CHAPTER 12

Applications of the Fourier Transform

There are two applications of the Fourier transform that are important to us. First, the convolution theorem has a number of interesting and useful consequences. Second, the Fourier transform explains what is lost in sampling and why aliasing occurs.

12.1 USING THE CONVOLUTION THEOREM

The *convolution theorem* (line 12 of Table **??**) says convolution in the signal domain is the same as multiplication in the Fourier domain. This makes it possible to visualize the effect of a linear filter in the Fourier domain. Because the inverse Fourier transform is a Fourier transform (up to a flip, above), the convolution theorem works both ways. Multiplication in the signal domain is the same as convolution in the Fourier domain.

12.1.1 Making Big Filters Faster with the FFT

One application of the convolution theorem illustrates some possible difficulties building filters. Write

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{\left[x^2+y^2\right]}{2\sigma^2}\right)}$$

then

$$\mathcal{F}(g_\sigma(x,y)) = C g_{\frac{1}{2\pi\sigma}}(u,v)$$

(where the constant C depends on σ). There is a big point here: a gaussian that is spread out in x, y is concentrated in u, v, and vice versa. This is a rather distant manifestation of Heisenberg's uncertainty principle. Now consider building a low pass filter that accepts a very small range of spatial frequencies. This could be modelled as multiplying the Fourier transform of the image by a gaussian with very small σ. The convolution kernel that implements this filter is the inverse Fourier transform of this gaussian – which has very large σ. You would need a very large convolution to implement this filter without further tricks.

This suggests, correctly, that in some cases, rather than convolving the image with a filter, it is actually better: to apply an FFT to the image; multiply the result by the FFT of the filter; then apply an inverse FFT to the result. Cases where there is an efficiency gain exist, but are rather special.

Most practical applications of very large filter kernels involve very aggressive smoothing. But if an image is going to be heavily smoothed, it will lose a lot of detail, and the detailed form of the smoother might not matter much. Further, applying a very large filter kernel to smooth is very expensive. If you are willing to accept a Gaussian smoother (the usual case), significant efficiency gains are available. Recall that

$$g_{\sigma_1} * g_{\sigma_2} = g_{\sqrt{\sigma_1^2+\sigma_2^2}}$$

(**exercises**). Equivalently, smoothing with a big Gaussian is equivalent to smoothing with a smaller Gaussian, then smoothing again with that smaller Gaussian. But once you have smoothed with a Gaussian, you can subsample, suggesting that to smooth heavily, you should smooth lightly, subsample, smooth lightly again, and so on. This is the gaussian pyramid of Section 3.2.5. Each layer of the gaussian pyramid is obtained by convolving the previous layer with a gaussian, then downsampling. For the moment, ignore the downsampling, and write $\mathcal{I}$ for the image. Then layer 0 is $\mathcal{I}$ and layer N is $g_\sigma * g_\sigma * \ldots * \mathcal{I}$ which is the same as $g_{\sigma\sqrt{N}} * \mathcal{I}$. Downsampling doesn't really affect this argument (which is why I omitted it), but just makes the convolution more efficient by removing redundant values.

These scaling effects are interesting for more than just gaussians. Imagine you wish to find large stripes in a large image (which you could do by applying a large convolution kernel to that image). A natural strategy is to downsample both kernel and image, and apply the small version of the kernel to the small image. Further, you could find many different sizes of stripe efficiently by applying one stripe filter to each layer of a gaussian pyramid. Responses at the early layers give fine stripes, and at the later layers give coarse stripes.

Line 8 of the table together with the convolution theorem supports this idea. Imagine you have a filter $f(x, y)$ that detects a small pattern. Then (say) $f(x/10, y/10)$ will detect a larger version of this pattern. Now line 8 shows that the Fourier transform of this new scaled filter will shrink by a factor of 10 in u, v space. In turn, the value depends on only low spatial frequencies. In turn, not much will be lost if you apply the scaled filter to a low pass filtered version of the image. Further, applying the scaled filter to a low pass filtered version of the image will be equivalent to applying the original filter to a scaled version of the image (line 8 again). But this is equivalent to applying the original filter to a downsampled layer of the gaussian pyramid.

Remember this: *You very seldom need to apply a very large filter to an image. When you do need to, the FFT may be more efficient than convolution. If you are using a very large filter to find a very large pattern, it is more efficient to downsample the image and look for a smaller pattern in the lower resolution image.*

12.1.2 Ringing

Recall the ringing effect of Figure 3.8. Here smoothing by just computing an unweighted average managed to create some unexpected fine details. A Fourier transform offers an easy explanation – the magnitude of the Fourier transform of the unweighted averaging filter falls off much more slowly with frequency than that of

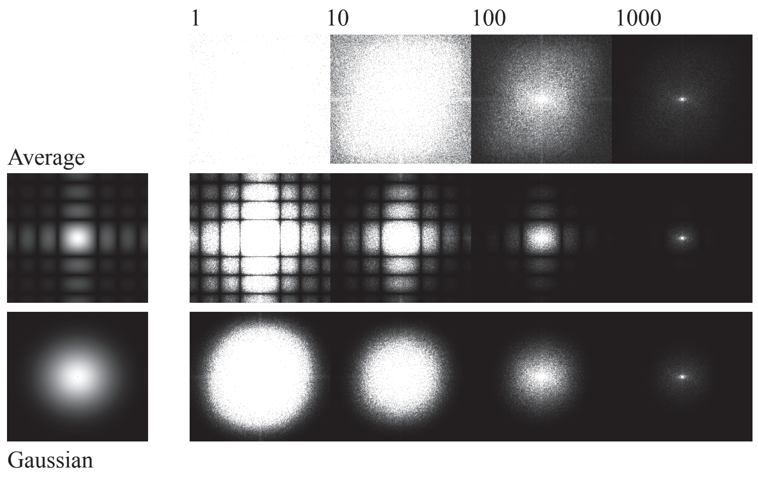

FIGURE 12.1: *A Fourier transform explains where the ringing of Figure 3.8 comes from. The top row shows the magnitude spectrum of the beard image. I have shown the actual magnitude (rather than log magnitude), because doing so makes the issue clearer. The first image shows the magnitude clipped to the range 0-1 (this is all light); the second the magnitude clipped to the range 0-10; the third, clipped to the range 0-100; and the fourth, clipped to the range 0-1000.* **Center left** *shows the magnitude of the Fourier transform of the unweighted averaging filter, and the rest of the images are the magnitude of the Fourier transform of the filtered images, clipped to the ranges as for the original image.* **Bottom left** *shows the magnitude of the Fourier transform of the Gaussian filter, and the rest of the images are the magnitude of the Fourier transform of the filtered images, clipped to the ranges as for the original image.*

the Gaussian filter. In turn, the unweighted averaging filter preserves some high frequencies, which are the ringing effect. The effect is quite difficult to see if one looks at the log of the Fourier transform magnitude, so Figure 12.1 shows the magnitude. Because the magnitude has very large dynamic range, I have shown the magnitude clipped to a variety of different ranges. Notice how the unweighted averaging filter has some high frequency terms that are much larger than Gaussian filter terms at the same frequency. These terms mean the image filtered with the unweighted average filter has considerable high frequency energy at some frequencies – these terms are the ringing.

Remember this: *Smoothing by just computing an unweighted local average creates unexpected fine details. This ringing, which occurs with other filters as well, is explained by the convolution theorem and the Fourier transform of the filter kernel.*

12.2 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 you work on a discrete pixel grid, but what? From Figure 3.4, the problem has to do with the number of samples relative to the function. You can formalize this rather precisely using Fourier transforms.

12.2.1 Modelling a Sampled Function

A crucial step is a reasonable model of a sampled function. 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*. Sampling must lose information about the original function (for example, see Figure 3.4). Accounting for what is lost requires building a model of the sampling process quite carefully.

Write

$$\texttt{sample}_{2D}(f)$$

for an operation that takes a continuous function in 2D and returns a sampled version. The sampled version should represent the values of f at all integer points (you can get any other uniform grid with a scale). It is highly desirable that $\texttt{sample}_{2D}(f)$ produce a result that is compatible with integration. In particular, that

$$\int g(u,v)\texttt{sample}_{2D}(f)dudv \approx \int g(u,v)f(u,v)dudv$$

to the extent possible for any $g(u)$. Recall the definition of the δ function in 2D (Definition 11.3). It turns out that the right choice for $\texttt{sample}_{2D}(f)$ is

$$\texttt{sample}_{2D}(f) = \sum_{ij} f(i,j)\delta(x-i, y-j)$$

The grid is infinite in each dimension to avoid having to write ranges, etc. (Figure 12.2). The δ function is a conceptual device to make the mathematical plumbing work properly. There is no need to place one at each sample function in an array inside your programs (and you can't – you'd have to have an opinion about the value of $\delta(0)$, which isn't going to work out). This definition yields a model which behaves well for integrals. In particular,

$$\int g(u)\texttt{sample}_{2D}(f)du = \sum_{ij} f(i,j)g(i,j)$$

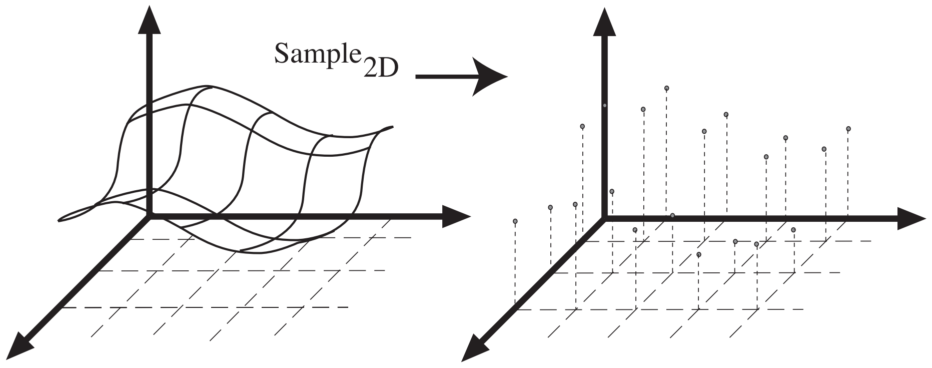

FIGURE 12.2: *Sampling in 2D takes a function and returns an array; again, we allow the array to be infinite dimensional and to have negative as well as positive indices.*

which is the best approximation to the integral that you will get if you know $f(u,v)$ only at integer points (**exercises**).

A regular grid is enough of a model for our purposes, though some practical systems have samples that are not evenly spaced. Older television sets had an aspect ratio of 4:3 (width:height), though 16:9 is commone for more recent sets. Cameras quite often accommodated this effect by spacing sample points slightly farther apart horizontally than vertically (in jargon, they had *non-square pixels*). It is unusual to encounter these effects now.

The sampling model may look strange to you, but respects convolution in an interesting way. Choose some continuous convolution kernel $g(x,y)$. If you convolve $\texttt{sample}_{2D}(\mathcal{I})$ with $g(x,y)$, then sample the result, you get what you would have gotten if you convolve $\mathcal{I}$ with $\texttt{sample}_{2D}(g)$ and sampled that **exercises** .

12.2.2 Interpolation: Passing from Discrete to Continuous

Recall the interpolate of Section 3.1 had the form

$$\mathcal{I}(x,y) = \sum_{i,j} \mathcal{I}_{ij} b(x-i, y-j).$$

Here b is some function with the properties $b(0,0) = 1$ and $b(u,v) = 0$ for u and v any other grid point. This is linear and shift invariant (**exercises**) so it must be a convolution. The way to see the convolution is to use the model of sampling, above. This exposes the convolution in interpolation. Check that

$$\begin{aligned}
\texttt{sample}_{2D}(\mathcal{I}) * b &= \int\int\sum_{ij}\mathcal{I}_{ij}\delta(x-u-i, y-v-j)b(u,v)dudv \\
&= \sum_{ij}\mathcal{I}_{ij}\int\int\delta(x-u-i, y-v-j)b(u,v)dudv \\
&= \sum_{i,j}\mathcal{I}_{ij}b(x-i, y-j) \text{ from the property of a } \delta \text{ function}
\end{aligned}$$

which is the form of an interpolate.

Remember this: *The process of sampling a function is modelled using δ functions. These ensure that integrals of the sampled function have sensible values. Interpolation is a process of convolution that takes a sampled function to a continuous function.*

12.2.3 The Fourier Transform of a Sampled Signal

As Section 12.2.2 showed, 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. You 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). 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 (**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. Formally,

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

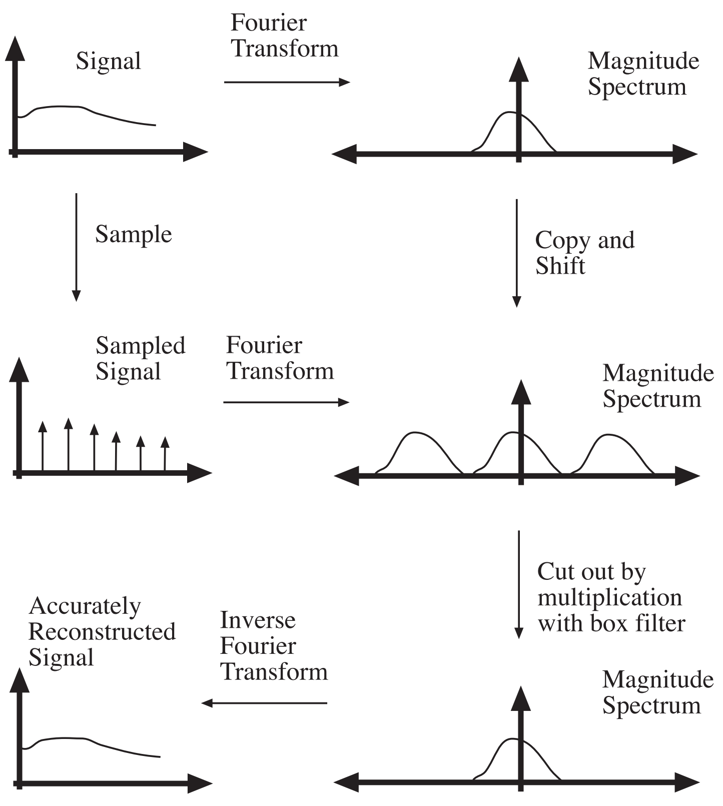

FIGURE 12.3: *The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. If the shifted copies do not intersect with each other (as in this case), the original signal can be reconstructed from the sampled signal (just cut out one copy of the Fourier transform and inverse transform it).*

(where $\mathcal{F}(f)(u-i, v-j)$ is the Fourier transform of f, evaluated at $u-i$, $v-j$). If the support of these shifted versions of the Fourier transform of the signal does not intersect, reconstructing the signal from the sampled version is straightforward. 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 12.3).

However, if the support regions *do* overlap, you are not able to reconstruct the signal because you 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 Figures 12.5, 12.6 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 you 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

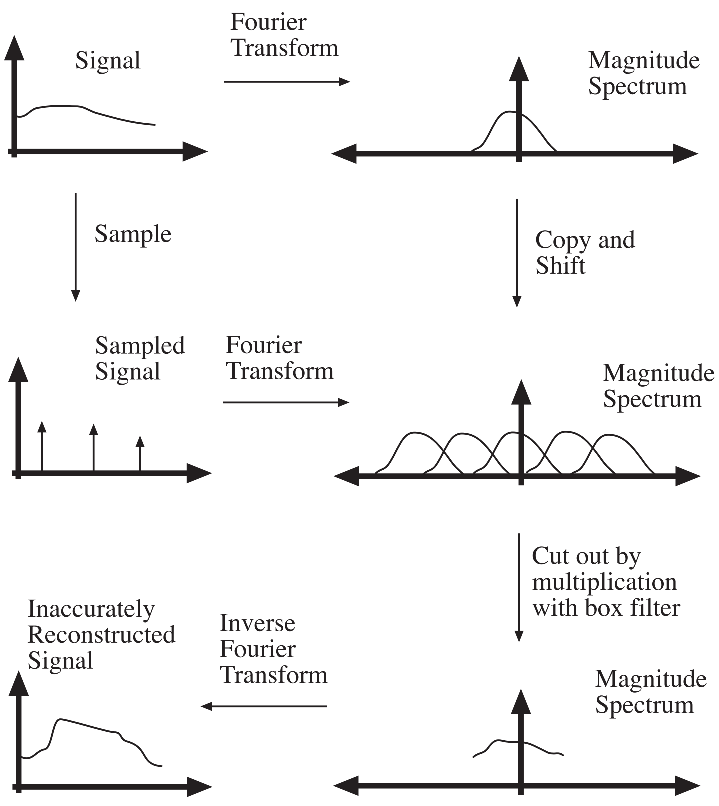

FIGURE 12.4: *The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. If the shifted copies intersect (as in this figure), the intersection region is added, and so you cannot 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.*

if we sample at a frequency of at least 2ω.

Nyquist's theorem means that, to avoid aliasing, you should either sample a continuous function at a high enough sampling rate (the Nyquist limit – twice the highest frequency present in the function) or low pass filter the function before you sample it. This filter should (at least!) remove all frequencies above half the sampling rate. You can't do this exactly, making exact reconstruction at the Nyquist limit unobtainable. You may think you could reconstruct exactly by multiplying the function's Fourier transform by a scaled 2D box function, but doing so is equivalent to convolving the function with a kernel that has infinite support (convolution theorem, and line 7 of table **??**), which is impossible.

A gaussian is a low-pass filter because its response at high spatial frequencies is low and its response at low spatial frequencies is high, so the downsampling

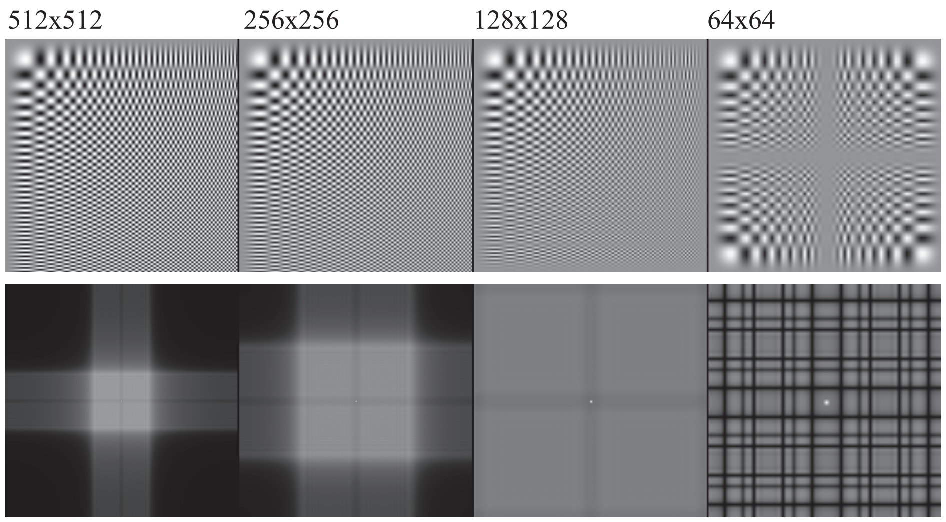

FIGURE 12.5: *The* **top row** *shows sampled versions of an image of a 512x512 grid obtained by multiplying two sinusoids with linearly increasing frequency—one in x and one in y. The other images in the series are obtained by resampling by factors of two without smoothing. These have been scaled to 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 Fourier transforms of these images, again scaled to be the same size. Notice how with downsampling by two, the Fourier transform looks like the center block of the Fourier transform of the original image. When the downsampling is more aggressive, the Fourier transform becomes very different – the overlaps are now so pronounced that the sum of shifted original Fourier transforms is very different from the original Fourier transform.*

process of Section 3.2.3 is justified. In fact, the Gaussian is not a particularly good low-pass filter. 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? Mostly, the advantages of being able to use a gaussian pyramid and the complexities of better filter design mean that, in practice, smoothing for subsampling is done with a gaussian.

12.2.4 Smoothing and Downsampling

It is easier to explain sampling and aliasing in the context of passing from a continuous signal to a sampled signal. But in practice, you have an image that has been sampled already and you want to downsample it. Nyquist's theorem applies here, too. The Fourier transform of the sampled image consists of a set of copies of some original Fourier transform, with centers shifted to integer points in u, v space. This

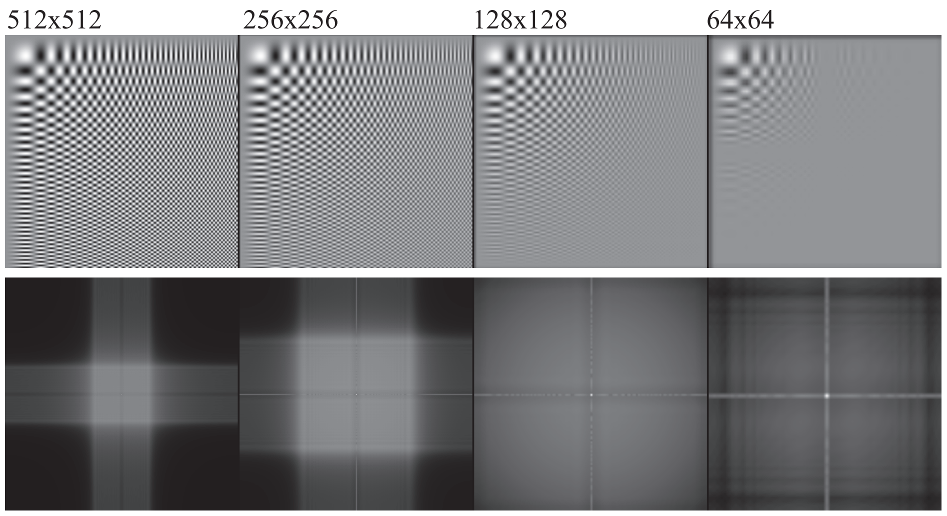

FIGURE 12.6: *The* **top row** *shows sampled versions of an image of a 512x512 grid obtained by multiplying two sinusoids with linearly increasing frequency—one in x and one in y. The other images in the series are obtained by smoothing with a Gaussian of $\sigma = 1$, then resampling by factors of two without smoothing. These have been scaled to the same size. Compare this figure with Figure 12.5. The smallest image is now a better representation of the large image. The* **bottom row** *shows the Fourier transforms of these images, again scaled to be the same size. Notice how with downsampling by two, the Fourier transform looks like the center block of the Fourier transform of the original image. When the downsampling is more aggressive, the Fourier transform still looks a bit like the center block, but now low-pass filtered.*

is true whether there is aliasing or not.

If the sampled image is downsampled by two, for example, the copies now have centers on the half-integer points in u, v space. You don't know the original Fourier transform and can only infer it from the sampled image's Fourier transform (you could use the procedure of Figure 12.3). This means that if the original sampled image has aliasing you can't get rid of it by downsampling (Figure 12.4). You can make the aliasing worse, however. If you downsample, you are moving the shifted copies closer together, and risking an overlap. To avoid this overlap, requires applying a filter that strongly reduces the content of the original Fourier transform outside the range $|u| < 1/2$, $|v| < 1/2$ *before you resample the signal.*

Remember this: *Fourier theory explains aliasing. The Fourier transform of a sampled signal is the sum of a set of shifted copies of the Fourier transform of the original signal. If these copies overlap, then you can't reconstruct the original signal from the sampled signal. If they don't, you can. To avoid aliasing, either sample often enough or apply a low-pass filter before sampling. All these observations apply to resampling as well as to sampling.*

12.3 YOU SHOULD

12.3.1 remember these definitions:

12.3.2 be able to:

- Apply the convolution theorem to explain ringing.
- Understand Nyquist's theorem
- Recognize interpolation as a convolution.

EXERCISES

QUICK CHECKS

12.1. Is $f * g * h$ the same as $g * f * h$? (use the convolution theorem).

12.2. Convolution in the Fourier domain is equivalent to what in the signal domain?

12.3. Section 12.1 has: "But if an image is going to be heavily smoothed, it will lose a lot of detail, and the detailed form of the smoother might not matter much. " Explain.

12.4. Section 12.1 has: "Imagine you have a filter $f(x, y)$ that detects a small pattern. Then (say) $f(x/10, y/10)$ will detect a larger version of this pattern." Explain.

12.5. Finding a pattern in a smoothed and downsampled version of the image is largely equivalent to finding a large version of the pattern in the original image. Explain.

12.6. Will ringing affect a gradient estimate?

12.7. Check that $\int g(u)\texttt{sample}_{2D}(f)du = \sum_{ij} f(i,j)g(i,j)$.

12.8. Section 22.6 has:"This definition yields a model which behaves well for integrals. In particular,

$$\int g(u)\texttt{sample}_{2D}(f)du = \sum_{ij} f(i,j)g(i,j)$$

which is the best approximation to the integral that you will get if you know $f(u, v)$ only at integer points." Explain.

12.9. Write an expression for what you would get if you convolve $\texttt{sample}_{2D}(\mathcal{I})$ with $g(x, y)$, then sample the result.

12.10. Write an expression for what you would get if you convolve $\mathcal{I}$ with $\texttt{sample}_{2D}(g)$, then sample the result.

12.11. Section 12.2.3 has:"convolving a function with a shifted δ-function merely shifts the function". Show this is true.

12.12. Section 12.2.4 has: "If the sampled image is downsampled by two, for example, the copies now have centers on the half-integer points in u, v space." Explain.

LONGER PROBLEMS

12.13. Write

$$g_\sigma(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{[x^2+y^2]}{2\sigma^2}\right)}.$$

(a) Show that

$$\mathcal{F}(g_\sigma(x, y)) = C g_{\frac{1}{2\pi\sigma}}(u, v).$$

What is C?

(b) Assume ξ_1 is a random variable with a normal distribution with mean 0 and standard deviation σ_1; and ξ_2 is a random variable with normal distribution with mean 0 and standard deviation σ_2. Then $\xi_1 + \xi_2$ is normal. Show that this has mean 0 and standard deviation $\sqrt{\sigma_1^2 + \sigma_2^2}$.

(c) Show that $\xi_1 + \xi_2$ is distributed as

$$\int_u g_{\sigma_1}(u) g_{\sigma_2}(x - u) du.$$

(d) Show that

$$g_{\sigma_1} * g_{\sigma_2} = g_{\sqrt{\sigma_1^2+\sigma_2^2}}.$$

12.14. The δ function isn't really a function (what value does it take at 0?) but it can be seen as a limit of a variety of functions. As one example, show that

$$\lim_{\epsilon \to 0} \int \left(\frac{1}{\epsilon^2} \mathrm{box}_{2D}(x/\epsilon, y/\epsilon) \right) f(x,y) dx dy = f(0,0)$$

(assuming that the limit exists).

12.15. **(a)** Check

$$\texttt{sample}_{2d}(\mathcal{I}) * g = \sum_{i,j} \mathcal{I}_{ij} g(x-i, y-j).$$

(b) Check

$$\mathcal{I} * \texttt{sample}_{2d}(g) = \sum_{i,j} \mathcal{I}_{ij} g(x-i, y-j).$$

(c)

PROGRAMMING EXERCISES

12.16. There are (at least!) two ways to apply a gaussian filter to an image. You could convolve with the gaussian, or you could FFT the image, multiply the result by a different gaussian, then inverse FFT. You expect that, for reasonably small σ, convolution is faster than the FFT strategy. It is possible that if σ is large enough, the FFT strategy is better. Use whatever API appeals (I used numpy) to compare the computational costs. Is there a value of σ for your API where the FFT strategy is faster? How does the size of the image affect this value?

12.17. Find an image with fine scale spatial details like the beard of Figure 3.8, and use it to reproduce and explain Figure 12.1.