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 and is given by

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

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 from the derivative estimation filter).

Canny (1986) 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.
- **Localization**—the filter response should reach a maximum close to $x = 0$.
- **Low false positives**—there should be only one maximum of the response in a reasonable neighborhood of $x = 0$.

Once a continuous filter has been found, it is discretized. 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 derived by most combinations of these criteria tend to look a great deal like Gaussians—this is intuitively reasonable because 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.

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 means that the filter responds on each side of the bar, and we are able to resolve the rising and falling edges of the bar. If the filter width is much greater, the bar is smoothed into the background and the bar generates little or no response (Figure 8.5).

8.2.4 Why Smooth with a Gaussian?

Although a Gaussian is not the only possible blurring kernel, it is convenient because it has a number of important properties. First, if we convolve a Gaussian with a Gaussian, 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 heavily smoothed images by resmoothing smoothed images. This is a significant property because discrete convolution can be an expensive operation (particularly if the kernel of the filter is large), and it is common to want versions of an image smoothed by different amounts.

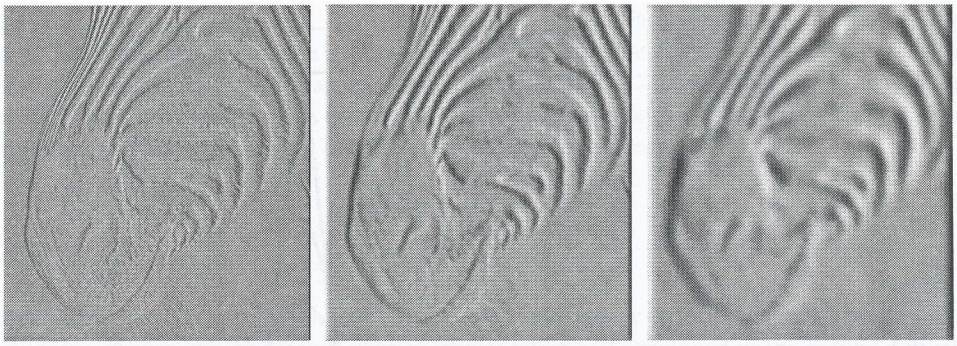

Figure 8.5 The scale (i.e., σ) of the Gaussian used in a derivative of Gaussian filter has significant effects on the results. The three images show estimates of the derivative in the x direction of an image of the head of a zebra obtained using a derivative of Gaussian filter with σ one pixel, three pixels, and seven pixels (moving to the right). Note how images at a finer scale show some hair, the animal's whiskers disappear at a medium scale, and the fine stripes at the top of the muzzle disappear at the coarser scale.

Efficiency Consider convolving an image with a Gaussian kernel with σ one pixel. Although the Gaussian kernel is nonzero 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 5×5 integer grid centered at the origin have values less than $e^{-4} = 0.0184$, and points outside a 7×7 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 (5×5 or 7×7 according to taste and the number of bits you allocate to represent the kernel).

However, if σ is 10 pixels, we may need a 50×50 array or worse. A back of the envelope count of operations should convince you that convolving a reasonably sized image with a 50×50 array is an unattractive 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 neighbors' values). As a result, some pixels can be discarded. We then have a quite efficient strategy: smooth, subsample, smooth, subsample, and so on. The result is an image that has the same information as a heavily smoothed image, but is much smaller and easier to obtain. We explore the details of this approach in Section 7.7.1.

The Central Limit Theorem Gaussians have another significant property that we do not prove but illustrate in Figure 8.6. For an important family of functions, convolving any member of that family of functions with itself repeatedly eventually yields a Gaussian. With the associativity of convolution, this implies that if we choose a different smoothing kernel and apply it repeatedly to the image, the result eventually looks like we smoothed the image with a Gaussian.

Gaussians are Separable Finally, an isotropic Gaussian can be factored as

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

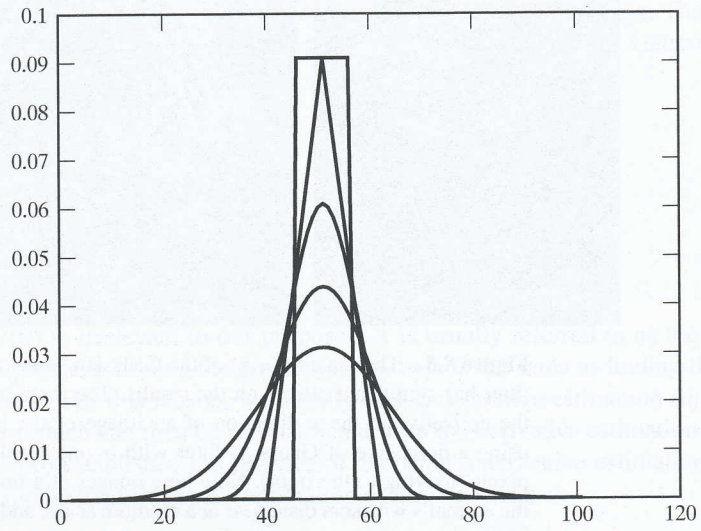

Figure 8.6 The central limit theorem states that repeated convolution of a positive kernel with itself eventually limits toward a kernel that is a scaling of a Gaussian. The graph illustrates this effect for 1D convolution; the triangle is obtained by convolving a box function with itself; each succeeding stage is obtained by convolving the previous stage with itself.

and this is a product of two 1D 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 useful property indeed. In particular, convolving with a filter kernel that is separable is the same as convolving with two 1D 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 discretized separable filter kernel, 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 because the rank of the matrix $\mathcal{H}$ must be one. Commercial convolution packages often test the kernel to see whether 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. Many kernels can be approximated in a useful way as a sum of separable kernels. If the number of kernels is sufficiently small, then the approximation can represent a practical saving in convolution. This is a particularly attractive strategy if one wishes to convolve an image with many different filters; in this case, one tries to obtain a representation of each of these filters as a weighted sum of separable kernels, which are tensor products of a small number of basis elements. It is then possible to convolve the images with the basis elements and then form different weighted sums of the result to obtain convolutions of the image with different filters.

Aliasing in Subsampled Gaussians The discussion of aliasing gives us some insight into available smoothing parameters. Any Gaussian kernel that we use is 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 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 is badly aliased; see the exercises).

8.3 DETECTING EDGES

The two main strategies for detecting edges both model edges as fast changes in brightness. In the first, we observe that the fastest change occurs when a 2D analogue of the second derivative vanishes (Section 8.3.1). Although this approach is historically important, it is no longer popular. The alternative is to explicitly search for points where the magnitude of the gradient is extremal (Section 8.3.2).

8.3.1 Using the Laplacian to Detect Edges

In one dimension, the second derivative of a signal is zero when the derivative magnitude is extremal. This means that, if we wish to find large changes, a good place to look is where the second derivative is zero. This approach extends to two dimensions. We now need a sensible analogue to the second derivative. This needs to 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 is natural to smooth the image before applying a Laplacian. Notice that the Laplacian is a linear operator (if you're not sure about this, you should check), meaning that we could represent taking the Laplacian as convolving the image with some kernel (which we write as K_{∇^2}. Because convolution is associative, we have that

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

The reason this is important is that, just as for first derivatives, smoothing an image and then applying the Laplacian is the same as convolving the image with the Laplacian of the kernel used for smoothing. Figure 8.7 shows the resulting kernel.

This leads to a simple and historically important edge detection strategy illustrated in Figure 8.8. 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. The method is due to Marr and Hildreth (1980).

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 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. This is roughly the same as filtering the image with a difference of Gaussians, multiplying the result by a small constant, and adding this back to the original image. Now the difference between two Gaussian kernels looks similar to a Laplacian of Gaussian kernel, and it is quite common to replace one with the other. This means that unsharp masking adds an edge term back to the image.