Chapter 8

LINEAR FILTERS

Pictures of zebras and of dalmatians have black and white pixels, and in about the same number, too. The differences between the two have to do with the characteristic appearance of small groups of pixels, rather than individual pixel values. In this chapter, we introduce methods for obtaining descriptions of the appearance of a small group of pixels.

Our main strategy will be to use weighted sums of pixel values, using different patterns of weights to find different image patterns. This process, despite its simplicity, is extremely useful. It allows us to smooth noise in images, and to find edges and other image patterns. We discuss noise in some detail. We also describe some useful non-linear functions of image neighbourhoods.

8.1 Linear Filters and Convolution

Construct a new array, the same size as the image. Fill each location of this new array with a weighted sum of the pixel values from the locations surrounding the corresponding location in the image, *using the same set of weights each time.* The result of this procedure is **shift-invariant** — meaning that the value of the output depends on the pattern in an image neighbourhood, rather than the position of the neighbourhood — and **linear** — meaning that the output for the sum of two images is the same as the sum of the outputs obtained for the images separately. The procedure itself is known as **linear filtering**.

8.1.1 Convolution

We introduce some notation at this point. The pattern of weights used for a linear filter is usually referred to as the **kernel** of the filter. The process of applying the filter is usually referred to as **convolution**. There is a catch: for reasons that will appear later (section 8.2.1), it is convenient to write the process in a non-obvious way.

In particular, given a filter kernel $\mathcal{H}$, the convolution of the kernel with image

$\mathcal{F}$ is an image $\mathcal{R}$. The i, j'th component of $\mathcal{R}$ are given by:

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

This form is similar to the form given for blurring by taking a local average in section 8.1.2, though you will need to change the variables in the summation to demonstrate this.

8.1.2 Example: Smoothing by Averaging

Images typically have the property that the value of a pixel is usually similar to that of its neighbour. Assume that the image is affected by noise of a form where we can reasonably expect that this property is preserved; for example, there might be occasional dead pixels, or small random numbers are added to the pixel values. It is natural to attempt to reduce the effects of this noise by replacing each pixel with a weighted average of its neighbours, a process often referred to as **smoothing** or **blurring**.

At first guess, we could model the blurring process as replacing the value of a function at each point with an unweighted (or uniform) average taken over a fixed region. For example, we could average all pixels within a $2k+1 \times 2k+1$ block of the pixel of interest. For an input image $\mathcal{F}$, this gives an output

$$\mathcal{R}_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{u=i+k} \sum_{v=j-k}^{v=j+k} \mathcal{F}_{uv}$$

This is convolution with a kernel that is a $2k+1 \times 2k+1$ block of ones, multiplied by a constant.

This process is a poor model of blurring — its output does not look like that of a defocussed camera (figure 8.1. The reason is clear. Assume that we have an image in which every point but the center point was zero, and the center point was one. If we blur this image by forming an unweighted average at each point, the result will look like a small bright box — but this is not what defocussed cameras do. We want a blurring process that takes a very small bright dot to a circularly symmetric region of blur, brighter at the center than at the edges and fading slowly to darkness. As figure 8.1 suggests, a set of weights of this form produces a much more convincing defocus model.

8.1.3 Example: Smoothing with a Gaussian

A good formal model for this fuzzy blob is the **symmetric Gaussian kernel**

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

illustrated in figure 8.2. σ is referred to as the standard deviation of the Gaussian (or its "sigma"!); the units are inter-pixel spaces, usually referred to as pixels. The

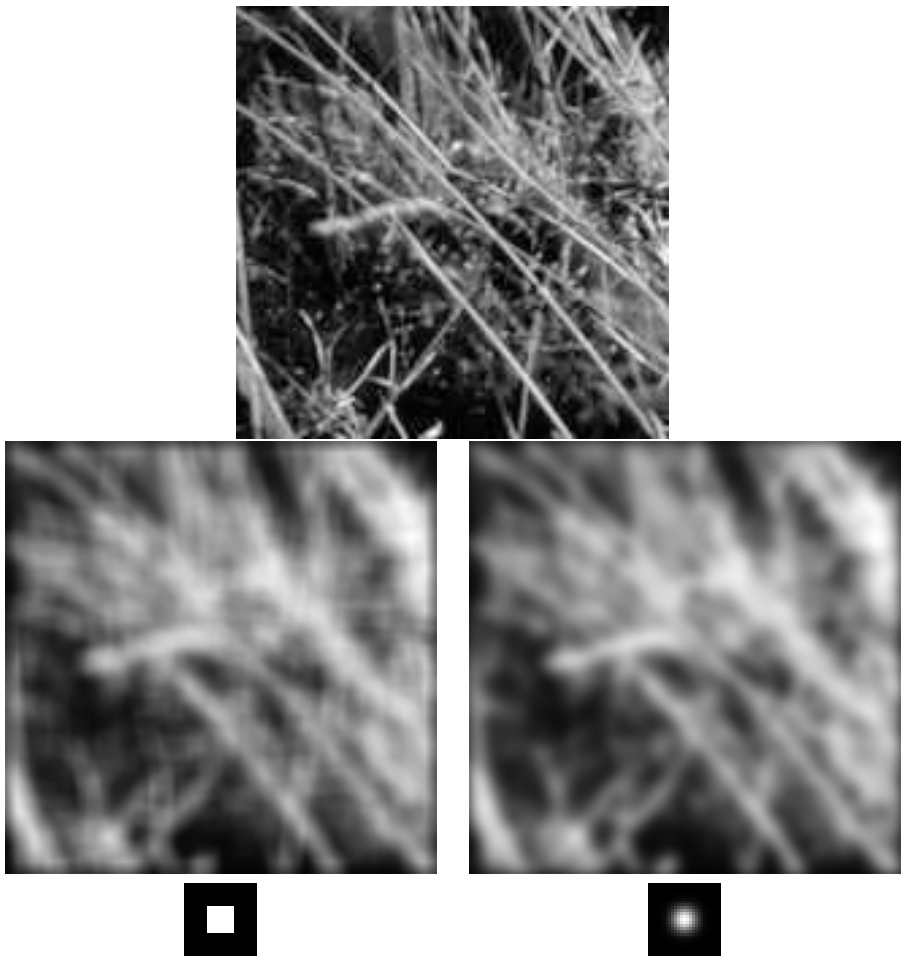

Figure 8.1. Although a uniform local average may seem to give a good blurring model, it generates effects that are not usually seen in defocussing a lens. The images above compare the effects of a uniform local average with weighted average. The image at the top shows a view of grass. On the left in the second row, the result of blurring this image using a uniform local model and on the right, the result of blurring this image using a set of Gaussian weights. The degree of blurring in each case is about the same, but the uniform average produces a set of narrow vertical and horizontal bars — an effect often known as **ringing**. The bottom row shows the weights used to blur the image, themselves rendered as an image; bright points represent large values and dark points represent small values (in this example the smallest values are zero).

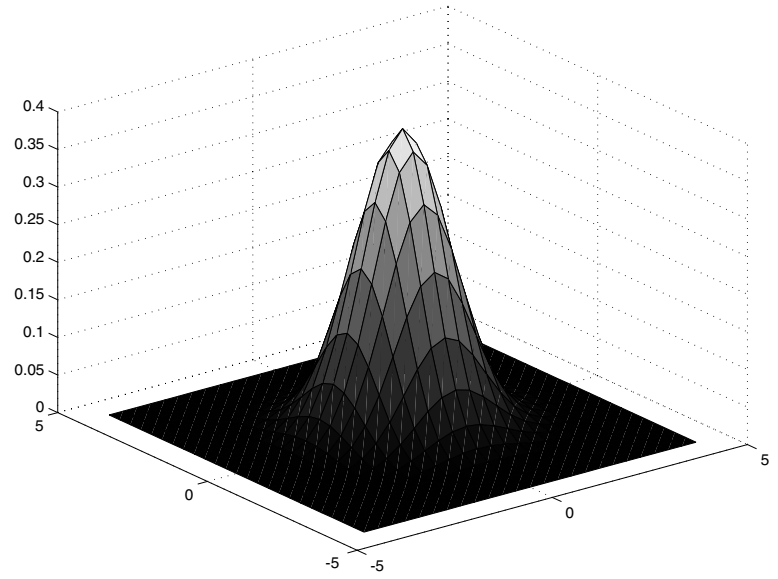

Figure 8.2. The symmetric Gaussian kernel in 2D. This view shows a kernel scaled so that its sum is equal to one; this scaling is quite often omitted. The kernel shown has $\sigma = 1$. Convolution with this kernel forms a weighted average which stresses the point at the center of the convolution window, and incorporates little contribution from those at the boundary. Notice how the Gaussian is qualitatively similar to our description of the point spread function of image blur; it is circularly symmetric, has strongest response in the center, and dies away near the boundaries.

constant term makes the integral over the whole plane equal to one and is often ignored in smoothing applications. The name comes from the fact that this kernel has the form of the probability density for a 2D Gaussian random variable with a particular covariance.

This smoothing kernel forms a weighted average, that weights pixels at its center much more strongly than at its boundaries. One can justify this approach qualitatively: smoothing suppresses noise by enforcing the requirement that pixels should look like their neighbours; but the requirement that a pixel look like its neighbours should be less strongly imposed for distant neighbours — we can do this by down-weighting distant neighbours in the average (figure 8.3 illustrates that Gaussian smoothing is effective at suppressing noise).

In applications, a discrete smoothing kernel can be obtained by constructing a $2k + 1 \times 2k + 1$ array whose i, j'th value is

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{((i-k-1)^2 + (j-k-1)^2)}{2\sigma^2}\right)$$

Notice that some care must be exercised with σ; if σ is too small, then only one element of the array will have a non-zero value. If σ is large, then k must be large, too, otherwise we will be ignoring contributions from pixels that should contribute with substantial weight. A particular property of smoothing with this kernel is that if one smoothes an image containing all zeros, except for a single one at the center, the response will be a Gaussian (exercises). This means that the kernel is behaving like a defocussed lens — taking a bright spot to a fuzzy blob.

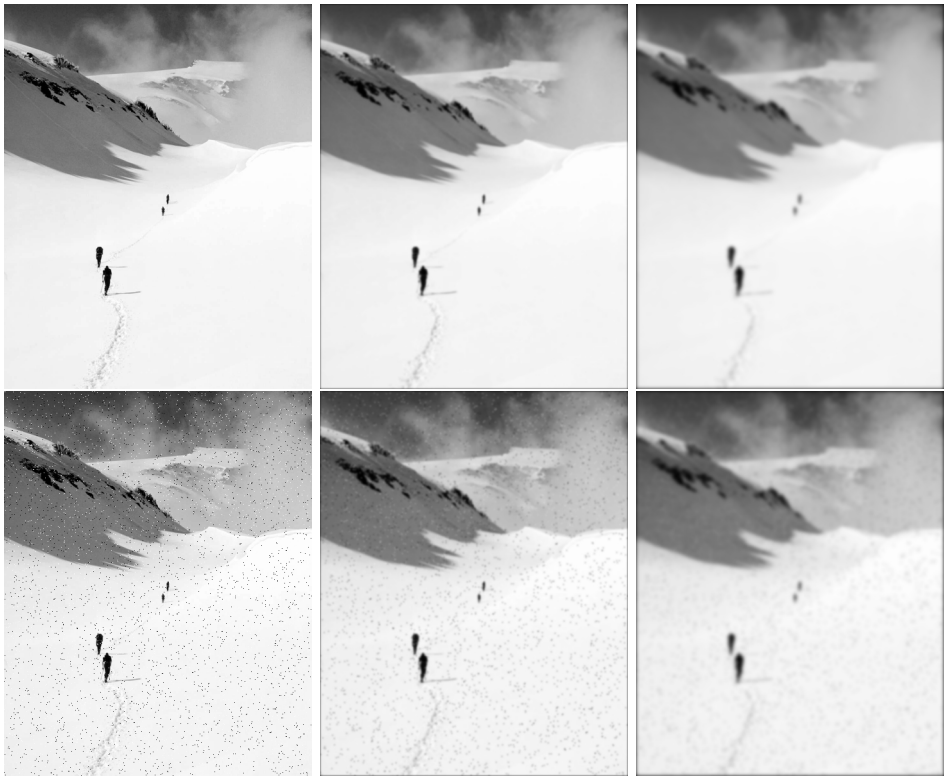

Figure 8.3. In salt-and-pepper noise, we choose pixels uniformly at random, and uniformly at random make them either black or white. Gaussian smoothing is particularly effective at suppressing the effects of salt-and-pepper noise. The top row shows an image, and versions smoothed by a symmetric Gaussian with σ two pixels and four pixels. The second row shows an image corrupted by this noise model and corresponding smoothed versions of the image. Notice that, as the smoothing increases, detail is lost, but the effects of the noise diminish, too — the smoothed versions of the noisy images look very much like the smoothed version of the noise-free images.

Convolving an image with a Gaussian is, in essence, the same as forming a weighted local average. A qualitative analysis gives:

- if standard deviation of the Gaussian is very small — say smaller than one pixel — the smoothing will have very little effect, because the weights for all pixels off the center will be very small;

- for a larger standard deviation, the neighbouring pixels will have larger weights in the weighted average, which means in turn that the average will be strongly biased toward a consensus of the neighbours — this will be a good estimate of a pixel's value, and the noise will largely disappear, at the cost of some blurring;

- finally, a kernel that has very large standard deviation will cause much of the image detail to disappear along with the noise.

8.2 Shift invariant linear systems

Most imaging systems have, to a good approximation, three significant properties:

- **Superposition:** we expect that

$$R(f+g) = R(f) + R(g)$$

  that is, the response to the sum of stimuli is the sum of the individual responses.

- **Scaling:** the response to a zero input is zero. Taken with superposition, we have that the response to a scaled stimulus is a scaled version of the response to the original stimulus, i.e.

$$R(kf) = kR(f)$$

  A device that exihibits superposition and scaling is **linear**.

- **Shift invariance:** in a **shift invariant** 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, we should see the same small bright blob, only translated.

A device that is linear and shift invariant is known as a **shift invariant linear system**, or often just as a **system**.

The response of a shift invariant linear system to a stimulus is obtained by convolution. We will demonstrate this first for systems that take discrete inputs — say vectors or arrays — and produce discrete outputs. We then use this to describe the behaviour of systems which operate on continuous functions of the line or the plane, and from this analysis obtain some useful facts about convolution.

8.2.1 Discrete Convolution

In the 1D case, we have a shift invariant linear system that 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, we assume that the input and output are infinite dimensional. This allows us to ignore some minor issues that arise at the boundaries of the input — we'll deal with these later (section **??**).

Discrete Convolution in One Dimension

We have an input vector $\boldsymbol{f}$. For convenience, we will assume that the vector is infinite, and its elements are indexed by the integers (i.e. there is a -1'th element, etc.). The i'th component of this vector is f_i. Now $\boldsymbol{f}$ is a weighted sum of basis elements. A convenient basis is a set of elements that have a one in one component, and zeros elsewhere. We write

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

(this is a data vector that has a 1 in the zero'th place, and zeros elsewhere). Define a shift operation, where $\texttt{Shift}(\boldsymbol{f}, i)$ is a vector whose j'th component is the $j-i$'th component of $\boldsymbol{f}$. For example, $\texttt{Shift}(\boldsymbol{e}_0, 1)$ has a zero in the first component. Now we can write

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

We write the response of our system to a vector $\boldsymbol{f}$ as

$$R(\boldsymbol{f})$$

Now because the system is shift invariant, we have that

$$R(\texttt{Shift}(\boldsymbol{f}, k)) = \texttt{Shift}(R(\boldsymbol{f}), k)$$

Furthermore, because it is linear, we have that

$$R(k\boldsymbol{f}) = kR(\boldsymbol{f})$$

This means that

$$\begin{aligned} R(\boldsymbol{f}) &= R(\sum_i f_i \texttt{Shift}(\boldsymbol{e}_0, i)) \\ &= \sum_i R(f_i \texttt{Shift}(\boldsymbol{e}_0, i)) \\ &= \sum_i f_i R(\texttt{Shift}(\boldsymbol{e}_0, i)) \\ &= \sum_i f_i \texttt{Shift}(R(\boldsymbol{e}_0), i)) \end{aligned}$$

This means that to obtain the system's response to any data vector, we need to know only its response to $\boldsymbol{e}_0$. This is usually called the system's **impulse response**. Assume that the impulse response can be written as $\boldsymbol{g}$. We have

$$R(\boldsymbol{f}) = \sum_i h_i \texttt{Shift}(\boldsymbol{g}, j) = \boldsymbol{g} * \boldsymbol{h}$$

This defines an operation — the 1D, discrete version of convolution — which we write with a $*$.

This is all very well, but does not give us a particularly easy expression for the output. If we consider the i'th element of $R(\boldsymbol{f})$, which we write as R_i, we must have:

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

which conforms to (and explains the origin of) the form we used in section 8.1.1.

Discrete Convolution in Two Dimensions

We now use an array of values, and write the i, j'th element of the array $\mathcal{D}$ is D_{ij}. The appropriate analogy to an impulse response is the response to a stimulus that looks like:

$$\mathcal{E}_{00} = \begin{array}{ccccc} \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & 0 & 0 & 0 & \ldots \\ \ldots & 0 & 1 & 0 & \ldots \\ \ldots & 0 & 0 & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{array}$$

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:

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

which we write as

$$\mathcal{R} = \mathcal{G} * * \mathcal{H}$$

8.2.2 Continuous Convolution

There are shift invariant linear systems that produce a continuous response to a continuous input; for example, a camera lens takes a set of radiances and produces another set, and many lenses are approximately shift invariant. A brief study of these systems will allow us to study the information that is lost by approximating a continuous function — the incoming radiance values across an image plane — by a discrete function — the value at each pixel.

The natural description is in terms of the system's response to a rather unnatural function — the δ-function, which is not a function in formal terms. We will do the derivation first in one dimension, to make the notation easier.

Convolution in One Dimension

We will obtain an expression for the response of a continuous shift invariant linear system from our expression for a discrete system. We can take a discrete input, and replace each value with a box, straddling the value — this gives a continuous input function. We will then make the boxes narrower, and consider what happens in the limit.

Our system takes a function of one dimension and returns a function of one dimension. Again, we write the response of the system to some input $f(x)$ as $R(f)$; when we need to emphasize that f is a function, we write $R(f(x))$. The response is also a *function*; occasionally, when we need to emphasize this fact, we will write $R(f)(u)$. We can express the linearity property in this notation by writing

$$R(kf) = kR(f)$$

(for k some constant) and the shift invariance property by introducing a `Shift` operator, which takes functions to functions:

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

With this `Shift` operator, we can write the shift invariance property as:

$$R(\texttt{Shift}(f, c)) = \texttt{Shift}(R(f), c)$$

We define the *box* function as:

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

The value of $box_\epsilon(\epsilon)$ does not matter for our purposes. The input function is $f(x)$. We construct an even grid of points x_i, where $x_{i+1} - x_i = \epsilon$. We now construct a vector $\boldsymbol{f}$, whose i'th component (written f_i) is $f(x_i)$.

This means that we can approximate f by $\sum_i f_i \texttt{Shift}(box_\epsilon, x_i)$ We apply this input to a shift-invariant linear system; the response is a weighted sum of shifted responses to box functions as in figure **??**. This means that

$$\begin{aligned} R(\sum_i f_i \texttt{Shift}(box_\epsilon, x_i)) &= \sum_i R(f_i \texttt{Shift}(box_\epsilon, x_i)) \\ &= \sum_i f_i \texttt{Shift}(R(\frac{box_\epsilon}{\epsilon}\epsilon), x_i) \\ &= \sum_i f_i \texttt{Shift}(R(\frac{box_\epsilon}{\epsilon}), x_i)\epsilon \end{aligned}$$

(so far, everything has followed our derivation for discrete functions). We now have something that looks like an approximate integral, if $\epsilon \to 0$.

We introduce a new device, called a δ-function, to deal with the term box_ϵ/ϵ. Define

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

The δ-function is:

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

We don't attempt to evaluate this limit, so we need not discuss the value of $\delta(0)$. 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. If we make the light smaller and brighter, while ensuring the total energy is constant, we expect to see a small but finite spot due to the defocus of the lens. The δ-function is the natural analogue for $\boldsymbol{e}_0$ in the continuous case.
This means that the expression for the response of the system,

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

will turn into an integral as ϵ limits to zero. We obtain

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

where we have written $R(\delta)$ — which is usually called the **impulse response** of the system — as g and have omitted the limits of the integral. These 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 we write the expression above as

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

Convolution is **symmetric**, meaning

$$(g * h)(x) = (h * g)(x)$$

Convolution is *associative*, meaning that

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

This latter property means that we can find a single shift-invariant linear system that behaves like the composition of two different systems. This will come in useful when we discuss sampling.

Convolution in Two Dimensions

The derivation of convolution in two dimensions requires more notation — a box function is now given by $box_\epsilon(x, y) = box_\epsilon(x)box_\epsilon(y)$; we now have

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

The δ-function is the limit of $d_\epsilon(x, y)$ function as $\epsilon \to 0$; and there are more terms in the sum. All this activity will result in the expression:

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

Where we have used two $*$'s to indicate a two dimensional convolution. Convolution *in 2D* is **symmetric** — that is $(g * *h) = (h * *g)$ — and associative.

A natural model for the impulse response of two dimensional system is to think of the pattern seen in a camera viewing a very small, distant light source (which subtends a very small viewing angle). In practical lenses, this view results in some form of fuzzy blob, justifying the name **point spread function** which is often used for the impulse response of a 2D system. The point spread function of a linear system is often known as its **kernel**.

8.2.3 Edge Effects in Discrete Convolutions

In practical systems we cannot have infinite arrays of data. This means that, when we compute the convolution, we need to contend with the edges of the image; at the edges, there are pixel locations where computing the value of the convolved image requires image values that don't exist. There are a variety of strategies we can adopt:

- **Ignore these locations** — this means that we report only values for which every required image location exists. This has the advantage of probity, but the disadvantage that the output is smaller than the input — repeated convolutions can cause the image to shrink quite drastically.
- **Pad the image with constant values** — this means that, as we look at output values closer to the edge of the image, the extent to which the output of the convolution depends on the image goes down. This is a convenient trick, because we can ensure that the image doesn't shrink, but it has the disadvantage that it can create the appearance of substantial gradients near the boundary.
- **Pad the image in some other way** — for example, we might think of the image as a doubly periodic function, so that, if we have an $n \times m$ image, then column $m + 1$ — required for the purposes of convolution — would be the same as column $m - 1$. This can also create the appearance of substantial gradients near the boundary.

8.3 Spatial Frequency and Fourier Transforms

We have used the trick of thinking of a signal $g(x, y)$ an weighted sum of a very large (or infinite) number of very 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 several problems so far left open:

- we need an explanation for the tendency of differentiation to emphasize noise and the tendency of smoothing to deemphasize it;
- while 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 k'th pixel — this could turn a checkerboard image all white or all black — and we should 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 will 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 will be obvious, because they will correspond to large amounts of high frequency sinusoids in the new basis.

8.3.1 Fourier Transforms

We will concentrate on Fourier transforms in the continuous domain, because we use Fourier transforms mainly as a conceptual device.

Fourier Transforms in the Continuous Domain

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).

Figure 8.4. 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 $[-1,1]x[-1,1]$, with the origin at the center of the image. On the left, $(u,v) = (1,2)$; in the center, $(u,v) = (10,-5)$ and on the right $(u,v) = (32,-32)$. These are sinusoids of various frequencies and orientations, described in the text.

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

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

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 will be 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 8.4.

The integral should be seen as a dot product. 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 can be seen as measuring the "amount" of the sinusoid with given frequency and orientation in the signal. The Fourier transform is linear:

$$\begin{aligned} \mathcal{F}(g(x,y)+h(x,y)) &= \mathcal{F}(g(x,y)) + \mathcal{F}(h(x,y)) \\ &\text{and} \\ \mathcal{F}(kg(x,y)) &= k\mathcal{F}(g(x,y)) \end{aligned}$$

Function	Fourier transform
$g(x,y)$	$\int\int_{-\infty}^{\infty} g(x,y)e^{-i2\pi(ux+vy)}dxdy$
$\delta(x,y)$	1
$\frac{\partial f}{\partial x}(x,y)$	$u\mathcal{F}(f)(u,v)$
$0.5\delta(x+a,y)+0.5\delta(x-a,y)$	$\cos 2\pi au$
$e^{-\pi(x^2+y^2)}$	$e^{-\pi(u^2+v^2)}$
$bar_1(x,y)$	$\frac{\sin u}{u}\frac{\sin v}{v}$
$f(ax,by)$	$\frac{\mathcal{F}(f)(u/a,v/b)}{ab}$
$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)$	$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(u-i,v-j)$
$(f**g)(x,y)$	$\mathcal{F}(f)\mathcal{F}(g)(u,v)$
$f(x-a,y-b)$	$e^{-i2\pi(au+bv)}\mathcal{F}(f)$
$f(x\cos\theta-y\sin\theta, x\sin\theta+y\cos\theta)$	$\mathcal{F}(f)(u\cos\theta-v\sin\theta, u\sin\theta+v\cos\theta)$

Table 8.1. 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 may 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 []).

The Inverse Fourier Transform

It is useful to be able to recover a signal from its Fourier transform. This is another change of basis, with the form

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

Fourier Transform Pairs

Fourier transforms are known in closed form for a variety of useful cases; a large set of examples appears in [?]. We list a few in table 8.1 for reference, with a drawing in

figure **??**. The last line of table 8.1 contains the **convolution theorem**; convolution in the signal domain is the same as multiplication in the Fourier domain. We will use this important fact several times in what follows.

Phase and Magnitude

The Fourier transform consists of a real and a complex component.

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

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. Firstly, a local change in the function — for example, 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. Secondly, the magnitude spectra of images tends to be similar, and magnitude is surprisingly uninformative (see figure 8.5 for an example).

8.3.2 Differentiation and Noise

From table **??**, differentiating a function is the same as multiplying its Fourier transform by a frequency variable; this means that the high spatial frequency components are heavily emphasized at the expense of the low frequency components. This is intuitively plausible — differentiating a function must set the constant component to zero, and the amplitude of the derivative of a sinusoid goes up with its frequency. Furthermore, this property is the reason we are interested in derivatives; we are discussing the derivative precisely because fast changes (which generate high spatial frequencies) have large derivatives.

It is possible to show that stationary additive Gaussian noise has uniform energy at each frequency; but if we differentiate the noise, we will emphasize the high frequencies. If we do not attempt to ameliorate this situation, the gradient magnitude map is likely to have occasional large values due to noise. Filtering with a Gaussian filter suppresses these high spatial frequencies.

The discussion of aliasing gives us some insight into available smoothing parameters as well. Any Gaussian kernel that we use will be a sampled approximation to a Gaussian, sampled on a single pixel grid. This means that, for the original kernel to be reconstructed from the sampled approximation, it should contain no components

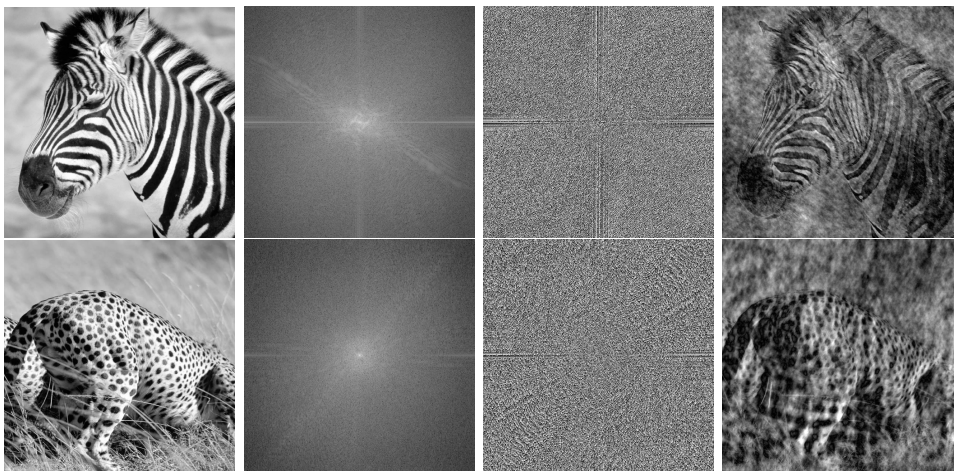

Figure 8.5. The second image in each row shows the log of the magnitude spectrum for the first image in the row; the third image shows the phase spectrum, scaled so that $-\pi$ is dark and π is light. The final images are obtained by swapping the magnitude spectra. While this swap leads to substantial 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.

of spatial frequency greater than 0.5pixel^{-1}. This isn't possible with a Gaussian, because its Fourier transform is also Gaussian, and hence isn't bandlimited. The best we can do is insist that the quantity of energy in the signal that is aliased is below some threshold — in turn, this implies a minimum value of σ that is available for a smoothing filter on a discrete grid (for values lower than this minimum, the smoothing filter itself is badly aliased — see exercise **??**).

8.4 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?

8.4.1 Sampling

Passing from a continuous function to a collection of values on a discrete grid is referred to as **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 shall

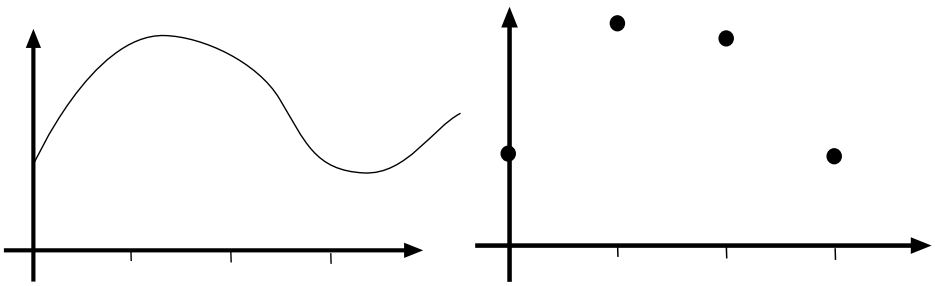

Figure 8.6. Sampling in 1D takes a function, and returns a vector whose elements are values of that function at the sample points, as the top figures show. For our purposes, it is enough that the sample points be integer values of the argument. We allow the vector to be infinite dimensional, and have negative as well as positive indices.

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:

$$\text{sample}_{1D}(f(x)) = \boldsymbol{f}$$

We will 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. This means that the i'th component of $\boldsymbol{f}$ is $f(x_i)$.

Sampling in Two Dimensions

Sampling in 2D is very similar to sampling in 1D. Although sampling can occur on non-regular grids (the best example being the human retina), we will 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. This array we allow to have negative indices in both dimensions, and can then write

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

where the i, j'th 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 accomodate this effect by spacing sample points slightly further apart horizontally than vertically (in jargon, they have **non-square pixels**).

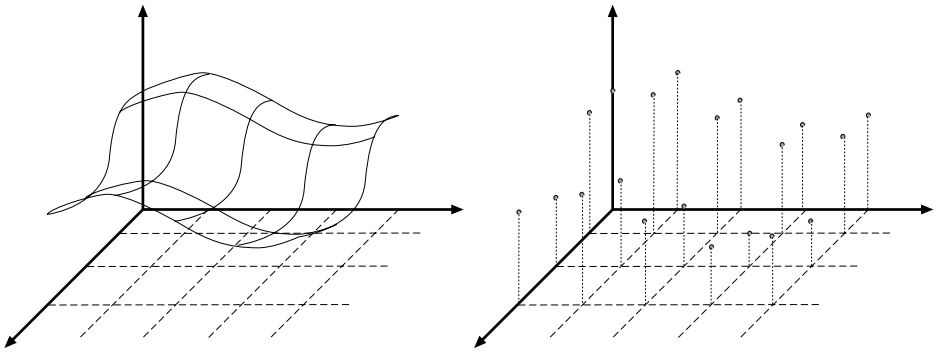

Figure 8.7. Sampling in 2D takes a function and returns an array; again, we allow the array to be infinite dimensional and to have negative as well as positive indices.

A Continuous Model of a Sampled Signal

We need a continuous model of a sampled signal. Generally, this model will be 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:

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

(where 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 proce-

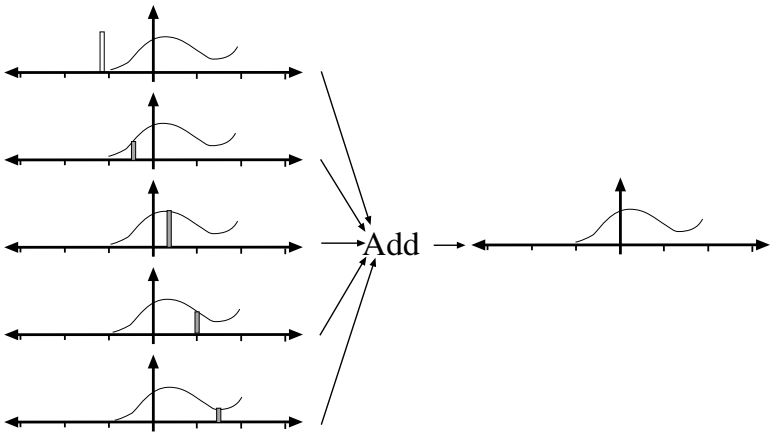

Figure 8.8. Convolving a δ-function with an arbitrary function can be thought of in terms of convolving a d_ϵ function, and taking the limit. The shaded region contributes to the integral, and as ϵ gets smaller — and so the region averaged gets narrower — the result limits to the original function.

dure gets

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

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.

8.4.2 Aliasing

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

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 (figure 8.8) (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.

$$\mathcal{F}(\text{sample2d}(g(x,y))) = \mathcal{F}\left(g(x,y)\left\{\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)\right\}\right) \quad (8.4.1)$$

$$= \mathcal{F}(g(x,y)) * *\mathcal{F}\left(\left\{\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)\right\}\right) \quad (8.4.2)$$

$$= \sum_{i=-\infty}^{\infty} G(u-i,v-j) \quad (8.4.3)$$

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

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 8.9).

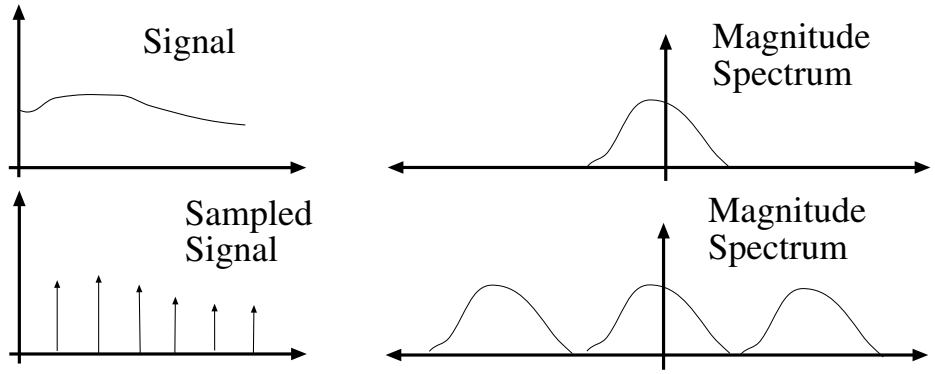

Figure 8.9. The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. Two possibilities occur. If the shifted copies do not intersect with each other (as in this case), the original 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 figure 8.10) the intersection region is added, and so we cannot obtain a separate copy of the Fourier transform, and the signal has aliased.

However, if the support regions *do* overlap, we will not be able to reconstruct the signal because we will not be able to 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 8.11 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.

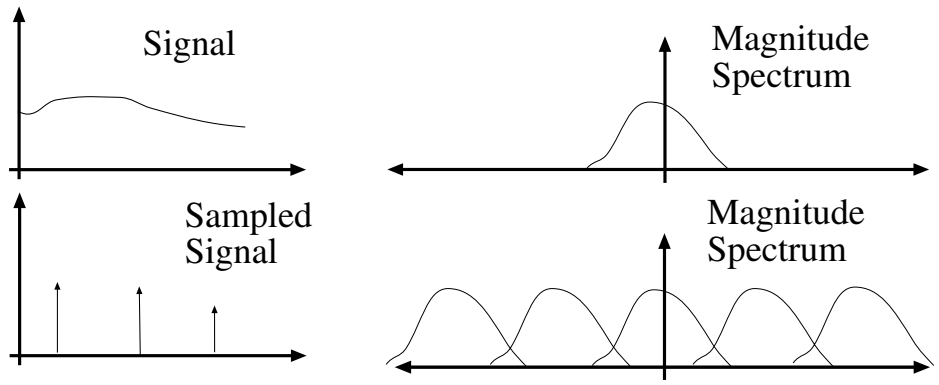

Figure 8.10. The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. Two possibilities occur. If the shifted copies do not intersect with each other (as in figure 8.9), the original 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.

8.4.3 Smoothing and Resampling

Nyquist's theorem means it is dangerous to shrink an image by simply taking every k'th pixel (as figure **??** 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 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 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.

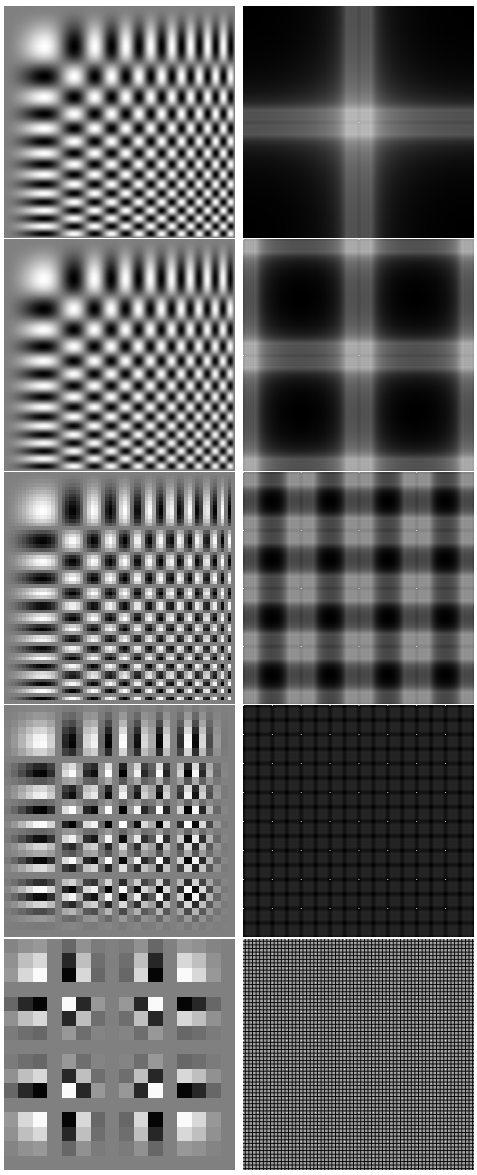

Figure 8.11. Left: At the top is a 256x256 pixel image showing a grid obtained by multiplying two sinusoids with linearly increasing frequency — one in x and one in y. The other images in the series are obtained by resampling by factors of two, without smoothing (i.e. the next is a 128x128, then a 64x64, 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. **Right:** 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 of aliasing.

If we resample this signal, the copies will now have centers on the half-integer points in u, v space. This means that, to avoid aliasing, we need to apply a low pass 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 may lose information, too.

Gaussians die away fairly quickly and so can be used as low pass filters. The choice of Gaussian depends on the application; if σ is large, the kernel must be large and, while there is less aliasing (because the value of the kernel outside our range is very small), 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 **??** and **??** illustrate the effects of different choices of σ.

8.5 Human: Filters and Primate Early Vision

There is quite a lot of information about the early stages of the primate visual system. The "wiring" can be studied using stains that carry from cell to cell; the response of individual cells can be studied by displaying patterns and recording the electrical behaviour of the cell; and some structural information can be elicited using psychophysical experiments. All this evidence suggests that spatiotemporal filters yield a rather good model of early visual processing.

8.5.1 The Visual Pathway

The anatomical details of how visual information is passed into the brain, and what happens in the early stages, are quite well understood. Information about the connections along this pathway can be obtained by staining methods; typically, a cell is stained with a substance that moves in known ways (along the body of the cell; across connections; etc.) and one looks to see where the stain ends up.

The stages in the **visual pathway** are:

- The **retina**, where photoreceptive cells transduce irradiance to electrical spikes. These signals are processed by a variety of layers of cells. **Retinal ganglion cells** connect to the final layer.
- The **optic nerve** consists of the fibers of the retinal ganglion cells, and connects the retina to the brain through the the **optic chiasma**. This is a crossing point; the left-hand side of *each* retina is connected to the left half of the brain, and the right-hand side to the right half.
- There are now two pathways; some information is fed to the **superior colliculus** (which we shall ignore), but most goes to the **lateral geniculate nucleus**.
- The lateral geniculate nucleus is connected to the **visual cortex**, one of the best studied regions in the primate brain. The visual cortex consists of a series of quite well defined layers. Much of early vision occurs in this structure, which

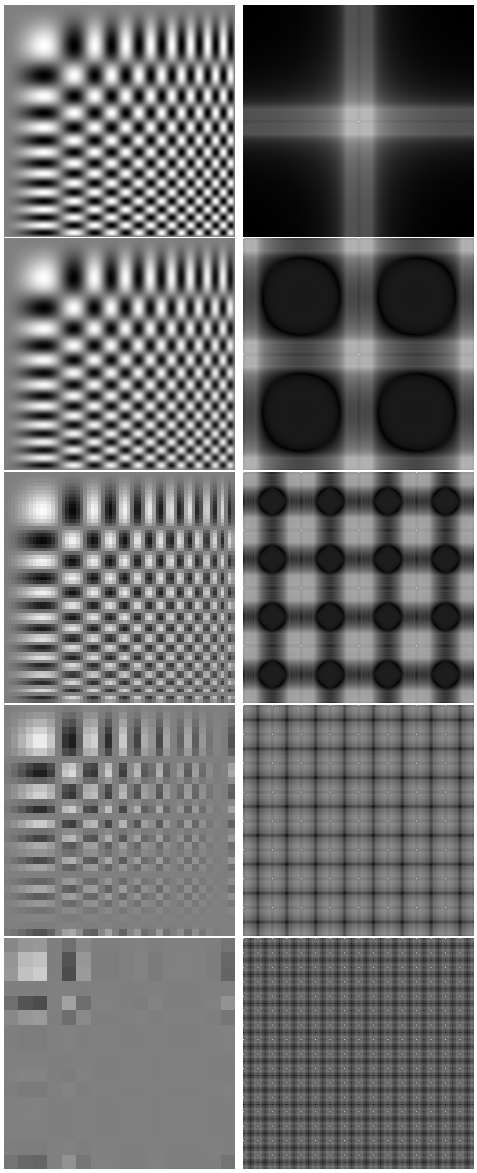

Figure 8.12. Left: Resampled versions of the image of figure 8.11, again by factors of two, but this time each image is smoothed with a Gaussian of σ one pixel before resampling. This filter is a low-pass filter, and so suppresses high spatial frequency components, reducing aliasing. **Right:** The effect 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.

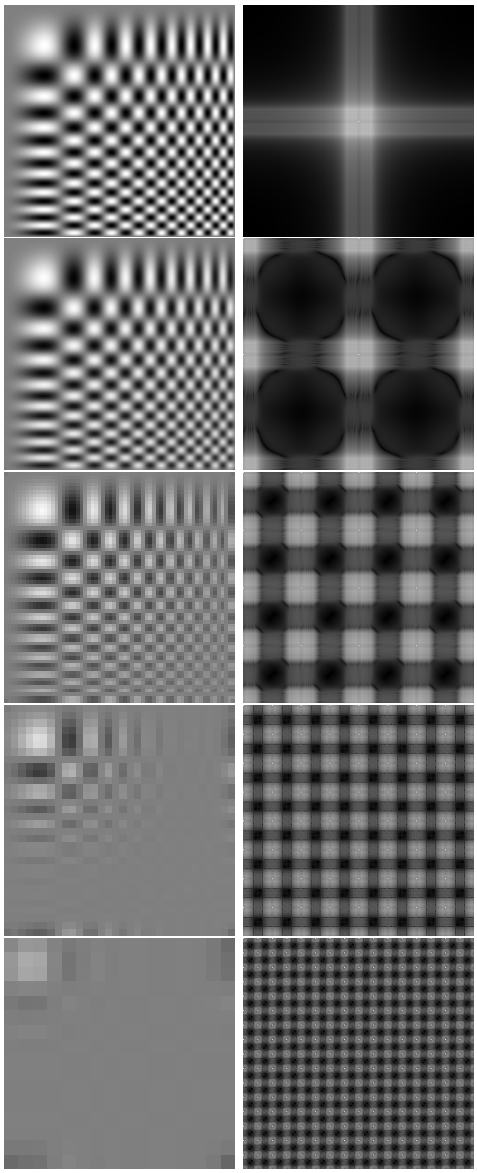

Figure 8.13. **Left:** Resampled versions of the image of figure 8.11, again by factors of two, but this time each image is smoothed with a Gaussian of σ two pixels before resampling. This filter suppresses high spatial frequency components more aggressively than that of figure 8.12. **Right:** The effect 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.

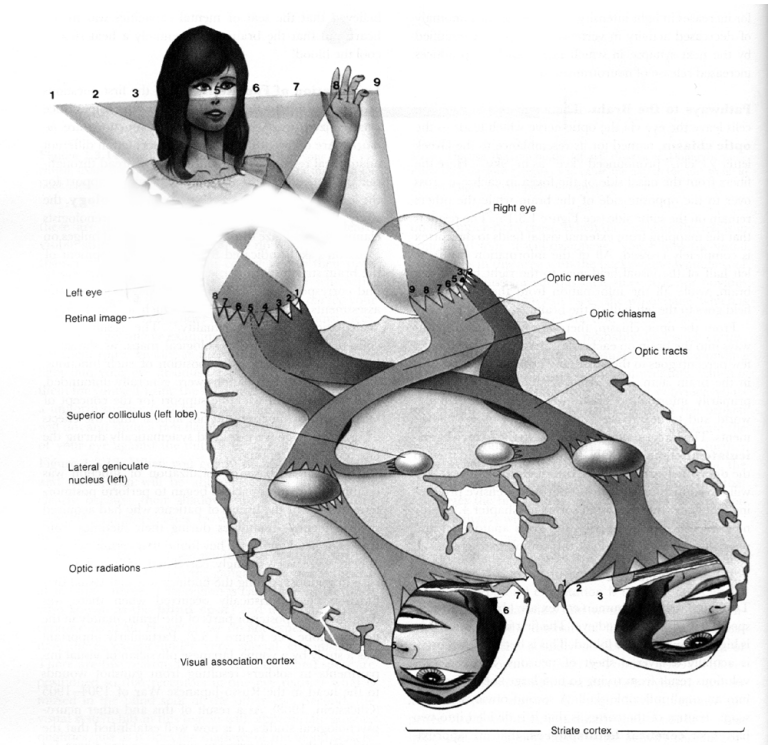

Figure 8.14. A diagram of the visual pathway, viewed from the underside of the brain. Elements of the right hand visual field are imaged on both left and right retinas; through the optic chiasm, the left hand visual field is projected to the right lateral geniculate nucleus (and vice versa — the dashed lines indicate the pathway followed by the right visual field). Signals then leave the LGN for the visual cortex, at the back of the brain. The behaviour of cells up to and including the cortex is quite well understood, and is usually modelled as a system of filters. Notice that the outputs of retinal cells map to the cortex in a spatially organised fashion, but that the central portion of the visual field maps to a larger area of cortex than the peripheral portions; this reflects the fact that the eye has higher spatial resolution in the central portion of the visual field. *figure from Palmer, Vision Science, p 36, in the fervent hope that permission will be granted*

contains a large selection of different representations of an image, organised in fairly well understood structures.

- Visual information leaves the visual cortex for the **parietal cortex** and the **temporal cortex**. The temporal cortex appears to be involved in determining *what* an object is, and the parietal cortex in determining *where* it is [] and possibly *how* it can be grasped []. This information must be reunited somewhere, but it isn't currently known where.

8.5.2 Experimental Methods

There are quite detailed models of how cells respond along this pathway, up to the point where signals leave the visual cortex (there is rather little information after this point). These models are supported by large quantities of evidence on the responses elicited from cells in the brain by particular patterns of light and motion. Both the visual cortex and the lateral geniculate nucleus are quite conveniently sited on a primate brain, so that it is possible to insert electrodes into these structures without traumatising the brain so badly that the results are meaningless.

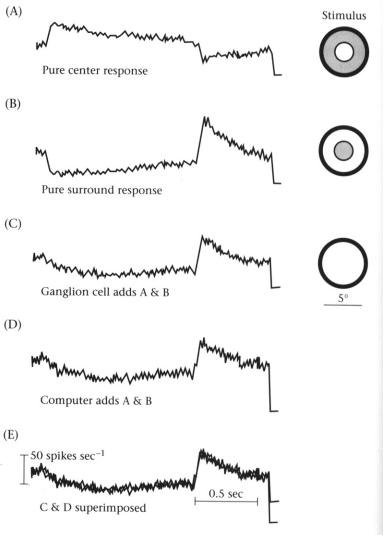

Figure 8.15. The linearity of the response of a ganglion cell can be tested by showing different spatial distributions of brightness to that cell, and then checking that the sum of responses is the same as the response to the sum of stimuli. In the experiment sketched here, the cell is shown a stimulus at the center, and then one at the surrounding annulus; the responses are added and compared to the response to a disk. The strength of response of a cell is indicated by the rate at which it produces electrical spikes — there is typically a resting rate; higher rates than the rest rate indicate excitation and lower rates indicate inhibition. *Figure from Brian Wandell's book, "Foundations of Vision", page132, in the fervent hope that permission will be granted*, after Enroth-Cugell and Pinto, 1970

In a typical experiment, an electrode is inserted along a path through some structure to record electrical signals from cells along that path. Patterns of light, colour and motion are shown to an experimental animal, and the response from the electrode — which is hopefully a response from a nearby cell — is recorded along with the depth to which the electrode has penetrated. When sufficient information has been obtained, the electrode is moved deeper along its path and the process continues. Recordings may be taken from several different paths. Eventually, the

experimental animal is sacrificed and its brain cut in sections to determine from what precise regions the electrode recorded information.

Typically, neurons in the visual pathway are discussed in terms of their **receptive field** — this is a record of the spatial distribution of the effect of illumination on the neuron's output. The response of cells is often determined by their response to a grating. A **spatial grating** is a pattern of the form $m(1+\cos 2\pi fx)$, where x is a convenient spatial coordinate across the visual field. These gratings can be used to investigate the spatial response of a cell; the spatio-temporal response is studied using a **spatio-temporal grating** is a pattern of the form $m(1+a\cos 2\pi fx\cos 2\pi gt))$ — this is a moving sinusoid. Many cells temporal components to their response as well, and such cells can be described in terms of their contrast sensitivity to a spatio-temporal grating (figure 8.16).

8.5.3 The Response of Retinal Cells

The earliest cells in the visual pathway are **retinal ganglion cells**, which collect outputs from the layers of retinal cells receptors. Typically, light falling in the center of a ganglion cell's receptive field increases its firing rate — often called its response — and light falling in the outside decreases the firing rate. A cell that responds like this is referred to as an **on-center, off-surround** cell; there are also **off-center, on-surround** cells.

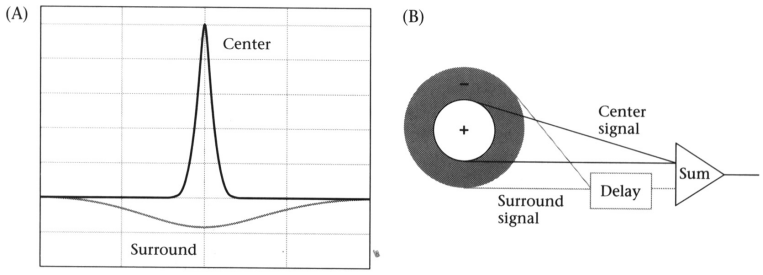

Figure 8.16. The basic model for a linear retinal ganglion cell is a difference of Gaussian model — there is a center field that has a spatial sensitivity of the form of a narrow Gaussian, and a surround field that has the form of a broad Gaussian. One field excites, the other inhibits (as indicated by the positive/negative signs in the figure). The response of the cell can be predicted by adding the temporal response of the center to the temporal response of the surround (which is slower than that of the center). *Figure from Brian Wandell's book, "Foundations of Vision", page145, in the fervent hope that permission will be granted* after Enroth-Cugell et al.

Ganglion cells are typically investigated using gratings. *For a fixed mean value* m, ganglion cells appear to sum their response over the receptive field. This can be tested by comparing the sum of the responses to individual stimuli and the response

to a sum of these stimuli, as in figure 8.15.

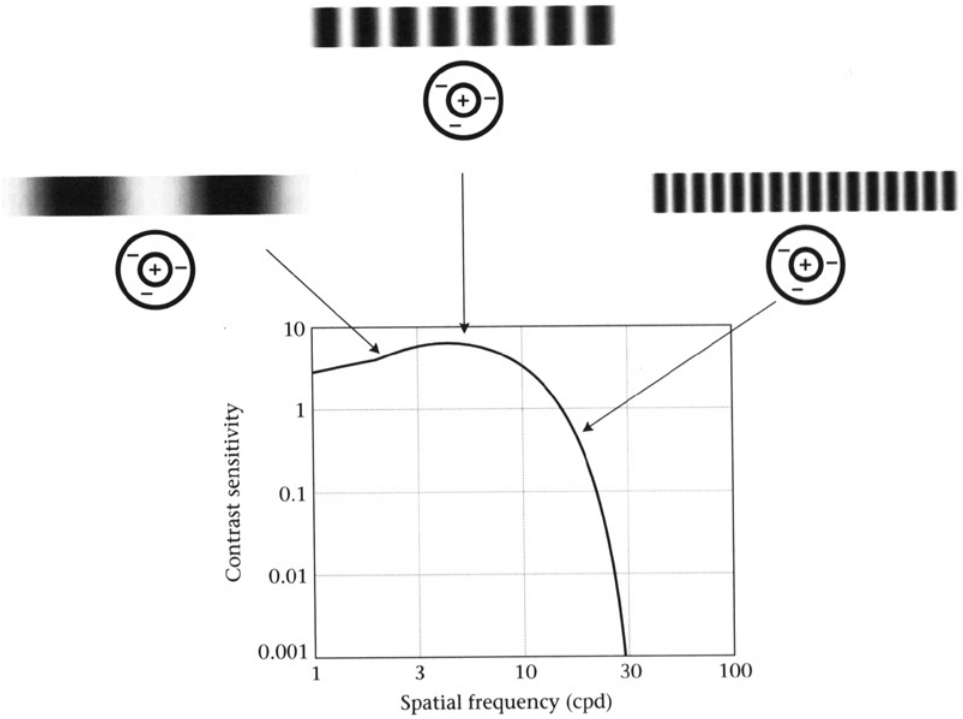

Figure 8.17. The contrast sensitivity function can be used to describe the behaviour of linear cells. In a center-surround cell, described in the figure, the response to a constant stimulus is zero. The receptive field is symmetric, meaning that its behaviour can be described in terms of its response to a signal of the form $m(1 + a\cos(2\pi f x))$, where f is the spatial frequency and x is spatial position in a convenient system of units; a is referred to the contrast of the signal. The contrast sensitivity is obtained by fixing some level of response, and taking the inverse of the contrast required to reach that level of response. Center-surround cells are tuned in spatial frequency; if the spatial frequency of the signal is low, then the signal is nearly constant over the receptive field of the cell (**left**), and the contrast sensitivity is lowered. If the frequency of the signal is high, then the excitatory and inhibitory responses tend to cancel (**right**).

This linearity can be exploited to measure response using periodic functions, which are usually spatial sinusoids. It is usual to study neurons by fixing some level of response, and then determining the **contrast** (amplitude of the sinusoid) required to elicit that level of response — the **contrast threshold**. Typically, one plots the **contrast sensitivity** (the inverse of the contrast threshold) against some interesting variable. Figure 8.17 plots the contrast sensitivity of a center-surround neuron against a measure of spatial frequency. In this case, the stimulus shown was

fixed, and the cell's response measured after some long time, giving an assymptotic response. The contrast sensitivity function is different for different mean values, an effect known as **adaptation** (figure 8.18).

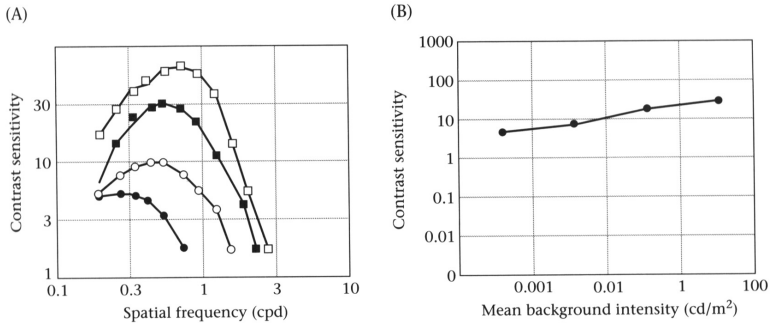

Figure 8.18. These figures illustrate adaptation in the cat. On the left, the curves show contrast sensitivity functions obtained at different mean brightnesses (m in our expressions). There are two orders of magnitude in brightness between each pair of stimuli; the brightest stimulus (top) is six orders of magnitude more intense than the darkest (bottom). The contrast sensitivity functions change, but not much compared with the change in mean brightness. On the right, the contrast sensitivity of a particular target measured over a range of mean intensities; as the mean intensity moves through six orders of magnitude, the contrast sensitivity moves through only one order of magnitude. *Figure from Brian Wandell's book, "Foundations of Vision", page149, in the fervent hope that permission will be granted*, after Pasternak and Merigan.

8.6 Laplacians and edges

One diagnostic for a large gradient magnitude is a zero of a "second derivative" at a point where the gradient is large. A sensible 2D analogue to the 1D second derivative must be rotationally invariant; it is not hard to show that the **Laplacian** has this property. The Laplacian of a function in 2D is defined as:

$$(\nabla^2 f)(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

It turns out that smoothing an image and then applying the Laplacian is the same as convolving the image with the Laplacian of a Gaussian (section 9.3.4 explains this fact for first derivatives, but it works for any linear differential operator).

This leads to a simple and historically important edge detection strategy, illustrated in figure 8.19. We convolve an image with a **Laplacian of Gaussian** at some scale, and mark the points where the result has value zero — the **zero crossings**. These points should be checked to ensure that the gradient magnitude is large.

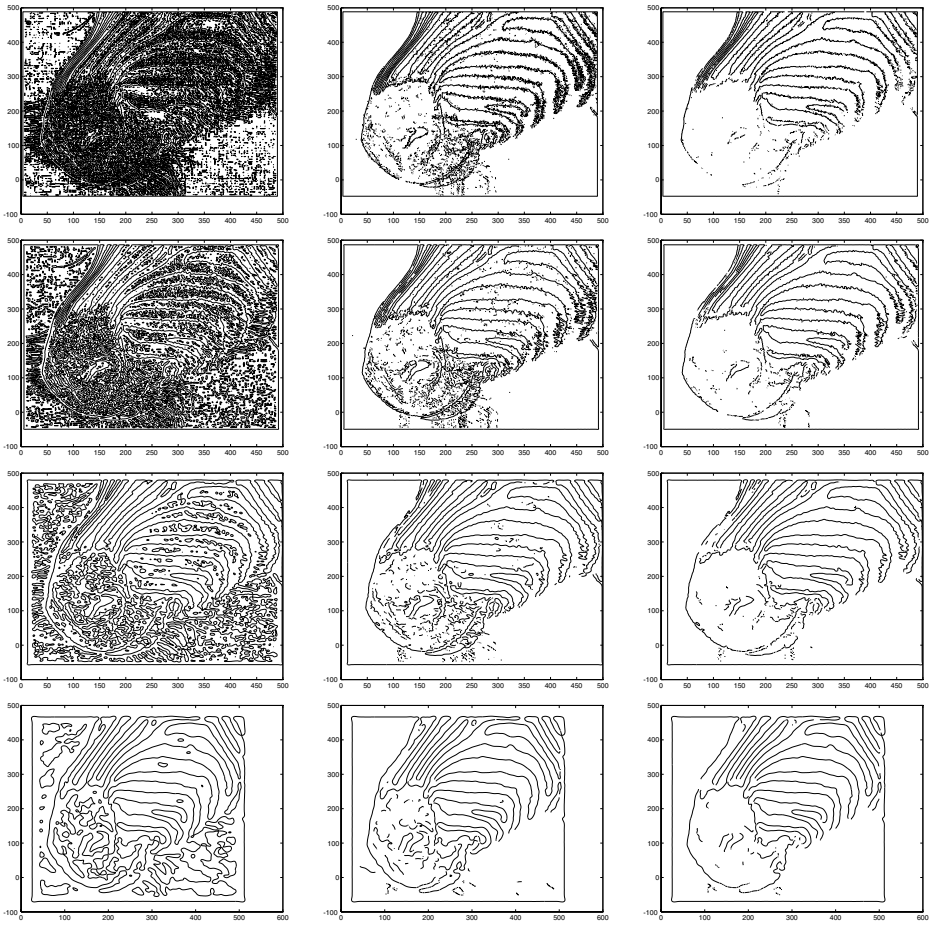

Figure 8.19. Zero crossings of the Laplacian of Gaussian for various scales and at various gradient magnitude thresholds. Each row shows a fixed scale, with the threshold on gradient magnitude increasing as one moves to the right (by a factor of two from image to image). Each column shows a fixed threshold, with scale increasing from σ one pixel to σ eight pixels, by factors of two. Notice that the fine scale, low threshold edges contain a quantity of detailed information that may or may not be useful (depending on one's interest in the hairs on the zebra's nose). As the scale increases, the detail is suppressed; as the threshold increases, small regions of edge drop out. No scale or threshold gives the outline of the zebra's head; all respond to its stripes, though as the scale increases, the narrow stripes on the top of the muzzle are no longer resolved.

The response of a Laplacian of Gaussian filter is positive on one side of an edge and negative on another. This means that adding some percentage of this response

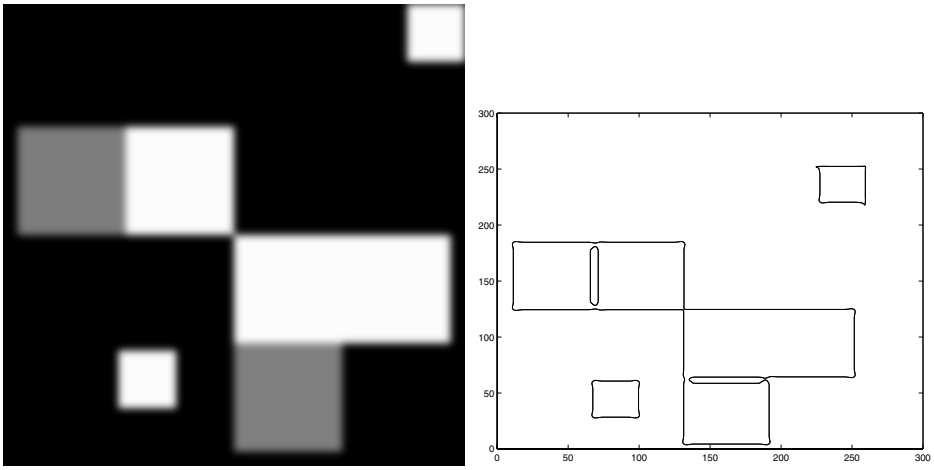

Figure 8.20. Zero crossings of Laplacian of Gaussian output can behave strangely at corners. Firstly, at a right angled corner, the zero crossing bulges out at the corner (but passes through the vertex). This effect is not due to digitisation or to quantization, but can be shown to occur in the continuous case as well. At corners where three or more edges meet, contours behave strangely, with the details depending on the structure of the contour marking algorithm — this algorithm (the one shipped with Matlab) produces curious loops. This effect can be mitigated with careful design of the contour marking process, which needs to incorporate a fairly detailed vertex model.

back to the original image yields a picture in which edges have been sharpened, and detail is more easy to see. This observation dates back to a photographic developing technique called *unsharp masking*, where a blurred positive is used to increase visibility of detail in bright areas by "subtracting" a local average of the brightness in that area. Unsharp masking essentially applies a difference of Gaussian kernel to an image; as exercise **??** indicates, the difference between two Gaussian kernels looks very similar to a Laplacian of Gaussian kernel. It is quite common to replace one with the other.

Laplacian of Gaussian edge detectors have fallen into some disfavour. Because the Laplacian of Gaussian filter is not oriented, its response is composed of an average across an edge and one along the edge. This means that their behaviour at corners — where the direction along the edge changes — is poor. They mark the boundaries of sharp corners quite inaccurately; furthermore, at trihedral or greater vertices, they have difficulty recording the topology of the corner correctly, as figure 8.20 illustrates. Secondly, the components along the edge tend to contribute to the response of the filter to noise but not necessarily to an edge; this means that zero-crossings may not lie exactly on an edge.

8.7 Discussion

The book (hah! mere mathematics has a book - vision has a library) in which the secrets of vision are recorded must contain many volumes on the subject of filters; we have had to omit topics very aggressively to produce a survey of reasonable length. The topics whose omission will most disturb the informed are probably (i) the design of filters and (ii) techniques to obtain filtered representations efficiently. The first three sections of this chapter are essential if you want to study these topics; it wouldn't be possible to read the literature on filters in vision without these ideas.

8.7.1 Real Imaging Systems vs Shift-Invariant Linear Systems

Imaging systems are only approximately linear. Film is not linear — it does not respond to very weak stimuli, and it saturates for very bright stimuli — but one can usually get away with a linear model within a reasonable range. CCD cameras are linear within a working range. They give a very small, but non-zero response to a zero input, as a result of thermal noise (which is why astronomers cool their cameras) and they saturate for very bright stimuli. CCD cameras often contain electronics that transforms their output to make them behave more like film, because consumers are used to film. Shift invariance is approximate as well, because lenses tend to distort responses near the image boundary. Some lenses — fish-eye lenses are a good example — are not shift-invariant.

8.7.2 Scale

There is a very large body of work on scale space and scaled representations. We have given only the briefest picture here, because the analysis tends to be quite tricky. The usefulness of the techniques is currently hotly debated, too. The general idea is as follows: a representation of an object like a tree should probably occur at multiple scales — at a fine scale, we consider each twig and leaf, and at a coarse scale there are a couple of puffs of leaves at the top of a trunk. One technique that might It is something of a stretch from a representation of the behaviour of brightness maxima/minima to (say) a description of a tree. Generally, high maxima and deep minima will give blobs that last over a large range of scales, so that dark leaves on a bright sky may lead to a very large collection of small blobs which slowly merge over scales to end up with a single dark blob (the puff at the top of the tree). Much of the detailed gymnastics of the blobs as they merge is irrelevant — we really care only about the statistics of the blobs at the finest scale, and the size of the blob at the coarsest. There is little formalised knowledge about which bits of the representation carry cogent information and which do not.

Assignments

Exercises

1. Show that convolving a function with a shifted δ function shifts the function.

2. Aliasing takes high spatial frequencies to low spatial frequencies. Explain why following effects occur:

   - In old cowboy films that show wagons moving, the wheel often seems to be stationary or moving in the wrong direction (i.e. the wagon moves from left to right and the wheel seems to be turning anticlockwise).
   - White shirts with thin dark pinstripes often generate a shimmering array of colours on television.
   - In ray-traced pictures, soft shadows generated by area sources look blocky.

Programming Assignments

1. A sampled Gaussian kernel must alias, because the kernel contains components at arbitrarily high spatial frequencies. Assume that the kernel is sampled on an infinite grid. As the standard deviation gets smaller, the aliased energy must increase. Plot the energy that aliases against the standard deviation of the Gaussian kernel in pixels. Now assume that the Gaussian kernel is given on a 7x7 grid. If the aliased energy must be of the same order of magnitude as the error due to truncating the Gaussian, what is the smallest standard deviation that can be expressed on this grid?

2. Median filters can smooth corners unacceptably.

   - Demonstrate this effect by applying a median filter to a variety of images; what are the qualitative effects at corners and in textured regions?
   - Explain these effects.
   - Show that the multi-stage median filter results in less heavily smoothed corners in practice.
   - Explain this effect.
   - On occasion, it is attractive to suppress texture; median filters can do this rather well. Read [**?**].