CHAPTER 5

Convolution in Detail

5.1 THE PROPERTIES OF CONVOLUTION

Most imaging systems have, to a good approximation, three significant properties. Write $R(f)$ for the response of the system to input f. Then the properties are:

- **Superposition:** the response to the sum of stimuli is the sum of the individual responses, so
$$R(f+g) = R(f) + R(g);$$
- **Scaling:** the response to a scaled stimulus is a scaled version of the response to the original stimulus, so
$$R(kf) = kR(f).$$
An operation that exihibits superposition and scaling is *linear*.
- **Shift invariance:** In a shift invariant linear system, the response to a translated stimulus is just a translation of the response to the stimulus. This means that, for example, if a view of a small light aimed at the center of the camera is a small, bright blob, then if the light is moved to the periphery, the response is same small, bright blob, only translated.

A device that is linear and shift invariant is known as a *shift invariant linear system*. The operation represented by the device is a *shift invariant linear operation*.

Some systems accept a continuous signal and produce a continuous signal. A natural example is a lens, which takes a pattern of light and produces a pattern of light. Others accept a discrete signal (a vector; an array) and produce a discrete signal. A natural example would be smoothing with a Gaussian. Either kind of system can be linear, and either kind of system can be shift invariant. It turns out that any operation that is shift invariant and linear can be represented by a convolution (Section **??** has the details). I have already shown an expression for convolution for discrete signals. There is an analogous expression for convolution for continuous signals, developed below. Mostly, I will overload the term convolution and the $*$ operation to refer to either case.

Convolution is:

- linear, by construction;
- shift-invariant, by construction;
- *commutative* (meaning
$$(g * h)(x) = (h * g)(x)$$
exercises);

- *associative* (meaning that

$$(f * (g * h)) = ((f * g) * h)$$

exercises).

5.1.1 Application: Derivative of Gaussian Filters

Because convolution is associative, smoothing an image and then differentiating it is the same as convolving it with the derivative of a smoothing kernel. First, differentiation is linear and shift invariant. This means that there is some kernel that differentiates. Given a function $I(x, y)$,

$$\frac{\partial I}{\partial x} = K_{(\partial/\partial x)} * I.$$

Write the convolution kernel for the smoothing as S. Now

$$(K_{(\partial/\partial x)} * (S * I)) = (K_{(\partial/\partial x)} * S) * I = (\frac{\partial S}{\partial x}) * I.$$

Usually, the smoothing function is a gaussian, so an estimate of the derivative can be obtained by convolving with the derivative of the gaussian (rather than convolve and then differentiate), yielding

$$\begin{aligned}
\frac{\partial g_\sigma}{\partial x} &= \frac{1}{2\pi\sigma^2}\left[\frac{-x}{2\sigma^2}\right]\exp -\left(\frac{x^2+y^2}{2\sigma^2}\right) \\
\frac{\partial g_\sigma}{\partial y} &= \frac{1}{2\pi\sigma^2}\left[\frac{-y}{2\sigma^2}\right]\exp -\left(\frac{x^2+y^2}{2\sigma^2}\right)
\end{aligned}$$

As they should (Section **??**), derivative of gaussian filters look like the effects they are 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.

Large gradients in images are interesting (Chapters **??** and **??**) because they tend to occur on the outlines of objects, at shadow boundaries, and so on. Generally, if there is a large x derivative at a pixel, there will be a large x derivative at neighboring pixels. Smoothing *across* the direction of the derivative may result in smeared or blurred derivatives; but smoothing *along* the direction of the derivative will tend to average the value at points with similar derivatives and improve the noise resistance. It is quite usual to use

$$\begin{aligned}
\frac{\partial g_\sigma}{\partial x} &= \frac{1}{2\pi\sigma_s\sigma_b}\left[\frac{-x}{2\sigma_s^2}\right]\exp -\left(\frac{x^2}{2\sigma_b^2}+\frac{y^2}{2\sigma_s^2}\right) \\
\frac{\partial g_\sigma}{\partial y} &= \frac{1}{2\pi\sigma_s\sigma_b}\left[\frac{-y}{2\sigma_s^2}\right]\exp -\left(\frac{x^2}{2\sigma_b^2}+\frac{y^2}{2\sigma_s^2}\right)
\end{aligned}$$

where $\sigma_b > \sigma_s$. Smoothing results in much smaller noise responses from the derivative estimates, and more smoothing yields less noisy, but more blurry, gradients (Figure 5.1 and 5.2).

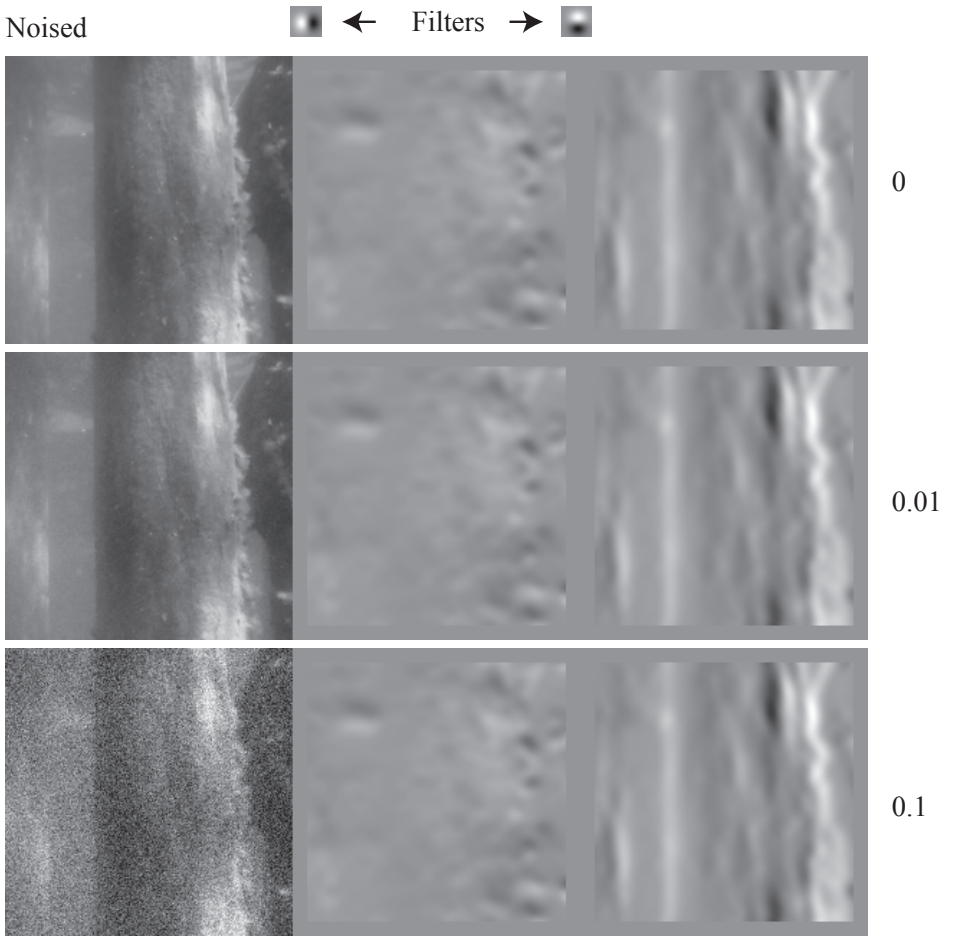

FIGURE 5.1: *Derivative of gaussian filters, equivalent to applying a finite difference to a smoothed image, very significantly improve estimates of derivatives. Compare this figure with Figure 4.5.* **Rows** *show image, horizontal derivative and vertical derivative, where derivatives are estimated by convolution with difference of gaussian filters. The filters are shown at the top. As you should expect, one looks like a dark bar next to a light bar, the other looks like a dark bar below a light bar. As should be apparent from the filters, the smoothing is the same in each direction.* **First row** *is noise free image; others have additive Gaussian noise added, with standard deviation shown on the* **right***. Notice how this noise hardly affects derivatives. The derivatives are scaled so that positive values are bright, negative values are dark, and 0 is mid-range. Although the scale is chosen* per row*, the derivative images look the same from row to row – this means that each row has about the same largest magnitude value.* Image credit: *Figure shows Robert Forsyth's photograph of historical dock pilings in Lake Michigan.*

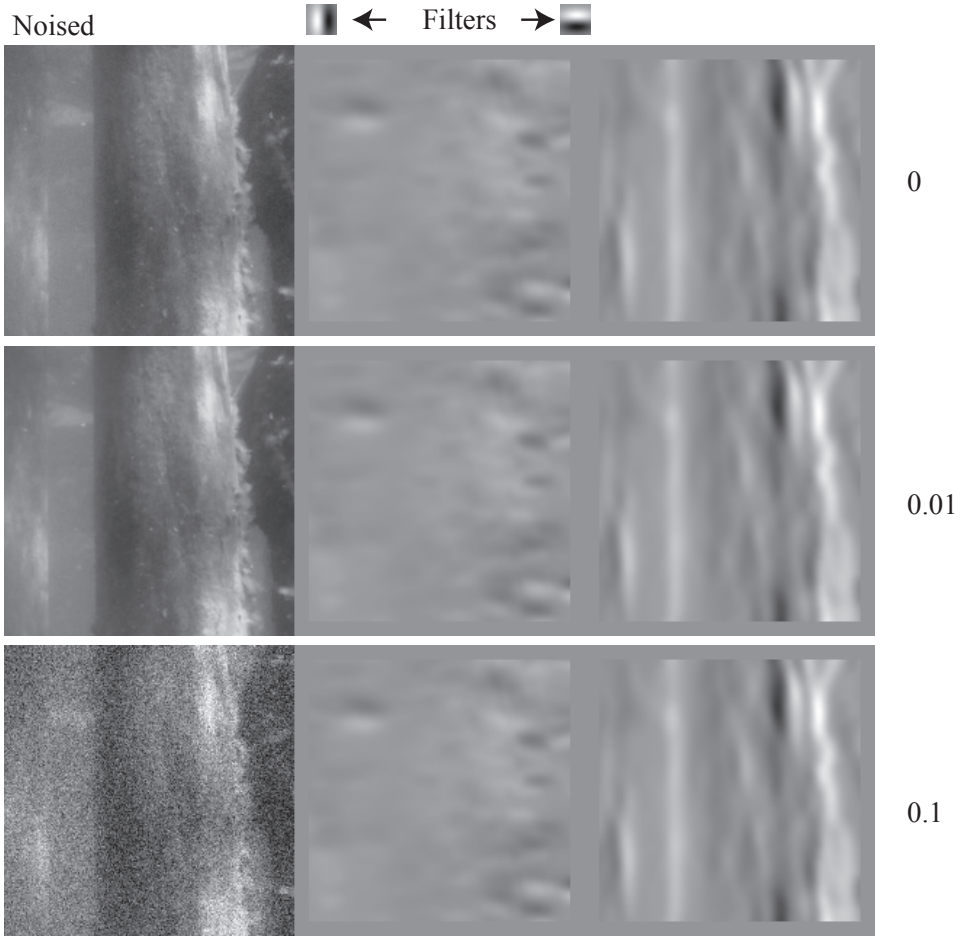

FIGURE 5.2: *Derivative of gaussian filters, equivalent to applying a finite difference to a smoothed image, very significantly improve estimates of derivatives. Compare this figure with Figure 4.5. As should be apparent from the filters, the smoothing is over a much larger range along the derivative direction than across it (compare Figure 5.1).* Image credit: *Figure shows Robert Forsyth's photograph of historical dock pilings in Lake Michigan.*

5.2 SHIFT INVARIANT LINEAR ≡ CONVOLUTION

5.2.1 Shift Invariant Linear ≡ Convolution, 1D Discrete Case

I have already shown convolution is shift invariant and linear. In fact, it is easily shown by construction that the response of a shift invariant linear system to a stimulus is obtained by convolution. In the 1D case, a shift invariant linear system takes a vector and responds with a vector. This case is the easiest to handle because there are fewer indices to look after. The 2D case—a system that takes an array

and responds with an array—follows easily. In each case, assume that the input and output are infinite dimensional, so there are no issues at the boundaries of the input.

For the 1D case, the input vector is $\mathbf{f}$. For convenience, assume that the vector has infinite length and its elements are indexed by the integers (i.e., there is an element with index -1, say). The ith component of this vector is f_i. Now $\mathbf{f}$ is a weighted sum of basis elements. A convenient basis is a set of elements that have a one in a single component and zeros elsewhere. We write

$$\mathbf{e}_0 = \ldots 0, 0, 0, 1, 0, 0, 0, \ldots$$

This is a data vector that has a 1 in the zeroth place, and zeros elsewhere. Define a shift operation, which takes a vector to a shifted version of that vector. In particular, the vector $\texttt{Shift}(\mathbf{f}, i)$ has, as its jth component, the $j - i$th component of $\mathbf{f}$. For example, $\texttt{Shift}(\mathbf{e}_0, 1)$ has a zero in the first component (and so is $\mathbf{e}_1$). Then

$$\mathbf{f} = \sum_i f_i \texttt{Shift}(\mathbf{e}_0, i).$$

Because the system is linear and shift invariant

$$\begin{aligned} R(\mathbf{f}) &= R\left(\sum_i f_i \texttt{Shift}(\mathbf{e}_0, i)\right) \\ &= \sum_i R(f_i \texttt{Shift}(\mathbf{e}_0, i)) \\ &= \sum_i f_i R(\texttt{Shift}(\mathbf{e}_0, i)) \text{ (the system is linear)} \\ &= \sum_i f_i \texttt{Shift}(R(\mathbf{e}_0), i)). \text{ (the system is shift invariant)} \end{aligned}$$

So the system's response to any data vector can be obtained from its response to $\mathbf{e}_0$. This is usually called the system's *impulse response*. Write $\mathbf{g}$ for the impulse response, yielding

$$R(\mathbf{f}) = \sum_i f_i \texttt{Shift}(\mathbf{g}, i) = \mathbf{g} * \mathbf{f}.$$

This defines an operation—the 1D, discrete version of convolution—which we write with a $*$. This doesn't yield a particularly easy expression for the output. Write R_j for the jth element of $R(\mathbf{f})$, and find

$$R_j = \sum_i g_{j-i} f_i.$$

5.2.2 Shift Invariant Linear ≡ Convolution, 2D Discrete Case

Now use an array of values and write the i, jth element of the array $\mathcal{D}$ as D_{ij}. The appropriate analogy to an impulse response is the response to a stimulus that looks

like

$$\mathcal{E}_{00} = \begin{matrix} \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & 0 & 0 & \dots \\ \dots & 0 & 1 & 0 & \dots \\ \dots & 0 & 0 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{matrix}$$

If $\mathcal{G}$ is the response of the system to this stimulus, the same considerations as for 1D convolution yield a response to a stimulus $\mathcal{F}$, that is,

$$R_{ij} = \sum_{u,v} G_{i-u,j-v} F_{uv}, = \mathcal{G} * \mathcal{F}$$

It should be clear that this construction works for any dimension, with minor adjustments to indexing, etc. So, *by construction*, any shift-invariant linear operation applied to a discrete N-d signal can be represented as convolution with some kernel.

5.2.3 Shift Invariant Linear ≡ Convolution, 1D Continuous case

Understanding aliasing requires understanding what happened when a continuous signal was sampled to produce a discrete image. It turns out that shift-invariant linear operations applied to continuous signals are important. All the logic of the previous section applies, but some new notation is required. The key question is obtaining some object analogous to $\mathbf{e}_0$ in the discrete case. It turns out this object is a rather unnatural function, the δ-function (which is not a function in formal terms). The process will be:

- Represent a continous input function as a weighted sum of a set of narrow boxes, shifted to sit on discrete sample points. The weights form a discrete vector.
- Apply the expressions for discrete convolution to represent the output.
- Now make the boxes narrow, and consider the limit.

Representing the input function: Define the `box` function by:

$$\texttt{box}_\epsilon(x) = \begin{cases} 0 & abs(x) > \frac{\epsilon}{2} \\ 1 & abs(x) < \frac{\epsilon}{2} \end{cases} .$$

The value of $box_\epsilon(\epsilon/2)$ does not matter. The input function is $f(x)$. Construct an even grid of points x_i, where $x_{i+1} - x_i = \epsilon$. Now construct a vector $\mathbf{f}$ whose ith component (written f_i) is $f(x_i)$. Define `Shift` operator, which takes functions to functions:

$$\texttt{Shift}(f, c) = f(u - c).$$

This vector can be used to represent the function, f as

$$\sum_i f_i \texttt{Shift}(box_\epsilon, x_i).$$

Apply discrete convolution: A shift invariant linear system applied to this function will yield a weighted sum of shifted responses to box functions, so

$$\begin{aligned} R\left(\sum_i f_i \texttt{Shift}(\texttt{box}_\epsilon, x_i)\right) &= \sum_i R(f_i \texttt{Shift}(\texttt{box}_\epsilon, x_i)) \\ &= \sum_i f_i R(\texttt{Shift}(\texttt{box}_\epsilon, x_i)) \\ &= \sum_i f_i \texttt{Shift}(R(\frac{\texttt{box}_\epsilon}{\epsilon}\epsilon), x_i) \\ &= \sum_i f_i \texttt{Shift}(R(\frac{\texttt{box}_\epsilon}{\epsilon}), x_i)\epsilon. \end{aligned}$$

So far, everything has followed our derivation for discrete functions. The result looks like an approximate integral if $\epsilon \to 0$, assuming that $\frac{\texttt{box}_\epsilon}{\epsilon}$ can be interpreted.

A new device, called a δ-function, deals with the term box_ϵ/ϵ. Define

$$d_\epsilon(x) = \frac{box_\epsilon(x)}{\epsilon}.$$

The δ-function is:

$$\delta(x) = \lim_{\epsilon \to 0} d_\epsilon(x).$$

There is no need to discuss the value of $\delta(0)$, but notice that

$$\begin{aligned} \int_{-\infty}^{\infty} \delta(x) f(x) dx &= \lim_{\epsilon \to 0} \int_{-\infty}^{\infty} d_\epsilon(x) f(x) dx \\ &= \lim_{\epsilon \to 0} \int_{-\infty}^{\infty} \frac{\texttt{box}_\epsilon(x)}{\epsilon} (f(x)) dx \\ &= \lim_{\epsilon \to 0} \sum_{i=-\infty}^{\infty} \frac{\texttt{box}_\epsilon(x)}{\epsilon} (f(i\epsilon) \texttt{box}_\epsilon(x - i\epsilon)\epsilon \\ &= f(0). \end{aligned}$$

The δ-function is the natural analogue for $\mathbf{e}_0$ in the continuous case. One interesting feature of this function is that, for practical shift invariant linear systems, the response of the system to a δ-function exists and has *compact support* (i.e., is zero except on a finite number of intervals of finite length). For example, a good model of a δ-function in 2D is an extremely small, extremely bright light. The result of making the light smaller and brighter while ensuring the total energy is constant is a small but finite spot due to the defocus of the lens.

This means that the expression for the response of the system,

$$\sum_i f_i \texttt{Shift}(R(\frac{box_\epsilon}{\epsilon}), x_i)\epsilon,$$

turns into an integral as ϵ limits to zero. The integral is

$$\begin{aligned} R(f) &= \int \{R(\delta)(u - x')\} f(x')dx' \\ &= \int g(u - x')f(x')dx', \end{aligned}$$

where $R(\delta)$—which is usually called the **impulse response** of the system— was written g. The limits of the integral could be from $-\infty$ to ∞, but more stringent limits could apply if g and h have compact support. This operation is called **convolution** (again), and can be written

$$R(f) = (g * f).$$

5.2.4 Shift Invariant Linear ≡ Convolution, 2D Continuous case

The derivation of convolution in two dimensions requires a little more notation. A box function is now given by $box_{\epsilon^2}(x, y) = box_\epsilon(x)box_\epsilon(y)$ and

$$d_\epsilon(x, y) = \frac{box_{\epsilon^2}(x, y)}{\epsilon^2}.$$

The δ-function is the limit of $d_\epsilon(x, y)$ function as $\epsilon \to 0$. Again, discussion of the value of $\delta(0)$ is better avoided, but notice

$$\int_{-\infty}^{\infty} g(x, y)\delta(x, y)dxdy = g(0, 0).$$

Finally, there are more terms in the sum. All this activity results in the expression

$$\begin{aligned} R(h)(x, y) &= \int\int g(x - x', y - y')h(x', y')dxdy \\ &= (g * h)(x, y), \end{aligned}$$

The impulse response of a 2D shift-invariant linear system is sometimes called its *point spread function.*

5.3 SAMPLING, INTERPOLATION AND CONVOLUTION

5.3.1 Sampling: Passing from Continuous to Discrete

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.* For sampling in one dimension, the most important case involves sampling on a uniform discrete grid. Assume that the samples are defined at integer points, yielding a process that takes some function and returns a vector of values:

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

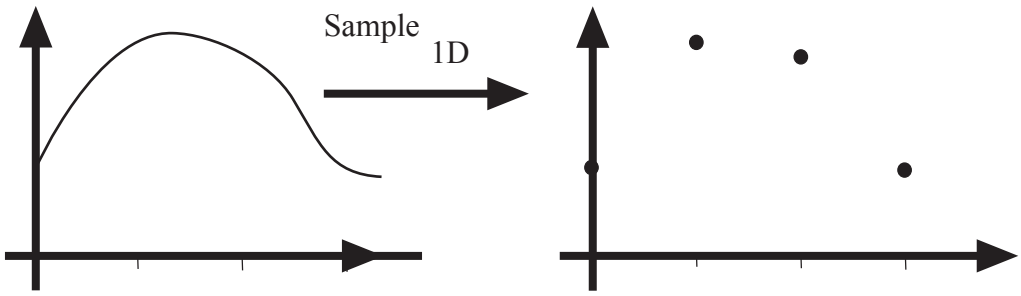

FIGURE 5.3: *Sampling in 1D takes a function and returns a vector whose elements are values of that function at all integer points. The vector is infinite to avoid having to write indices, etc.*

Here the ith component of $\mathbf{f}$ is $f(x_i)$, and $\mathbf{f}$ is an infinite vector to avoid having to write indices, etc. (Figure 43.2).

Sampling in 2D is very like sampling in 1D. Although sampling can occur on nonregular grids (the best example being the human retina), the most important case has samples on a uniform grid of integer coordinates. This gives

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

where the i, jth element of the array $\mathcal{F}$ is $F(x_i, y_j) = F(i,j)$. The grid is infinite in each dimension to avoid having to write ranges, etc. (Figure 43.4). Notice that the kernel of Section 2.3.3 and 4.2.3 is a sampled version of a continuous function. That kernel was

$$w_{i,j} = \frac{\exp - \left(\frac{(i-k)^2 + (j-k)^2}{2\sigma^2} \right)}{C}$$

which is $\texttt{sample}_{2D}(g_\sigma(x,y))$ for

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

(which is the probability density function of an isotropic bivariate normal distribution with mean at $(0,0)$).

Samples are not always evenly spaced in practical systems. This is quite often due to the pervasive effect of television; older television sets had an aspect ratio of 4:3 (width:height), though 16:9 is commone for more recent sets. Cameras quite often accommodate this effect by spacing sample points slightly farther apart horizontally than vertically (in jargon, they have *non-square pixels*).

5.3.2 A Continuous Model of a Sampled Function

Understanding aliasing will require a continuous model of a sampled signal. Write $C(\mathcal{I})$ for the operation that maps a sampled image $\mathcal{I}$ to this continuous model. This

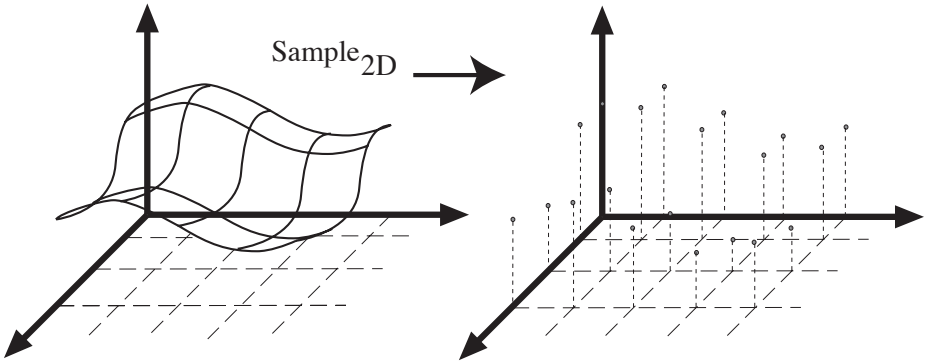

FIGURE 5.4: *Sampling in 2D takes a function and returns an array; again, we allow the array to be infinite dimensional and to have negative as well as positive indices.*

model should respect convolution and sampling in a sensible way. Choose some continuous convolution kernel $g(x, y)$ A desirable property of this model is that if you convolve $C(\mathcal{I})$ with $g(x, y)$, then sample the result, you get what you would have gotten if you convolve $\mathcal{I}$ with $\texttt{sample}_{2D}(g)$. To write this out, it is helpful to distinguish discrete convolution (I will write $*_d$) and continuous convolution (I will write $*_c$). The property is:

$$\texttt{sample}_{2D}(C(\mathcal{I}) *_c g) = \mathcal{I} *_d \texttt{sample}_{2D}(g).$$

Now $C(\mathcal{I}$ cannot just be a function that takes the value of the signal at integer points and is zero everywhere else, because this model has a zero integral so the left hand side will be zero. Instead, use

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

and find

$$C(\mathcal{I}) *_c g = \sum_{i,j} \mathcal{I}_{ij} g(x - x_i, y - y_j)$$

so that the u, v'th component of

$$\texttt{sample}_{2D}(C(\mathcal{I}) *_c g) \text{ is } \sum_{i,j} \mathcal{I}_{ij} g(x_u - x_i, y_v - y_j)$$

and the property holds.

5.3.3 Interpolation: Passing from Discrete to Continuous

Recall the interpolate of Section 2.2 had the form

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

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 continuous model of the sampled image. This exposes the convolution in interpolation. Notice that

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

which is the form of an interpolate.