Chapter 9

EDGE DETECTION

9.1 Estimating Derivatives with Finite Differences

Estimates of derivatives are generally important in vision, because changes in images are interesting. A sharp change in an image could be associated with the boundary of an object, or with markings on the object; and such changes are associated with large gradients.

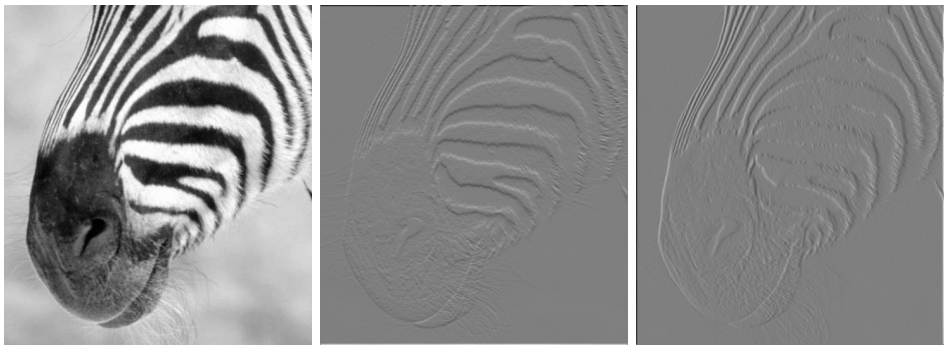

Figure 9.1. Finite differences are one way to obtain an estimate of a derivative. The image at the left shows a detail from a picture of a zebra. The center image shows the partial derivative in the y-direction — which responds strongly to horizontal stripes and weakly to vertical stripes — and the right image shows the partial derivative in the x-direction — which responds strongly to vertical stripes and weakly to horizontal stripes.

To estimate a derivative of an image represented by a discrete set of pixels, we need to resort to an approximation. Derivatives are rather naturally approximated by **finite differences**. Because

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x, y)}{\epsilon}$$

we might estimate a partial derivative as a symmetric difference:

$$\frac{\partial h}{\partial x} \approx h_{i+1,j} - h_{i-1,j}$$

This is the same as a convolution, where the convolution kernel is

$$\mathcal{G} = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{Bmatrix}$$

Notice that this kernel could be interpreted as a template — it will give a large positive response to an image configuration that is positive on one side and negative on the other, and a large negative response to the mirror image.

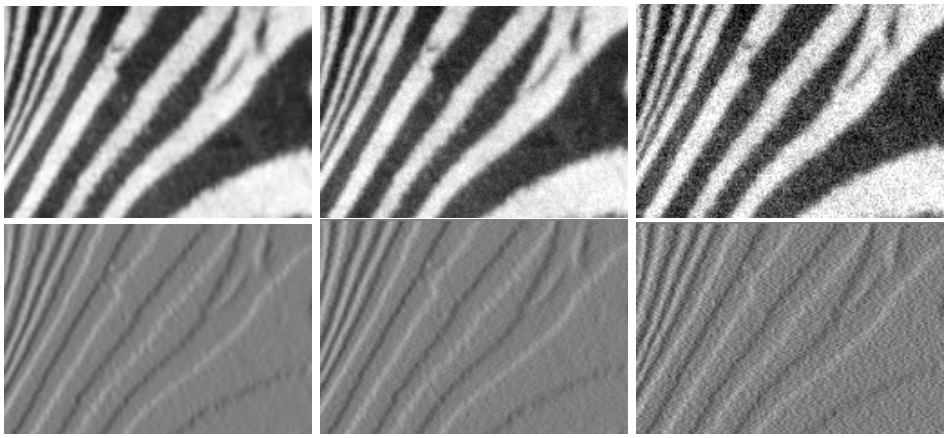

Figure 9.2. Finite differences respond strongly to noise. The image at top left shows a detail from a picture of a zebra; the next image in the row is obtained by adding a random number with zero mean and normal distribution ($\sigma = 0.03$) to each pixel; and the third image is obtained by adding a random number with zero mean and normal distribution ($\sigma = 0.09$) to each pixel. The second row shows the partial derivative in the x-direction of the image at the head of the row. Notice how strongly the differentiation process emphasizes image noise.

As figure 9.2 suggests, finite differences give a most unsatisfactory estimate of the derivative. This is because this filter has a strong response to fast changes due to noise, as well as those due to signal. For example, if we had bought a discount camera with some pixels that were stuck at either black or white, the filter would produce a strong response at those pixels because they will, in general, be substantially different from their neighbours. All this suggests that some form of smoothing is appropriate before differentiation; the details appear in section 9.3.1.

9.2 Noise

We have asserted that smoothing suppresses some kinds of noise. To be more precise we need a model of noise. Usually, by the term **noise**, we mean image measurements from which we do not know how to extract information, or from which we do not care to extract information; all the rest is **signal**. It is wrong to believe that noise does not contain information — for example, we should be able to extract some estimate of the camera temperature by taking pictures in a dark room with the lens-cap on. Furthermore, since we cannot say anything meaningful about noise without a noise model, it is wrong to say that noise is not modelled. Noise is everything we don't wish to use, and that's all there is to it.

9.2.1 Additive Stationary Gaussian Noise

In the **additive stationary Gaussian noise** model, each pixel has added to it a value chosen independently from the same Gaussian probability distribution. Almost always, the mean of this distribution is zero. The standard deviation is a parameter of the model. The model is intended to describe thermal noise in cameras.

Linear Filter Response to Additive Gaussian Noise

Assume we have a discrete linear filter whose kernel is $\mathcal{G}$, and we apply it to a noise image $\mathcal{N}$ consisting of stationary additive Gaussian noise with mean μ and standard deviation σ. The response of the filter at some point i, j will be:

$$R(\mathcal{N})_{i,j} = \sum_{u,v} G_{i-u,j-v} N_{u,v}$$

Because the noise is stationary, the expectations that we compute will not depend on the point, and we assume that i and j are zero, and dispense with the subscript. Assume the kernel has finite support, so that only some subset of the noise variables contributes to the expectation; write this subset as $n_{0,0}, \ldots, n_{r,s}$. The expected value of this response must be:

$$\begin{aligned} \mathtt{E}[R(\mathcal{N})] &= \int_{-\infty}^{\infty} \{R(\mathcal{N})\} p(N_{0,0}, \ldots, N_{r,s}) dN_{0,0} \ldots dN_{r,s} \\ &= \sum_{u,v} G_{-u,-v} \{ \int_{-\infty}^{\infty} N_{u,v} p(N_{u,v}) dN_{u,v} \} \end{aligned}$$

where we have done some aggressive moving around of variables, and integrated out all the variables that do not appear in each expression in the sum. Since all the $N_{u,v}$ are independent identically distributed Gaussian random variables with mean μ, we have that:

$$\mathtt{E}[R(\mathcal{N})] = \mu \sum_{u,v} G_{i-u,j-v}$$

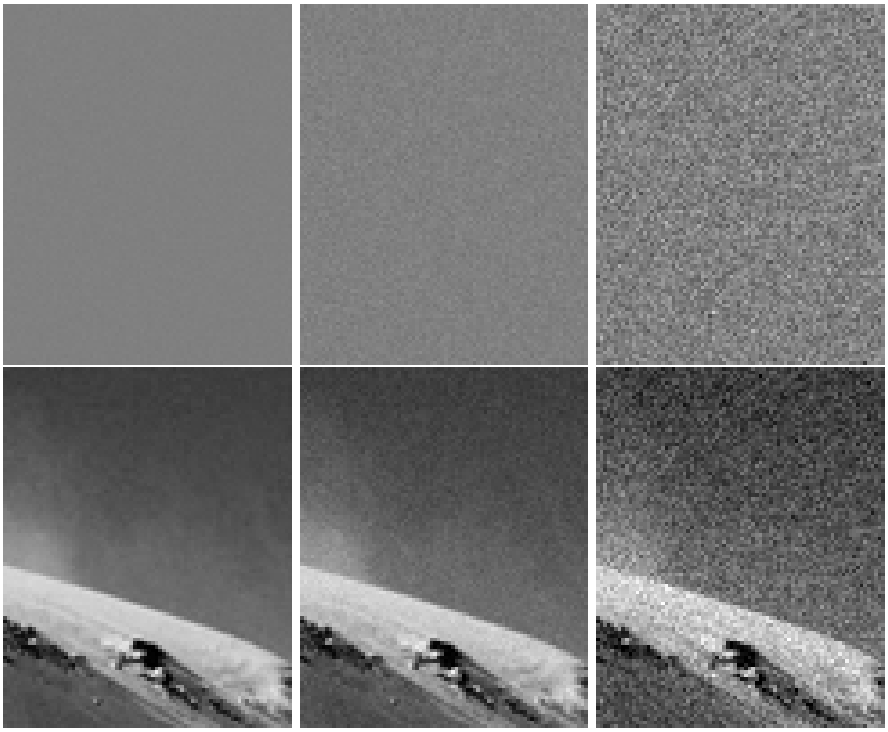

Figure 9.3. The top row shows three realisations of a stationary additive Gaussian noise process. We have added half the range of brightnesses to these images, so as to show both negative and positive values of noise. From left to right, the noise has standard deviation 1/256, 4/256 and 16/256 of the full range of brightness respectively. This corresponds roughly to bits zero, two and five of a camera that has an output range of eight bits per pixel. The lower row shows this noise added to an image. In each case, values below zero or above the full range have been adjusted to zero or the maximum value accordingly.

The variance of the noise response is obtained as easily. We want to determine

$$\begin{aligned}\mathtt{E}[\{R(\mathcal{N})_{i,j} - \mathtt{E}[R(\mathcal{N})_{i,j}]\}^2]) &= \int \{\{R(\mathcal{N})_{i,j} - \mathtt{E}[R(\mathcal{N})_{i,j}]\}^2 p(N_{0,0}, \ldots, N_{r,s}) dN_{0,0} \ldots dN_{r,s} \\ &= \int \{\sum_{u,v} G_{-u,-v}(N_{u,v} - \mu)\}^2 p(N_{0,0}, \ldots, N_{r,s}) dN_{0,0} \ldots dN_{r,s}\end{aligned}$$

This expression expands into a sum of two kinds of integral. Terms of the form

$$\int G^2_{-u,-v}(N_{u,v} - \mu)^2 p(N_{0,0}, \ldots, N_{r,s}) dN_{0,0} \ldots dN_{r,s}$$

(for some u, v) can be integrated easily, because each $N_{u,v}$ is independent; the integral is $\sigma^2 G^2_{-u,-v}$ where σ is the standard deviation of the noise. Terms of the

form

$$\int G_{-u,-v}G_{-a,-b}(N_{u,v}-\mu)(N_{a,b}-\mu)p(N_{0,0},\ldots,N_{r,s})dN_{0,0}\ldots dN_{r,s}$$

(for some u, v and a, b) integrate to zero, again because each noise term is independent. We now have:

$$\mathtt{E}[\{R(\mathcal{N})_{i,j}-\mathtt{E}[R(\mathcal{N})_{i,j}]\}^2]=\sigma^2\sum G_{u,v}^2$$

Finite Difference Filters and Gaussian Noise

From these results, we get some insight into the noise behaviour of finite differences. Assume we have an image of stationary Gaussian noise of zero mean, and consider the variance of the response to a finite difference filter that estimates derivatives of increasing order. We shall use the kernel

$$\begin{matrix}0 & 0\\ 1 & -1\\ 0 & 0\end{matrix}$$

to estimate the first derivative. Now a second derivative is simply a first derivative applied to a first derivative, so the kernel will be:

$$\begin{matrix}0 & 0 & 0\\ 1 & -2 & 1\\ 0 & 0 & 0\end{matrix}$$

With a little thought, you can convince yourself that under this scheme, the kernel coefficients of a k'th derivative come from the $k+1$'th row of Pascal's triangle, with appropriate flips of sign. For each of these derivative filters, the mean response to Gaussian noise is zero; but the variance of this response goes up sharply; for the k'th derivative it is the sum of squares of the $k+1$'th row of Pascal's triangle times the standard deviation. Figure 9.4 illustrates this result.

Smoothing alleviates this effect, but the explanation needs a little thought. Assume we smooth a noisy image, and then differentiate it. Firstly, the variance of the noise will tend to be reduced by a smoothing kernel. This is because we tend to use smoothing kernels which are positive, and for which

$$\sum_{uv}G_{uv}=1$$

which means that

$$\sum_{uv}G_{uv}^2\leq 1$$

Secondly, pixels will have a greater tendency to look like neighbouring pixels — if we take stationary additive Gaussian noise, and smooth it, the pixel values of the

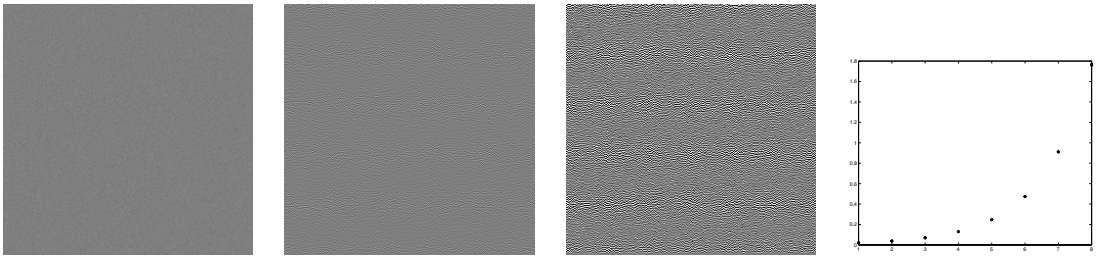

Figure 9.4. Finite differences can accentuate additive Gaussian noise substantially, following the argument in section **??**. On the left, an image of zero mean Gaussian noise with standard deviation 4/256 of the full range. The second figure shows a finite difference estimate of the third derivative in the x direction, and the third shows the sixth derivative in the x direction. In each case, the image has been centered by adding half the full range to show both positive and negative deviations. The images are shown using the same grey level scale; in the case of the sixth derivative, some values exceed the range of this scale. The rightmost image shows the standard deviations of these noise images compared with those predicted by the Pascal's triangle argument.

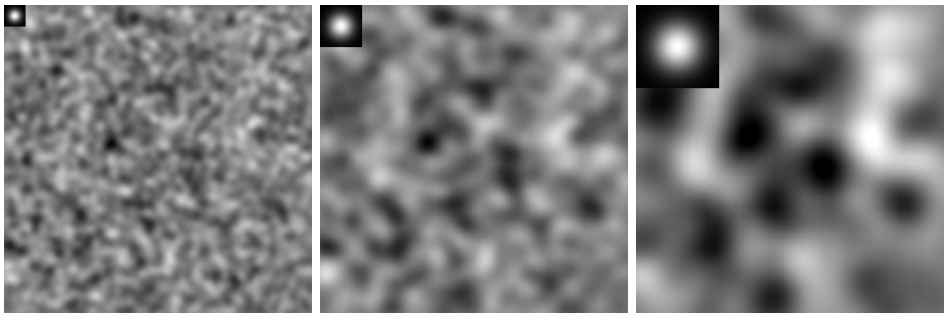

Figure 9.5. Smoothing stationary additive Gaussian noise results in signals where pixel values tend to be increasingly similar to the value of neighbouring pixels. This occurs at about the scale of the filter kernel, because the filter kernel causes the correlations. The figures show noise smoothed with increasingly large Gaussian smoothing kernels. Grey pixels have zero value, darker values are negative and brighter values are positive. The kernels are shown in the top right hand corners of the figures, to indicate the spatial scale of the kernel (we have scaled the brightness of the kernels, which are Gaussians, so that the center pixel is white and the boundary pixels are black). Smoothed noise tends to look like natural texture, as the figures indicate.

resulting signal *are no longer independent*. In some sense, this is what smoothing was about — recall we introduced smoothing as a method to predict a pixel's value from the values of its neighbours. However, if pixels tend to look like their neighbours, then derivatives must be smaller (because they measure the tendency

of pixels to look different from their neighbours).

Smoothed noise has applications. As figure 9.5 indicates, smoothed noise tends to look like some kinds of natural texture, and smoothed noise is quite widely used as a source of textures in computer graphics applications []).

Difficulties with the Additive Stationary Gaussian Noise Model

Taken literally, the additive stationary Gaussian noise model is poor model of image noise. Firstly, the model allows positive (and, more alarmingly, *negative!*) pixel values of arbitrary magnitude. With appropriate choices of standard deviation for typical current cameras operating indoors or in daylight, this doesn't present much of a problem, because these pixel values are extremely unlikely to occur in practice. In rendering noise images, the problematic pixels that do occur are fixed at zero or full output respectively.

Secondly, noise values are completely independent, so this model does not capture the possibility of groups of pixels that have correlated responses, perhaps because of the design of the camera electronics or because of hot spots in the camera integrated circuit. This problem is harder to deal with, because noise models that do model this effect tend to be difficult to deal with analytically. Finally, this model does not describe "dead pixels" (pixels that consistently report no incoming light, or are consistently saturated) terribly well. If the standard deviation is quite large and we threshold pixel values, then dead pixels will occur, but the standard deviation may be too large to model the rest of the image well. A crucial advantage of additive Gaussian noise is that it is easy to estimate the response of filters to this noise model. In turn, this gives us some idea of how effective the filter is at responding to signal and ignoring noise.

9.3 Edges and Gradient-based Edge Detectors

Sharp changes in image brightness are interesting for many reasons. Firstly, object boundaries often generate sharp changes in brightness — a light object may lie on a dark background, or a light object may lie on a dark background. Secondly, reflectance changes often generate sharp changes in brightness which can be quite distinctive — zebras have stripes and leopards have spots. Cast shadows can also generate sharp changes in brightness. Finally, sharp changes in surface orientation are often associated with sharp changes in image brightness.

Points in the image where brightness changes particularly sharply are often called **edges** or **edge points**. We should like edge points to be associated with the boundaries of objects and other kinds of meaningful changes. It is hard to define precisely the changes we would like to mark — is the region of a pastoral scene where the leaves give way to the sky the boundary of an object? Typically, it is hard to tell a semantically meaningful edge from a nuisance edge, and to do so requires a great deal of high-level information. Nonetheless, experience building vision systems suggests that very often, interesting things are happening in an image

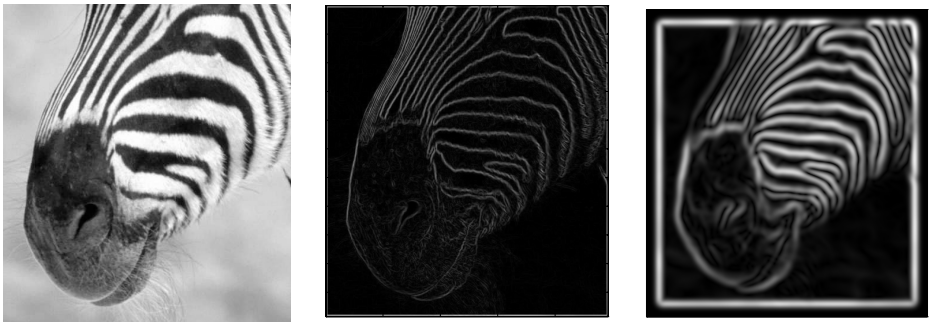

Figure 9.6. The gradient magnitude can be estimated by smoothing an image and then differentiating it. This is equivalent to convolving with the derivative of a smoothing kernel. The extent of the smoothing affects the gradient magnitude; in this figure, we show the gradient magnitude for the figure of a zebra at different scales. On the left, gradient magnitude estimated using the derivatives of a Gaussian with $\sigma = 1$ pixel and on the right gradient magnitude estimated using the derivatives of a Gaussian with $\sigma = 2$ pixel. Notice that large values of the gradient magnitude form thick trails.

at an edge and it is worth knowing where the edges are.

We will proceed with a rather qualitative model of an edge as a point where the change of image brightness is distinctive and large. One sign of a sharp change in an image is a large gradient magnitude. Typically, the gradient magnitude can be large along a thick trail in an image (figure 9.6). Object outlines are curves however, and we should like to obtain a curve of the most distinctive points on this trail.

A natural approach is to look for points where the gradient magnitude is a maximum along the direction perpendicular to the edge. For this approach, the direction perpendicular to the edge can be estimated using the direction of the gradient (figure 9.7). These considerations yield algorithm 1. Most current edgefinders follow these lines, but there remain substantial debates about the proper execution of the details.

9.3.1 Estimating Gradients

As figure 9.2 indicates, simple finite difference filters tend to give strong responses to noise, so that applying two finite difference filters is a poor way to estimate a gradient. However, we expect that any change of significance to us has effects over a pool of pixels. For example, the contour of an object can result in a long chain of points where the image derivative is large. For many kinds of noise model, large image derivatives due to noise are an essentially local event. This means that smoothing a differentiated image would tend to pool support for the changes we are interested in, and to suppress the effects of noise. An alternative interpretation of the point is that the changes we are interested in will not be suppressed by some

```
form an estimate of the image gradient

obtain the gradient magnitude from this estimate

identify image points where the value
of the gradient magnitude is maximal
in the direction perpendicular to the edge
and also large; these points are edge points
```

Algorithm 9.1: *Gradient based edge detection.*

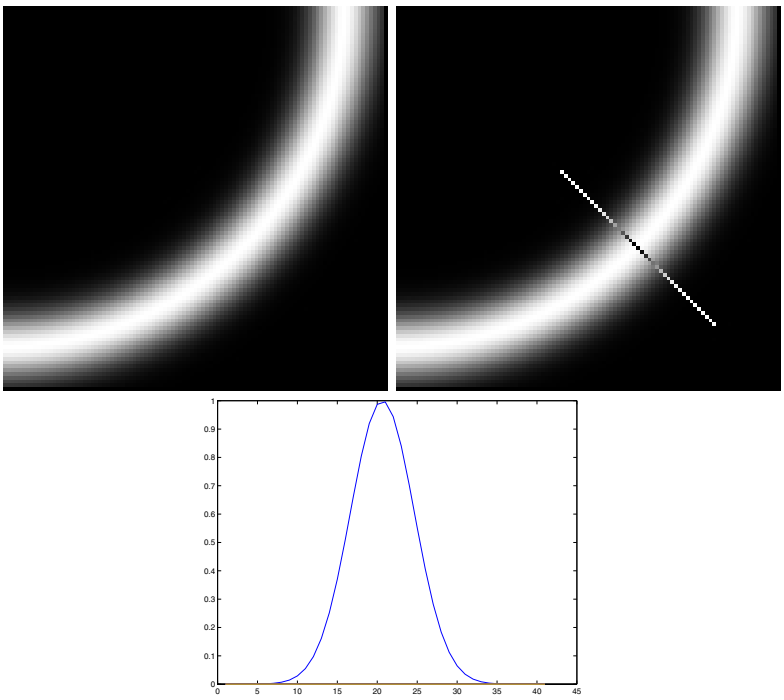

Figure 9.7. The gradient magnitude tends to be large along thick trails in an image. Typically, we would like to condense these trails into curves of representative edge points. A natural way to do this is to cut the trail perpendicular to its direction and look for a peak. We will use the gradient direction as an estimate of the direction in which to cut. The top left figure shows a trail of large gradient magnitude; the figure on the top right shows an appropriate cutting direction; and below, we show the peak in this direction.

smoothing, which will tend to suppress the effects of noise. There is no difference in

principle between differentiating a smoothed image, or smoothing a differentiated image.

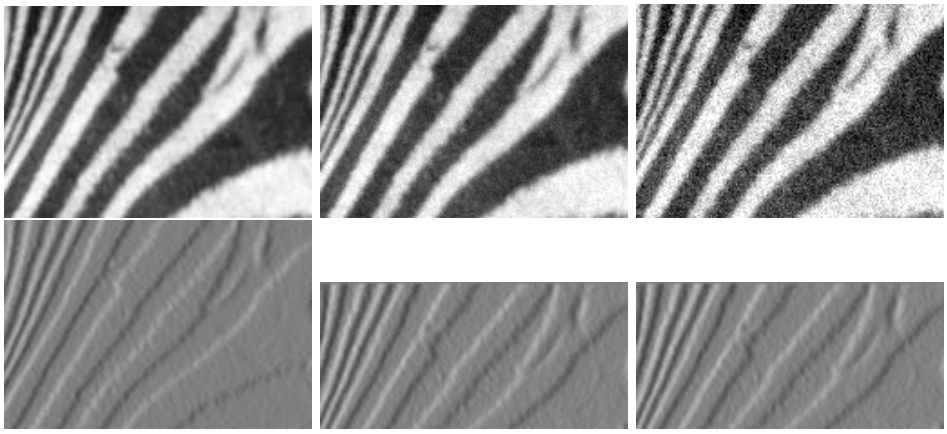

Figure 9.8. Derivative of Gaussian filters are less extroverted in their response to noise than finite difference filters. The image at top left shows a detail from a picture of a zebra corrupted by zero mean stationary additive Gaussian noise, with $\sigma = 0.03$ (pixel values range from 0 to 1). Bottom left shows the same image corrupted by zero mean stationary additive Gaussian noise, with $\sigma = 0.09$. The second column shows the partial derivative in the y-direction, and the third column shows the partial derivative in the x-direction, of the image at the head of the row, in each case estimated by a derivative of Gaussian filter with σ one pixel. Notice how the smoothing helps to reduce the impact of the noise.

In practice, it is usual to differentiate a smoothed image.

9.3.2 Choosing a Smoothing Filter

The smoothing filter can be chosen by taking a model of an edge and then using some set of criteria to choose a filter that gives the best response to that model. It is difficult to pose this problem as a two dimensional problem, because edges in 2D can be curved. Conventionally, the smoothing filter is chosen by formulating a one-dimensional problem, and then using a rotationally symmetric version of the filter in 2D.

The one-dimensional filter must be obtained from a model of an edge. The usual model is a step function of unknown height, in the presence of stationary additive Gaussian noise:

$$edge(x) = AU(x) + n(x)$$

where

$$U(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$$

(the value of $U(0)$ is irrelevant to our purpose). A is usually referred to as the **contrast** of the edge. In the 1D problem, finding the gradient magnitude is the

same as finding the square of the derivative response. For this reason, we usually seek a derivative estimation filter rather than a smoothing filter (which can then be reconstructed by integrating the derivative estimation filter).

Canny established the practice of choosing a derivative estimation filter by using the continuous model to optimize a combination of three criteria:

- **Signal to noise ratio** — the filter should respond more strongly to the edge *at* $x = 0$ than to noise.
- **Localisation** — the filter response should reach a maximum very close to $x = 0$.
- **Low false positives** — there should be only one maximum of the response in a reasonable neighbourhood of $x = 0$.

Once a continuous filter has been found, it is discretised. The criteria can be combined in a variety of ways, yielding a variety of somewhat different filters. It is a remarkable fact that the optimal smoothing filters that are derived by most combinations of these criteria tend to look a great deal like Gaussians — this is intuitively reasonable, as the smoothing filter must place strong weight on center pixels and less weight on distant pixels, rather like a Gaussian. In practice, optimal smoothing filters are usually replaced by a Gaussian, with no particularly important degradation in performance.

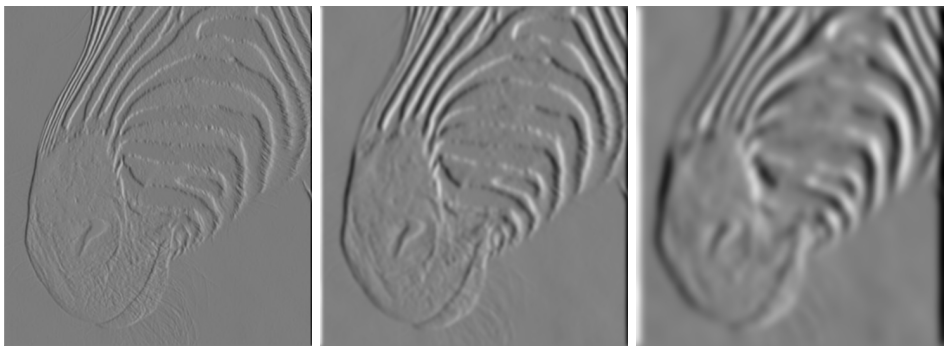

Figure 9.9. The scale (i.e. σ) of the Gaussian used in a derivative of Gaussian filter has significant effects on the results. The three images show estimates of the derivative in the x direction of an image of the head of a zebra, obtained using a derivative of Gaussian filter with σ one pixel, three pixels and seven pixels (moving to the right). Note how images at a finer scale show some hair and the animal's whiskers disappear at a medium scale, and the fine stripes at the top of the muzzle disappear at the coarser scale.

The choice of σ used in estimating the derivative is often called the *scale* of the smoothing. Scale has a substantial effect on the response of a derivative filter. Assume we have a narrow bar on a constant background, rather like the zebra's

whisker. Smoothing on a scale smaller than the width of the bar will mean that the filter responds on each side of the bar, and we will be able to resolve the rising and falling edges of the bar. If the filter width is much greater, the bar will be smoothed into the background, and the bar will generate little or no response (as figure 9.9).

9.3.3 Why Smooth with a Gaussian?

While a Gaussian is not the only possible blurring kernel, it is convenient because it has a number of important properties. Firstly, if we convolve a Gaussian with a Gaussian, and the result is another Gaussian:

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

This means that it is possible to obtain very heavily smoothed images by resmoothing smoothed images. This is a significant property, firstly because discrete convolution can be an expensive operation (particularly if the kernel of the filter is large), and secondly because it is common to want to see versions of an image smoothed by different amounts.

Efficiency

Consider convolving an image with a Gaussian kernel with σ one pixel. Although the Gaussian kernel is non zero over an infinite domain, for most of that domain it is extremely small because of the exponential form. For σ one pixel, points outside a 5x5 integer grid centered at the origin have values less than $e^{-4} = 0.0184$ and points outside a 7x7 integer grid centered at the origin have values less than $e^{-9} = 0.0001234$. This means that we can ignore their contributions, and represent the discrete Gaussian as a small array (5x5 or 7x7, according to taste and the number of bits you allocate to representing the kernel).

However, if σ is 10 pixels, we may need a 50x50 array or worse. A back of the envelope count of operations should convince you that convolving a reasonably sized image with a 50x50 array is an unattractice prospect. The alternative — convolving repeatedly with a much smaller kernel — is much more efficient, *because we don't need to keep every pixel in the interim.* This is because a smoothed image is, to some extent, redundant (most pixels contain a significant component of their neighbours' values). As a result, some pixels can be discarded. We then have a strategy which is quite efficient: smooth, subsample, smooth, subsample, etc. The result is an image that has the same information as a heavily smoothed image, but is very much smaller and is easier to obtain. We explore the details of this approach in section 9.4.1.

The Central Limit Theorem

Gaussians have another significant property which we shall not prove but illustrate in figure 9.10. For an important family of functions, convolving any member of that family of functions with itself repeatedly will eventually yield a Gaussian. With

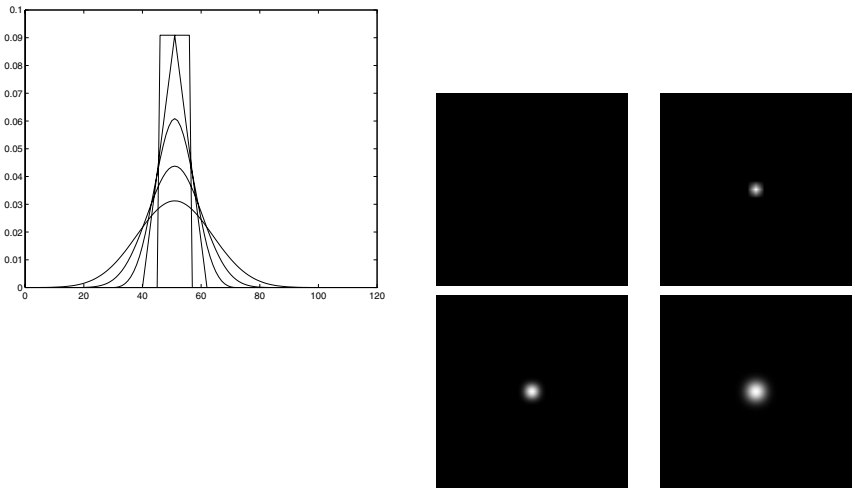

Figure 9.10. The central limit theorem states that repeated convolution of a positive kernel with itself will eventually limit towards a kernel that is a scaling of a Gaussian. The top figure illustrates this effect for 1D convolution; the triangle is obtained by convolving a box function with itself; each succeding stage is obtained by convolving the previous stage with itself. The four images show a 2D box function convolved with itself 0, 1, 2 and 4 times (clockwise).

the associativity of convolution, this implies that if we choose a different smoothing kernel, and apply it repeatedly to the image, the result will eventually look as though we had smoothed the image with a Gaussian anyhow.

Gaussians are Separable

Finally, a Gaussian can be factored as

$$\begin{aligned} G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x^2)}{2\sigma^2}\right)\right) \times \left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y^2)}{2\sigma^2}\right)\right) \end{aligned}$$

and this is a product of two 1-D Gaussians. Generally, a function $f(x,y)$ that factors as $f(x,y) = g(x)h(y)$ is referred to as a **tensor product**. It is common to refer to filter kernels that are tensor products as **separable kernels**. Separability is a very useful property indeed. In particular, convolving with a filter kernel that is separable is the same as convolving with two 1-D kernels, one in the x direction and another in the y direction (exercises).

Many other kernels are separable. Separable filter kernels result in discrete representations that factor as well. In particular, if $\mathcal{H}$ is a discretised separable

filter kernel, then there are some vectors $\boldsymbol{f}$ and $\boldsymbol{g}$ such that

$$H_{ij} = f_i g_j$$

It is possible to identify this property using techniques from numerical linear algebra; commercial convolution packages often test the kernel to see if it is separable before applying it to the image. The cost of this test is easily paid off by the savings if the kernel does turn out to be separable.

9.3.4 Derivative of Gaussian Filters

Smoothing an image and then differentiating it is the same as convolving it with the derivative of a smoothing kernel. This fact is most easily seen by thinking about continuous convolution.

Firstly, differentiation is linear and shift invariant. This means that there is some kernel — we dodge the question of what it looks like — that differentiates. That is, given a function $I(x, y)$

$$\frac{\partial I}{\partial x} = K_{\frac{\partial}{\partial x}} * * I$$

Now we want the derivative of a smoothed function. We write the convolution kernel for the smoothing as S. Recalling that convolution is associative, we have

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

This fact appears in its most commonly used form when the smoothing function is a Gaussian; we can then write

$$\frac{\partial \left(G_\sigma * * I\right)}{\partial x} = (\frac{\partial G_\sigma}{\partial x}) * * I$$

i.e. we need only convolve with the derivative of the Gaussian, rather than convolve and then differentiate. A similar remark applies to the Laplacian. Recall that the Laplacian of a function in 2D is defined as:

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

Again, because convolution is associative, we have that

$$(K_{\nabla^2} * * (G_\sigma * * I)) = (K_{\nabla^2} * * G_\sigma) * * I = (\nabla^2 G_\sigma) * * I$$

This practice results in much smaller noise responses from the derivative estimates (figure 9.8).

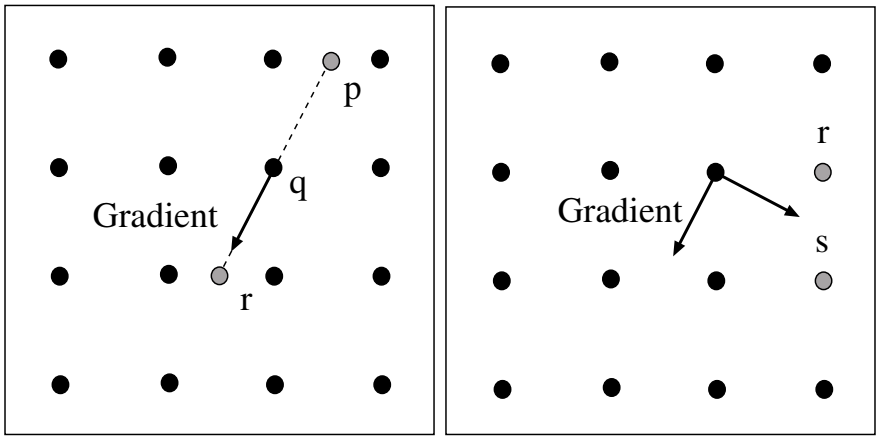

Figure 9.11. Non-maximum suppression obtains points where the gradient magnitude is at a maximum *along the direction of the gradient*. The figure on the left shows how we reconstruct the gradient magnitude. The dots are the pixel grid. We are at pixel $\boldsymbol{q}$, attempting to determine whether the gradient is at a maximum; the gradient direction through $\boldsymbol{q}$ does not pass through any convenient pixels in the forward or backward direction, so we must interpolate to obtain the values of the gradient magnitude at $\boldsymbol{p}$ and $\boldsymbol{r}$; if the value at $\boldsymbol{q}$ is larger than both, $\boldsymbol{q}$ is an edge point. Typically, the magnitude values are reconstructed with a linear interpolate, which in this case would use the pixels to the left and right of $\boldsymbol{p}$ and $\boldsymbol{r}$ respectively to interpolate values at those points. On the right, we sketch how to find candidates for the next edge point, given that $\boldsymbol{q}$ is an edge point; an appropriate search direction is perpendicular to the gradient, so that points $\boldsymbol{s}$ and $\boldsymbol{t}$ should be considered for the next edge point. Notice that, in principle, we don't need to restrict ourselves to pixel points on the image grid, because we know where the predicted position lies between $\boldsymbol{s}$ and $\boldsymbol{t}$, so that we could again interpolate to obtain gradient values for points off the grid.

9.3.5 Identifying Edge Points from Filter Outputs

Given estimates of gradient magnitude, we would like to obtain edge points. Again, there is clearly no objective definition, and we proceed by reasonable intuition. The gradient magnitude can be thought of as a chain of low hills. Marking local extrema would mark isolated points — the hilltops in the analogy. A better criterion is to slice the gradient magnitude along the gradient direction — which should be perpendicular to the edge — and mark the points along the slice where the magnitude is maximal. This would get a chain of points along the crown of the hills in our chain; the process is called **non-maximum suppression**.

Typically, we expect edge points to occur along curve-like chains. The significant steps in non maximum suppression are:

- determining whether a given point is an edge point;

- and, if it is, finding the next edge point.

Once these steps are understood, it is easy to enumerate all edge chains. We find the first edge point, mark it, expand all chains through that point exhaustively, marking all points along those chains, and continue to do this for all unmarked edge points.

```
While there are points with high gradient
that have not been visited

  Find a start point that is a local maximum in the
    direction perpendicular to the gradient
    erasing points that have been checked

  while possible, expand a chain through
    the current point by:
      1) predicting a set of next points, using
        the direction perpendicular to the gradient

      2) finding which (if any) is a local maximum
        in the gradient direction

      3) testing if the gradient magnitude at the
        maximum is sufficiently large

      4) leaving a record that the point and
        neighbours have been visited

      record the next point, which becomes the current point

  end

end
```

Algorithm 9.2: *Non-maximum suppression.*

The two main steps are simple. For the moment, assume that edges are to be marked at pixel locations (rather than, say, at some finer subdivision of the pixel grid). We can determine whether the gradient magnitude is maximal at any pixel by comparing it with values at points some way backwards and forwards *along the gradient direction*. This is a function of distance along the gradient; typically we step forward to the next row (or column) of pixels and backwards to the previous to

determine whether the magnitude at our pixel is larger (figure 9.11). The gradient direction does not usually pass through the next pixel, so we must interpolate to determine the value of the gradient magnitude at the points we are interested in; a linear interpolate is usual.

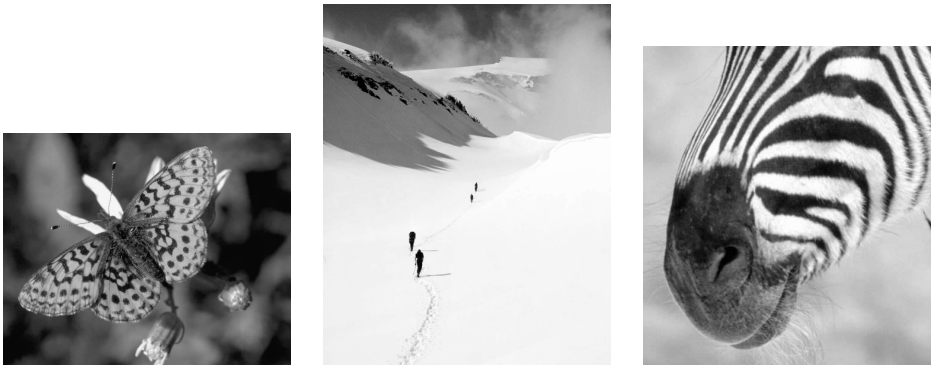

Figure 9.12. We use these three images to illustrate properties of a gradient based edge detector. The butterfly is on a blurred background; there is strong contrast between the figures on the snow and the background; and the zebra's nose has fine scale detail — its whiskers — as well as coarse scale detail.

If the pixel turns out to be an edge point, the next edge point in the curve can be guessed by taking a step perpendicular to the gradient. This step will not, in general, end on a pixel; a natural strategy is to look at the neighbouring pixels that lie close to that direction (see figure 9.11. This approach leads to a set of curves that can be represented by rendering them in black on a white background, as in figure 9.13.

There are too many of these curves to come close to being a reasonable representation of object boundaries. This is, in part, because we have marked maxima of the gradient magnitude without regard to how large these maxima are. It is more usual to apply a threshold test, to ensure that the maxima are greater than some lower bound. This in turn leads to broken edge curves (figure **??**). The usual trick for dealing with this is to use **hysteresis**; we have two thresholds, and refer to the *larger* when starting an edge chain and the *smaller* while following it. The trick often results in an improvement in edge outputs (exercises)

9.4 Technique: Scale and Image Pyramids

Images look quite different at different scales. For example, the zebra's nose in figure **??** can be described in terms of individual hairs — which might be coded in terms of the response of oriented filters that operate at a scale of a small number of pixels — or in terms of the stripes on the zebra.

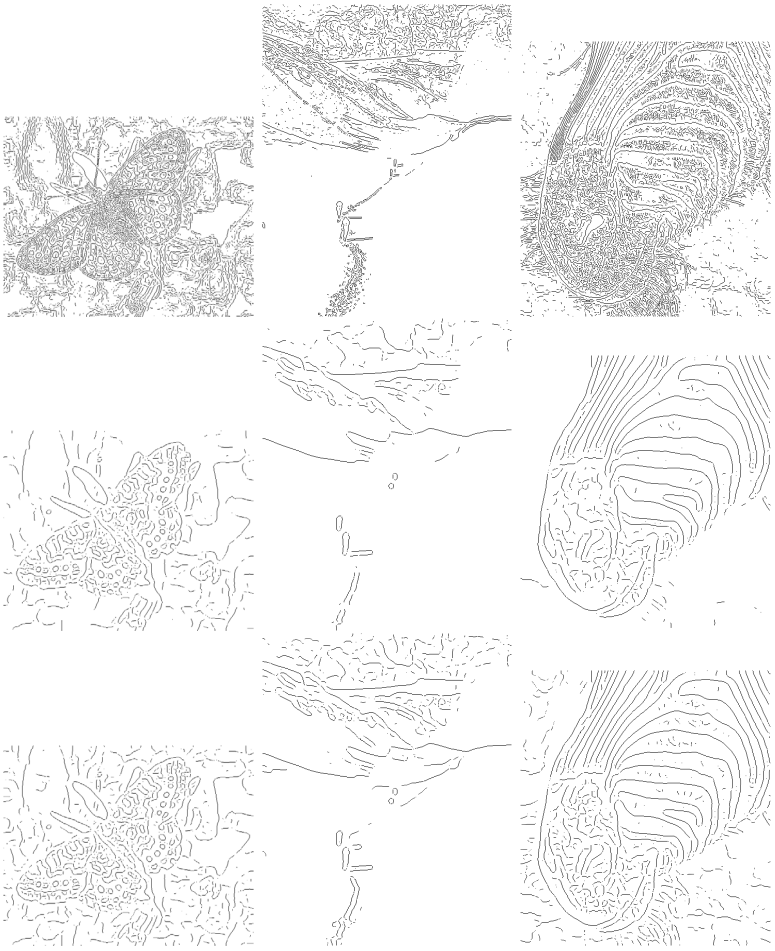

Figure 9.13. Edge points marked on the pixel grid for the three images shown in 9.12. The top row shows edge points obtained using a Gaussian smoothing filter at σ one pixel, and the center and bottom rows show points obtained using a smoothing filter at σ four pixels. For the top two cases, gradient magnitude has been tested against a high threshold to determine whether a point is an edge point or not; for the bottom row, gradient magnitude was tested against a low threshold. At a fine scale, fine detail at high contrast generates edge points, which disappear at the coarser scale — for example, the zebra's whiskers disappear. When the threshold is high, curves of edge points are often broken because the gradient magnitude dips below the threshold; for the low threshold, a variety of new edge points of dubious significance are introduced.

9.4.1 The Gaussian Pyramid

A **pyramid** is a collection of representations of an image. The name pyramid comes from a visual analogy. Typically, each layer of the pyramid is half the width and

Figure 9.14. A Gaussian pyramid of images, running from 512x512 to 8x8. Details in the text.

half the height of the previous layer, and if we were to stack the layers on top of each other a pyramid would result. In a **Gaussian pyramid**, each layer is smoothed by a symmetric Gaussian kernel and resampled to get the next layer (figure 9.14). These pyramids are most convenient if the image dimensions are a power of two, or a multiple of a power of two. The smallest image is the most heavily smoothed; the layers are often referred to as **coarse scale** versions of the image.

There are a variety of other kinds of pyramid, best understood with the aid of a little notation. To simplify things, assume that the original image is square, with image dimensions 2^k. The operator $S^{\downarrow}$ downsamples an image; in particular, the j, k'th element of $S^{\downarrow}(\mathcal{I})$ is the $2j$, $2k$'th element of $\mathcal{I}$. The n'th level of a pyramid $P(\mathcal{I})$ is denoted $P(\mathcal{I})_n$.

We can now write simple expressions for the layers of a Gaussian pyramid:

$$P_{\text{Gaussian}}(\mathcal{I})_{n+1} = S^{\downarrow}(G_\sigma ** P_{\text{Gaussian}}(\mathcal{I})_n) \quad (9.4.1)$$
$$= (S^{\downarrow} G_\sigma) P_{\text{Gaussian}}(\mathcal{I})_n \quad (9.4.2)$$

(where we have written G_σ for the linear operator that takes an image to the convolution of that image with a Gaussian). The finest scale layer is the original image

$$P_{\text{Gaussian}}(\mathcal{I})_1 = \mathcal{I}$$

A simple, immediate use for a Gaussian pyramid is to obtain zero-crossings of a Laplacian of Gaussian (or a DOG) at various levels of smoothing.

Algorithm

```
Set the finest scale layer to the image

For each layer, going from next to finest to coarsest

  Obtain this layer by smoothing the next finest
  layer with a Gaussian, and then subsampling it

end
```

9.3: *Forming a Gaussian pyramid*

9.4.2 Applications of Scaled Representations

Gaussian pyramids are useful, because they make it possible to extract representations of different types of structure in an image. For example, in the case of the zebra, we would not want to apply very large filters to find the stripes. This is because these filters are inclined to spurious precision — we don't wish to have to represent the disposition of each hair on the stripe — inconvenient to build, and slow to apply.

A more practical approach than applying very large filters is to apply smaller filters to a less detailed version of the image. We expect effects to appear at a variety of scales, and so represent the image in terms of several smoothed and sampled versions. Continuing with the zebra analogy, this means we can represent the hair on the animal's nose, the stripes of its body, dark legs, and entire zebras, each as bars of different sizes.

Scale and Spatial Search

Another application is spatial search, a common theme in computer vision. Typically, we have a point in one image and are trying to find a point in a second image that corresponds to it. This problem occurs in stereopsis — where the point has moved because the two images are obtained from different viewing positions — and in motion analysis — where the image point has moved either because the camera moved, or because it is on a moving object.

Searching for a match in the original pairs of images is inefficient, because we may have to wade through a great deal of detail. A better approach, which is now pretty much universal, is to look for a match in a heavily smoothed and resampled image, and then refine that match by looking at increasingly detailed versions of the image. For example, we might reduce 1024x1024 images down to 4x4 versions, match those and then look at 8x8 versions (because we know a rough match, it is easy to refine it); we then look at 16x16 versions, etc. all the way up to 1024x1024. This gives an extremely efficient search, because a step of a single pixel in the 4x4 version is equivalent to a step of 256 pixels in the 1024x1024 version. We will explore this strategy of **coarse-to-fine matching** in greater detail in chapters **??**.

Edge Tracking

Most edges found at coarse levels of smoothing tend are associated with large, high contrast image events, because for an edge to be marked at a coarse scale a large pool of pixels need to agree that there is a high contrast edge. Typically, these edges understate the extent of a feature — the contrast might decay along the edge, for example — and their localisation can be quite poor — a single pixel error in a coarse-scale image represents a multiple pixel error in a fine-scale image.

At fine scales, there are many edges, some of which are associated with smaller, low contrast events. One strategy for improving a set of edges obtained at a fine scale is to track edges across scales to a coarser scale, and accept only the fine scale edges that have identifiable parents at a coarser scale. This strategy in principle can suppress edges resulting from textured regions (often referred to as "noise") and edges resulting from real noise.

9.4.3 Scale Space

Coarse scale components give the overall structure of a signal, and fine scale components give detailed information, as figure **??** suggests. This approach allows us to think about representing such objects as trees, which appear to exist at several distinct scales; we would want to be able to represent a tree both as a puff of foliage on top of a stalk (coarse scale) and as a collection of individual leaves and twigs (fine scale).

Gaussian pyramids are not an ideal tool for this representation, because they sample the range of smoothed images quite coarsely. Instead of a discrete pyramid of images, we might consider a one parameter family of images (or, equivalently, a

Missing Figure

Figure 9.15. Edge tracking

function of three dimensions)

$$\Phi(x, y, u) = G_{\sigma(u)} * \mathcal{I}(x, y)$$

where the extent of the smoothing is now a continuous parameter. For a 1D signal, we can draw the behaviour of features as the scaling parameter changes, and this drawing gives us a simple and sometimes quite informative description of the signal (figure **??**). If we define a "feature" to be a zero-crossing of the Laplacian, then it is possible to show that in this family of images, features are not created by smoothing. This means that all coarse-scale zero-crossings will have finer-scale events corresponding to them, so that there are quite simple rules for what these drawings look like (figure **??**). These drawings and the underlying representations are often referred to as **scale space**.

If we think of the signal as being composed of a set of parametric patches joined at these feature points, we obtain a different decomposition of the signal at each scale. Instead of representing the signal with a continuous family of decompositions where the feature points move, we could fix the location of these feature points and then use the simple rules for constructing scale-space drawings to obtain a strip-like decomposition (as in figure **??**). This is not a decomposition with a canonical form — we get to choose what the features are and the nature of the parametric patches — but it is often rather useful in practice, mainly because the decomposition of the signal appears to reflect the structure apparent to humans.

2D scale space

It is possible to extend these decompositions from 1D to 2D. Again, the choice of features is somewhat open, but a reasonable choice is the points of maximum or minimum brightness. Smoothing an image with a symmetric Gaussian cannot create local maxima or minima in brightness, but it can (and does) destroy them. Assume we have a scale value σ_{die} where a maximum (or minimum — we will just talk about the one case, for simplicity) is destroyed. If we now *reduce* the scale, a standard pattern will appear — there will be a corresponding maximum surrounded

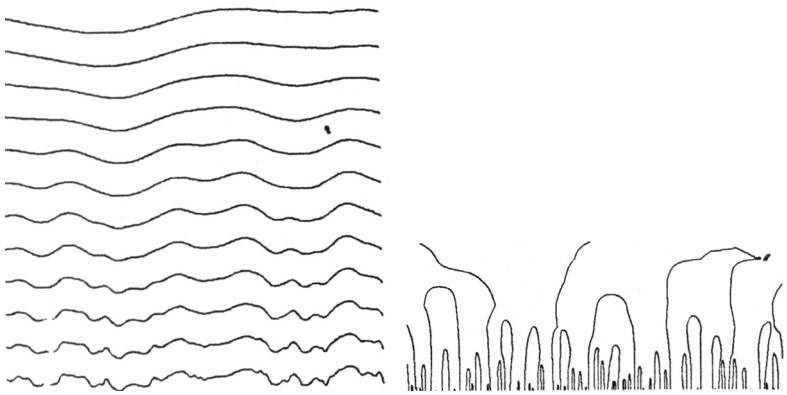

Figure 9.16. On the left, a 1D signal smoothed using Gaussian filters of increasing σ (scale increases vertically; the signal is a function of the horizontal coordinate). As the signal is smoothed, extrema merge and vanish. The smoothest versions of the signal can be seen as an indication of the "overall trend" of the signal, and the finer versions have increasing amounts of detail imposed. The representation on the left marks the position of zero crossings of the second derivative of the smoothed signal, as the smoothing increases (again, scale increases vertically). Notice that zero crossings can meet and obliterate one another as the signal is smoothed but no new zero-crossing is created. This means that the figure shows the characteristic structure of either vertical curves or inverted "u" curves. An inverted pitchfork shape is also possible — where three extrema meet and become one — but this requires special properties of the signal and usually becomes an inverted u next to a vertical curve. Notice also that the position of zero crossings tends to shift as the signal is smoothed. *Figure from "Scale-space filtering," A.P. Witkin, Proc. 8'th IJCAI, 1983, page 1020, in the fervent hope, etc.*

by a curve of equal brightness that has a self intersection. Thus, corresponding to each maximum (or minimum) at any scale, we have a **blob**, which is the region of the image marked out by this curve of equal brightness. Typically, maxima are represented by light blobs and minima by dark blobs. All of this gives us a representation of an image in terms of blobs growing (or dying) as the scale is decreased (or increased).

9.4.4 Anisotropic Scaling

One important difficulty with scale space models is that the symmetric Gaussian smoothing process tends to blur out edges rather two aggressively for comfort. For example, if we have two trees near one another on a skyline, the large scale blobs corresponding to each tree may start merging before all the small scale blobs have finished. This suggests that we should smooth differently at edge points than at other points. For example, we might make an estimate of the magnitude and orientation of the gradient: for large gradients, we would then use an oriented smoothing

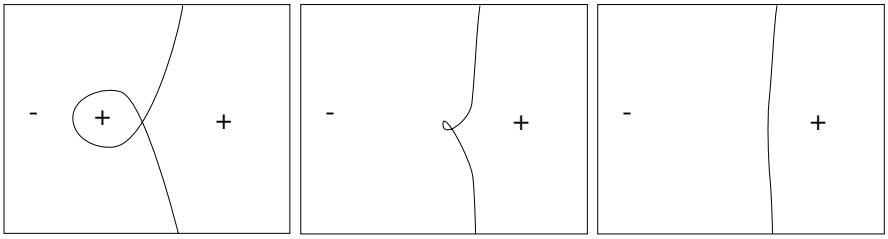

Figure 9.17. Smoothing an image with a symmetric Gaussian cannot create local maxima or local minima of brightness. However, local extrema can be extinguished. What happens is that a local maximum shrinks down to the value of the surrounding pixels. This is most easily conveyed by drawing a contour plot — the figure shows a contour plot with a local maximum which dies as the image is smoothed (the curve is a contour of constant brightness; the regions marked + have brightness greater than that of the contour, so the blob must have a maximum in it; the regions marked - have brightness less than that of the contour). Recording the details of these disappearances — where the maximum that disappears is, the contour defining the blob around the maximum, and the scale at which it disappears — yields a scale-space representation of the image.

operator that smoothed aggressively perpendicular to the gradient and very little along the gradient; for small gradients, we might use a symmetric smoothing operator. This idea used to be known as **edge preserving smoothing**.

In the modern, more formal version (details in in []), we notice the scale space representation family is a solution to the **diffusion equation**

$$\begin{aligned}\frac{\partial \Phi}{\partial \sigma} &= \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} \\ &= \nabla^2 \Phi \\ \Phi(x, y, 0) &= \mathcal{I}(x, y)\end{aligned}$$

If this equation is modified to have the form

$$\begin{aligned}\frac{\partial \Phi}{\partial \sigma} &= \nabla \cdot (c(x, y, \sigma) \nabla \Phi) \\ &= c(x, y, \sigma) \nabla^2 \Phi + (\nabla c(x, y, \sigma)) \cdot (\nabla \Phi) \\ \Phi(x, y, 0) &= \mathcal{I}(x, y)\end{aligned}$$

then if $c(x, y, \sigma) = 1$, we have the diffusion equation we started with, and if $c(x, y, \sigma) = 0$ there is no smoothing. We will assume that c does not depend on σ. If we knew where the edges were in the image, we could construct a mask that consisted of regions where $c(x, y) = 1$, isolated by patches along the edges where $c(x, y) = 0$; in this case, a solution would smooth *inside* each separate region, but not over the edge. While we do not know where the edges are — the exercise would be empty if we did — we can obtain reasonable choices of $c(x, y)$ from the

magnitude of the image gradient. If the gradient is large, then c should be small, and vice-versa.

9.5 Human Vision: More of the Visual Pathway

9.5.1 The Lateral Geniculate Nucleus

The LGN is a layered structure, consisting of many sheets of neurons. The layers are divided into two classes — those consisting of cells with large cell bodies (**magnocellular layers**), and those consisting of cells with small cell bodies (**parvocellular layers**). The behaviour of these cells differs as well.

Each layer in the LGN receives input from a single eye, and is laid out like the retina of the eye providing input (an effect known as **retinotopic mapping**). Retinotopic mapping means that nearby regions on the retina end up near one another in the layer, and so we can think of each layer as representing some form of feature map. Neurons in the lateral geniculate nucleus display similar receptive field behaviour to retinal neurons. The role of the LGN is unclear; it is known that the LGN receives input from the cortex and from other regions of the brain, which may modify the visual signal travelling from the retina to the cortex.

9.5.2 The Visual Cortex

Most visual signals arrive at an area of the cortex called **area V1** (or the **primary visual cortex**, or the **striate cortex**). This area is highly structured and has been intensively studied. Most physiological information about the cortex comes from studies of cats or monkeys (which are known to react differently from one another and from humans if the cortex is damaged). The cortex is also retinotopically mapped. It has a highly structured layered architecture, with regions organised by the eye of origin of the signal (often called **ocular dominance columns**). Within these columns, cells are arranged so that their receptive fields move smoothly from the center to the periphery of the visual field. Neurons in the primary visual cortex have been extensively studied. Two classes are usually recognised — **simple cells** and **complex cells**.

Simple cells have **orientation selective** receptive fields, meaning that a particular cell will respond more strongly to an oriented structure. To a good approximation, the response of a simple cell is linear, so that the behaviour of these cells can be modelled with spatial filters. The structure of typical receptive fields means that these cells can be thought of as edge and bar detectors [**?**], or as first and second derivative operators. Some simple cells have more lobes to their receptive field, and can be thought of as detecting higher derivatives. The preferred orientation of a cell varies fairly smoothly in a principled way that depends on the cell's position.

Complex cells typically are highly non-linear, and respond to moving edges or bars. Typically, the cells display **direction selectivity**, in that the response of a cell to a moving bar depends strongly on both the orientation of the bar and the direction

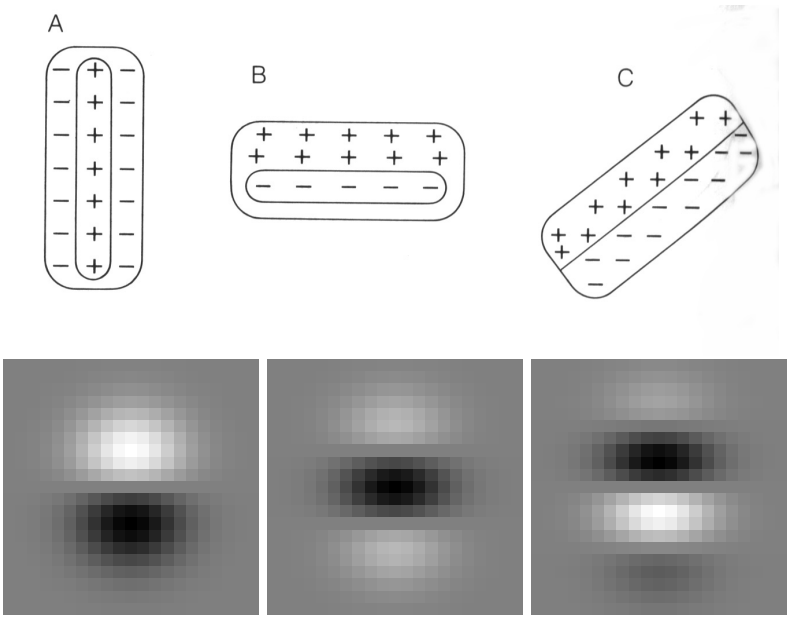

Figure 9.18. Receptive fields of some cortical simple cells, which can be modelled as linear. '+' signs mean that light in the area excites the cell; '-' signs means that it inhibits the cell. Notice that these cells are orientation selective; the cell on the left responds most strongly to a bright bar on a dark background, the cell on the center will respond most strongly to a dark bar off center on a light background, and the cell on the right will respond strongly to an oriented edge. *Figure from John Dowling's book "Neurons and Networks - an introduction to neuroscience", page 336, in the fervent hope that permission will be granted* Notice the similarity to the derivative of Gaussian filters plotted on below.

of the motion (figure 9.19). Some complex cells, often called **hypercomplex cells** respond preferentially to bars of a particular length.

One strong distinction between simple and complex cells appears when one considers the time-course of a response to a **contrast reversing pattern** — a spatial sinusoid whose amplitude is a sinusoidal function of time. Exposed to such a stimulus, a simple cell responds strongly for the positive contrast and not at all for the negative contrast — it is trying to be linear, but because the resting response of the cell is low, there is a limit to the extent to which the cell's output can be inhibited. In contrast, complex cells respond to both phases (figure 9.20). Thus, one can think of a simple cell as performing half-wave rectification — it responds to the positive half of the amplitude signal — and a complex cell as performing full wave rectification — it gives a response to both the positive and negative half of the amplitude signal.

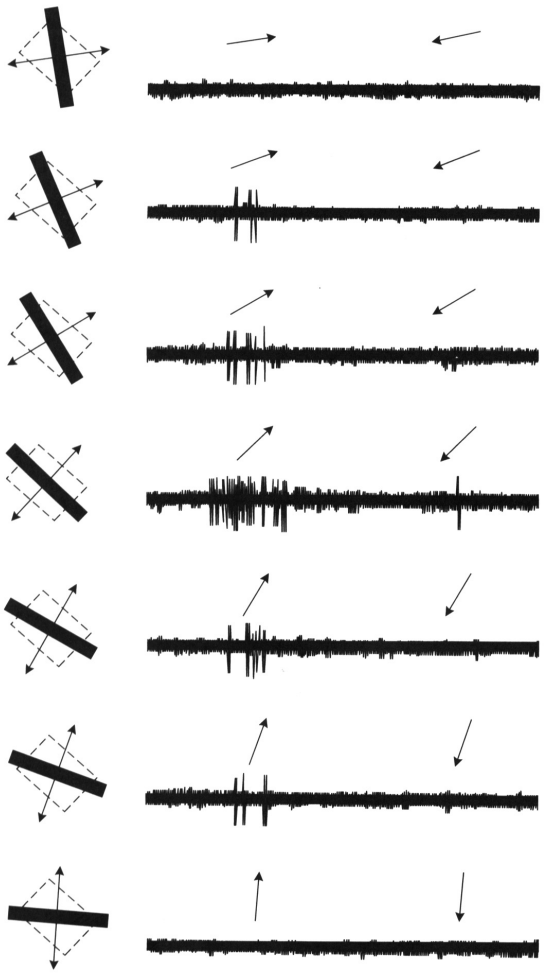

Figure 9.19. Cortical cells give stronger or weaker responses to moving bars, depending on the direction in which the bar moves. The figure shows the response of a cell to a moving bar; note that, as the bar moves forward in the direction perpendicular to its extent, the response is stronger than when the bar moves backward. Similarly, the strength of the response depends on the orientation of the bar. *Figure from Brian Wandell's book, "Foundations of Vision", page171, in the fervent hope that permission will be granted* after Hubel and Wiesel, 1968.

9.5.3 A Model of Early Spatial Vision

We now have a picture of the early stages of primate vision. The retinal image is transformed into a series of retinotopic maps, each of which contains the output of

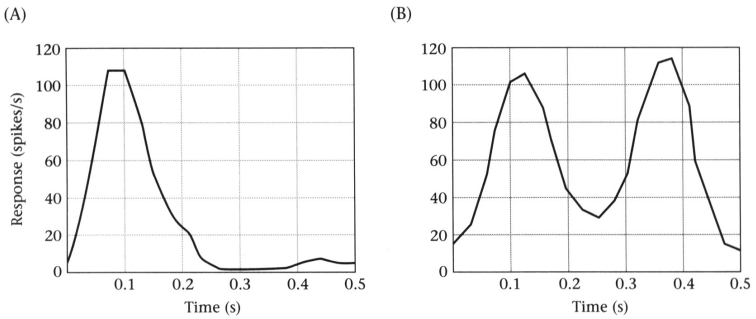

Figure 9.20. Cortical cells are classified simple or complex, depending on their response to a time-reversing grating. In a simple cell (on the left), the response grows as the intensity of the grating grows, and then falls off as the contrast of the grating reverses. The negative response is weak, because the cell's resting output is low so that it cannot code much inhibition. The response is what one would expect from a linear system with a lower threshold on its response. On the right, the response of a complex cell, which looks like full wave rectification; the cell responds similarly to a grating with positive and reversed contrast. *Figure from Brian Wandell's book, "Foundations of Vision", page172, in the fervent hope that permission will be granted* after DeValois et al 82

a linear filter which may have spatial or spatio-temporal support. The retinotopic structure means that each map can be thought of as an image which is filtered version of the retinal image. The filters themselves are oriented filters that look rather a lot like various derivatives of a Gaussian, at various orientations. The retinotopic maps are subjected to some form of non-linearity (to get the output of the complex cells).

This model can be refined somewhat, with the aid of psychophysical studies of adaptation. Adaptation is a term that is used fairly flexibly; generally, the response of an observer to a stimulus declines if the stimulus is maintained and stays depressed for some time afterwards. Adaptation can be used to determine components of a signal that are coded differently — or follow different psychophysical channels, in the jargon — if we adopt the model that channels adapt independently. Observer sensitivity to gratings can be measured by the **contrast sensitivity function**, which codes the contrast that a periodic signal must have with a constant background before it is visible (figure **??**).

In experiments by Blakemore and Campbell [], observers were shown spatial frequency gratings until they are adapted to that spatial frequency. It turns out that the observer's contrast sensitivity is decreased for a range of spatial frequencies around the adapting frequency. This suggests that the observer is sensitive to several spatial frequency channels; the contrast sensitivity function can be seen as a superposition of several contrast sensitivity functions, one for each channel.

This is a **multiresolution model**. The current best model of human early

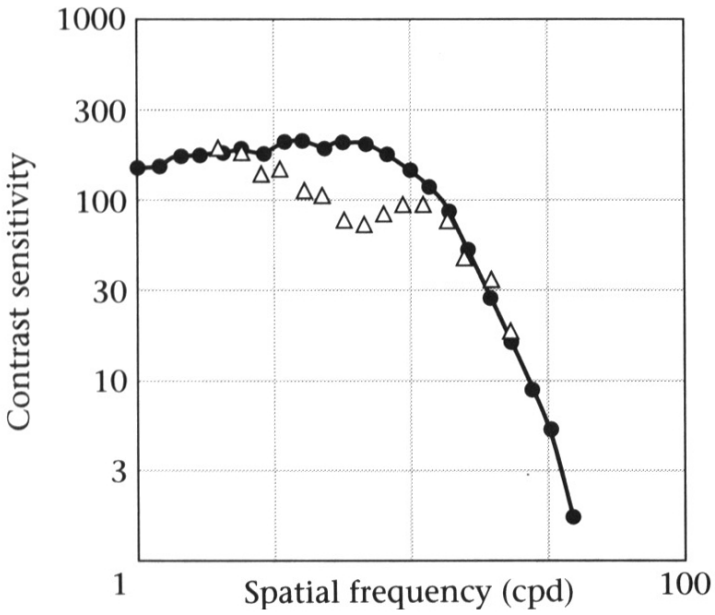

Figure 9.21. The figure shows the effect on contrast sensitivity of viewing a grating for a long period of time (adaptation). The contrast sensitivity shown as a filled curve is measured before the observer adapts; after adaptation, the contrast sensitivity is lowered (open curve). Notice that adaptation affects only a range of spatial frequencies around the adapting frequency. This effect supports a multiresolution theory. *Figure from Brian Wandell's book, "Foundations of Vision", page214, in the fervent hope that permission will be granted*, after Blakemore and Campbell, 1969.

vision is that the visual signal is split into several spatial frequency bands (rather as in section **??**); each band is then subjected to a set of oriented linear filters, and the responses of these filters in turn are subjected to a non-linearity (as in figure 9.22). The response of this model can be used quite successfully to predict various forms of pattern sensitivity for simple patterns (it clearly doesn't explain, say, recognition); we will see it again in discussions of texture.

9.6 Commentary

Edge detection is a subject that is alive with controversy, much of it probably empty. We have hardly scratched the surface. There are many optimality criteria for edge detectors, and rather more "optimal" edge detectors. At the end of the day, most boil down to smoothing the image with something that looks a lot like a Gaussian before measuring the gradient. Our choice of Canny's approach to expound here and in the next chapter is based mainly on tradition.

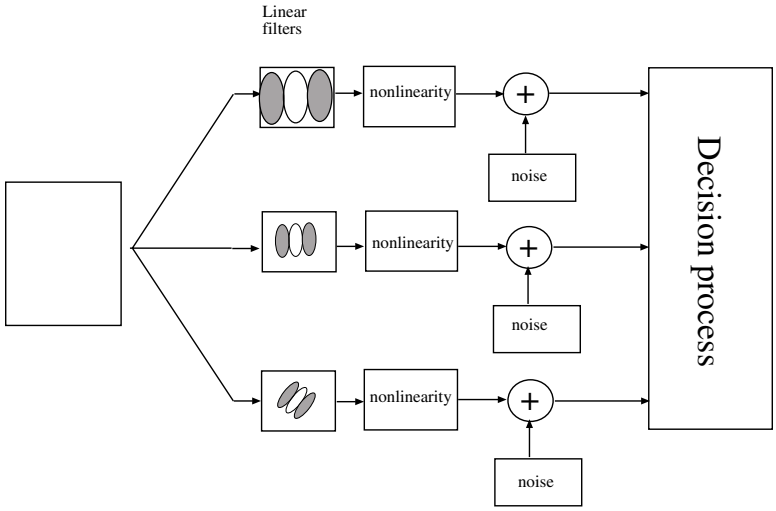

Figure 9.22. An overview of a multiresolution model of human pattern sensitivity. The stimulus is convolved with linear filters at a variety of scales and orientations — we show three scales, and only one orientation per scale; this is *not* a commitment to a number — and then subjected to a nonlinearity. The results have noise added, and are passed to a decision process. The details of the number of scales, the number of orientations, the choice of non-linearity, etc. vary from author to author. This class of model is now quite successful at predicting responses to simple patterns. (after *Figure from Brian Wandell's book, "Foundations of Vision", page222, in the fervent hope that permission will be granted*, who got it from Spillman and Werner)

Object boundaries are not the same as sharp changes in image values. Firstly, objects may not have a strong contrast with their backgrounds through sheer bad luck. Secondly, objects are often covered with texture or markings which generate edges of their own; often so many that it is hard to wade through them to find the relevant pieces of object boundary. Finally, shadows and the like may generate edges that have no relation to object boundaries. There are some strategies for dealing with these difficulties.

Firstly, some applications allow management of illumination; if it is possible to choose the illumination, a careful choice can make a tremendous difference in the contrast and eliminate shadows. Secondly, by setting smoothing parameters large and contrast thresholds high it is often possible to ensure that edges due to texture are smoothed over and not marked. This is a dubious business, firstly because it can be hard to choose reliable values of the smoothing and the thresholds and secondly because it is perverse to regard texture purely as a nuisance, rather than a source of information.

There are other ways to handle the uncomfortable distinction between edges and object boundaries. Firstly, one might work to make better edge detectors. This

approach is the root of a huge literature, dealing with matters like localisation, corner topology and the like. We incline to the view that returns are diminishing rather sharply in this endeavour.

Secondly, one might deny the usefulness of edge detection entirely. This approach is rooted in the observation that some stages of edge detection, particularly non-maximum suppression, discard information that is awfully difficult to retrieve later on. This is because a hard decision — testing against a threshold — has been made. Instead, the argument proceeds, one should keep this information around in a "soft" (a propaganda term for probabilistic) way. Attactive as these arguments sound, we are inclined to discount this view because there are currently no practical mechanisms for handling the volumes of soft information so obtained.

Finally, one might regard this as an issue to be dealt with by overall questions of system architecture — the fatalist view that almost every visual process is going to have obnoxious features, and the correct approach to this problem is to understand the integration of visual information well enough to construct vision systems that are tolerant to this. Although it sweeps a great deal of dust under the carpet — precisely *how* does one construct such architectures? — we find this approach most attractive and will discuss it again and again.

All edge detectors behave badly at corners; only the details vary. This has resulted in two lively strands in the literature (i - what goes wrong; ii - what to do about it). There are a variety of quite sophisticated corner detectors, mainly because corners make quite good point features for correspondence algorithms supporting such activities as stereopsis, reconstruction or structure from motion. This has led to quite detailed practical knowledge of the localisation properties of corner detectors (e.g. []).

A variety of other forms of edge are quite common, however. The most commonly cited example is the **roof edge**, which can result from the effects of interreflections (figure 3.17). Another example that also results from interreflections is a composite of a step and a roof (figure **??**). It is possible to find these phenomena by using essentially the same steps as outlined above (find an "optimal" filter, and do non-maximum suppression on its outputs). In practice, this is seldom done. There appear to be two reasons. Firstly, there is no comfortable basis in theory (or practice) for the models that are adopted — what particular composite edges are worth looking for? the current answer — those for which optimal filters are reasonably easy to derive — is most unsatisfactory. Secondly, the semantics of roof edges and more complex composite edges is even vaguer than that of step edges — there is little notion of what one would *do* with roof edge once it had been found.

Edges are poorly defined and usually hard to detect, but one can solve problems with the output of an edge detector. Roof edges are similarly poorly defined and similarly hard to detect; we have never seen problems solved with the output of a roof edge detector. The real difficulty here is that there seems to be no reliable mechanism for predicting, in advance, what will be worth detecting. We will scratch the surface of this very difficult problem in section **??**.

Assignments

Exercises

1. Show that forming unweighted local averages — which yields an operation of the form
$$\mathcal{R}_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{u=i+k} \sum_{v=j-k}^{v=j+k} \mathcal{F}_{uv}$$
is a convolution. What is the kernel of this convolution?

2. Write $\mathcal{E}_0$ for an image that consists of all zeros, with a single one at the center. Show that convolving this image with the kernel
$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{((i-k-1)^2 + (j-k-1)^2)}{2\sigma^2}\right)$$
(which is a discretised Gaussian) yields a circularly symmetric fuzzy blob.

3. Show that convolving an image with a discrete, separable 2D filter kernel is equivalent to convolving with two 1D filter kernels. Estimate the number of operations saved for an NxN image and a $2k+1 \times 2k+1$ kernel.

4. We said "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" in section 8.6. Why is this true?

5. Each pixel value in 500×500 pixel image $\mathcal{I}$ is an independent normally distributed random variable with zero mean and standard deviation one. Estimate the number of pixels that where the absolute value of the x derivative, estimated by forward differences (i.e. $|I_{i+1,j} - I_{i,j}|$ is greater than 3.

6. Each pixel value in 500×500 pixel image $\mathcal{I}$ is an independent normally distributed random variable with zero mean and standard deviation one. $\mathcal{I}$ is convolved with the $2k+1 \times 2k+1$ kernel $\mathcal{G}$. What is the covariance of pixel values in the result? (There are two ways to do this; on a case-by-case basis — e.g. at points that are greater than $2k+1$ apart in either the x or y direction, the values are clearly independent — or in one fell swoop. Don't worry about the pixel values at the boundary.)

7. For a set with $2k+1$ elements, the median is the $k+1$'th element of the sorted set of values. For a set with $2k$ elements, the median is the average of the k and the $k+1$'th element of the sorted set. Show that it does not matter whether the set is sorted in increasing or decreasing order.

8. Assume that we wish to remove salt-and-pepper noise from a uniform background. Show that up to half of the elements in the neighbourhood could be noise values and a median filter would still give the same (correct) answer.

Programming Assignments

- One way to obtain a Gaussian kernel is to convolve a constant kernel with itself, many times. Compare this strategy with evaluating a Gaussian kernel.
    1. How many repeated convolutions will you need to get a reasonable approximation? (you will need to establish what a reasonable approximation is; you might plot the quality of the approximation against the number of repeated convolutions).
    2. Are there any benefits that can be obtained like this? (hint: not every computer comes with an FPU)
- Why is it necessary to check that the gradient magnitude is large at zero crossings of the Laplacian of an image? Demonstrate a series of edges for which this test is significant.
- The Laplacian of a Gaussian looks similar to the difference between two Gaussians at different scales. Compare these two kernels for various values of the two scales — which choices give a good approximation? How significant is the approximation error in edge finding using a zero-crossing approach?
- Obtain an implementation of Canny's edge detector (you could try the vision home page at `http://www.somewhereorother`) and make a series of images indicating the effects of scale and contrast thresholds on the edges that are detected. How easy is it to set up the edge detector to mark only object boundaries? Can you think of applications where this would be easy?
- It is quite easy to defeat hysteresis in edge detectors that implement it — essentially, one sets the lower and higher thresholds to have the same value. Use this trick to compare the behaviour of an edge detector with and without hysteresis. There are a variety of issues to look at:
    1. What are you trying to do with the edge detector output? it is sometimes very helpful to have linked chains of edge points — does hysteresis help significantly here?
    2. Noise suppression: we often wish to force edge detectors to ignore some edge points and mark others. One diagnostic that an edge is useful is high contrast (it is by no means reliable). How reliably can you use hysteresis to suppress low contrast edges without breaking high contrast edges?
- Build a normalised correlation matcher. The hand finding application is a nice one, but another may occur to you.
    1. How reliable is it?
    2. How many different patterns can you tell apart in practice?
    3. How sensitive is it to illumination variations? shadows? occlusion?

Chapter 10

NON-LINEAR FILTERS

10.1 Filters as Templates

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

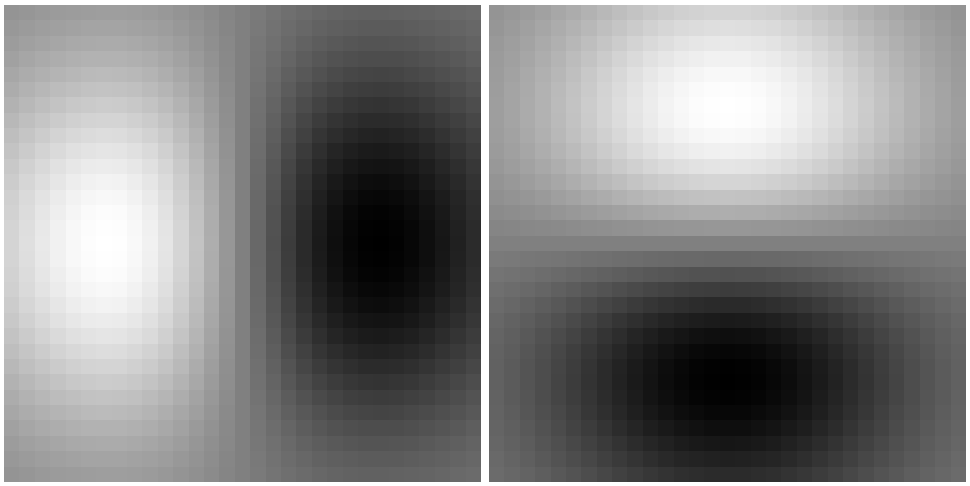

Figure 10.1. Filter kernels "look like" the effects they are intended to detect. On the left, an smoothed derivative of Gaussian filter that looks for large changes in the x-direction (such as a dark blob next to a light blob); on the right, a smoothed derivative of Gaussian filter that looks for large changes in the y-direction.

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