reduced number of broken edges; for instance, compare the 45° edges in 10.20(a) and (b). Of course, edges whose intensity values were severely atten due to blurring (e.g., the edges in the tile roof) are likely to be totally elimi by thresholding. We return to the problem of broken edges in Section 10.2.7

10.2.6 More Advanced Techniques for Edge Detection

The edge-detection methods discussed in the previous section are based ply on filtering an image with one or more masks, with no provisions made for edge characteristics and noise content. In this section, we di more advanced techniques that make an attempt to improve on simple detection methods by taking into account factors such as image noise an nature of edges themselves.

The Marr-Hildreth edge detector

One of the earliest successful attempts at incorporating more sophistic analysis into the edge-finding process is attributed to Marr and Hildreth [1 Edge-detection methods in use at the time were based on using small oper (such as the Sobel masks), as discussed in the previous section. Marr and Hild argued (1) that intensity changes are not independent of image scale and so detection requires the use of operators of different sizes; and (2) that a sudde tensity change will give rise to a peak or trough in the first derivative or, equ lently, to a zero crossing in the second derivative (as we saw in Fig. 10.10).

To convince yourself that edge detection is not independent of scale, consider, for example, the roof edge in Fig. 10.8(c). If the scale of the image is reduced, the edge will appear thinner.

These ideas suggest that an operator used for edge detection should two salient features. First and foremost, it should be a differential operato pable of computing a digital approximation of the first or second derivati every point in the image. Second, it should be capable of being "tuned" to at any desired scale, so that large operators can be used to detect blurry e and small operators to detect sharply focused fine detail.

Marr and Hildreth argued that the most satisfactory operator fulfi these conditions is the filter $\nabla^2 G$ where, as defined in Section 3.6.2, ∇^2 i Laplacian operator, $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$, and G is the 2-D Gaussian function

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2}$$

with standard deviation σ (sometimes σ is called the *space constant*). To an expression for $\nabla^2 G$ we perform the following differentiations:

It is customary for Eq. (10.2-21) to differ from the definition of a 2-D Gaussian PDF by the constant term $1/2\pi\sigma^2$. If an exact expression is desired in a given application, then the multiplying constant can be appended to the final result in Eq. (10.2-23).

$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \\ &= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \qquad (10.2 \\ &= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}\end{aligned}$$

ollecting terms gives the final expression:

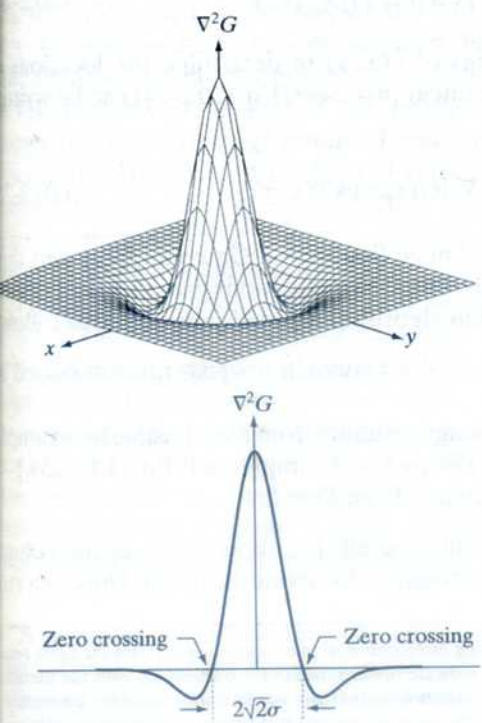
$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (10.2\text{-}23)$$

his expression is called the *Laplacian of a Gaussian* (LoG).

Figures 10.21(a) through (c) show a 3-D plot, image, and cross section of the *gative* of the LoG function (note that the zero crossings of the LoG occur at $+ y^2 = 2\sigma^2$, which defines a circle of radius $\sqrt{2}\sigma$ centered on the origin). ecause of the shape illustrated in Fig. 10.21(a), the LoG function sometimes called the *Mexican hat* operator. Figure 10.21(d) shows a 5×5 mask that pproximates the shape in Fig. 10.21(a) (in practice we would use the *negative* this mask). This approximation is not unique. Its purpose is to capture the sential *shape* of the LoG function; in terms of Fig. 10.21(a), this means a pos- ive, central term surrounded by an adjacent, negative region whose values in- ease as a function of distance from the origin, and a zero outer region. The efficients must sum to zero so that the response of the mask is zero in areas constant intensity.

Masks of arbitrary size can be generated by sampling Eq. (10.2-23) and scal- g the coefficients so that they sum to zero. A more effective approach for enerating a LoG filter is to sample Eq. (10.2-21) to the desired $n \times n$ size and

Note the similarity between the cross section in Fig. 10.21(c) and the highpass filter in Fig. 4.37(d). Thus, we can expect the LoG to behave as a highpass filter.



0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

then convolve† the resulting array with a Laplacian mask, such as the mask in Fig. 10.4(a). Because convolving an image array with a mask whose coefficients sum to zero yields a result whose elements also sum to zero (see Problems 3.16 and 10.14), this approach automatically satisfies the requirement that the sum of the LoG filter coefficients be zero. We discuss the issue of selecting the size of LoG filter later in this section.

There are two fundamental ideas behind the selection of the operator $\nabla^2 G$. First, the Gaussian part of the operator blurs the image, thus reducing the intensity of structures (including noise) at scales much smaller than σ. Unlike averaging of the form discussed in Section 3.5 and used in Fig. 10.18, the Gaussian function is smooth in both the spatial and frequency domains (see Section 4.8.3), and is thus less likely to introduce artifacts (e.g., ringing) not present in the original image. The other idea concerns ∇^2, the second derivative part of the filter. Although first derivatives can be used for detecting abrupt changes in intensity, they are directional operators. The Laplacian, on the other hand, has the important advantage of being isotropic (invariant to rotation), which not only corresponds to characteristics of the human visual system (Marr [1982]) but also responds equally to changes in intensity in any mask direction, thus avoiding having to use multiple masks to calculate the strongest response at any point in the image.

The Marr-Hildreth algorithm consists of convolving the LoG filter with an input image, $f(x, y)$,

$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y) \tag{10.2-24}$$

and then finding the zero crossings of $g(x, y)$ to determine the locations of edges in $f(x, y)$. Because these are linear processes, Eq. (10.2-24) can be written also as

$$g(x, y) = \nabla^2 [G(x, y) \star f(x, y)] \tag{10.2-25}$$

This expression is implemented in the spatial domain using Eq. (3.4-2). It can be implemented also in the frequency domain using Eq. (4.7-1).

indicating that we can smooth the image first with a Gaussian filter and then compute the Laplacian of the result. These two equations give identical results.

The Marr-Hildreth edge-detection algorithm may be summarized as follows:

1. Filter the input image with an $n \times n$ Gaussian lowpass filter obtained by sampling Eq. (10.2-21).
2. Compute the Laplacian of the image resulting from Step 1 using, for example, the 3×3 mask in Fig. 10.4(a). [Steps 1 and 2 implement Eq. (10.2-25).]
3. Find the zero crossings of the image from Step 2.

To specify the size of the Gaussian filter, recall that about 99.7% of the volume under a 2-D Gaussian surface lies between $\pm 3\sigma$ about the mean. Thus, as a rule

†The LoG is a symmetric filter, so spatial filtering using correlation or convolution yields the same result. We use the convolution terminology here to indicate linear filtering for consistency with the literature on this topic. Also, this gives you exposure to terminology that you will encounter in other contexts. It is important that you keep in mind the comments made at the end of Section 3.4.2 regarding this topic.

of thumb, the size of an $n \times n$ LoG discrete filter should be such that n is the smallest odd integer greater than or equal to 6σ. Choosing a filter mask smaller than this will tend to "truncate" the LoG function, with the degree of truncation being inversely proportional to the size of the mask; using a larger mask would make little difference in the result.

One approach for finding the zero crossings at any pixel, p, of the filtered image, $g(x, y)$, is based on using a 3×3 neighborhood centered at p. A zero crossing at p implies that the *signs* of at least two of its opposing neighboring pixels must differ. There are four cases to test: left/right, up/down, and the two diagonals. If the values of $g(x, y)$ are being compared against a threshold (a common approach), then not only must the signs of opposing neighbors be different, but the absolute value of their numerical difference must also exceed the threshold before we can call p a zero-crossing pixel. We illustrate this approach in Example 10.7 below.

Attempting to find the zero crossings by finding the coordinates (x, y), such that $g(x, y) = 0$ is impractical because of noise and/or computational inaccuracies.

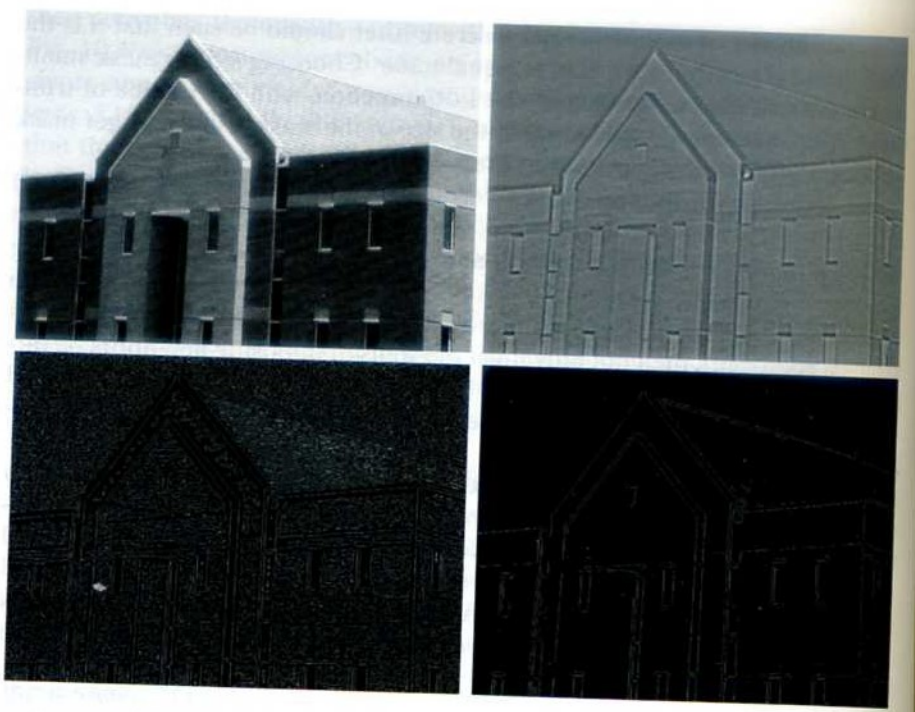
Zero crossings are the key feature of the Marr-Hildreth edge-detection method. The approach discussed in the previous paragraph is attractive because of its simplicity of implementation and because it generally gives good results. If the accuracy of the zero-crossing locations found using this method is inadequate in a particular application, then a technique proposed by Huertas and Medioni [1986] for finding zero crossings with subpixel accuracy can be employed.

EXAMPLE 10.7: Illustration of the Marr-Hildreth edge-detection method.

Figure 10.22(a) shows the original building image used earlier and Fig. 10.22(b) is the result of Steps 1 and 2 of the Marr-Hildreth algorithm, using $\sigma = 4$ (approximately 0.5% of the short dimension of the image) and $n = 25$ (the smallest odd integer greater than or equal to 6σ, as discussed earlier). As in Fig. 10.5, the gray tones in this image are due to scaling. Figure 10.22(c) shows the zero crossings obtained using the 3×3 neighborhood approach discussed above with a threshold of zero. Note that all the edges form closed loops. This so-called "spaghetti" effect is a serious drawback of this method when a threshold value of zero is used (Problem 10.15). We avoid closed-loop edges by using a positive threshold.

Figure 10.22(d) shows the result of using a threshold approximately equal to 4% of the maximum value of the LoG image. Note that the majority of the principal edges were readily detected and "irrelevant" features, such as the edges due to the bricks and the tile roof, were filtered out. As we show in the next section, this type of performance is virtually impossible to obtain using the gradient-based edge-detection techniques discussed in the previous section. Another important consequence of using zero crossings for edge detection is that the resulting edges are 1 pixel thick. This property simplifies subsequent stages of processing, such as edge linking. ■

A procedure used sometimes to take into account the fact mentioned earlier that intensity changes are scale dependent is to filter an image with various values of σ. The resulting zero-crossings edge maps are then combined by keeping only the edges that are common to all maps. This approach can yield



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

useful information, but, due to its complexity, it is used in practice mostly as a design tool for selecting an appropriate value of σ to use with a single filter.

Marr and Hildreth [1980] noted that it is possible to approximate the LoG filter in Eq. (10.2-23) by a difference of Gaussians (DoG):

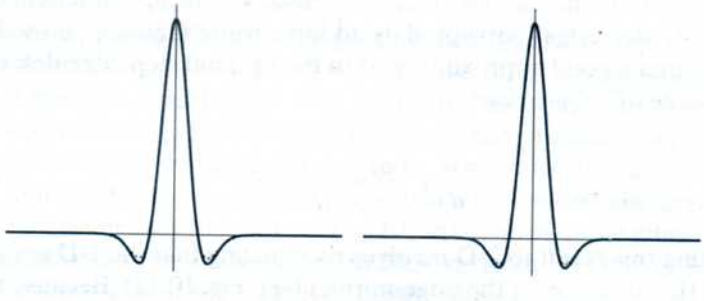
The difference of Gaussians is a highpass filter, as discussed in Section 4.7.4.

$$\mathrm{DoG}(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \tag{10.2-26}$$

with $\sigma_1 > \sigma_2$. Experimental results suggest that certain "channels" in the human vision system are selective with respect to orientation and frequency, and can be modeled using Eq. (10.2-26) with a ratio of standard deviations of 1.75:1. Marr and Hildreth suggested that using the ratio 1.6:1 preserves the basic characteristics of these observations and also provides a closer "engineering" approximation to the LoG function. To make meaningful comparisons between the LoG and DoG, the value of σ for the LoG must be selected as in the following equation so that the LoG and DoG have the same zero crossings (Problem 10.17):

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \tag{10.2-27}$$

Although the zero crossings of the LoG and DoG will be the same when this value of σ is used, their amplitude scales will be different. We can make them compatible by scaling both functions so that they have the same value at the origin.



a b

FIGURE 10.23 (a) Negatives of LoG (solid) and DoG (dotted) profiles using standard deviation ratio of 1.75:1. (b) Profiles obtained using a ratio of

The profiles in Figs. 10.23(a) and (b) were generated with standard deviation ratios of 1:1.75 and 1:1.6, respectively (by convention, the curves shown are inverted, as in Fig. 10.21). The LoG profiles are shown as solid lines while the DoG profiles are dotted. The curves shown are intensity profiles through the center of LoG and DoG arrays generated by sampling Eq. (10.2-23) (with the constant in $1/2\pi\sigma^2$ in front) and Eq. (10.2-26), respectively. The amplitude of all curves at the origin were normalized to 1. As Fig. 10.23(b) shows, the ratio 1:1.6 yielded a closer approximation between the LoG and DoG functions.

Both the LoG and the DoG filtering operations can be implemented with 1-D convolutions instead of using 2-D convolutions directly (Problem 10.19). For an image of size $M \times N$ and a filter of size $n \times n$, doing so reduces the number of multiplications and additions for each convolution from being proportional to n^2MN for 2-D convolutions to being proportional to nMN for 1-D convolutions. This implementation difference is significant. For example, if $n = 25$, a 1-D implementation will require on the order of 12 times fewer multiplication and addition operations than using 2-D convolution.

The Canny edge detector

Although the algorithm is more complex, the performance of the Canny edge detector (Canny [1986]) discussed in this section is superior in general to the edge detectors discussed thus far. Canny's approach is based on three basic objectives:

1. *Low error rate.* All edges should be found, and there should be no spurious responses. That is, the edges detected must be as close as possible to the true edges.
2. *Edge points should be well localized.* The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
3. *Single edge point response.* The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exists.

The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solutions to these formulations. In general, it is difficult (or impossible) to find a closed-form solution

that satisfies all the preceding objectives. However, using numerical optimization with 1-D step edges corrupted by additive white Gaussian noise led to the conclusion that a good approximation† to the optimal step edge detector is the *first derivative of a Gaussian:*

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} \tag{10.2-28}$$

Recall that *white noise* is noise having a frequency spectrum that is continuous and uniform over a specified frequency band. White Gaussian noise is white noise in which the distribution of amplitude values is Gaussian. Gaussian white noise is a good approximation of many real-world situations and generates mathematically tractable models. It has the useful property that its values are statistically independent.

Generalizing this result to 2-D involves recognizing that the 1-D approach *still applies* in the direction of the edge normal (see Fig. 10.12). Because the direction of the normal is unknown beforehand, this would require applying the 1-D edge detector in all possible directions. This task can be approximated by first smoothing the image with a *circular* 2-D Gaussian function, computing the gradient of the result, and then using the gradient magnitude and direction to estimate edge strength and direction at every point.

Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2-29}$$

We form a smoothed image, $f_s(x, y)$, by convolving G and f:

$$f_s(x, y) = G(x, y) \star f(x, y) \tag{10.2-30}$$

This operation is followed by computing the gradient magnitude and direction (angle), as discussed in Section 10.2.5:

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{10.2-31}$$

and

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{10.2-32}$$

with $g_x = \partial f_s/\partial x$ and $g_y = \partial f_s/\partial y$. Any of the filter mask pairs in Fig. 10.14 can be used to obtain g_x and g_y. Equation (10.2-30) is implemented using an $n \times n$ Gaussian mask whose size is discussed below. Keep in mind that $M(x, y)$ and $\alpha(x, y)$ are arrays of the same size as the image from which they are computed.

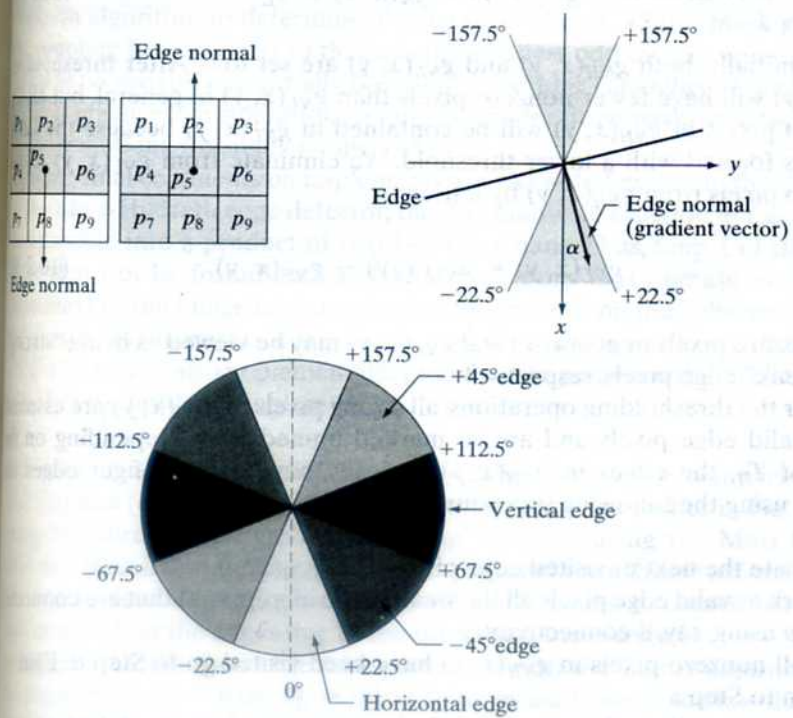
Because it is generated using the gradient, $M(x, y)$ typically contains wide ridges around local maxima (recall the discussion in Section 10.2.1 regarding edges obtained using the gradient). The next step is to thin those ridges. One approach is to use *nonmaxima suppression*. This can be done in several ways, but the essence of the approach is to specify a number of discrete orientations

†Canny [1986] showed that using a Gaussian approximation proved only about 20% worse than using the optimized numerical solution. A difference of this magnitude generally is imperceptible in most applications.

f the edge normal (gradient vector). For example, in a 3 × 3 region we can efine four orientations† for an edge passing through the center point of the egion: horizontal, vertical, +45° and −45°. Figure 10.24(a) shows the situation r the two possible orientations of a horizontal edge. Because we have to uantize all possible edge directions into four, we have to define a range of di- ctions over which we consider an edge to be horizontal. We determine edge irection from the direction of the edge normal, which we obtain directly from e image data using Eq. (10.2-32). As Fig. 10.24(b) shows, if the edge normal is the range of directions from −22.5° to 22.5° or from −157.5° to 157.5°, we ll the edge a horizontal edge. Figure 10.24(c) shows the angle ranges corre- ponding to the four directions under consideration.

Let d_1, d_2, d_3, and d_4 denote the four basic edge directions just discussed for 3 × 3 region: horizontal, −45°, vertical, and +45°, respectively. We can for- ulate the following nonmaxima suppression scheme for a 3 × 3 region cen- ered at *every* point (x, y) in $\alpha(x, y)$:

1. Find the direction d_k that is closest to $\alpha(x, y)$.
2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$



a b
c

FIGURE 10.24 (a) Two possible orientations of a horizontal edge (in gray) in a 3 × 3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3 × 3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

eep in mind that every edge has two possible orientations. For example, an edge whose normal is ori- nted at 0° and an edge whose normal is oriented at 180° are the same *horizontal* edge.

where $g_N(x, y)$ is the nonmaxima-suppressed image. For example, with reference to Fig. 10.24(a), letting (x, y) be at p_5 and assuming a horizontal edge through p_5, the pixels in which we would be interested in Step 2 are p_2 and p_8. Image $g_N(x, y)$ contains only the thinned edges; it is equal to $M(x, y)$ with the nonmaxima edge points suppressed.

The final operation is to threshold $g_N(x, y)$ to reduce false edge points. In Section 10.2.5 we did this using a single threshold, in which all values below the threshold were set to 0. If we set the threshold too low, there will still be some false edges (called *false positives*). If the threshold is set too high, then actual valid edge points will be eliminated (*false negatives*). Canny's algorithm attempts to improve on this situation by using *hysteresis thresholding* which, as we discuss in Section 10.3.6, uses two thresholds: a low threshold, T_L, and a high threshold, T_H. Canny suggested that the ratio of the high to low threshold should be two or three to one.

We can visualize the thresholding operation as creating two additional images

$$g_{NH}(x, y) = g_N(x, y) \geq T_H \tag{10.2-33}$$

and

$$g_{NL}(x, y) = g_N(x, y) \geq T_L \tag{10.2-34}$$

where, initially, both $g_{NH}(x, y)$ and $g_{NL}(x, y)$ are set to 0. After thresholding, $g_{NH}(x, y)$ will have fewer nonzero pixels than $g_{NL}(x, y)$ in general, but all the nonzero pixels in $g_{NH}(x, y)$ will be contained in $g_{NL}(x, y)$ because the latter image is formed with a lower threshold. We eliminate from $g_{NL}(x, y)$ all the nonzero pixels from $g_{NH}(x, y)$ by letting

$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y) \tag{10.2-35}$$

The nonzero pixels in $g_{NH}(x, y)$ and $g_{NL}(x, y)$ may be viewed as being "strong" and "weak" edge pixels, respectively.

After the thresholding operations, all strong pixels in $g_{NH}(x, y)$ are assumed to be valid edge pixels and are so marked immediately. Depending on the value of T_H, the edges in $g_{NH}(x, y)$ typically have gaps. Longer edges are formed using the following procedure:

(a) Locate the next unvisited edge pixel, p, in $g_{NH}(x, y)$.
(b) Mark as valid edge pixels all the weak pixels in $g_{NL}(x, y)$ that are connected to p using, say, 8-connectivity.
(c) If all nonzero pixels in $g_{NH}(x, y)$ have been visited go to Step d. Else, return to Step a.
(d) Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

At the end of this procedure, the final image output by the Canny algorithm is formed by appending to $g_{NH}(x, y)$ all the nonzero pixels from $g_{NL}(x, y)$.

We used two additional images, $g_{NH}(x, y)$ and $g_{NL}(x, y)$, to simplify the cussion. In practice, hysteresis thresholding can be implemented directly ing nonmaxima suppression, and thresholding can be implemented directly $g_N(x, y)$ by forming a list of strong pixels and the weak pixels connected to m.

Summarizing, the Canny edge detection algorithm consists of the following ic steps:

1. Smooth the input image with a Gaussian filter.
2. Compute the gradient magnitude and angle images.
3. Apply nonmaxima suppression to the gradient magnitude image.
4. Use double thresholding and connectivity analysis to detect and link edges.

hough the edges after nonmaxima suppression are thinner than raw gradi- edges, edges thicker than 1 pixel can still remain. To obtain edges 1 pixel k, it is typical to follow Step 4 with one pass of an edge-thinning algorithm Section 9.5.5).

As mentioned earlier, smoothing is accomplished by convolving the input ge with a Gaussian mask whose size, $n \times n$, must be specified. We can use approach discussed in the previous section in connection with the Marr- lreth algorithm to determine a value of n. That is, a filter mask generated ampling Eq. (10.2-29) so that n is the smallest odd integer greater than or al to 6σ provides essentially the "full" smoothing capability of the Gaussian r. If practical considerations require a smaller filter mask, then the tradeoff ss smoothing for smaller values of n.

ome final comments on implementation: As noted earlier in the discussion ie Marr-Hildreth edge detector, the 2-D Gaussian function in Eq. (10.2-29) parable into a product of two 1-D Gaussians. Thus, Step 1 of the Canny rithm can be formulated as 1-D convolutions that operate on the rows umns) of the image one at a time and then work on the columns (rows) of result. Furthermore, if we use the approximations in Eqs. (10.2-12) and 2-13), we can also implement the gradient computations required for Step 2 D convolutions (Problem 10.20).

EXAMPLE 10.8: Illustration of the Canny edge-detection method.

igure 10.25(a) shows the familiar building image. For comparison, Figs. 5(b) and (c) show, respectively, the results obtained earlier in Fig. 10.20(b) g the thresholded gradient and Fig. 10.22(d) using the Marr-Hildreth ctor. Recall that the parameters used in generating those two images were ted to detect the principal edges while attempting to reduce "irrelevant" ures, such as the edges due to the bricks and the tile roof.

igure 10.25(d) shows the result obtained with the Canny algorithm using arameters $T_L = 0.04$, $T_H = 0.10$ (2.5 times the value of the low threshold), 4 and a mask of size 25×25, which corresponds to the smallest odd inte- greater than 6σ. These parameters were chosen interactively to achieve objectives stated in the previous paragraph for the gradient and Marr- lreth images. Comparing the Canny image with the other two images, we

a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

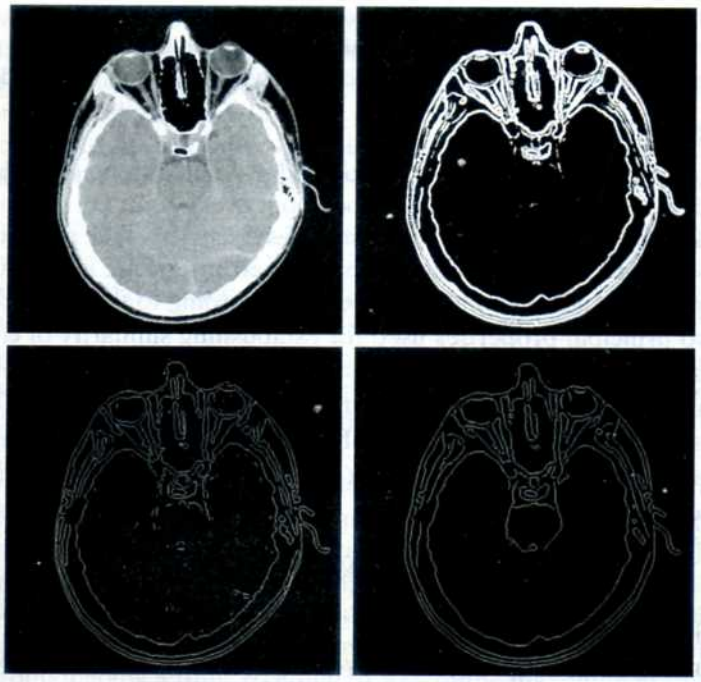
The threshold values given here should be considered only in relative terms. Implementation of most algorithms involves various scaling steps, such as scaling the range of values of the input image to the range [0, 1]. Different scaling schemes obviously would require different values of thresholds from those used in this example.

see significant improvements in detail of the principal edges and, at the same time, more rejection of irrelevant features in the Canny result. Note, for example, that both edges of the concrete band lining the bricks in the upper section of the image were detected by the Canny algorithm, whereas the thresholded gradient lost both of these edges and the Marr-Hildreth image contains only the upper one. In terms of filtering out irrelevant detail, the Canny image does not contain a single edge due to the roof tiles; this is not true in the other two images. The quality of the lines with regard to continuity, thinness, and straightness is also superior in the Canny image. Results such as these have made the Canny algorithm a tool of choice for edge detection. ■

EXAMPLE 10.9: Another illustration of the three principal edge detection methods discussed in this section.

■ As another comparison of the three principal edge-detection methods discussed in this section, consider Fig. 10.26(a) which shows a 512 × 512 head CT image. Our objective in this example is to extract the edges of the outer contour of the brain (the gray region in the image), the contour of the spinal region (shown directly behind the nose, toward the front of the brain), and the outer contour of the head. We wish to generate the thinnest, continuous contours possible, while eliminating edge details related to the gray content in the eyes and brain areas.

Figure 10.26(b) shows a thresholded gradient image that was first smoothed with a 5 × 5 averaging filter. The threshold required to achieve the result shown was 15% of the maximum value of the gradient image. Figure 10.26(c) shows the result obtained with the Marr-Hildreth edge-detection algorithm with a threshold of 0.002, $\sigma = 3$, and a mask of size 19 × 19 pixels. Figure 10.26(d) was obtained using the Canny algorithm with $T_L = 0.05$, $T_H = 0.15$ (3 times the



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

value of the low threshold), $\sigma = 2$, and a mask of size 13 × 13, which, as in the Marr-Hildreth case, corresponds to the smallest odd integer greater than 6σ.

The results in Fig. 10.26 correspond closely to the results and conclusions in the previous example in terms of edge quality and the ability to eliminate irrelevant detail. Note also that the Canny algorithm was the only procedure capable of yielding a totally unbroken edge for the posterior boundary of the brain. It was also the only procedure capable of finding the best contours while eliminating all the edges associated with the gray matter in the original image. ■

As might be expected, the price paid for the improved performance of the Canny algorithm is a more complex implementation than the two approaches discussed earlier, requiring also considerably more execution time. In some applications, such as real-time industrial image processing, cost and speed requirements usually dictate the use of simpler techniques, principally the thresholded gradient approach. When edge quality is the driving force, then the Marr-Hildreth and Canny algorithms, especially the latter, offer superior alternatives.

10.2.7 Edge Linking and Boundary Detection

Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edges completely because of noise, breaks in the edges due to nonuniform illumination, and other effects that introduce spurious discontinuities in intensity values. Therefore, edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries. In this section, we discuss three fundamental approaches to edge linking that are representative of techniques used in practice.