CHAPTER 7

Sampling and Aliasing

7.1 SPATIAL FREQUENCY AND FOURIER TRANSFORMS

We have used the trick of thinking of a signal $g(x, y)$ as a weighted sum of a large (or infinite) number of small (or infinitely small) box functions. This model emphasizes that a signal is an element of a vector space. The box functions form a convenient basis, and the weights are coefficients on this basis. We need a new technique to deal with two related problems so far left open:

- Although it is clear that a discrete image version cannot represent the full information in a signal, we have not yet indicated what is lost.
- It is clear that we cannot shrink an image simply by taking every kth pixel—this could turn a checkerboard image all white or all black—and we would like to know how to shrink an image safely.

All of these problems are related to the presence of fast changes in an image. For example, shrinking an image is most likely to miss fast effects because they could slip between samples; similarly, the derivative is large at fast changes.

These effects can be studied by *a change of basis*. We change the basis to be a set of sinusoids and represent the signal as an infinite weighted sum of an infinite number of sinusoids. This means that fast changes in the signal are obvious, because they correspond to large amounts of high-frequency sinusoids in the new basis.

7.1.1 Fourier Transforms

The change of basis is effected by a *Fourier transform*. We define the Fourier transform of a signal $g(x, y)$ to be

$$\mathcal{F}(g(x,y))(u,v) = \int_{-\infty}^{\infty}\int g(x,y)e^{-i2\pi(ux+vy)}dxdy.$$

Assume that appropriate technical conditions are true to make this integral exist. It is sufficient for all moments of g to be finite; a variety of other possible conditions are available [**?**]. The process takes a complex valued function of x, y and returns a complex valued function of u, v (images are complex valued functions with zero imaginary component).

For the moment, fix u and v, and let us consider the meaning of the value of the transform at that point. The exponential can be rewritten

$$e^{-i2\pi(ux+vy)} = \cos(2\pi(ux+vy)) + i\sin(2\pi(ux+vy)).$$

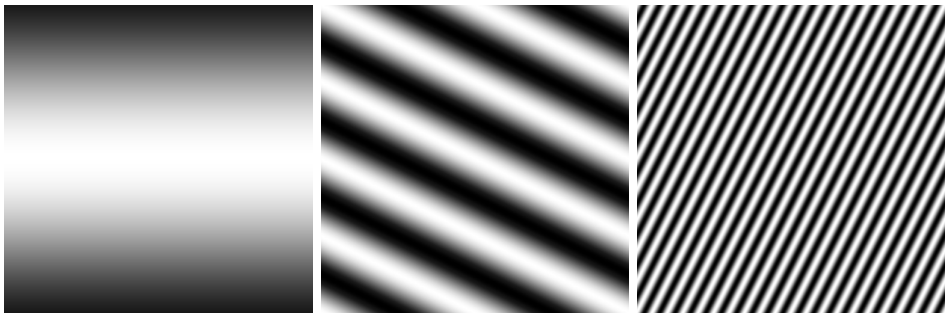

FIGURE 7.1: *The real component of Fourier basis elements shown as intensity images. The brightest point has value one, and the darkest point has value zero. The domain is* $[-1,1] \times [-1,1]$*, with the origin at the center of the image. On the* **left***,* $(u,v) = (0, 0.4)$*; in the* **center***,* $(u,v) = (1,2)$*; and on the* **right** $(u,v) = (10,-5)$*. These are sinusoids of various frequencies and orientations described in the text.*

These terms are sinusoids on the x, y plane, whose orientation and frequency are given by u, v. For example, consider the real term, which is constant when $ux+vy$ is constant (i.e., along a straight line in the x, y plane whose orientation is given by $\tan\theta = v/u$). The gradient of this term is perpendicular to lines where $ux + vy$ is constant, and the frequency of the sinusoid is $\sqrt{u^2+v^2}$. These sinusoids are often referred to as *spatial frequency components*; a variety are illustrated in Figure 7.1.

The integral should be seen as a dot product. If we fix u and v, the value of the integral is the dot product between a sinusoid in x and y and the original function. This is a useful analogy because dot products measure the amount of one vector in the direction of another.

In the same way, the value of the transform at a particular u and v can be seen as measuring the amount of the sinusoid with given frequency and orientation in the signal. The transform takes a function of x and y to the function of u and v whose value at any particular (u,v) is the amount of that particular sinusoid in the original function. This view justifies the model of a Fourier transform as a change of basis.

Linearity

The Fourier transform is linear:

$$\mathcal{F}(g(x,y) + h(x,y)) = \mathcal{F}(g(x,y)) + \mathcal{F}(h(x,y))$$

and

$$\mathcal{F}(kg(x,y)) = k\mathcal{F}(g(x,y)).$$

The Inverse Fourier Transform It is useful to recover a signal from its Fourier transform. This is another change of basis with the form

$$g(x,y) = \int_{-\infty}^{\infty}\int \mathcal{F}(g(x,y))(u,v)e^{i2\pi(ux+vy)}dudv.$$

Fourier Transform Pairs Fourier transforms are known in closed form

TABLE 7.1: *A variety of functions of two dimensions and their Fourier transforms. This table can be used in two directions (with appropriate substitutions for u, v and x, y) because the Fourier transform of the Fourier transform of a function is the function. Observant readers might suspect that the results on infinite sums of δ functions contradict the linearity of Fourier transforms. By careful inspection of limits, it is possible to show that they do not (see, for example, ?). Observant readers also might have noted that an expression for $\mathcal{F}(\frac{\partial f}{\partial y})$ can be obtained by combining two lines of this table.*

Function	Fourier transform
$g(x,y)$	$\int_{-\infty}^{\infty}\int g(x,y)e^{-i2\pi(ux+vy)}dxdy$
$\int_{-\infty}^{\infty}\int \mathcal{F}(g(x,y))(u,v)e^{i2\pi(ux+vy)}dudv$	$\mathcal{F}(g(x,y))(u,v)$
$\delta(x,y)$	1
$\frac{\partial f}{\partial x}(x,y)$	$u\mathcal{F}(f)(u,v)$
$0.5\delta(x+a,y)+0.5\delta(x-a,y)$	$\cos 2\pi au$
$e^{-\pi(x^2+y^2)}$	$e^{-\pi(u^2+v^2)}$
$box_1(x,y)$	$\frac{\sin u}{u}\frac{\sin v}{v}$
$f(ax,by)$	$\frac{\mathcal{F}(f)(u/a,v/b)}{ab}$
$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)$	$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(u-i,v-j)$
$(f**g)(x,y)$	$\mathcal{F}(f)\mathcal{F}(g)(u,v)$
$f(x-a,y-b)$	$e^{-i2\pi(au+bv)}\mathcal{F}(f)$
$f(x\cos\theta-y\sin\theta, x\sin\theta+y\cos\theta)$	$\mathcal{F}(f)(u\cos\theta-v\sin\theta, u\sin\theta+v\cos\theta)$

for a variety of useful cases; a large set of examples appears in **?**. We list a few in Table 7.1 for reference. The last line of Table 7.1 contains the *convolution theorem*; convolution in the signal domain is the same as multiplication in the Fourier domain.

Phase and Magnitude The Fourier transform consists of a real and a

complex component:

$$\begin{aligned}\mathcal{F}(g(x,y))(u,v) &= \int\int_{-\infty}^{\infty} g(x,y)\cos(2\pi(ux+vy))dxdy + \\ &\quad i\int\int_{-\infty}^{\infty} g(x,y)\sin(2\pi(ux+vy))dxdy \\ &= \Re(\mathcal{F}(g)) + i * \Im(\mathcal{F}(g)) \\ &= \mathcal{F}_R(g) + i * \mathcal{F}_I(g).\end{aligned}$$

It is usually inconvenient to draw complex functions of the plane. One solution is to plot $\mathcal{F}_R(g)$ and $\mathcal{F}_I(g)$ separately; another is to consider the *magnitude* and *phase* of the complex functions, and to plot these instead. These are then called the *magnitude spectrum* and *phase spectrum*, respectively.

The value of the Fourier transform of a function at a particular u, v point depends on the whole function. This is obvious from the definition because the domain of the integral is the whole domain of the function. It leads to some subtle properties, however. First, a local change in the function (e.g., zeroing out a block of points) is going to lead to a change *at every point* in the Fourier transform. This means that the Fourier transform is quite difficult to use as a representation (e.g., it might be very difficult to tell whether a pattern was present in an image just by looking at the Fourier transform). Second, the magnitude spectra of images tends to be similar. This appears to be a fact of nature, rather than something that can be proven axiomatically. As a result, the magnitude spectrum of an image is surprisingly uninformative (see Figure 7.2 for an example).

7.2 SAMPLING AND ALIASING

The crucial reason to discuss Fourier transforms is to get some insight into the difference between discrete and continuous images. In particular, it is clear that some information has been lost when we work on a discrete pixel grid, but what? A good, simple example comes from an image of a checkerboard, and is given in Figure 7.3. The problem has to do with the number of samples relative to the function; we can formalize this rather precisely given a sufficiently powerful model.

7.2.1 Sampling

Passing from a continuous function—like the irradiance at the back of a camera system—to a collection of values on a discrete grid —like the pixel values reported by a camera—is referred to as *sampling*. We construct a model that allows us to obtain a precise notion of what is lost in sampling.

Sampling in One Dimension

Sampling in one dimension takes a function and returns a discrete set of values. The most important case involves sampling on a uniform discrete grid, and we assume that the samples are defined at integer points. This means we have a process that takes some function and returns a vector of values:

$$\texttt{sample}_{1D}(f(x)) = \mathbf{f}.$$

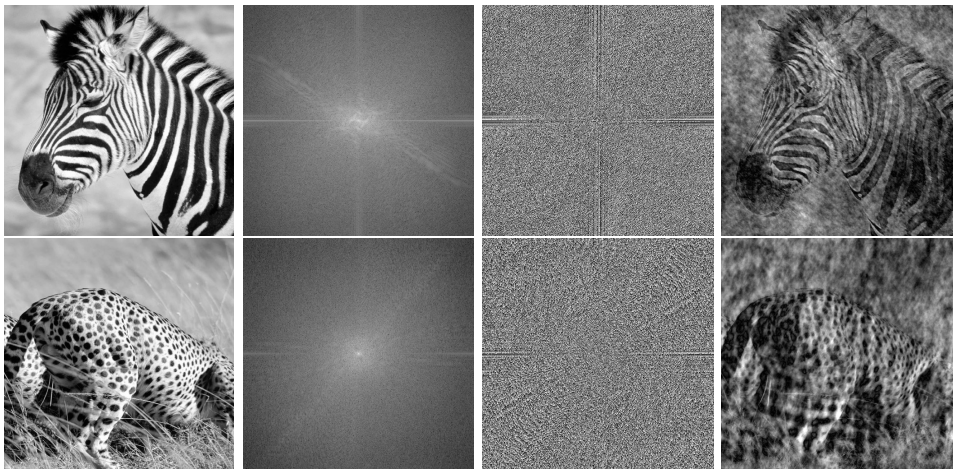

FIGURE 7.2: *The second image in each row shows the log of the magnitude spectrum for the first image in the row; the third image shows the phase spectrum scaled so that $-\pi$ is dark and π is light. The final images are obtained by swapping the magnitude spectra. Although this swap leads to substantial image noise, it doesn't substantially affect the interpretation of the image, suggesting that the phase spectrum is more important for perception than the magnitude spectrum.*

We model this sampling process by assuming that the elements of this vector are the values of the function $f(x)$ at the sample points and allowing negative indices to the vector (Figure 7.4). This means that the ith component of $\mathbf{f}$ is $f(x_i)$.

Sampling in Two Dimensions

Sampling in 2D is very similar to sampling in 1D. Although sampling can occur on nonregular grids (the best example being the human retina), we proceed on the assumption that samples are drawn at points with integer coordinates. This yields a uniform rectangular grid, which is a good model of most cameras. Our sampled images are then rectangular arrays of finite size (all values outside the grid being zero).

In the formal model, we sample a function of two dimensions, instead of one, yielding an array (Figure 7.5). We allow this array to have negative indices in both dimensions, and can then write

$$\texttt{sample}_{2D}(F(x,y)) = \mathcal{F},$$

where the i, jth element of the array $\mathcal{F}$ is $F(x_i, y_j) = F(i,j)$.

Samples are not always evenly spaced in practical systems. This is quite often due to the pervasive effect of television; television screens have an aspect ratio of 4:3 (width:height). Cameras quite often accommodate this effect by spacing sample points slightly farther apart horizontally than vertically (in jargon, they have *non-square pixels*).

A Continuous Model of a Sampled Signal

We need a continuous model of a sampled signal. Generally, this model is used to evaluate integrals; in particular, taking a Fourier transform involves integrating the product of our model with a complex exponential. It is clear how this integral should behave: the value of the integral should be obtained by adding up values at each integer point. This means we cannot model a sampled signal as a function that is zero everywhere except at integer points (where it takes the value of the signal), because this model has a zero integral.

An appropriate continuous model of a sampled signal relies on an important property of the δ function:

$$\begin{aligned}\int_{-\infty}^{\infty} a\delta(x)f(x)dx &= a\lim_{\epsilon\to 0}\int_{-\infty}^{\infty} d(x;\epsilon)f(x)dx \\ &= a\lim_{\epsilon\to 0}\int_{-\infty}^{\infty} \frac{bar(x;\epsilon)}{\epsilon}(f(x))dx \\ &= a\lim_{\epsilon\to 0}\sum_{i=-\infty}^{\infty} \frac{bar(x;\epsilon)}{\epsilon}(f(i\epsilon)bar(x-i\epsilon;\epsilon))\epsilon \\ &= af(0).\end{aligned}$$

Here we have used the idea of an integral as the limit of a sum of small strips.

An appropriate continuous model of a sampled signal consists of a δ-function at each sample point weighted by the value of the sample at that point. We can obtain this model by multiplying the sampled signal by a set of δ-functions, one at each sample point. In one dimension, a function of this form is called a *comb function* (because that's what the graph looks like). In two dimensions, a function of this form is called a *bed-of-nails function* (for the same reason).

Working in 2D and assuming that the samples are at integer points, this procedure gets

$$\begin{aligned}\texttt{sample}_{2D}(f) &= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f(i,j)\delta(x-i,y-j) \\ &= f(x,y)\left\{\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)\right\}.\end{aligned}$$

This function is zero except at integer points (because the δ-function is zero except at integer points), and its integral is the sum of the function values at the integer points.

7.2.2 Aliasing

Sampling involves a loss of information. As this section shows, a signal sampled too slowly is misrepresented by the samples; high spatial frequency components of the original signal appear as low spatial frequency components in the sampled signal—an effect known as *aliasing*.

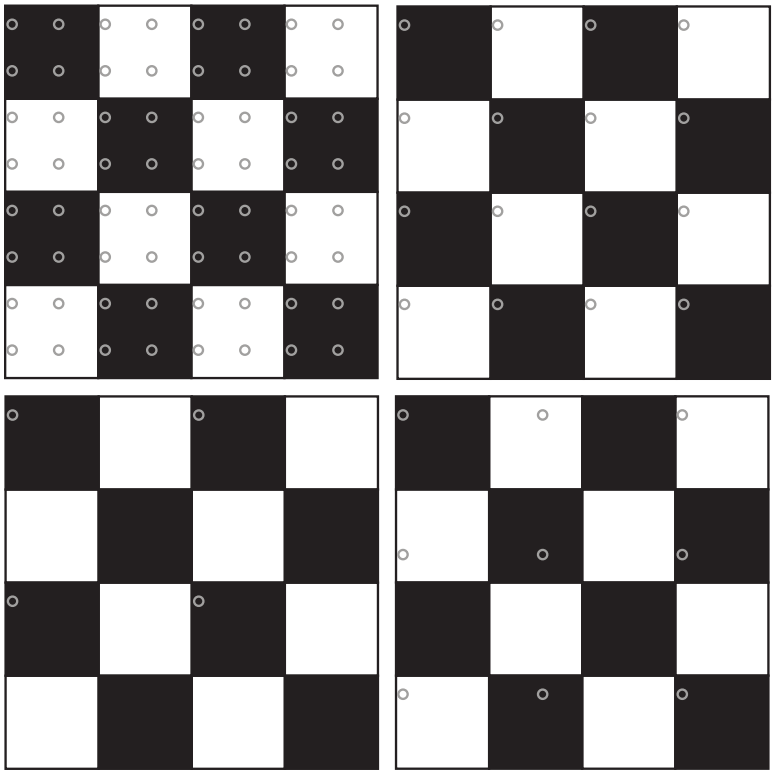

FIGURE 7.3: *The two checkerboards on the* **top** *illustrate a sampling procedure that appears to be successful (whether it is or not depends on some details that we will deal with later). The gray circles represent the samples; if there are sufficient samples, then the samples represent the detail in the underlying function. The sampling procedures shown on the* **bottom** *are unequivocally unsuccessful; the samples suggest that there are fewer checks than there are. This illustrates two important phenomena: first, successful sampling schemes sample data often enough; and second, unsuccessful sampling schemes cause high-frequency information to appear as lower-frequency information.*

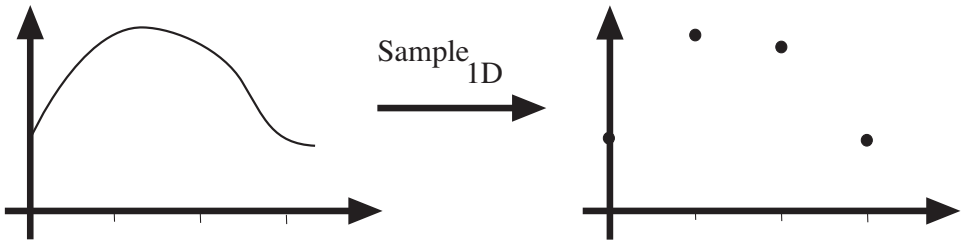

FIGURE 7.4: *Sampling in 1D takes a function and returns a vector whose elements are values of that function at the sample points. For our purposes, it is enough that the sample points be integer values of the argument. We allow the vector to be infinite dimensional and have negative as well as positive indices.*

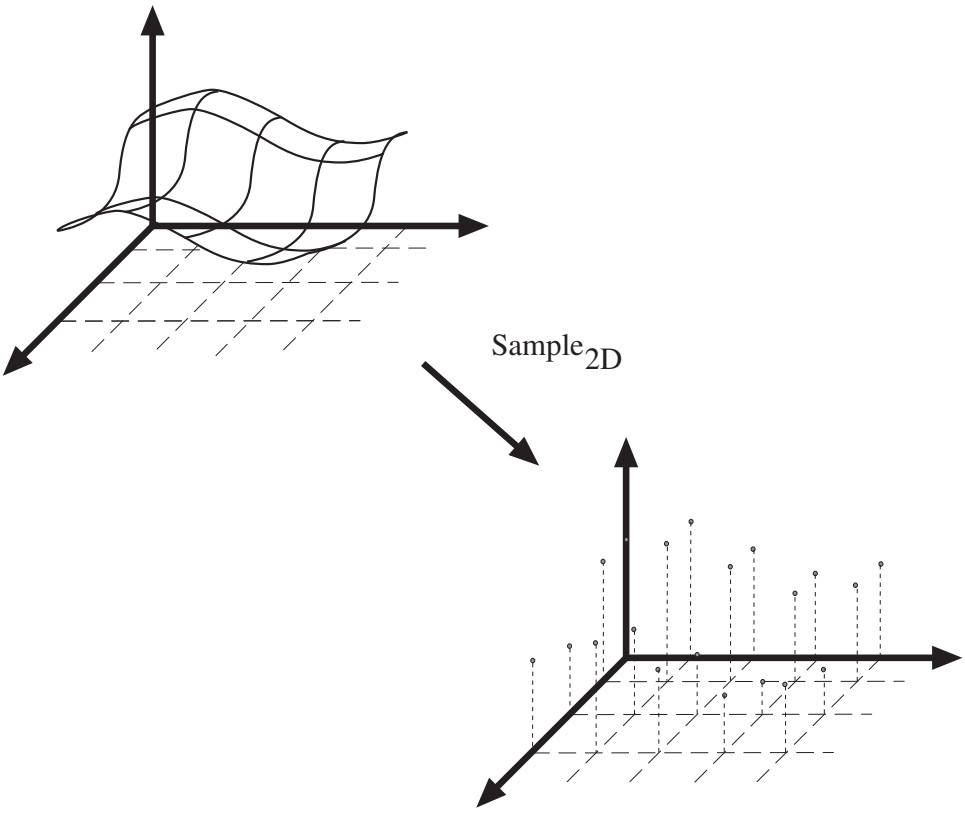

FIGURE 7.5: *Sampling in 2D takes a function and returns an array; again, we allow the array to be infinite dimensional and to have negative as well as positive indices.*

The Fourier Transform of a Sampled Signal

A sampled signal is given by a product of the original signal with a bed-of-nails function. By the convolution theorem, the Fourier transform of this product is the convolution of the Fourier transforms of the two functions. This means that the Fourier transform of a sampled signal is obtained by convolving the Fourier transform of the signal with another bed-of-nails function.

Now convolving a function with a shifted δ-function merely shifts the function (see exercises). This means that the Fourier transform of the sampled signal is the sum of a collection of shifted versions of the Fourier transforms of the signal, that is,

$$\begin{aligned}\mathcal{F}(\texttt{sample}_{2D}(f(x,y))) &= \mathcal{F}\left(f(x,y)\left\{\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)\right\}\right)\\ &= \mathcal{F}(f(x,y)) * *\mathcal{F}\left(\left\{\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)\right\}\right)\\ &= \sum_{i=-\infty}^{\infty}F(u-i,v-j),\end{aligned}$$

where we have written the Fourier transform of $f(x,y)$ as $F(u,v)$.

If the support of these shifted versions of the Fourier transform of the signal does not intersect, we can easily reconstruct the signal from the sampled version. We take the sampled signal, Fourier transform it, and cut out one copy of the Fourier transform of the signal and Fourier transform this back (Figure 7.6).

However, if the support regions *do* overlap, we are not able to reconstruct the signal because we can't determine the Fourier transform of the signal in the regions of overlap, where different copies of the Fourier transform will add. This results in a characteristic effect, usually called *aliasing*, where high spatial frequencies appear to be low spatial frequencies (see Figure 7.8 and exercises). Our argument also yields *Nyquist's theorem*: the sampling frequency must be at least twice the highest frequency present for a signal to be reconstructed from a sampled version. By the same argument, if we happen to have a signal that has frequencies present only in the range $[2k-1\Omega, 2k+1\Omega]$, then we can represent that signal exactly if we sample at a frequency of at least 2Ω.

7.2.3 Smoothing and Resampling

Nyquist's theorem means it is dangerous to shrink an image by simply taking every kth pixel (as Figure 7.8 confirms). Instead, we need to filter the image so that spatial frequencies above the new sampling frequency are removed. We could do this exactly by multiplying the image Fourier transform by a scaled 2D bar function, which would act as a low-pass filter. Equivalently, we would convolve the image with a kernel of the form $(\sin x \sin y)/(xy)$. This is a difficult and expensive (a polite way of saying *impossible*) convolution because this function has infinite support.

The most interesting case occurs when we want to halve the width and height of the image. We assume that the sampled image has no aliasing (because if it

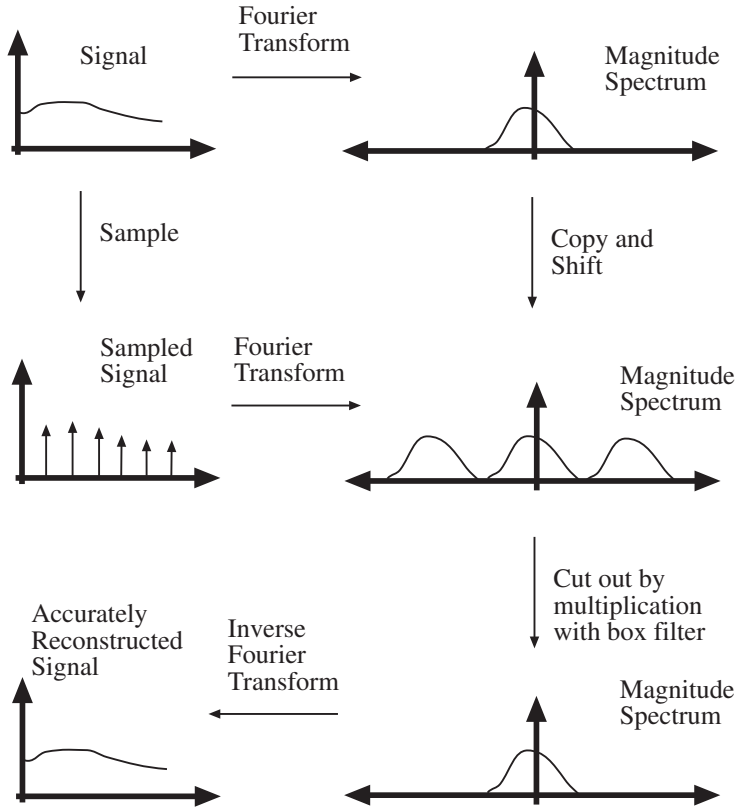

FIGURE 7.6: *The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. Two possibilities occur. If the shifted copies do not intersect with each other (as in this case), the original signal can be reconstructed from the sampled signal (we just cut out one copy of the Fourier transform and inverse transform it). If they do intersect (as in Figure 7.7), the intersection region is added, and so we cannot obtain a separate copy of the Fourier transform, and the signal has aliased.*

did, there would be nothing we could do about it anyway; once an image has been sampled, any aliasing that is going to occur has happened, and there's not much we can do about it without an image model). This means that the Fourier transform of the sampled image is going to consist of a set of copies of some Fourier transform, with centers shifted to integer points in u, v space.

If we resample this signal, the copies now have centers on the half-integer points in u, v space. This means that, to avoid aliasing, we need to apply a filter that strongly reduces the content of the original Fourier transform outside the range $|u| < 1/2$, $|v| < 1/2$. Of course, if we reduce the content of the signal *inside* this range, we might lose information, too. Now the Fourier transform of a Gaussian is a Gaussian, and Gaussians die away fairly quickly. Thus, if we were to convolve the

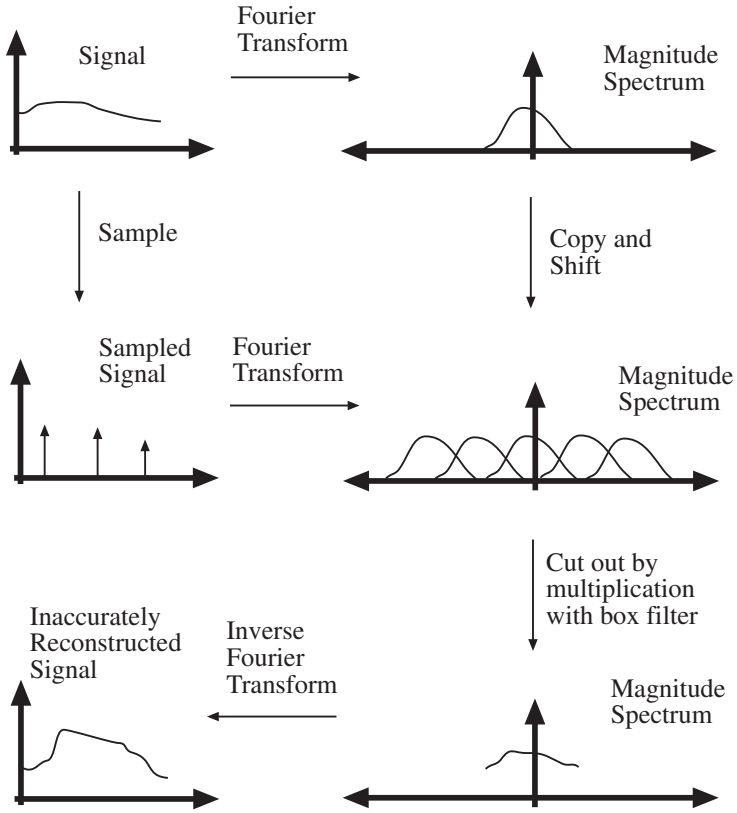

FIGURE 7.7: *The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. Two possibilities occur. If the shifted copies do not intersect with each other (as in Figure 7.6), the original signal can be reconstructed from the sampled signal (we just cut out one copy of the Fourier transform and inverse transform it). If they do intersect (as in this figure), the intersection region is added, and so we cannot obtain a separate copy of the Fourier transform, and the signal has aliased. This also explains the tendency of high spatial frequencies to alias to lower spatial frequencies.*

image with a Gaussian—or multiply its Fourier transform by a Gaussian, which is the same thing—we could achieve what we want.

The choice of Gaussian depends on the application. If σ is large, there is less aliasing (because the value of the kernel outside our range is very small), but information is lost because the kernel is not flat within our range; similarly, if σ is small, less information is lost within the range, but aliasing can be more substantial. Figures 7.9 and 7.10 illustrate the effects of different choices of σ.

We have been using a Gaussian as a low-pass filter because its response at high spatial frequencies is low and its response at low spatial frequencies is high. In fact, the Gaussian is not a particularly good low-pass filter. What one wants

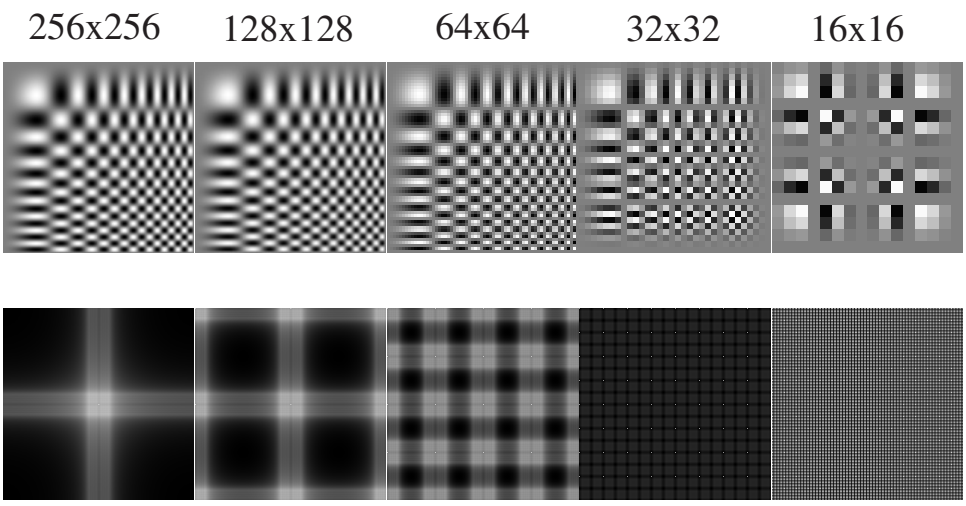

FIGURE 7.8: *The* **top row** *shows sampled versions of an image of a grid obtained by multiplying two sinusoids with linearly increasing frequency—one in x and one in y. The other images in the series are obtained by resampling by factors of two without smoothing (i.e., the next is a 128x128, then a 64x64, etc., all scaled to the same size). Note the substantial aliasing; high spatial frequencies alias down to low spatial frequencies, and the smallest image is an extremely poor representation of the large image. The* **bottom row** *shows the magnitude of the Fourier transform of each image displayed as a log to compress the intensity scale. The constant component is at the center. Notice that the Fourier transform of a resampled image is obtained by scaling the Fourier transform of the original image and then tiling the plane. Interference between copies of the original Fourier transform means that we cannot recover its value at some points; this is the mechanism underlying aliasing.*

is a filter whose response is pretty close to constant for some range of low spatial frequencies—the pass band—and whose response is also pretty close to zero—for higher spatial frequencies—the stop band. It is possible to design low-pass filters that are significantly better than Gaussians. The design process involves a detailed compromise between criteria of ripple—how flat is the response in the pass band and the stop band?—and roll-off—how quickly does the response fall to zero and stay there? The basic steps for resampling an image are given in Algorithm 7.1.

7.3 FILTERS AS TEMPLATES

It turns out that filters offer a natural mechanism for finding simple patterns because filters respond most strongly to pattern elements that look like the filter. For example, smoothed derivative filters are intended to give a strong response at a point where the derivative is large. At these points, the kernel of the filter looks like the effect it is intended to detect. The x-derivative filters look like a vertical light blob next to a vertical dark blob (an arrangement where there is a large x-derivative), and so on.

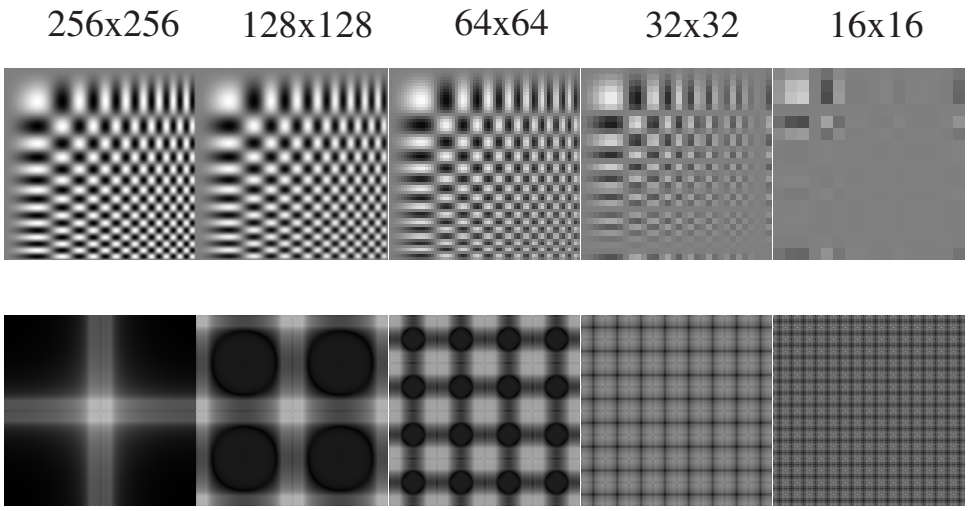

FIGURE 7.9: **Top:** *Resampled versions of the image of Figure 7.8, again by factors of two, but this time each image is smoothed with a Gaussian of σ one pixel before resampling. This filter is a low-pass filter, and so suppresses high spatial frequency components, reducing aliasing.* **Bottom:** *The effect of the low-pass filter is easily seen in these log-magnitude images; the low-pass filter suppresses the high spatial frequency components so that components interfere less, to reduce aliasing.*

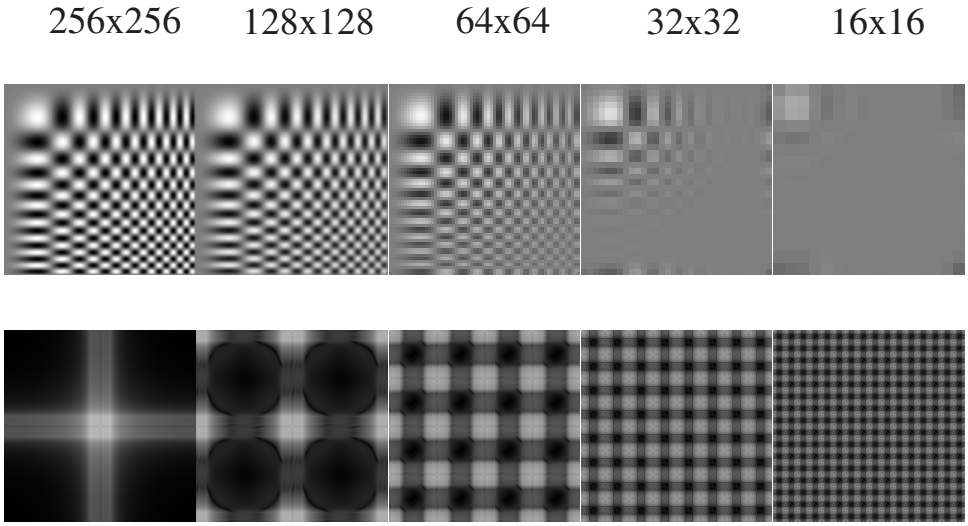

FIGURE 7.10: **Top:** *Resampled versions of the image of Figure 7.8, again by factors of two, but this time each image is smoothed with a Gaussian of σ two pixels before resampling. This filter suppresses high spatial frequency components more aggressively than that of Figure 7.9.* **Bottom:** *The effect of the low-pass filter is easily seen in these log-magnitude images; the low-pass filter suppresses the high spatial frequency components so that components interfere less, to reduce aliasing.*

Apply a low-pass filter to the original image
  (a Gaussian with a σ of between one
  and two pixels is usually an acceptable choice).
Create a new image whose dimensions on edge are half
  those of the old image
Set the value of the i, jth pixel of the new image to the value
  of the $2i$, $2j$th pixel of the filtered image

Algorithm 7.1: *Subsampling an Image by a Factor of Two.*

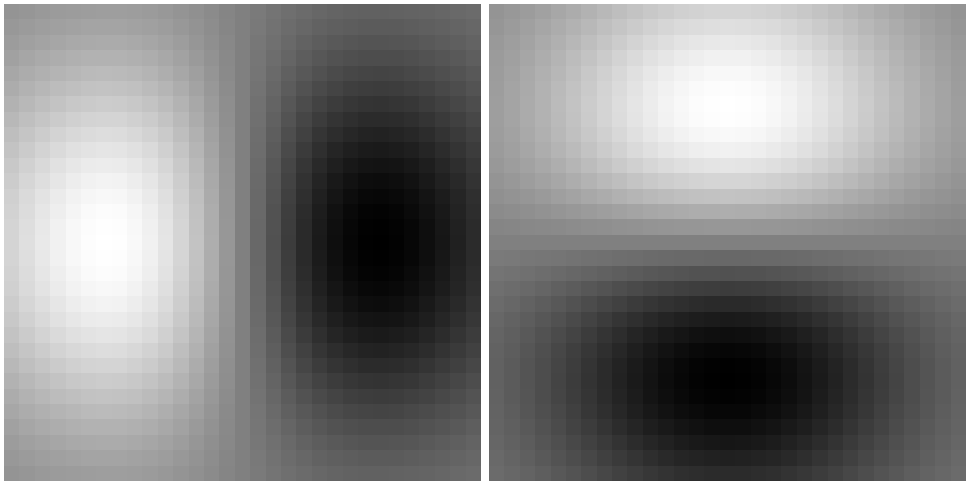

FIGURE 7.11: *Filter kernels look like the effects they are intended to detect. On the* **left**, *a smoothed derivative of Gaussian filter that looks for large changes in the x-direction (such as a dark blob next to a light blob); on the* **right**, *a smoothed derivative of Gaussian filter that looks for large changes in the y-direction.*

It is generally the case that filters intended to give a strong response to a pattern look like that pattern (Figure 7.11). This is a simple geometric result.

7.3.1 Convolution as a Dot Product

Recall from Section **??** that, for $\mathcal{G}$, the kernel of some linear filter, the response of this filter to an image $\mathcal{H}$ is given by

$$R_{ij} = \sum_{u,v} G_{i-u,j-v} H_{uv}.$$

Now consider the response of a filter at the point where i and j are zero. This is

$$R = \sum_{u,v} G_{-u,-v} H_{u,v}.$$

This response is obtained by associating image elements with filter kernel elements, multiplying the associated elements, and summing. We could scan the image into a vector and the filter kernel into another vector in such a way that associated elements are in the same component. By inserting zeros as needed, we can ensure that these two vectors have the same dimension. Once this is done, the process of multiplying associated elements and summing is precisely the same as taking a dot product.

This is a powerful analogy because this dot product, like any other, achieves its largest value when the vector representing the image is parallel to the vector representing the filter kernel. This means that a filter responds most strongly when it encounters an image pattern that looks like the filter. The response of a filter gets stronger as a region gets brighter, too.

Now consider the response of the image to a filter at some other point. Nothing significant about our model has changed. Again, we can scan the image into one vector and the filter kernel into another vector, such that associated elements lie in the same components. Again, the result of applying this filter is a dot product. There are two useful ways to think about this dot product.

7.3.2 Changing Basis

We can think of convolution as a dot product between the image and *a different vector* (because we have moved the filter kernel to lie over some other point in the image). The new vector is obtained by rearranging the old one so that the elements lie in the right components to make the sum work out. This means that, by convolving an image with a filter, we are representing the image on a new *basis* of the vector space of images—the basis given by the different shifted versions of the filter. The original basis elements were vectors with a zero in all slots except one. The new basis elements are shifted versions of a single pattern.

For many of the kernels discussed, we expect that this process will *lose* information—for the same reason that smoothing suppresses noise—so that the coefficients on this basis are redundant. This basis transformation is valuable in texture analysis. Typically, we choose a basis that consists of small, useful pattern components. Large values of the basis coefficients suggest that a pattern component is present, and texture can be represented by representing the relationships between these pattern components, usually with some form of probability model.