8

Edge Detection

Sharp changes in image brightness are interesting for many reasons. First, object boundaries often generate sharp changes in brightness—a light object may lie on a dark background or a dark object may lie on a light background. Second, reflectance changes often generate sharp changes in brightness, which can have quite distinctive patterns—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 interesting things often happen at an edge in an image and it is worth knowing where the edges are.

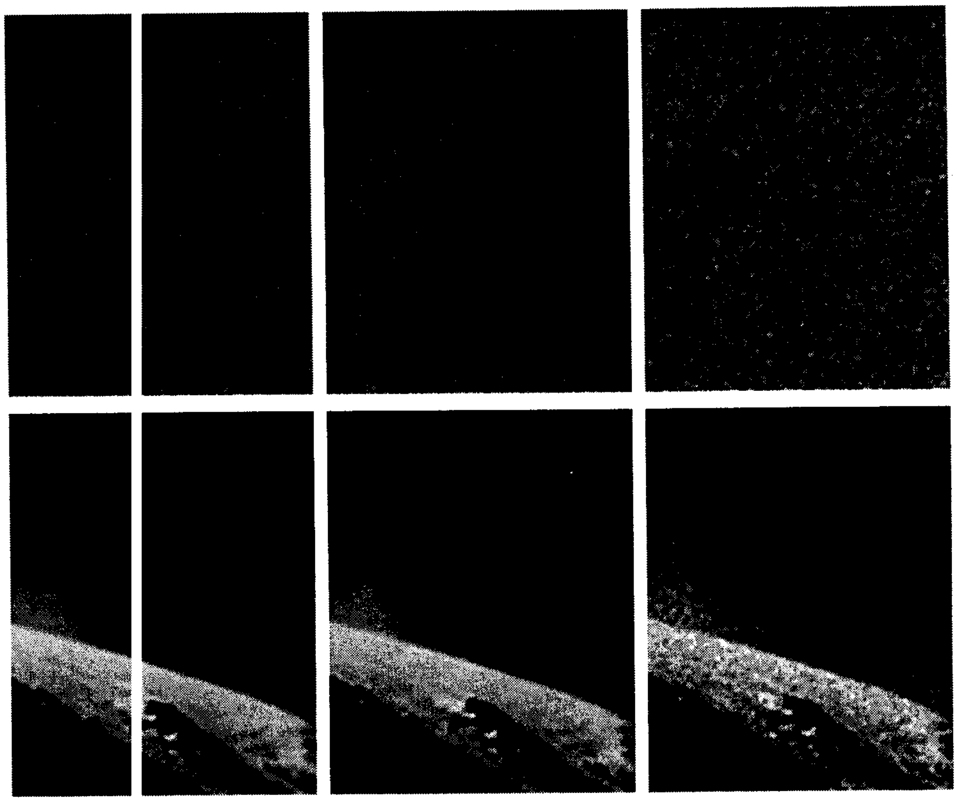
8.1 NOISE

A primary problem in edge detection is image noise. This is because edge detectors are constructed to respond strongly to sharp changes; but one way to get sharp changes in an image is to add noise to the pixels (because the noise values at each pixel are typically uncorrelated, meaning they can be very different). As Section 7.3 indicated, noise makes finite difference estimates of image derivatives unusable. We use this observation as an impetus to study image noise in general.

The term , *oise* usually means image measurements from which we do not know how to extract informat on or from which we do not care to extract information; all the rest is *signal*. It is wrong to b lieve that noise does not contain information—for example, we should be able to extract some stimate of the camera temperature by taking pictures in a dark room with the lens cap on. Fur hermore, since we cannot say anything meaningful about noise without a noise model, it is wro g to say that noise is not modeled. Noise is everything we don't wish to use, and that's all there i to it.

8.1.1 Additiv Stationary Gaussian Noise

In the *additive . tationary Gaussian noise* model, each pixel has added to it a value chosen independently fr n the same Gaussian probability distribution. Almost always the mean of this distribution is z ro. The standard deviation is a parameter of the model. The model is intended to describe ther ial noise in cameras. It is illustrated in Figure 8.1.



Fi ire 8.1 The top row shows three realizations of a stationary additive Gaussia noise process. We have added half the range of brightnesses to these images to iow 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, res ectively. 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 n image. In each case, values below zero or above the full range have been adj isted to zero or the maximum value accordingly.

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 do 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 $n_{0,0}, \dots, n_{r,s}$. The expected value of this response must be

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

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 and identically distributed Gaussian random variables with mean μ, we have

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

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

$$\mathrm{E}[\{R(\mathcal{N})_{i,j} - \mathrm{E}[R(\mathcal{N})_{i,j}]\}^2],$$

and this is the same as

$$\int \{R(\mathcal{N})_{i,j} - \mathrm{E}[R(\mathcal{N})_{i,j}]\}^2 p(N_{0,0}, \dots, N_{r,s}) dN_{0,0} \dots dN_{r,s},$$

which expands to

$$\int \left\{ \sum_{u} \sum_{v} G_{-u,-v}(N_{u,v} - \mu) \right\}^2 p(N_{0,0}, \dots, N_{r,s}) dN_{0,0} \dots dN_{r,s},$$

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

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

(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

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

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

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

Difficulties with the Additive Stationary Gaussian Noise Model Taken literally, the additive s ationary Gaussian noise model is a poor model of image noise. First, the model allows posi ve (and, more alarmingly, *negative!*) pixel values of arbitrary magnitude. With appropriate c oices of standard deviation for typical current cameras operating indoors or in daylight, this d esn't present much of a problem because these pixel values are extremely unlikely to occur i practice. In rendering noise images, the problematic pixels that do occur are fixed at zero or ful output, respectively.

Second, nois values are completely independent, so this model does not capture the possibility of groups f 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, beca se 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 re consistently saturated) terribly well. If the standard deviation is quite large and we threshold ixel 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 estim te the response of filters to this noise model. In turn, this gives us some idea of how effective the ilter is at responding to signal and ignoring noise.

8.1.2 Why Fini e Differences Respond to Noise

Our discussion o the response of linear filters to additive stationary Gaussian noise offers some insight into the oise behavior of finite differences. Assume we have an image of stationary Gaussian noise f zero mean, and consider the variance of the response to a finite difference filter that estima s derivatives of increasing order. We use the kernel

$$\begin{array}{cc} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{array}$$

to estimate the rst derivative. Now a second derivative is simply a first derivative applied to a first derivative, o the kernel is

$$\begin{array}{ccc} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{array}.$$

With a li tle thought, you can convince yourself that, under this scheme, the kernel coefficients of a th derivative come from the $k + 1$th row of Pascal's triangle with appropriate flips of sign. F r each of these derivative filters, the mean response to Gaussian noise is zero, but the varianc of this response goes up sharply; for the kth derivative, it is the sum of squares of the $k + 1$th row of Pascal's triangle times the standard deviation. Figure 8.2 illustrates this result.

There is n alternative explanation. From Table 7.1, differentiating a function is the same as multiplying its Fourier transform by a frequency variable; this means that the high spatial frequency com onents are heavily emphasized at the expense of the low-frequency components. This is intuitiv ly plausible—differentiating a function must set the constant component to zero, and the ampli de 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 c anges (which generate high spatial frequencies) have large derivatives.

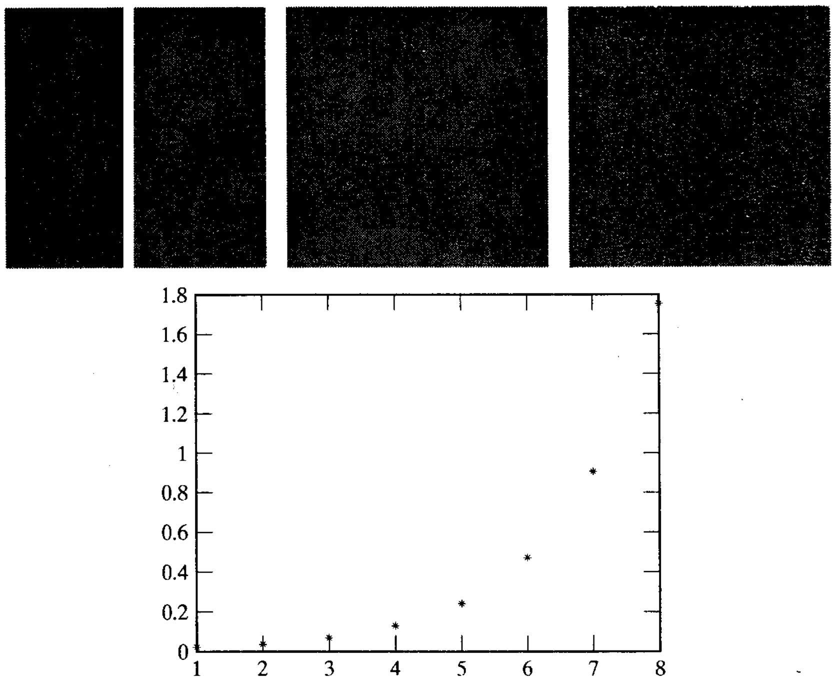


Figure 8 2 Finite differences can accentuate additive Gaussian noise substantially foll)wing the argument in Section 8.1.2. On the **top left** an image of zero mean Ga ıssian noise with standard deviation 4/256 of the full range. The **top center** fi ure shows a finite difference estimate of the third derivative in the x direction and the **top right** shows the sixth derivative in the x direction. In each case, the mage has been centered by adding half the full range to show both positive and ıegative deviations. The images are shown using the same gray-level scale; in he case of the sixth derivative, some values exceed the range of this scale. Th graph on the **bottom** shows the standard deviations of these noise images for he first eight derivatives (estimated using the argument based around Pascal's ı ·iangle).

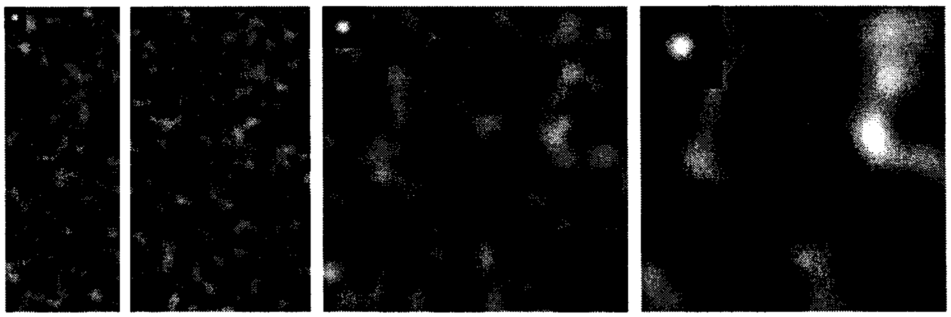
8.2 ESTIMATING DERIVATIVES

As Figure 7.4 indic ıtes, simple finite difference filters tend to give strong responses to noise so that applying tw ı finite difference filters (one in each direction) is a poor way to estimate a gradient. The wa) to deal with this problem is to smooth the image and then differentiate it (we could also smo ıth the derivative). In practice, the image is almost always smoothed with a Gaussian filter—i ı fact, the finite difference operator is smoothed. We discuss this practice first, and then for th ıse who want more information, we discuss why smoothing helps and why a Gaussian is a gooc choice of smoothing filter.

8.2.1 Derivative (f Gaussian Filters

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

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



Fi ure 8.3 Smoothing stationary additive Gaussian noise results in signals wh re pixel values tend to be increasingly similar to the value of neighboring pi els. This occurs at about the scale of the filter kernel because the filter ker- ne causes the correlations. The figures show noise smoothed with increasingly lar e Gaussian smoothing kernels. Gray pixels have zero value, darker values ar negative, and brighter values are positive. The kernels are shown in the top rig t-hand corners of the figures to indicate the spatial scale of the kernel (we ha e scaled the brightness of the kernels, which are Gaussians, so that the center pi el is white and the boundary pixels are black). Smoothed noise tends to look lik natural texture as the figures indicate.

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

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

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

This fact appe rs in its most commonly used form when the smoothing function is a Gaussian; we can then w ite

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

that is, we nee l only convolve with the derivative of the Gaussian, rather than convolve and then differentiate. moothing results in much smaller noise responses from the derivative estimates (Figure 8.4).

8.2.2 Why moothing Helps

In general, an change of significance to us has effects over a pool of pixels. For example, the contour of an bject can result in a long chain of points where the image derivative is large. For many kinds o noise model, large image derivatives due to noise are an essentially local event. This means th t smoothing a differentiated image tends to pool support for the changes we are interested in nd to suppress the effects of noise. An alternative interpretation of the point is that the chang s we are interested in will not be suppressed by some smoothing, which tends to suppress the e fects of noise.

There i an alternative explanation as to why smoothing may help. Assume we smooth a noisy image nd then differentiate it. First, the variance of the noise tends to be reduced by a

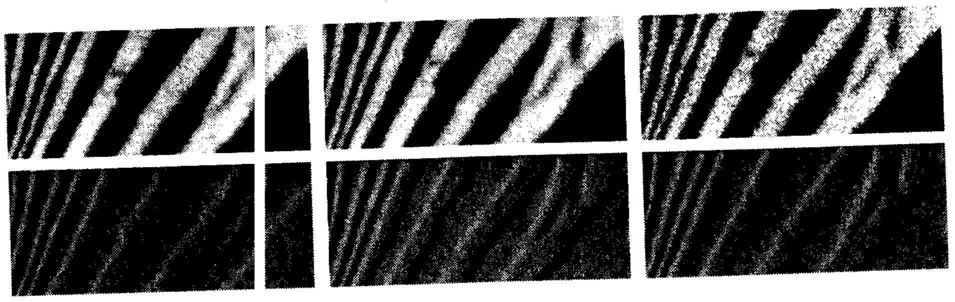


Figure 8.4 De ivative of Gaussian filters are less extroverted in their response to noise than fir te difference filters. The image at **top left** shows a detail from a picture of a ze)ra; **top center** shows the same image corrupted by zero mean stationary addit /e Gaussian noise, with $\sigma = 0.03$ (pixel values range from 0 to 1). **Top right** sl)ws the same image corrupted by zero mean stationary additive Gaussian noise, with $\sigma = 0.09$. The second row shows the partial derivative in the x-direction f each image, in each case estimated by a derivative of Gaussian filter with σ on pixel. Notice how the smoothing helps reduce the impact of the noise.

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

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

Second, pixels have a gı :ater tendency to look like neighboring pixels—if we take stationary additive Gaussian noise a ıd smooth it, the pixel values of the resulting signal *are no longer independent*. In some sense, t ıis is what smoothing was about— recall that we introduced smoothing as a method to predict a ı ixel's value from the values of its neighbors. However, if pixels tend to look like their neighbors then derivatives must be smaller (because they measure the tendency of pixels to look differen from their neighbors).

Another approach ıs to reason in terms of spatial frequencies. It is possible to show that stationary additive Gaus ian noise has uniform energy at each frequency. If we differentiate the noise, we 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 suppress s these high spatial frequencies as it does for resampling (Section 7.4.3).

Smoothed noise h ıs applications. As Figure 8.3 indicates, smoothed noise tends to look like some kinds of natu al texture, and smoothed noise is widely used as a source of textures in computer graphics a plications (Ebert, Musgrave, Peachey, Worley and Perlin, 1998, Perlin, 1985).

8.2.3 Choosing a Sr ıoothing Filter

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

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.

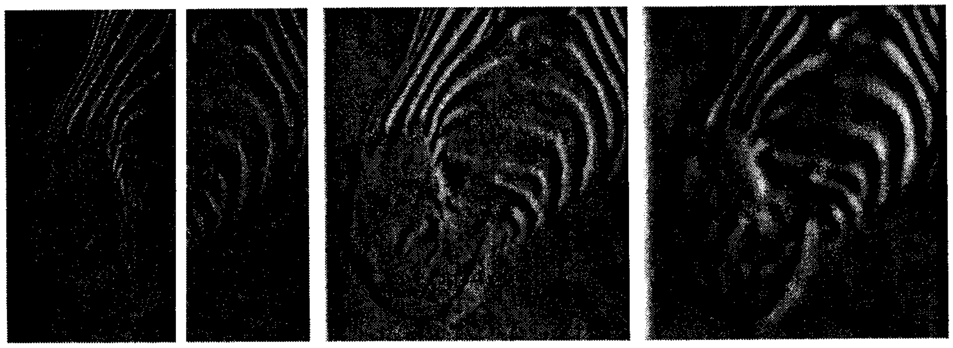


Figure ŧ 5 The scale (i.e., σ) of the Gaussian used in a derivative of Gaussian filter ha significant effects on the results. The three images show estimates of the deri ative in the x direction of an image of the head of a zebra obtained using a erivative of Gaussian filter with σ one pixel, three pixels, and seven pixels (r oving to the right). Note how images at a finer scale show some hair, the anim l's whiskers disappear at a medium scale, and the fine stripes at the top of the m zzle disappear at the coarser scale.

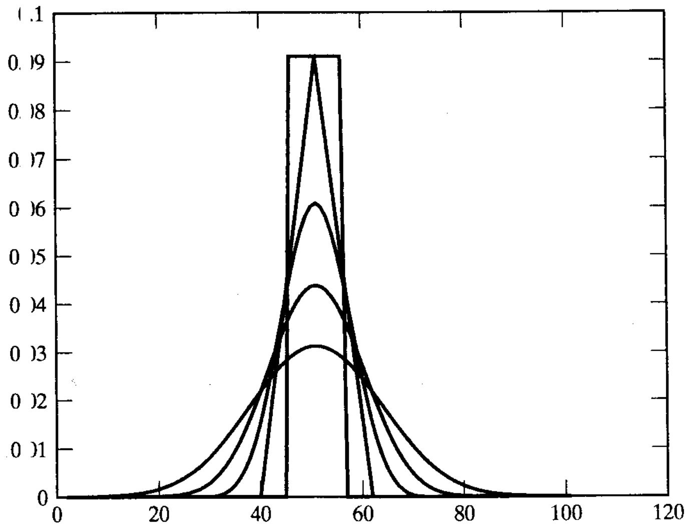
Efficiency Consider convolving an image with a Gaussian kernel with σ one pixel. Although the Gaus ian kernel is nonzero over an infinite domain, for most of that domain it is extremely small be ause of the exponential form. For σ one pixel, points outside a 5×5 integer grid centered at the origin have 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 nd represent the discrete Gaussian as a small array (5×5 or 7×7 according to taste and the nun ber 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 unattrac ive prospect. The alternative—convolving repeatedly with a much smaller kernel— is much n ore efficient *because we don't need to keep every pixel in the interim.* This is because a smoo ed 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 strate y: smooth, subsample, smooth, subsample, and so on. The result is an image that has the same i formation as a heavily smoothed image, but is much smaller and easier to obtain. We explore he details of this approach in Section 7.7.1.

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

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

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



Figur : 8.6 The central limit theorem states that repeated convolution of a positive l ernel with itself eventually limits toward a kernel that is a scaling of a Gaus: an. The graph illustrates this effect for 1D convolution; the triangle is obtainec by convolving a box function with itself; each succeeding stage is obtained by co volving the previous stage with itself.

and this is a prod' ct of two 1D Gaussians. Generally, a function $f(x, y)$ that factors as $f(x, y) = g(x)h(y)$ is refer ed to as a *tensor product*. It is common to refer to filter kernels that are tensor products as *sepa. able kernels*. Separability is a useful property indeed. In particular, convolving with a filter kern l that is separable is the same as convolving with two 1D kernels—one in the x direction and anc ther in the y direction (exercises).

Many othe kernels are separable. Separable filter kernels result in discrete representations that factor as wel . In particular, if $\mathcal{H}$ is a discretized separable filter kernel, there are some vectors f and g such tha

$$H_{ij} = f_i g_j.$$

It is possible to dentify this property using techniques from numerical linear algebra because the rank of the n atrix $\mathcal{H}$ 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 ı sum of separable kernels. If the number of kernels is sufficiently small, then the approximatic n can represent a practical saving in convolution. This is a particularly attractive strategy if one w ishes to convolve an image with many different filters; in this case, one tries to obtain a represe tation of each of these filters as a weighted sum of separable kernels, which are tensor products f a small number of basis elements. It is then possible to convolve the images with the basis ele ments and then form different weighted sums of the result to obtain convolutions of the image wit ı different filters.

Aliasing in Subsampled Gaussians The discussion of aliasing gives us some insight into availal le smoothing parameters. Any Gaussian kernel that we use is a sampled approximation to a Ga ssian sampled on a single pixel grid. This means that, for the original kernel to be reconstructe 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

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

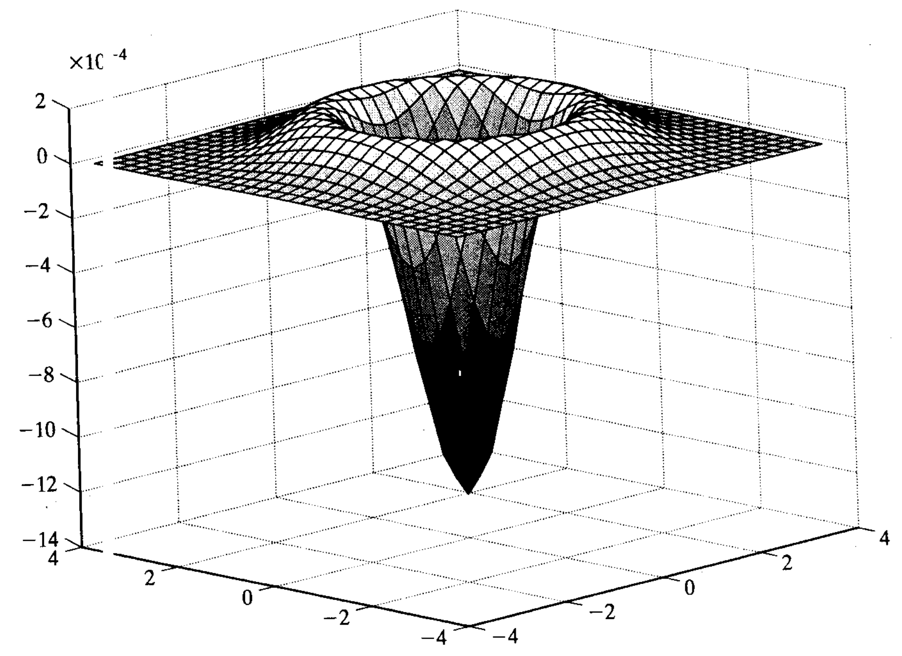
It is natural to smooth the image before applying a Laplacian. Notice that the Laplacian is a linear operator (if you're not sure about this, you should check), meaning that we could represent taking the Laplacian as convolving the image with some kernel (which we write as K_{∇^2}. Because convolution 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.



Fi ure 8.7 The Laplacian of Gaussian filter kernel, shown here for σ one pixel, ca be thought of as subtracting the center pixel from a weighted average of the su round (hence the analogy with unsharp masking described in the text). It is qu te common to replace this kernel with the difference of two Gaussians—one wi h a small value of σ and the other with a large value of σ.

Laplaci n of Gaussian edge detectors have fallen into some disfavor. Because the Laplacian of Gaussian fi ter is not oriented, its response is composed of an average across an edge and one along the ed . This means that the behavior at corners—where the direction along the edge changes—is oor. They mark the boundaries of sharp corners quite inaccurately. Furthermore, at trihedral or g ater vertices, they have difficulty recording the topology of the corner correctly, as Figure 8.9 ill strates. Second, the components along the edge tend to contribute to the response of the filter t noise but not necessarily to an edge; this means that zero crossings may not lie exactly on an edge.

8.3.2 Grad ent-Based Edge Detectors

In a gradient ased edge detector, we compute some estimate of the gradient magnitude—almost always usin a Gaussian as a smoothing filter—and use this estimate to determine the position of edge poir s. Typically, the gradient magnitude can be large along a thick trail in an image (Figure 8.10 . Object outlines are curves, however, and we should like to obtain a curve of the most distinc ve points on this trail.

A natu al 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 estin ated using the direction of the gradient (Figure 8.11). These considerations yield Algorithm 8 1. Most current edgefinders follow these lines, but there remain substantial debates about the pr per execution of the details.

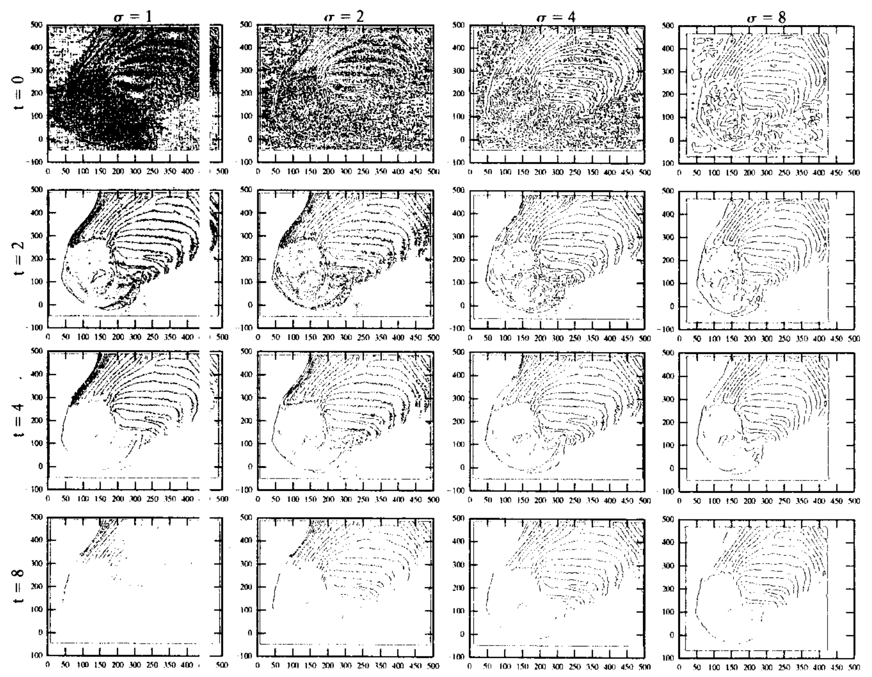
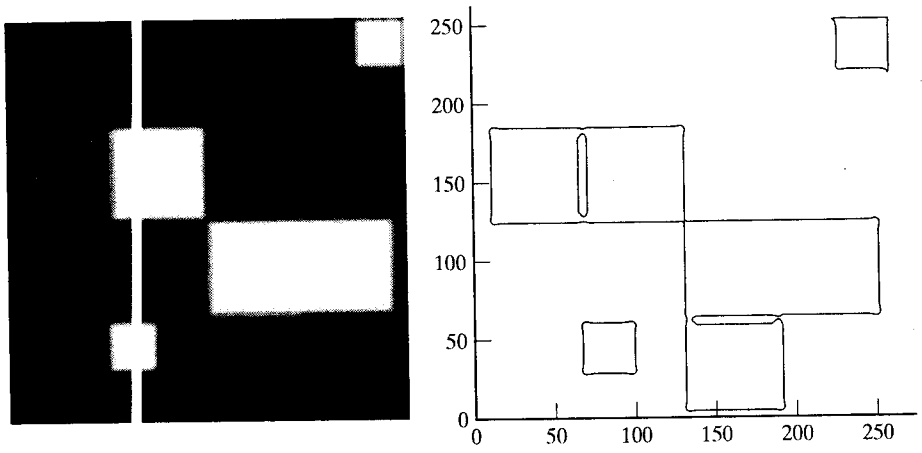


Figure 8.8 Zε o crossings of the Laplacian of Gaussian for various scales and at various gradi nt magnitude thresholds. Each column shows a fixed scale, with t, the threshold n gradient magnitude increasing as one moves down (by a factor of two from im ge to image). Each row shows a fixed t, with scale increasing from σ one pix l to σ eight pixels by factors of two. Notice that the fine-scale, low-threshold e ges contain a quantity of detailed information that may or may not be useful (d pending on one's interest in the hairs on the zebra's nose). As the scale increases, he detail is suppressed; as the threshold increases, small regions of edge drop ou . No scale or threshold gives the outline of the zebra's head; all respond to its s ipes, although as the scale increases, the narrow stripes on the top of the muzz are no longer resolved.

Algorithm 8.1: Gradier -Based Edge Detection

```
Form an estimate of the image gradient
Obtain the gradient ma nitude from this estimate
Identify image points w here the value
   of the gradient magn tude is maximal
   in the direction perpe ndicular to the edge
   and also large; these oints are edge points
```

Nonmaximum Su pression Given estimates of gradient magnitude, we would like to obtain edge points. Aga n, there is clearly no objective definition, and we proceed by reasonable intuition. The gradier magnitude can be thought of as a chain of low hills. Marking local extrema would mark isola ed points—the hilltops in the analogy. A better criterion is to slice



Figu e 8.9 Zero crossings of Laplacian of Gaussian output can behave stran ely at corners. First, at a right angled corner, the zero crossing bulges out at th corner (but passes through the vertex). This effect is not due to digitiza-tion r quantization, but can be shown to occur in the continuous case as well. At cc ners where three or more edges meet, contours behave strangely, with the detai s depending on the structure of the contour marking algorithm—this algo-rithn (the one shipped with Matlab) produces curious loops. This effect can be miti ated with careful design of the contour marking process, which needs to inco porate a fairly detailed vertex model.

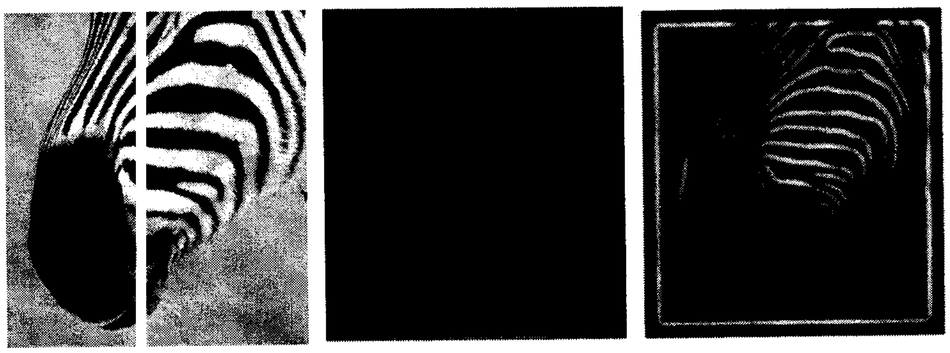


Fig re 8.10 The gradient magnitude can be estimated by smoothing an image and hen differentiating it. This is equivalent to convolving with the derivative of a sn oothing kernel. The extent of the smoothing affects the gradient magnitude; in th s figure, we show the gradient magnitude for the figure of a zebra at different scal s. At the **center**, gradient magnitude estimated using the derivatives of a Gau sian with $\sigma = 1$ pixel; and on the **right** gradient magnitude estimated using the erivatives of a Gaussian with $\sigma = 2$ pixel. Notice that large values of the grac ent magnitude form thick trails.

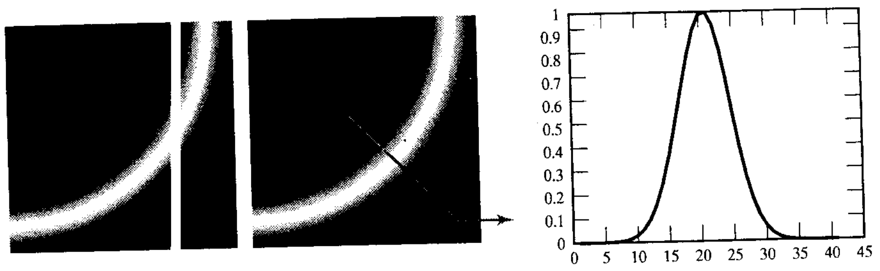


Figure 8.1] The gradient magnitude tends to be large along thick trails in an image. Typ :ally, we would like to condense these trails into curves of representative edge ›oints. A natural way to do this is to cut the trail perpendicular to its direction ar] look for a peak. We use the gradient direction as an estimate of the direction ir which to cut. The figure on the **left** shows a trail of large gradient magnitude; the figure at the **center** shows an appropriate cutting direction; the figure on tł : **right** shows the peak in this direction.

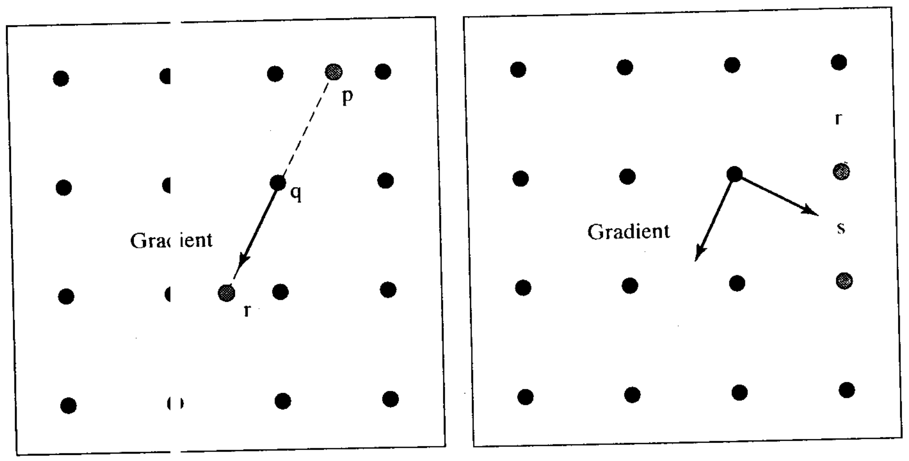


Figure 8. 2 Nonmaximum suppression obtains points where the gradient magnitude is ε a maximum *along the direction of the gradient*. The figure on the left shows ho / we reconstruct the gradient magnitude. The dots are the pixel grid. We are at ›ixel q, attempting to determine whether the gradient is at a maximum; the gradiε ıt direction through q does not pass through any convenient pixels in the forwa d or backward direction, so we must interpolate to obtain the values of the gra lient magnitude at p and r; if the value at q is larger than both, q is an edge poir . Typically, the magnitude values are reconstructed with a linear interpolate, w ıich in this case would use the pixels to the left and right of p and r, respectivı ly, to interpolate values at those points. On the right, we sketch how to find cand ‍dates for the next edge point given that q is an edge point; an appropriate searcl direction is perpendicular to the gradient, so that points s and t should be consiı ered for the next edge point. Notice that, in principle, we don't need to restric ourselves to pixel points on the image grid because we know where the predi ted position lies between s and t. Hence, we could again interpolate to obtain gr dient values for points off the grid.

the gradient mag itude along the gradient direction, which should be perpendicular to the edge, and mark the poi its along the slice where the magnitude is maximal. This would get a chain of points along the rown of the hills in our chain; the process is called *nonmaximum suppression* (Figure 8.12).

Edge Fol owing Typically, we expect edge points to occur along curve like chains. The following ai : the significant steps in nonmaximum suppression:

- determi ing whether a given point is an edge point;
- and, if i is, finding the next edge point.

Once these step are understood, it is easy to enumerate all edge chains. We find the first edge point, mark it, e :pand all chains through that point exhaustively, marking all points along those chains, and con nue to do this for all unmarked edge points.

Algorithm ٤ .2: Nonmaximum Suppression

```
While there ire points with high gradient
that have no been visited
  Find a sta t point that is a local maximum in the
    directi( n perpendicular to the gradient
    erasing points that have been checked
  While po sible, expand a chain through
    the cui ent point by:
      1) p edicting a set of next points, using
         tl : direction perpendicular to the gradient
      2) fi ıding which (if any) is a local maximum
         iı the gradient direction
      3) t sting if the gradient magnitude at the
         n aximum is sufficiently large
      4) l aving a record that the point and
         r :ighbors have been visited
      rec rd the next point, which becomes the current point
  end
end
```

The tw main steps are simple. For the moment, assume that edges are to be marked at pixel locatioı ; (rather than, say, at some finer subdivision of the pixel grid). We can determine whether the ; radient magnitude is maximal at any pixel by comparing it with values at points some way ba kward and forward *along the gradient direction* (Figure 8.11). This is a function of distance alon ; the gradient; typically we step forward to the next row (or column) of pixels and backward to ıe previous to determine whether the magnitude at our pixel is larger (Figure 8.12). The gradient lirection does not usually pass through the next pixel, so we must interpolate to determine the v ılue of the gradient magnitude at the points we are interested in; a linear interpolate is usual.

If the ɪ xel 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. In general, this step does not end on a pixel; a natural

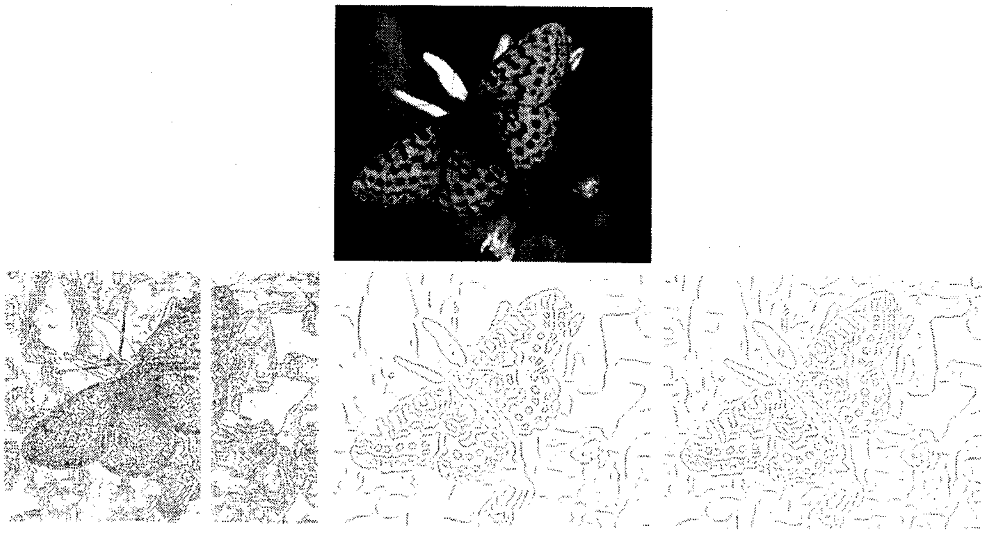


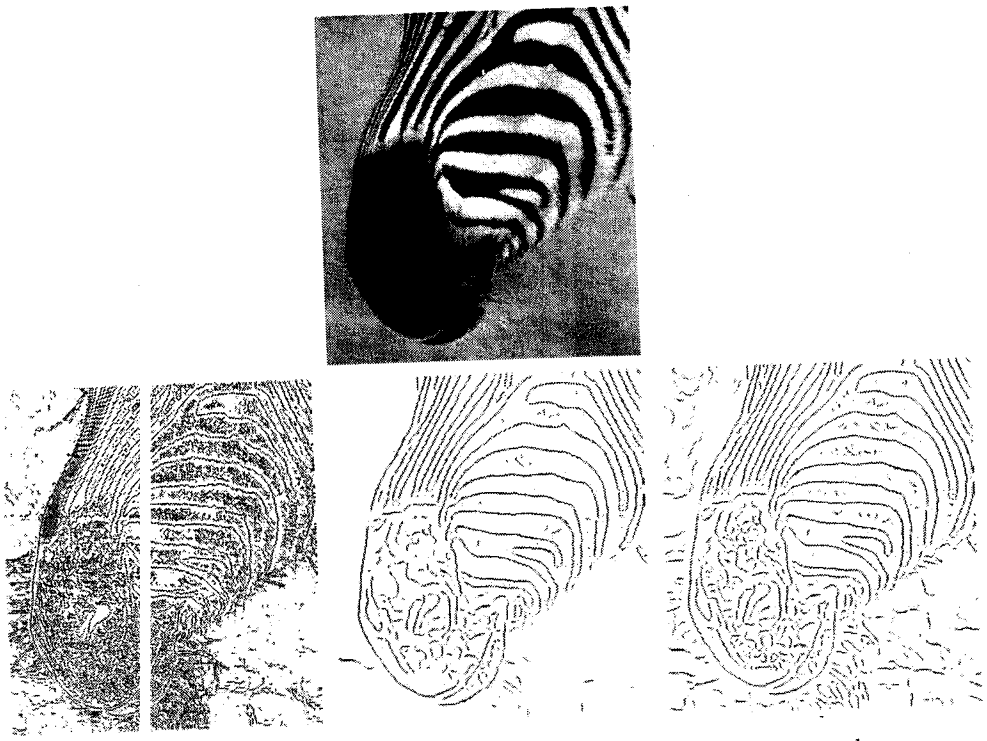
Figure 8.13 Edge points marked on the pixel grid for the image shown on the **top**. The edge points on the **left** are obtained using a Gaussian smoothing filter at σ one pixel, and gradient magnitude has been tested against a high threshold to determine whether a point is an edge point. The edge points on the **center** are obtained using a Gaussian smoothing filter at σ four pixels, and gradient-magnitude has been tested against a high threshold to determine whether a point is an edge point. The edge points on the **right** are obtained using a Gaussian smoothing filter at σ four pixels, and gradient magnitude has been tested against a low threshold to determine whether a point is an edge point. At a fine scale, fine detail at high contrast generates edge points, which disappear at the coarser scale. 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.

strategy is to look at the neighboring pixels that lie close to that direction (see Figure 8.12). This approach leads to a set of curves that can be represented by rendering them in black on a white background, as in Figures 8.13, 8.14 and 8.15.

Hysteresis 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 (look closely at Figures 8.13 to 8.15). 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 (see Exercises).

8.3.3 Technique: Orientation Representations and Corners

Edge detectors notoriously fail at corners because the assumption that estimates of the partial derivatives in the x and y direction suffice to estimate an oriented gradient becomes unsupportable. At sharp corners or unfortunately oriented corners, these partial derivative estimates are poor because their support will cross the corner. There are a variety of specialized corner detec-



Fi ure 8.14 Edge points marked on the pixel grid for the image shown on the **to** . The edge points on the **left** are obtained using a Gaussian smoothing filter at σ one pixel, and gradient magnitude has been tested against a high threshold to determine whether a point is an edge point. The edge points on the **center** ar obtained using a Gaussian smoothing filter at σ four pixels, and gradient m gnitude has been tested against a high threshold to determine whether a point is an edge point. The edge points on the **right** are obtained using a Gaussian sı oothing filter at σ four pixels, and gradient magnitude has been tested against a ow threshold to determine whether a point is an edge point. At a fine scale, fi e detail at high contrast generates edge points, which disappear at the coarser s ale. When the threshold is high, curves of edge points are often broken because tl gradient magnitude dips below the threshold; for the low threshold, a variety o new edge points of dubious significance are introduced.

tors, which l ok for image neighborhoods where the gradient swings sharply (Figure 8.16). More generally, th statistics of the gradient in an image neighborhood yields quite a useful description of the neighb orhood. There is a rough taxonomy of four qualitative types of image window:

- *con tant windows*, where the gray level is approximately constant;
- *edg windows*, where there is a sharp change in image brightness that runs along a single dire tion within the window;
- *flo windows*, where there are several fine parallel stripes—say hair or fur—within the wir low;
- and *2D windows*, where there is some form of 2D texture—say spots or a corner—within the window.

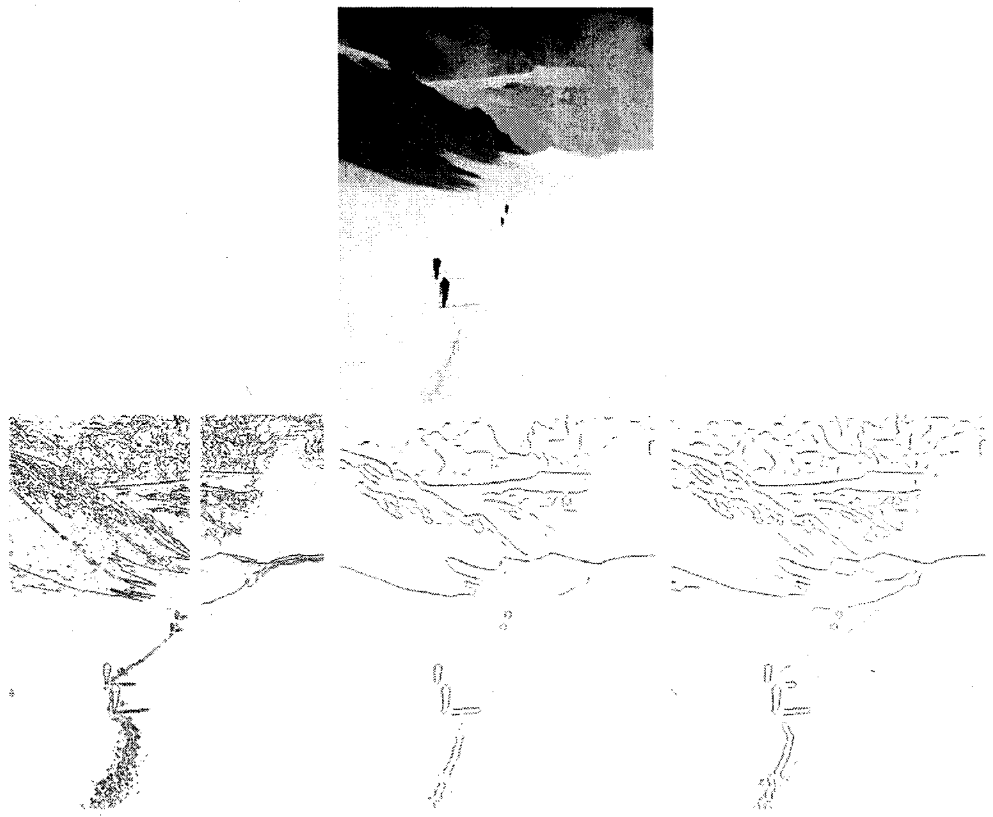


Figure **.15** Edge points marked on the pixel grid for the image shown on the **top**. Th edge points on the **left** are obtained using a Gaussian smoothing filter at σ on pixel, and gradient magnitude has been tested against a high threshold to deter nine whether a point is an edge point. The edge points on the **center** are obt ned using a Gaussian smoothing filter at σ four pixels, and gradient magnit le has been tested against a high threshold to determine whether a point is an ec ge point. The edge points on the **right** are obtained using a Gaussian smoothi ig filter at σ four pixels, and gradient magnitude has been tested against a low t eshold to determine whether a point is an edge point. At a fine scale, fine det il at high contrast generates edge points, which disappear at the coarser scale. W hen the threshold is high, curves of edge points are often broken because the grac ent magnitude dips below the threshold; for the low threshold, a variety of new dge points of dubious significance are introduced.

These cases corres ond to different kinds of behavior on the part of the image gradient. In constant windows, the gradient vector is short; in edge windows, there is a small number of long gradient vectors al pointing in a single direction; in flow windows, there are many gradient vectors pointing in wo directions; and in 2D windows, the gradient vector swings.

These distinc ions can be quite easily drawn by looking at variations in orientation within a window. In partic ılar, the matrix

Fi ;ure 8.16 An image of a joshua tree on the **left** and its orientations shown as ve :tors superimposed on the image on the **right**. The orientation is superimposed o the image as small vectors. Notice that around corners and in textured regions, th orientation vector swings sharply.

$$\mathcal{H} = \sum_{window} \left\{ (\nabla I)(\nabla I)^T \right\}$$

$$\approx \sum_{window} \left\{ \begin{array}{cc} \left(\frac{\partial G_\sigma}{\partial x} **\mathcal{I}\right)\left(\frac{\partial G_\sigma}{\partial x} **\mathcal{I}\right) & \left(\frac{\partial G_\sigma}{\partial x} **\mathcal{I}\right)\left(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}\right) \\ \left(\frac{\partial G_\sigma}{\partial x} **\mathcal{I}\right)\left(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}\right) & \left(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}\right)\left(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}\right) \end{array} \right\}$$

gives a good dea of the behavior of the orientation in a window. In a constant window, both eigenvalues o this matrix are small because all terms are small. In an edge window, we expect to see one large :igenvalue associated with gradients at the edge and one small eigenvalue because few gradient: run in other directions. In a flow window, we expect the same properties of the eigenvalues, ((cept that the large eigenvalue is likely to be larger because many edges contribute. Finally, in a 2) window, both eigenvalues are large.

The bel avior of this matrix is most easily understood by plotting the ellipses

$$(x, y)^T \mathcal{H}^{-1}(x, y) = \epsilon$$

for some sma l constant ϵ. These ellipses are superimposed on the image windows. Their major and minor ax(s are along the eigenvectors of $\mathcal{H}$, and the extent of the ellipses along their major or minor axes c(rresponds to the size of the eigenvalues; this means that a large circle corresponds to an edge w: 1dow and a narrow extended ellipse indicates an edge window (as in Figure 8.17 and Figure 8. 8). Thus, corners could be marked by marking points where the area of this ellipse is extremal a d large. The localization accuracy of this approach is limited by the size of the window and he behavior of the gradient. More accurate localization can be obtained at the price of prov ding a more detailed model of the corner sought (see, for example, Harris and Stephens, 19{ 3 or Schmid, Mohr and Bauckhage, 2000).

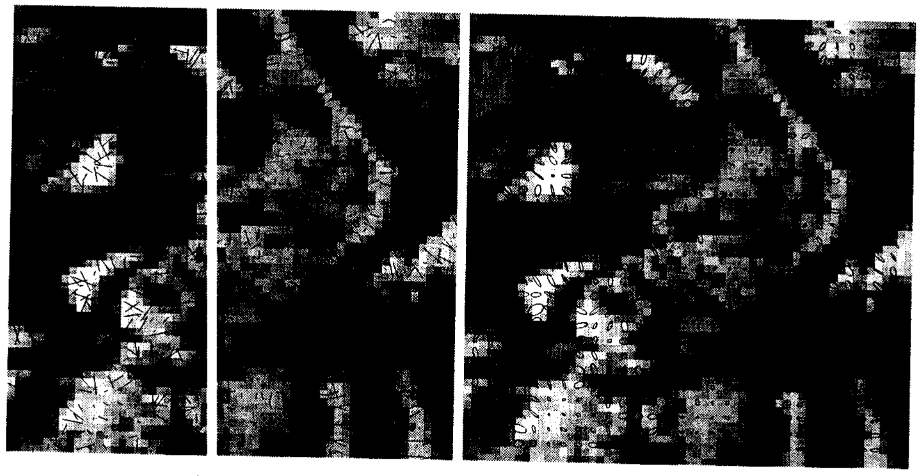


Figure 8.17 The orientation field for a detail of the joshua tree picture. On the **left**, the orien ıtions shown as vectors and superimposed on the image. Orientations have bee n censored to remove those where the gradient magnitude is too small. The **rig** ıt figure shows the ellipses described in the text for a 3x3 window.

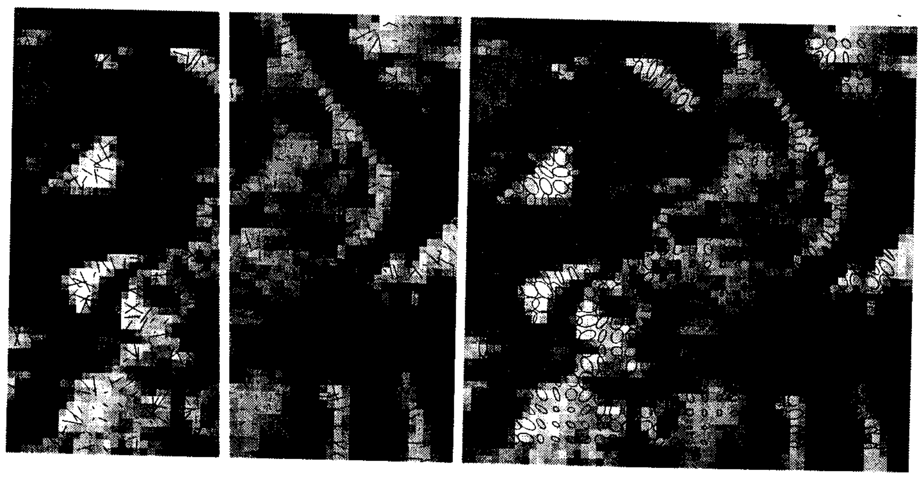


Figure 8.18 he orientation field for a detail of the joshua tree picture. On the **left**, the orienta ions shown as vectors and superimposed on the image. Orientations have beer censored to remove those where the gradient magnitude is too small. The **righ** figure shows the ellipses described in the text, for a 5x5 window.

8.4 NOTES

There is a huge edge detec ion literature. The earliest paper of which we are aware is Julez (1959) (yes, 1959!). Those wishir ; to be acquainted with the early literature in detail should start with a 1975 survey by Davis (197 5); Herskovits and Binford (1970); Horn (1971); and Hueckel (1971), who models edges and the ı detects the model.

Edge detection is a ubject alive with controversy, much of it probably empty. We have hardly scratched the surfa e. There are many optimality criteria for edge detectors, and rather more “optimal” ed e detectors. The key paper in this literature is by Canny (1986); significant variants are due to eriche (1987) and to Spacek (1986). Faugeras' textbook contains a detailed and accessible exp sition of the main issues (1993). At the end of the day, most variants boil down to smoothin the image with something that looks a lot like a Gaussian before measuring the gradient.

Object boun aries are not the same as sharp changes in image values. First, objects may not have a strong ontrast with their backgrounds through sheer bad luck. Second, objects are often covered with texture or markings that generate edges of their own—so many that it is often hard to wade thro gh them to find the relevant pieces of object boundary. Finally, shadows and the like may gener te edges that have no relation to object boundaries. There are some strategies for dealing with th se difficulties.

First, some pplications allow management of illumination; if it is possible to choose the illumination, a ca eful choice can make a tremendous difference in the contrast and eliminate shadows. Second, y setting smoothing parameters large and contrast thresholds high, it is often possible to ensure hat edges due to texture are smoothed over and not marked. This is a dubious business, because t can be hard to choose reliable values of the smoothing and the thresholds, and because it is rverse to regard texture purely as a nuisance, rather than a source of information.

There are o her ways to handle the uncomfortable distinction between edges and object boundaries. First, ne might work to make better edge detectors. This approach is the root of a huge literature, de ling with matters like localization, corner topology, and the like. We incline to the view that retu ns are diminishing rather sharply in this endeavor; we can provide only some pointers to this (st) literature. The reader could start with Bergholm (1987), Deriche (1990), Elder and Zucke (1998), Fleck (1992*a*), Kube and Perona (1996), Olson (1998), Perona and Malik (1990*a*,*b*), r Torre and Poggio (1986).

Second, on might deny the usefulness of edge detection entirely. This approach is rooted in the observatio that some stages of edge detection, particularly nonmaximum suppression, discard informati n that is awfully difficult to retrieve. This is because a hard decision—testing against a thresho l—has been made. Instead, the argument proceeds, one should keep this information around in “soft” (a propaganda term for probabilistic) way. Attactive as these arguments sound, we are inc ined to discount this view because there are currently no practical mechanisms for handling the olumes of soft information so obtained.

Finally, on 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 e ough to construct vision systems that are tolerant to this. Although it sweeps a great deal of d st under the carpet (precisely *how* does one construct such architectures?) we find this approac most attractive and discuss it again and again.

All edge d tectors behave badly at corners; only the details vary. In the case of zero crossings of the Lapl cian of Gaussian, the problem is well understood (Berzins, 1984). This bad behavior has res lted in two lively strands in the literature (What goes wrong? What to do about it?). There are a variety of quite sophisticated corner detectors, mainly because corners make quite good point eatures for correspondence algorithms supporting such activities as stereopsis, reconstruction, structure from motion. This has led to quite detailed practical knowledge of the localisation roperties of corner detectors (e.g., Schmid, Mohr and Bauckhage, 2000).

Another li ely strand in the literature is to determine how well edge detectors do. One may study locali zation accuracy (e.g., Kakarala and Hero, 1992, Lyvers and Mitchell, 1988) or stability (e.g., C o, Meer and Cabrera, 1997, 1998); one may compare with human preferences (e.g., Bowyer, K anenburg and Dougherty, 1999, Dougherty and Bowyer, 1998, Heath, Sarkar, Sanocki and Bc wyer, 1997) or compare performance in the context of a fixed task, such as structure from motion (e. ., Shin, Goldgof and Bowyer, 1998) or recognition (e.g., Shin, Goldgof and Bowyer, 1999). All e lge detectors share some difficulties (e.g., at corners Fleck, 1992*b*).

The edges that our edge detectors respond to are sometimes called *step edges* because they consist of a sharp, " liscontinuous" change in value that is sometimes modeled as a step. A variety of other forms f edge have been studied. The most commonly cited example is the *roof edge*, which consists f a rising segment meeting a falling segment, rather like some of the reflexes that can result fr m the effects of interreflections (Figure 5.16). Another example that also results from interrefl ctions is a composite of a step and a roof. It is possible to find these phenomena by using esse tially the same steps as outlined before (find an "optimal" filter, and do nonmaximum suppressior on its outputs) (Canny, 1986, Perona and Malik, 1990*a,b*). In practice, this is seldom done. There appear to be two reasons. First, there is no comfortable basis in theory (or practice) for the mode that are adopted. What particular composite edges are worth looking for? The easy answer—t se for which optimal filters are reasonably easy to derive—is most unsatisfactory. Second, th semantics of roof edges and more complex composite edges is even vaguer than that of step ec es. There is little notion of what one would *do* with roof edge once it had been found.

Edges are poorly de ined 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 probl ms solved with the output of a roof edge detector. The real difficulty here is that there seems t be no reliable mechanism for predicting, in advance, what is worth detecting. We scratch the urface of this very difficult problem in what follows.

ASSIGNMENTS

PROBLEMS

8.1. Each pixel value in 500 500 pixel image $\mathcal{I}$ is an independent, normally distributed random variable with zero mean and sta dard 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.

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

8.3. We have a camera that ca produce output values that are integers in the range from 0 to 255. Its spatial resolution is 1024 by 76 pixels, and it produces 30 frames a second. We point it at a scene that, in the absence of noise, would roduce the constant value 128. The output of the camera is subject to noise that we model as zero m an stationary additive Gaussian noise with a standard deviation of 1. How long must we wait befor the noise model predicts that we should see a pixel with a negative value? (Hint: You may find it he ful to use logarithms to compute the answer as a straightforward evaluation of $\exp(-128^2/2)$ will yi ld 0; the trick is to get the large positive and large negative logarithms to cancel.)

8.4. We said a sensible 2D a alogue to the 1D second derivative must be rotationally invariant in Section 8.3.1. Why is this tr ?

Programming Assi nments

8.5. Why is it necessary to ch ck that the gradient magnitude is large at zero crossings of the Laplacian of an image? Demonstrate a eries of edges for which this test is significant.

8.6. The Laplacian of a Gauss n looks similar to the difference between two Gaussians at different scales. Compare these two kerne for various values of the two scales. Which choices give a good approximation? How significant i the approximation error in edge finding using a zero-crossing approach?

8.7. Obtain an in plementation of Canny's edge detector (you could try the vision home page; MATLAB has an impl nentation in the image processing toolbox, too) and make a series of images indicating the effects c scale and contrast thresholds on the edges that are detected. How easy is it to set up the edge det ctor to mark only object boundaries? Can you think of applications where this would be easy?

8.8. It is quite ea y to defeat hysteresis in edge detectors that implement it—essentially, one sets the lower and higher hresholds to have the same value. Use this trick to compare the behavior of an edge detector wi 1 and without hysteresis. There are a variety of issues to look at:

(a) What a e you trying to do with the edge detector output? It is sometimes helpful to have linked chains f edge points. Does hysteresis help significantly here?

(b) Noise uppression: 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 reliabl can you use hysteresis to suppress low-contrast edges without breaking high-contrast edges?