

a b

FIGURE 10.23
 (a) Negatives of the LoG (solid) and DoG (dotted) profiles using a standard deviation ratio of 1.75:1.
 (b) Profiles obtained using a ratio of 1.6:1.

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

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.

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*:

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

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:

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

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

$$f_s(x, y) = G(x, y) \star f(x, y) \quad (10.2-30)$$

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

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

and

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

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.

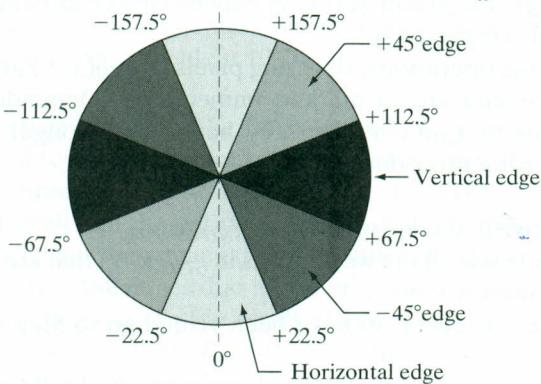
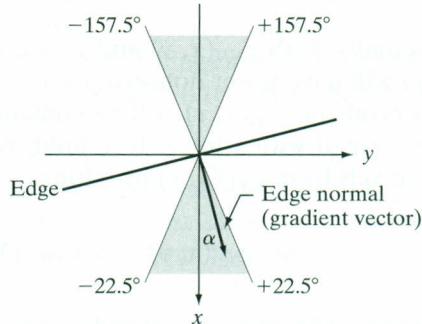
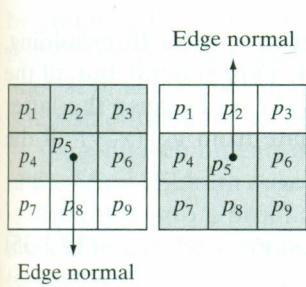
Because it is generated using the gradient, $M(x, y)$ typically contains wide ridges around local maxima (recall the discussion in Section 10.2.1 regarding edges obtained using the gradient). The next step is to thin those ridges. One approach is to use *nonmaxima suppression*. This can be done in several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

of the edge normal (gradient vector). For example, in a 3×3 region we can define four orientations[†] for an edge passing through the center point of the region: horizontal, vertical, $+45^\circ$ and -45° . Figure 10.24(a) shows the situation for the two possible orientations of a horizontal edge. Because we have to quantize all possible edge directions into four, we have to define a range of directions over which we consider an edge to be horizontal. We determine edge direction from the direction of the edge normal, which we obtain directly from the image data using Eq. (10.2-32). As Fig. 10.24(b) shows, if the edge normal is in the range of directions from -22.5° to 22.5° or from -157.5° to 157.5° , we call the edge a horizontal edge. Figure 10.24(c) shows the angle ranges corresponding 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 a 3×3 region: horizontal, -45° , vertical, and $+45^\circ$, respectively. We can formulate the following nonmaxima suppression scheme for a 3×3 region centered 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.

[†]Keep in mind that every edge has two possible orientations. For example, an edge whose normal is oriented 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

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

and

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

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

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

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 discussion. In practice, hysteresis thresholding can be implemented directly during nonmaxima suppression, and thresholding can be implemented directly on $g_N(x, y)$ by forming a list of strong pixels and the weak pixels connected to them.

Summarizing, the Canny edge detection algorithm consists of the following basic 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.

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

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

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

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

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

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

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.

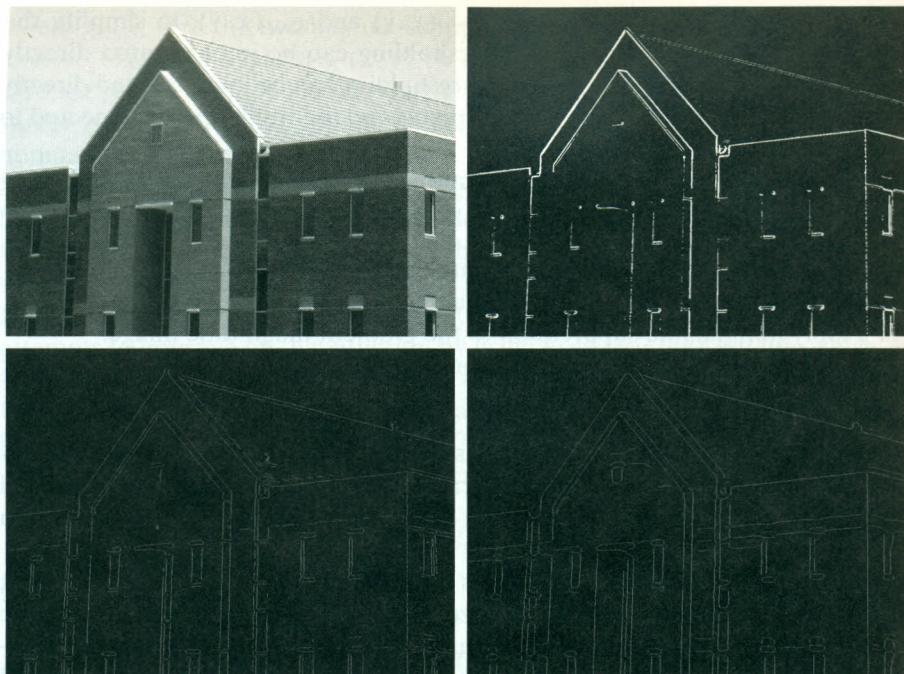
(c) Image obtained using the Marr-Hildreth algorithm.

(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.

EXAMPLE 10.9:

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

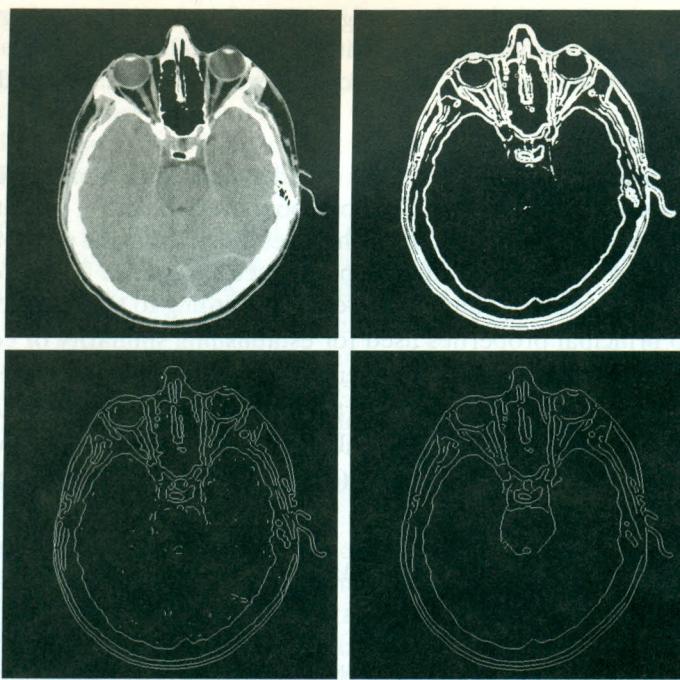


see significant improvements in detail of the principal edges and, at the same time, more rejection of irrelevant features in the Canny result. Note, for example, that both edges of the concrete band lining the bricks in the upper section of the image were detected by the Canny algorithm, whereas the 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.

■ 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

I
t
e
d
b
a
p



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.