]. Some of the methods are iterative, employing projections onto convex sets (POCS) or similar algorithms [12, 14, 20–23, 26]. Methods [15, 21] that have been primarily developed for interpolation of a high-resolution image from multiple frames of low resolution data can also be considered with the methods discussed here by assuming that there is only one low resolution frame available. In order to illustrate the kind of performance that can be achieved with methods of this type, we will briefly describe two approaches that have been reported in the literature, and show some experimental results.

4.2.1 Edge-Directed Interpolation

Figure 14 shows the framework within which the edge-directed interpolation algorithm operates. We will only sketch the highlights of the procedure here. For further details, the reader is directed to [12]. A subpixel edge estimation technique is used to generate a high resolution edge map from the low resolution image, and then the high resolution edge map is used to guide the interpolation of the low resolution image to the final high resolution version. Figure 15 shows the structure of the edge-directed interpolation algorithm itself. It consists of two phases: rendering and data correction. Rendering is based on a modified form of bilinear interpolation of the low resolution image data. An implicit assumption

underlying bilinear interpolation is that the low resolution data consists of point samples from the high resolution image. However, most sensors generate low resolution data by averaging the light incident at the focal plane over the unit cell corresponding to the low resolution sampling lattice. We iteratively compensate for this effect by feeding the interpolated image back through the sensor model and using the disparity between the resulting estimated sensor data and the true sensor data to correct the mesh values on which the bilinear interpolation is based. Reference [9] also embodies a sensor model.

To estimate the subpixel edge map shown in Fig. 14, we filter the low resolution image with a simple rectangular center-on-surround-off (COSO) filter with a constant positive center region embedded within a constant negative surround region. The relative heights are chosen to yield zero DC response. The filter coefficients are given by

$$h^{\mathrm{COSO}}(m,n) = \begin{cases} h_c, & |m|, |n| \le N_c, \\ h_s, & N_c < |m| \le N_s \text{ and } |n| \le N_s, \\ & N_c < |n| \le N_s \text{ and } |m| \le N_s, \\ 0, & \text{otherwise.} \end{cases} \tag{63}$$

This filter mimics the point spread function for the Laplacian-of-Gaussian (LoG) given by [27]

$$h^{\mathrm{LoG}}(m,n) = \frac{2}{\sigma^2}\left[1 - (m^2+n^2)/2\sigma^2\right]e^{-(m^2+n^2)/2\sigma^2}. \tag{64}$$

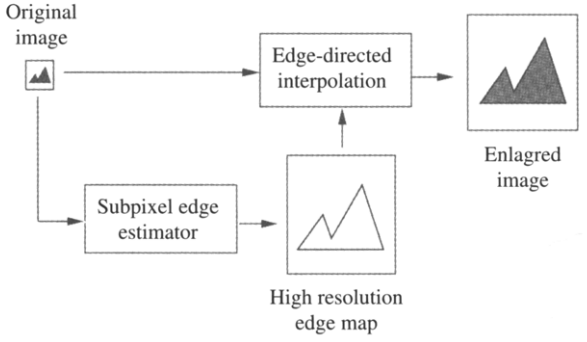


FIGURE 14 Framework for edge-directed interpolation algorithm.

as shown in Fig. 16. For a detailed treatment of the Laplacian-of-Guassian filter and its use, the reader is directed to Chapter 4.11.

The COSO filter results in a good approximation to the edge map generated with a true LoG filter, but requires only 9 additions/subtractions and 2 multiplies per output point when recursively implemented with row and column buffers. To determine the high resolution edge map, we linearly interpolate the COSO filter output between points on the low resolution lattice to estimate zero-crossing positions on the high resolution lattice. Figure 17 shows a subpixel edge map estimated using the COSO filter followed by piecewise linear interpolation, using the original low resolution image shown in Fig. 11. The interpolation factor was 4×. For comparison, we show a subpixel edge map obtained by upsampling the low-resolution edge map, followed by filtering with a LoG filter, and detection of zero crossings. The COSO edge map does not contain the fine detail that can be seen in the LoG edge map. However, it does show the major edges corresponding to significant gray value changes in the original image.

Now let us turn our attention to Fig. 15. The essential feature of the rendering step is that we modify bilinear interpolation on a pixel by pixel basis to prevent interpolation across edges. To illustrate the approach, let's consider interpolation at the high resolution pixel m in Fig. 18. We first determine whether or not any of the low resolution

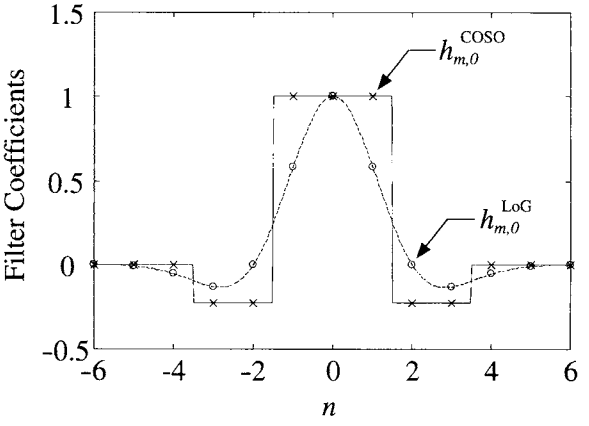


FIGURE 16 Point spread function of center-on-surround-off (COSO) and Laplacian of Gaussian (LoG) filters along the axes.

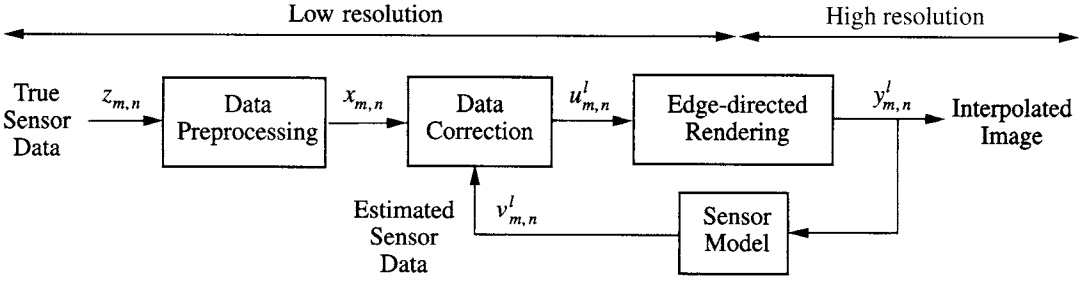


FIGURE 15 Structure of edge-directed interpolation algorithm.

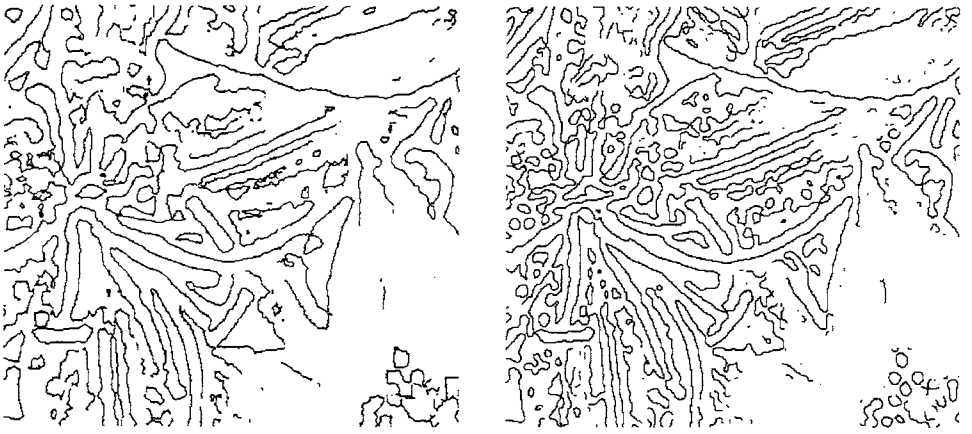


FIGURE 17 High resolution edge map interpolated 4× using rectangular, center-on-surroundoff (COSO) filter followed by piecewise linear interpolation of zero crossings (left) and high resolution edge map interpolated 4× using a Laplacian of Gaussian (LoG) filter after upsampling by 4×.

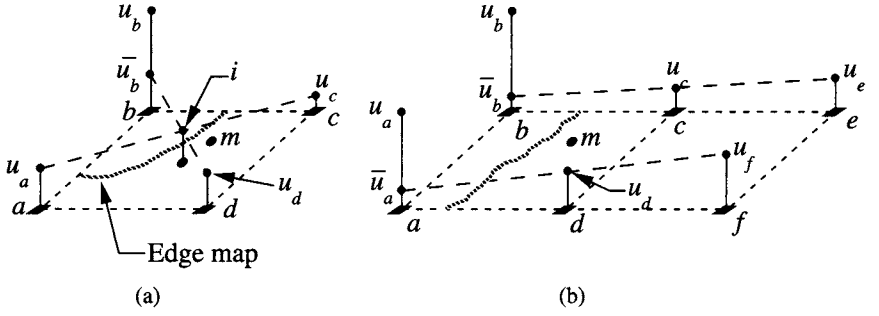


FIGURE 18 Computation of replacement values for the low resolution corner pixels to be used when bilinearly interpolating the image value at high resolution pixel m, The cases shown are: (a) replacement of one pixel, and (b) replacement of two adjacent pixels.

corner pixels $a, b, c,$ and d are separated from m by edges. For all those pixels that are, we compute replacement values according to a heuristic procedure that depends on the number and geometry of the pixels to be replaced. Figure 18a shows the situation where a single corner pixel u_b is to be replaced. In this case, we linearly interpolate to the midpoint i of the line $u_a - u_c$, and then extrapolate along the line $u_d - i$ to yield the replacement value $\bar{u}_b$. If two corner pixels are to be replaced, they can be either adjacent or not adjacent. Figure 18b shows the case where two adjacent pixels u_a and u_b must be replaced. In this case, we check to see if any edges cross the lines $e - c$ and $f - d$. If none does, we linearly extrapolate along the lines $u_e - u_c$ and $u_f - u_d$ to generate the replacement values $\bar{u}_b$ and $\bar{u}_a$, respectively. If an edge crosses $e - c$, we simply let $\bar{u}_b = u_c$. The cases where two non-adjacent pixels are to be replaced or where three pixels are to be replaced are treated similarly. The final case to be considered is that which occurs when the pixel m to be interpolated is separated from all four corner pixels $a, b, c,$ and d by edges. This case would only occur in regions of high spatial activity. In such areas, we assume that it is not possible to obtain a meaningful estimate of the high resolution edge map from just the four low resolution corner pixels; so the high resolution image will be rendered with unmodified bilinear interpolation. It is interesting to note that the process of bilinear interpolation, except across edges is very closely related to aniosotropic diffusion which is studied in detail in Chapter 4.12.

To describe the iterative correction procedure, we let l be the iteration index, and denote the true sensor data by $z(m, n)$, the preprocessed sensor data by $x(m, n)$, the corrected sensor data by $u^{(l)}(m, n)$, the edge-directed rendering step by the operator $\mathcal{R}$, the interpolated image by $y^{(l)}(m, n)$, the sensor model by the operator $\mathcal{S}$, and the estimated sensor data by $v^{(l)}(m, n)$. The sensor model $\mathcal{S}$ is a simple block average of the high resolution pixels in the unit cell for each pixel in the low resolution lattice.

With this notation, we may formally describe the procedure depicted in Fig. 15 by the following equations

$$y^{(l)}(m, n) = \mathcal{R}\left[u^{(l)}(m, n)\right], \tag{65}$$

$$v^{(l)}(m, n) = \mathcal{S}\left[y^{(l)}(m, n)\right], \tag{66}$$

$$u^{(l+1)}(m, n) = u^{(l)}(m, n) + \lambda\left(v^{(l)}(m, n) - x(m, n)\right), \tag{67}$$

(a) (b)

FIGURE 19 Image interpolated by 4× using edge-directed interpolation with (a) 0 and (b) 10 iterations. (See color insert.)

where λ is a constant that controls the gain of the correction process. The iteration is started with the initial condition $u^0(m, n) = x(m, n)$. Equations (65)–(67) represent a classic successive approximation procedure [28]. We can think of $v^{(l)}(m, n) - x(m, n)$ as the closed loop error when an image is interpolated and then decimated as it passes through the sensor model. If we have convergence in the iterative loop, i.e., if $u^{(l+1)}(m, n) = u^{(l)}(m, n)$, this implies that $v^{(l)}(m, n) = x(m, n)$. Hence the closed loop error is zero. Convergence of the iteration can be proved under mild restrictions on the location of edges [14].

Figure 19 shows the results of 4× interpolation using the edge directed interpolation algorithm after iterations 0 and 10. We see that edge-directed interpolation yields a much sharper result than bilinear interpolation. While some of the aliasing artifacts that occur with pixel replication can be seen in the edge-directed interpolation result, they are not nearly as prominent. The result after iteration 0 shows the effect of the edge-directed rendering alone, without data correction to account for the sensor model. While this image is sharper than that produced by bilinear interpolation, it lacks some of the crispness of the image resulting after 10 iterations of the algorithm.

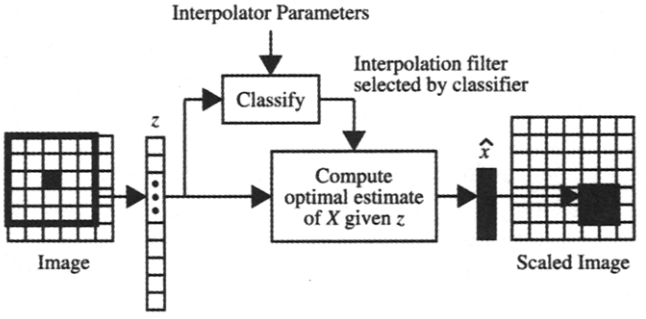


FIGURE 20 TBRS interplation by a factor of 2.

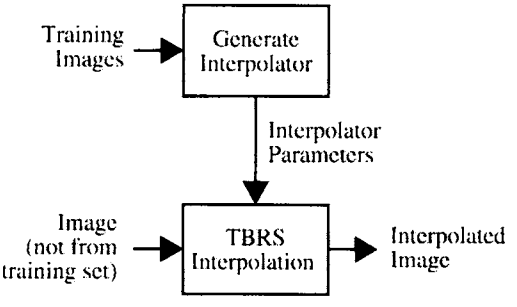


FIGURE 21 TBRS by a factor of 2.

4.2.2 Tree-based Resolution Synthesis

Tree-based resolution synthesis (TBRS) [16] works by first performing a fast local classification of a window around the pixel being interpolated, and then applying an interpolation filter designed for the selected class, as illustrated in Fig. 20. The idea behind TBRS is to use a regression tree as a piecewise linear approximation to the conditional mean estimator of the high-resolution image given the low-resolution image. Intuitively, having the different regions of linear operation allows for separate filtering of distinct behaviors like edges of different orientation and smoother gradient transitions.

An overview of the TBRS algorithm appears in Fig. 21. Note that before TBRS can be executed, we must already have generated the parameters for the regression tree by training on sample images. This training procedure requires considerable computation, but it only needs to be performed once. The resulting predictor may be used effectively on images that were not used in the training.

As illustrated in Fig. 20, we generate an $C \times C$ block of high-resolution pixels for every pixel in the low-resolution source image by filtering the corresponding $W \times W$ window of pixels

in the low resolution image, with the filter coefficients selected based on a classification. We have used $W = 5$. Thinking of the desired high-resolution pixels as an C^2-dimensional random vector X and the corresponding low-resolution pixels as the realization of a W^2-dimensional random vector Z, our approach is to use a regression tree which approximates the conditional mean estimator of $X|Z$, so that the vector $\hat{X}$ of interpolated pixels satisfies

$$\hat{X} = E[X|Z]. \tag{68}$$

It is well known that the conditional mean estimator minimizes the expected mean-squared error [29]. Here we will use capital letters to represent random quantities, and lowercase letters for their realizations. A closed-form expression for the true conditional mean estimator would be difficult to obtain for the present context. However, the regression tree T that we use provides a convenient and flexible piecewise linear approximation, with the M different linear regions being polygonal subsets which comprise a partition of the sample space Z of low-resolution vectors Z. These polygonal subsets, or classes, correspond to visually distinct behaviors like edges of different orientation.

With the main ideas in place, we return to Fig. 20 for a better look. To interpolate the shaded pixel in the low-resolution image, we first procure the vector z by stacking the pixels in the 5×5 window centered there. Then we obtain interpolated pixels as

$$\hat{x} = A_j z + \beta_j, \tag{69}$$

where A_j and β_j are, respectively, the $L^2 \times W^2$ matrix and L^2-dimensional vector comprising the interpolation filter for class j, and j is the index of the class obtained as

$$j = C_T(z), \tag{70}$$

where $C_T : \mathcal{Z} \to \{0, \ldots, M-1\}$ is a function which embodies the classifying action of T. To evaluate $C_T(z)$, we begin at the top and traverse down the tree T as illustrated in Fig. 22, making a decision to go right or left at each nonterminal node (circle), and taking the index j of the terminal node (square) which z lands in. Each decision has the form,

$$e_m^t(Z - U_m) \gtrless 0, \tag{71}$$

where m is the index of the node, e_m and U_m are W^2-dimensional vectors, and a superscript t denotes taking the transpose. This decision determines whether z is on one side of a hyperplane or the other, with U_m being a point in the hyperplane and with e_m specifying its orientation. By convention, we go left if the quantity on the left-hand side is negative, and we go right otherwise.

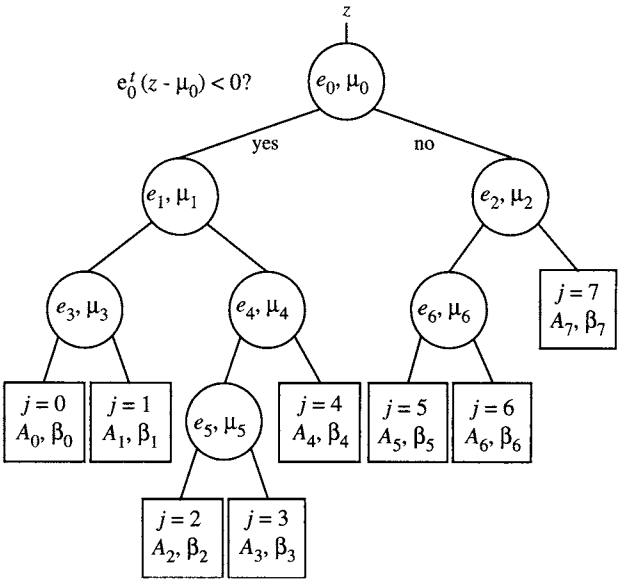


FIGURE 22 Binary tree structure used in TBRS.

FIGURE 23 Interpolation by 4× via TBRS. (See color insert.)

In order to complete the design of the TBRS algorithm, we must obtain numeric values for the integer number $M \geq 1$ of terminal nodes in the tree; the decision rules $\{(e_m, U_m)\}_{m=0}^{M-2}$ for the nonterminal nodes (assuming that $M > 1$); and the interpolation filters $\{(A_m, \beta_m)\}_{m=0}^{M-1}$ for the terminal nodes. To compute these parameters, a training procedure is used which is based on that given by Gelfand, Ravishankar, and Delp [30], suitably modified for the design of a regression tree rather than a classification tree. The training vector pairs are assumed to be independent realizations of (X, Z). Training vector pairs are extracted from low and high resolution renderings of the same image. For further details regarding the design process, the reader is directed to Ref. [16]. Figure 23 shows the flower

image interpolated by 4× using tree-based resolution synthesis. Comparing this image with those shown in Fig. 19, which were generated via edge-directed interpolation, we see that TBRS yields higher quality than edge-directed interpolation after 0 iterations, and quality that is comparable to that of edge-directed interpolation after 10 iterations.

5 Conclusion

Most systems for image capture may be categorized into one of two classes: flying spot scanners and focal plane arrays. Both these classes of systems may be modeled as a convolution with an aperture function, followed by multiplication by a sampling function. In the frequency domain, the spectrum of the continuous-parameter image is multiplied by the Fourier transform of the aperture function, resulting in attentuation of the higher frequencies in the image. This modified spectrum is then replicated on a lattice of points that is reciprocal to the sampling lattice. If the sampling frequency is sufficiently high, the replications will not overlap; and the original image may be reconstructed from its samples. Otherwise, the overlap of the spectral replications with the baseband term may cause aliasing artifacts to appear in the image.

In many image processing-applications, including printing of digital images, it is necessary to resize the image. This process may be analyzed within the framework of multirate signal processing. Decimation, which consists of low pass filtering followed by downsampling, results in expansion and replication of the spectrum of the original digital image. Interpolation, which consists of upsampling following by low pass filtering, causes the spectrum of the original digital image to contract; so that it occupies only a portion of the baseband spectral region. With this approach, the interpolated image is a linear function of the sampled data. Linear interpolation may blur edges and fine detail in the image. A variety of nonlinear approaches have been proposed that yield improved rendering of edges and detail in the image.

References

[1] H. S. Hou and H. C. Andrews, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoustics, Speech, and Signal Processing,* 26, 508–517, June 1978.

[2] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information," *IEEE Trans. Image Processing,* 4, 247–258, March 1995.

[3] V. R. Algazi, G. E. Ford, and R. Potharlanka, "Directional interpolation of images based on visual properties and rank order filtering," *Proc. 1991 International Conference on Acoustics, Speech, and Signal Processing,* Toronto, CN, May 14–17, 1991, 3005–3008.

[4] G. E. Ford, R. R. Estes, and H. Chen, "Space scale analysis for image sampling and interpolation," *Proc. 1992 International Conference on Acoustics, Speech, and Signal Processing,* San Francisco, CA, Mar. 23–26, 1992, III-165–III-168.

[5] B. Ayazifar and J. S. Lim, "Pel-adaptive model-based interpolation of spatially subsampled images," *Proc. 1992 International Conference on Acoustics, Speech, and Signal Processing,* San Francisco, CA, Mar. 23–26, 1992, III-181–III-184.

[6] K. Xue, A. Winans, and E. Walowit, "An edge-restricted_spatial interpolation algorithm," *J. Electtronic Imaging,* 1, 2, Apr. 1992.

[7] F. G. B. De Natale, G. S. Desoli, D. D. Guisto, and G. Vernazza, "A spline-like scheme for least-squares bilinear interpolation," *Proc. 1993 International Conference on Acoustics, Speech, and Signal Processing,* Minneapolis, MN, Apr. 27–30, 1993, V-141–V-144.

[8] S. W. Lee and J, K. Paik, "Image interpolation using adaptive fast B-spline filtering," *Proc: 1993 International Conference on Acoustics, Speech, and Signal Processing,* Minneapolis, MN, Apr. 27–30, 1993, V-177–V-180.

[9] R. R. Schultz and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," *IEEE Transactions on Image Processing,* 3, 233–242, May 1994.

[10] K. Jensen and D. Anastassiou, "Subpixel edge localization and the interpolation of still images," *IEEE Trans. Image Processing,* 4, 285–295, Mar. 1995.

[11] S. G. Chang, Z. Cvetkovic, and M. Vetterli, "Resolution enhancement of images using wavelet transform extrema extrapolation," *Proc. 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing,* 4, 2379–2382, 1995.

[12] J. P. Allebach and P. W. Wong, "Edge-directed interpolation," *Proc, 1996 IEEE International Conference on Image Processing,* Lausanne, Switzerland, Sep. 16–19, 1996, III-707–III-710.

[13] S. Carrato, G. Ramponi, and S. Marsi "A simple edge-sensitive image interpolation filter," *Proc. 1996 IEEE International Conference on Image Processing,* Lausanne, Switzerland, Sep. 16–19, 1996, III-711–III-714.

[14] P. W. Wong and J. P. Allebach, "Convergence of an iterative edge directed image interpolation algorithm," *Proc. of the 1997 IEEE International Symposium on Circuits and Systems,* Hong Kong, June 9–12, 1997, 2, 1173–1176.

[15] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, "Joint map registration and high-resolution image estimation using a sequence of undersampled images," *IEEE Transactions on Image Processing,* 6, 1621–1633, Dec. 1997.

[16] C. B. Atkins, C. A. Bouman, and J. P. Allebach, "Tree-based resolution synthesis," *Proceedings of PICS-99: the 1999 IS&T Image Processing, Image Quality, Image Capture Systems Conference,* Savannah, GA, Apr. 25–28, 1999, 405–410.

[17] G. Ramponi, "Warped distance for space-variant linear image interpolation," *IEEE Transactions on Image Processing,* 8, 629–639, May 1999.

[18] W. K. Carey, D. B. Chuang, and S. S. Hemami, "Regularity-preserving image interpolation," *IEEE Transactions on Image Processing,* 8, 1293–1297, Sep. 1999.

[19] B. Tao and M. T. Orchard "Wavelet-domain edge modeling with applications to image interpolation," *Image and Video Communications and Processing 2000,* B. Vasudev, T. R. Hsiang, A. G. Tescher, and R. L. Stevenson, eds. SPIE-3974, 537–548, 2000.

[20] L. S. DeBrunner, V. DeBrunner, and M. Yao, "Edge-retaining asymptotic projections onto convex sets for image interpolation," *Proc. 4th IEEE Southwest Symposium on Image Analysis and Interpretation,* April 2–4, 2000.

[21] A. J. Patti and Y. Altunbasak, "Artifact reduction for set theoretic super resolution image reconstruction with edge adaptive constraints and higher, order interpolants," *IEEE Transactions on Image Processing,* 10, 179–186, Jan. 2001.

[22] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing,* 10, 1521–1527, Oct. 2001.

[23] B. S. Morse and D. Schwartzwald, "Image magnification using level-set reconstruction," *Proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1, 333–340, Dec. 2001.

[24] H. F. Ates and M. T. Orchard, "Image interpolation using wavelet-based contour estimation," *Proc. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing,* 3, 109–112, April 2003.

[25] F. Zotta, P. Carrai, G. Ferretti, and G. Ramponi, "A directional edge-sensitive image interpolator," *Proc. SPIE-IS&T Electronic Imaging–Image Processing: Algorithms and Systems III,* E. R. Dougherty, J. T. Astola, K. O. Egiazarian, eds. SPIE-5298, 212–219, 2004.

[26] D. D. Muresan and T. W. Parks, "Adaptively quadratic (AQua) image interpolation," *IEEE Transactions on Image Processing,* 13, 690–698, May 2004

[27] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-8, 679–698, Nov. 1986.

[28] R. W. Schafer, R M. Mersereau, and M. A. Richard, "Constrained Iterative Restoration Algorithms," *Proc. IEEE,* 69, 432–450, Apr. 1981.

[29] L. L. Sharf, *Statistical Signal Processing,* Addison-Wesley, Reading, MA, 1991.

[30] S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Trans. PAMI,* 13, 163–174, 1991.

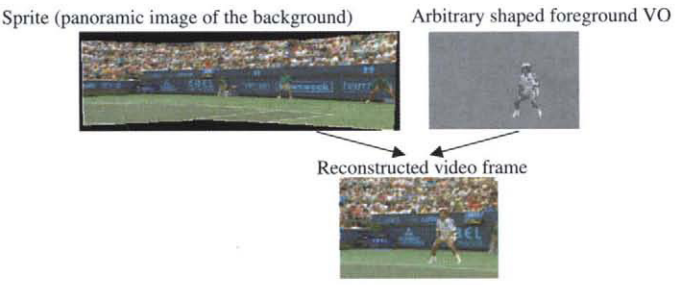


FIGURE 6.5.7 Sprite coding of a video sequence (courtesy of Dr. Thomas Sikora, Technical University of Berlin, [6]). VO, video object.

FIGURE 6.5.17 A decoded frame of the sequence *Foreman* (a) without the in-loop deblocking filtering applied and (b) with the in-loop deblocking filtering (the original sequence *Foreman* is courtesy of Siemens AG).

FIGURE 6.6.1 Embedded video codec aplication in handheld and portable multimedia products.

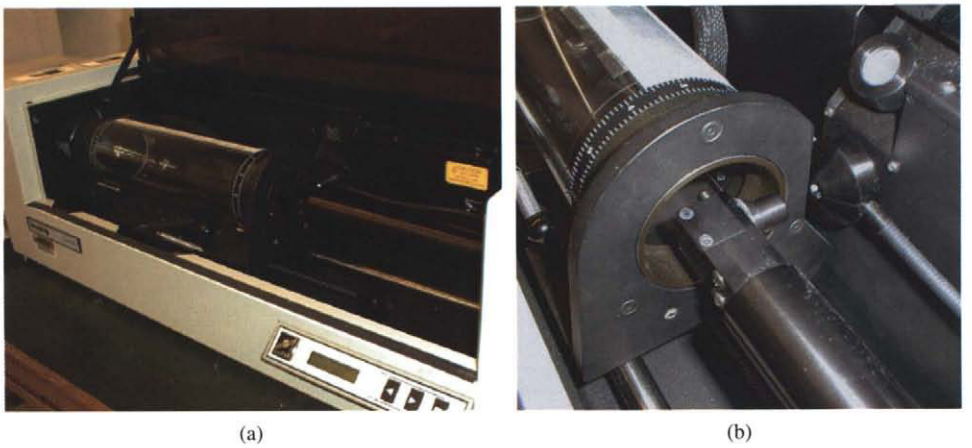


FIGURE 7.1.1 High resolution drum scanner: (a) scanner with cover open, and (b) closeup view showing screw-mounted "C" carriage with light source on inside arm, and detector optics on outside arm.

(a) (b)

FIGURE 7.1.11 Interpolation by pixel replication: (a) original image, (b) image interpolated 4×.

FIGURE 7.1.13 Interpolation by 4× via bilinear interpolation.

(a) (b)

FIGURE 7.1.19 Image interpolated by 4× using edge-directed interpolation with (a) 0 and (b) 10 iterations.

FIGURE 7.1.23 Interpolation by 4× via TBRS.