4.13

Gradient and Laplacian Edge Detection

Phillip A. Mlsna
Northern Arizona University, Flagstaff

Jeffrey J. Rodríguez
University of Arizona, Tucson

1 Introduction

One of the most fundamental image analysis operations is edge detection. Edges are often vital clues toward the analysis and interpretation of image information, both in biological vision and computer image analysis. Some sort of edge detection capability is present in the visual systems of a wide variety of creatures, so it is obviously useful in their abilities to perceive their surroundings.

For this discussion, it is important to define what is and is not meant by the term "edge". The everyday notion of an edge is usually a physical one, caused by either the shapes of physical objects in three dimensions or by their inherent material properties. Described in geometric terms, there are two types of physical edges: (1) the set of points along which there is an abrupt change in local orientation of a physical surface, and (2) the set of points describing the boundary between two or more materially distinct regions of a physical surface. Most of our perceptual senses, including vision, operate at a distance and gather information using receptors that work in, at most, two dimensions. Only the sense of touch, which requires direct contact to stimulate the skin's pressure sensors, is capable of direct perception of objects in three-dimensional (3D) space. However, some physical edges of the second type may not be perceptible by touch because material differences—for instance different colors of paint—do not always produce distinct tactile sensations. Everyone first develops a working understanding of physical edges in early childhood by touching and handling every object within reach.

The imaging process inherently performs a projection from a 3D scene to a two-dimensional (2D) representation of that scene, according to the viewpoint of the imaging device. Because of this projection process, edges in images have a somewhat different meaning than physical edges. Although the precise definition depends on the application context, an edge can generally be defined as a boundary or contour that separates adjacent image regions having relatively distinct characteristics according to some feature of interest. Most often this feature is gray level or luminance, but others, such as reflectance, color, or texture, are sometimes used. In the most common situation where luminance is of primary interest, edge pixels are those at the locations of abrupt gray-level change. To eliminate single-point impulses from consideration as edge pixels, one usually requires that edges be sustained along a contour; i.e., an edge point must be part of an edge structure having some minimum extent appropriate for the scale of interest. Edge detection is the process of determining which pixels are the edge pixels. The result of the edge detection process is typically an edge map, a new image that

Copyright © 2005 by Elsevier Academic Press.
All rights of reproduction in any form reserved.

describes each original pixel's edge classification and perhaps additional edge attributes, such as magnitude and orientation.

There is usually a strong correspondence between the physical edges of a set of objects and the edges in images containing views of those objects. Infants and young children learn this as they develop hand–eye coordination, gradually associating visual patterns with touch sensations as they feel and handle items in their vicinity. There are many situations, however, in which edges in an image do not correspond to physical edges. Illumination differences are usually responsible for this effect—for example, the boundary of a shadow cast across an otherwise uniform surface.

Conversely, physical edges do not always give rise to edges in images. This can also be caused by certain cases of lighting and surface properties. Consider what happens when one wishes to photograph a scene rich with physical edges—for example, a craggy mountain face consisting of a single type of rock. When this scene is imaged while the sun is directly behind the camera, no shadows are visible in the scene and hence shadow-dependent edges are nonexistent in the photo. The only edges in such a photo are produced by the differences in material reflectance, texture, or color. Since our rocky subject material has little variation of these types, the result is a rather dull photograph, because of the lack of apparent depth caused by the missing edges. Thus, images can exhibit edges having no physical counterpart, and they can also miss capturing edges that do. Although edge information can be very useful in the initial stages of such image processing and analysis tasks as segmentation, registration, and object recognition, edges are not completely reliable for these purposes.

If one defines an edge as an abrupt gray-level change, then the derivative, or gradient, is a natural basis for an edge detector. Figure 1 illustrates the idea with a continuous, one-dimensional (1D) example of a bright central region against a dark background. The left-hand portion of the gray-level function $f_c(x)$ shows a smooth transition from dark to bright as x increases. There must be a point x_0 that marks the transition from the low-amplitude region on the left to the adjacent high-amplitude region in the center. The gradient approach to detecting this edge is to locate x_0 where $|f_c'(x)|$ reaches a local maximum or, equivalently, $f_c'(x)$ reaches a local extremum, as shown in the second plot of Fig. 1. The second derivative, or Laplacian approach, locates x_0 where a zero-crossing of $f_c''(x)$ occurs, as in the third plot of Fig. 1. The right-hand side of Fig. 1 illustrates the case for a falling edge located at x_1.

To use the gradient or the Laplacian approaches as the basis for practical image edge detectors, one must extend the process to two dimensions, adapt to the discrete case, and somehow deal with the difficulties presented by real images. Relative to the 1D edges shown in Fig. 1, edges in 2D images have the additional quality of direction. One usually wishes to find edges regardless of direction, but a directionally sensitive edge detector can be useful at times. Also, the discrete nature

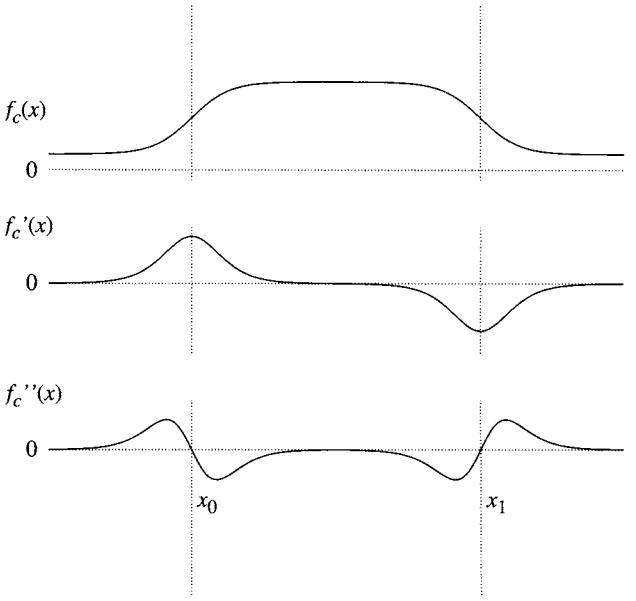

FIGURE 1 Edge detection in the 1D continuous case; changes in $f_c(x)$ indicate edges, and x_0 and x_1 are the edge locations found by local extrema of $f_c'(x)$ or by zero-crossings of $f_c''(x)$.

of digital images requires the use of an approximation to the derivative. Finally, there are a number of problems that can confound the edge detection process in real images. These include noise, crosstalk or interference between nearby edges, and inaccuracies resulting from the use of a discrete grid. False edges, missing edges, and errors in edge location and orientation are often the result.

Because the derivative operator acts as a high-pass filter, edge detectors based on it are sensitive to noise. It is easy for noise inherent in an image to corrupt the real edges by shifting their apparent locations and by adding many false edge pixels. Unless care is taken, seemingly moderate amounts of noise are capable of overwhelming the edge detection process, rendering the results virtually useless. The wide variety of edge detection algorithms developed over the past three decades exists, in large part, because of the many ways proposed for dealing with noise and its effects. Most algorithms employ noise-suppression filtering of some kind before applying the edge detector itself. Some decompose the image into a set of low-pass or band-pass versions, apply the edge detector to each, and merge the results. Still others use adaptive methods, modifying the edge detector's parameters and behavior according to the noise characteristics of the image data. Some recent work by Mathieu and others [20] on fractional derivative operators shows some promise for enriching the gradient and Laplacian possibilities for edge detection. Fractional derivatives may allow better control of noise sensitivity, edge localization, and error rate under various conditions.

An important tradeoff exists between correct detection of the actual edges and precise location of their positions. Edge detection errors can occur in two forms: false positives, in

which nonedge pixels are misclassified as edge pixels, and false negatives, which are the reverse. Detection errors of both types tend to increase with noise, making good noise suppression very important in achieving a high detection accuracy. In general, the potential for noise suppression improves with the spatial extent of the edge detection filter. Hence, the goal of maximum detection accuracy calls for a large-sized filter. Errors in edge localization also increase with noise. To achieve good localization, however, the filter should generally be of small spatial extent. The goals of detection accuracy and location accuracy are thus put into direct conflict, creating a kind of uncertainty principle for edge detection [28].

In this chapter, we cover the basics of gradient and Laplacian edge detection methods in some detail. Following each, we also describe several of the more important and useful edge detection algorithms based on that approach. While the primary focus is on gray-level edge detectors, some discussion of edge detection in color and multispectral images is included.

2 Gradient-based Methods

2.1 Continuous Gradient

The core of gradient edge detection is, of course, the gradient operator, ∇. In continuous form, applied to a continuous-space image, $f_c(x, y)$, the gradient is defined as

$$\nabla f_c(x, y) = \frac{\partial f_c(x, y)}{\partial x}\mathbf{i}_x + \frac{\partial f_c(x, y)}{\partial y}\mathbf{i}_y, \tag{1}$$

where $\mathbf{i}_x$ and $\mathbf{i}_y$ are the unit vectors in the x and y directions. Notice that the gradient is a vector, having both magnitude and direction. Its magnitude, $|\nabla f_c(x_0, y_0)|$, measures the maximum rate of change in the intensity at the location (x_0, y_0). Its direction is that of the greatest increase in intensity; i.e., it points "uphill."

To produce an edge detector, one may simply extend the 1D case described earlier. Consider the effect of finding the local extrema of $\nabla f_c(x, y)$ or the local maxima of

$$|\nabla f_c(x, y)| = \sqrt{\left(\frac{\partial f_c(x, y)}{\partial x}\right)^2 + \left(\frac{\partial f_c(x, y)}{\partial y}\right)^2}. \tag{2}$$

The precise meaning of "local" is very important here. If the maxima of Eq. (2) are found over a 2D neighborhood, the result is a set of isolated points rather than the desired edge contours. The problem stems from the fact that the gradient magnitude is seldom constant along a given edge, so finding the 2D local maxima yields only the locally strongest of the edge contour points. To fully construct edge contours, it is better to apply Eq. (2) to a 1D local neighborhood, namely a line segment, whose direction is chosen to cross the edge. The situation is then similar to that of Fig. 1, where the point of locally maximum gradient magnitude is the edge point. Now the issue becomes how to select the best direction for the line segment used for the search.

The most commonly used method of producing edge segments or contours from Eq. (2) consists of two stages: thresholding and thinning. In the thresholding stage, the gradient magnitude at every point is compared to a pre-defined threshold value, T. All points satisfying the following criterion are classified as candidate edge points:

$$|\nabla f_c(x, y)| \geq T. \tag{3}$$

The set of candidate edge points tends to form strips, which have positive width. Since the desire is usually for zero-width boundary segments or contours to describe the edges, a subsequent processing stage is needed to thin the strips to the final edge contours. Edge contours derived from continuous-space images should have zero width because any local maxima of $|\nabla f_c(x, y)|$, along a line segment that crosses the edge, cannot be adjacent points. For the case of discrete-space images, the nonzero pixel size imposes a minimum practical edge width.

Edge thinning can be accomplished in a number of ways, depending on the application, but thinning by nonmaximum suppression is usually the best choice. Generally speaking, we wish to suppress any point that is not, in a 1D sense, a local maximum in gradient magnitude. Since a 1D local neighborhood search typically produces a single maximum, those points that are local maxima will form edge segments only one point wide. One approach classifies an edge-strip point as an edge point if its gradient magnitude is a local maximum in at least one direction. However, this thinning method sometimes has the side effect of creating false edges near strong edge lines [17]. It is also somewhat inefficient because of the computation required to check along a number of different directions. A better, more efficient thinning approach checks only a single direction, the gradient direction, to test whether a given point is a local maximum in gradient magnitude. The points that pass this scrutiny are classified as edge points. Looking in the gradient direction essentially searches perpendicular to the edge itself, producing a scenario similar to the 1D case shown in Fig. 1. The method is efficient because it is not necessary to search in multiple directions. It also tends to produce edge segments having good localization accuracy. These characteristics make the gradient direction, local extremum method quite popular. The following steps summarize its implementation.

1. Using one of the techniques described in the next section, compute ∇f for all pixels.
2. Determine candidate edge pixels by thresholding all pixels' gradient magnitudes by T.
3. Thin by suppressing all candidate edge pixels whose gradient magnitude is not a local maximum along its

gradient direction. Those that survive nonmaximum supression are classified as edge pixels.

The order of the thinning and thresholding steps might be interchanged. If thresholding is accomplished first, the computational cost of thinning can be significantly reduced. However, it can become difficult to predict the number of edge pixels that will be produced by a given threshold value. By thinning first, there tends to be somewhat better predictability of the richness of the resulting edge map as a function of the applied threshold.

Consider the effect of performing the thresholding and thinning operations in isolation. If thresholding alone were done, the edges would show as strips or patches instead of thin segments. If thinning were done without thresholding, that is, if edge points were simply those having locally maximum gradient magnitude, many false edge points would likely result because of noise. Noise tends to create false edge points because some points in edge-free areas happen to have locally maximum gradient magnitudes. The thresholding step of Eq. (3) is often useful to reduce noise either prior to or following thinning. A variety of adaptive methods have been developed that adjust the threshold according to certain image characteristics, such as an estimate of local signal-to-noise ratio. Adaptive thresholding can often do a better job of noise suppression while reducing the amount of edge fragmentation.

The edge maps in Fig. 3, computed from the original image in Fig. 2, illustrate the effect of the thresholding and subsequent thinning steps.

The selection of the threshold value T is a tradeoff between the wish to fully capture the actual edges in the image and the desire to reject noise. Increasing T decreases sensitivity to noise at the cost of rejecting the weakest edges, forcing the edge segments to become more broken and fragmented. By decreasing T, one can obtain more connected and richer edge contours, but the greater noise sensitivity is likely to produce more false edges. If only thresholding is used, as in Eq. (3) and Fig. 3(a), the edge strips tend to narrow as T increases and widen as it decreases. Figure 4 compares edge maps obtained from several different threshold values.

Sometimes a directional edge detector is useful. One can be obtained by decomposing the gradient into horizontal and vertical components and applying them separately. Expressed in the continuous domain, the operators become:

$$\left|\frac{\partial f_c(x,y)}{\partial x}\right| \geq T \quad \text{for edges in the } y \text{ direction,}$$

$$\left|\frac{\partial f_c(x,y)}{\partial y}\right| \geq T \quad \text{for edges in the } x \text{ direction.}$$

FIGURE 2 Original cameraman image, 512 × 512 pixels.

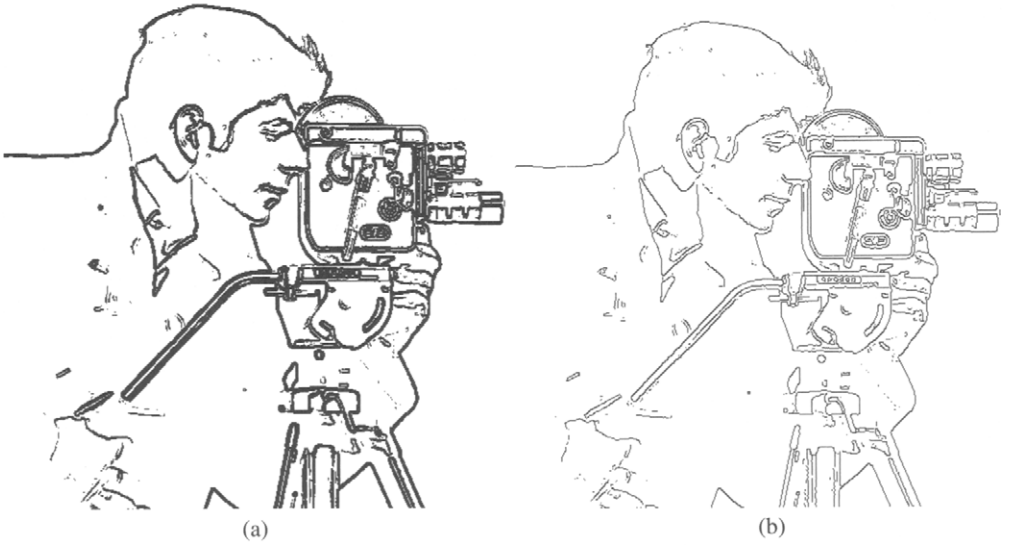

FIGURE 3 Gradient edge detection steps, using the Sobel operator: (a) After thresholding $|\nabla f|$; (b) after thinning (a) by finding the local maximum of $|\nabla f|$ along the gradient direction.

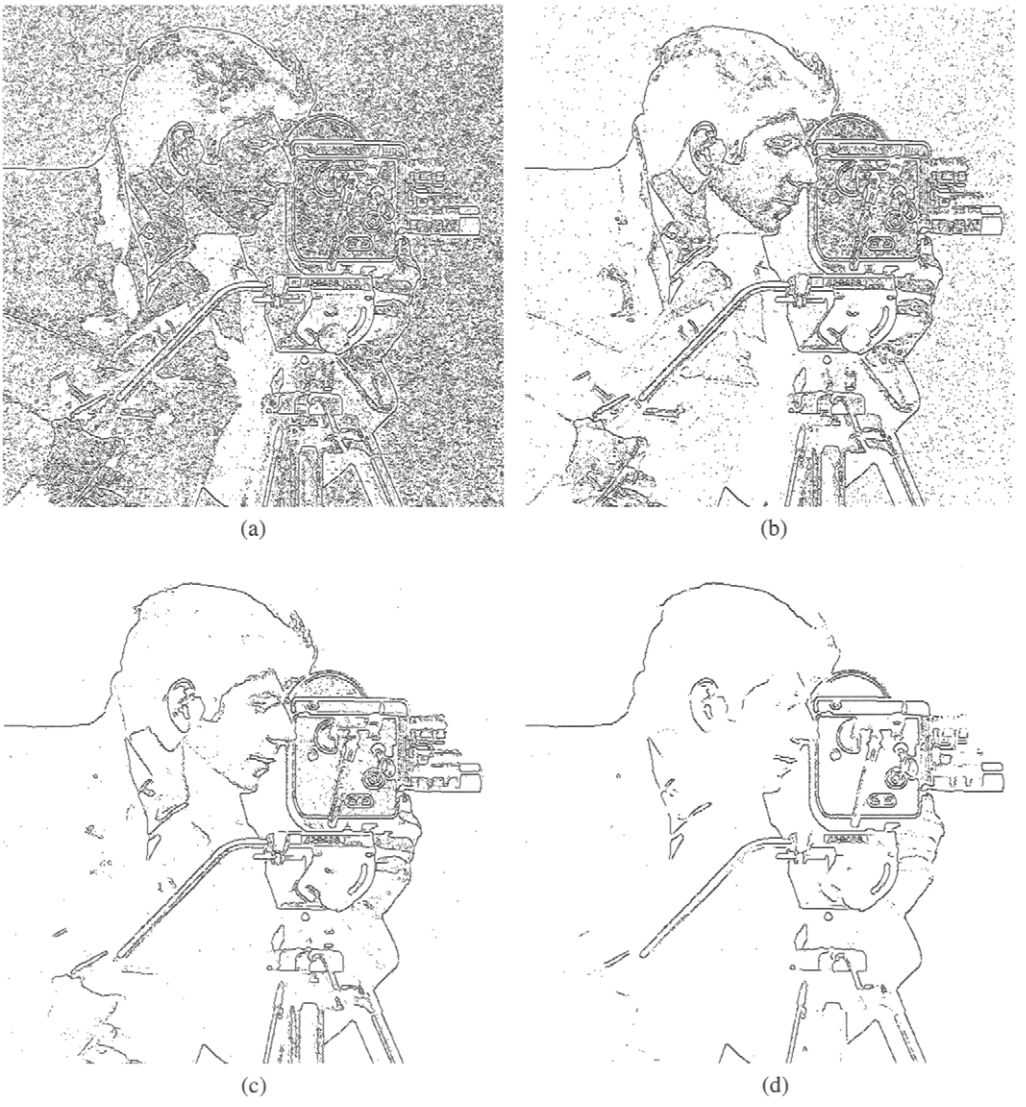

FIGURE 4 Roberts edge maps obtained by using various threshold values: (a) $T=5$, (b) $T=10$, (c) $T=20$, (d) $T=40$. As T increases, more noise-induced edges are rejected along with the weaker real edges.

An example of directional edge detection is illustrated in Fig. 5.

A directional edge detector can be constructed for any desired direction by using the directional derivative along a unit vector $\mathbf{n}$,

$$\frac{\partial f_c}{\partial \mathbf{n}} = \nabla f_c(x, y) \cdot \mathbf{n},$$
$$= \frac{\partial f_c(x, y)}{\partial x} \cos\theta + \frac{\partial f_c(x, y)}{\partial y} \sin\theta, \qquad (4)$$

where θ is the angle of $\mathbf{n}$ relative to the positive x axis. The directional derivative is most sensitive to edges perpendicular to $\mathbf{n}$.

The continuous-space gradient magnitude produces an isotropic or rotationally symmetric edge detector, equally sensitive to edges in any direction [17]. It is easy to show why $|\nabla f|$ is isotropic. In addition to the original X–Y coordinate system, let us introduce a new system, X'–Y', which is rotated by an angle of ϕ relative to X–Y. Let $\mathbf{n}_{x'}$ and $\mathbf{n}_{y'}$ be the unit vectors in the x' and y' directions, respectively. For the gradient magnitude to be isotropic, the same result must be produced in both coordinate systems, regardless of ϕ. Using Eq. (4) along with abbreviated notation, the partial derivatives with respect to the new coordinate axes are

$$f_{x'} = \nabla f \cdot \mathbf{n}_{x'} = f_x \cos\phi + f_y \sin\phi,$$
$$f_{y'} = \nabla f \cdot \mathbf{n}_{y'} = -f_x \sin\phi + f_y \cos\phi.$$

From this point, it's a simple matter of plugging into Eq. (2) to show the gradient magnitudes are identical in both coordinate systems, regardless of the rotation angle, ϕ.

(a) (b)

FIGURE 5 Directional edge detection comparison, using the Sobel operator: (a) results of horizontal difference operator; (b) results of vertical difference operator.

Occasionally, one may wish to reduce the computation load of Eq. (2) by approximating the square root with a computationally simpler function. Three possibilities are

$$|\nabla f_c(x,y)| \approx \max\left\{\left|\frac{\partial f_c(x,y)}{\partial x}\right|, \left|\frac{\partial f_c(x,y)}{\partial y}\right|\right\}, \tag{5}$$

$$\approx \left|\frac{\partial f_c(x,y)}{\partial x}\right| + \left|\frac{\partial f_c(x,y)}{\partial y}\right|, \tag{6}$$

$$\approx \max\left\{\left|\frac{\partial f_c(x,y)}{\partial x}\right|, \left|\frac{\partial f_c(x,y)}{\partial y}\right|\right\} + \frac{1}{4}\min\left\{\left|\frac{\partial f_c(x,y)}{\partial x}\right|, \left|\frac{\partial f_c(x,y)}{\partial y}\right|\right\}. \tag{7}$$

One should be aware that approximations of this type may alter the properties of the gradient somewhat. For instance, the approximated gradient magnitudes of Eqs. (5), (6), and (7) are not isotropic and produce their greatest errors for purely diagonally oriented edges. All three estimates are correct only for the pure horizontal and vertical cases. Otherwise, Eq. (5) consistently underestimates the true gradient magnitude while Eq. (6) overestimates it. This makes Eq. (5) biased against diagonal edges and Eq. (6) biased toward them. The estimate of Eq. (7) is by far the most accurate of the three.

2.2 Discrete Gradient Operators

In the continuous-space image, $f_c(x,y)$, let x and y represent the horizontal and vertical axes, respectively. Let the discrete-space representation of $f_c(x,y)$ be $f(n_1,n_2)$, with n_1 describing the horizontal position and n_2 describing the vertical. Let us assume that the positive directions of n_1 and n_2 are to the right and upward, respectively. For use on discrete-space images, the continuous gradient's derivative operators must be approximated in discrete form. The approximation takes the form of a pair of orthogonally oriented filters, $h_1(n_1,n_2)$ and $h_2(n_1,n_2)$, which must be separately convolved with the image. Based on Eq. (1), the gradient estimate is

$$\widehat{\nabla} f(n_1,n_2) = f_1(n_1,n_2)\,\mathbf{i}_{n_1} + f_2(n_1,n_2)\,\mathbf{i}_{n_2},$$

where

$$f_1(n_1,n_2) = f(n_1,n_2) * h_1(n_1,n_2)$$
$$f_2(n_1,n_2) = f(n_1,n_2) * h_2(n_1,n_2).$$

Two filters are necessary because the gradient requires the computation of an orthogonal pair of directional derivatives. The gradient magnitude and direction estimates can then be computed as follows:

$$\left|\widehat{\nabla} f(n_1,n_2)\right| = \sqrt{f_1^2(n_1,n_2) + f_2^2(n_1,n_2)},$$
$$\angle \widehat{\nabla} f(n_1,n_2) = \tan^{-1}\left(\frac{f_2(n_1,n_2)}{f_1(n_1,n_2)}\right). \tag{8}$$

Each of the filters implements a derivative and should not respond to a constant, so the sum of its coefficients must always be zero. A more general statement of this property is described later in this chapter by Eq. (12).

There are many possible derivative-approximation filters for use in gradient estimation. Let us start with the simplest

cases. Two simple approximation schemes for the derivative in one dimension are

$$\begin{aligned} &\text{First difference:} && f(n) - f(n-1), \\ &\text{Central difference:} && \tfrac{1}{2}[f(n+1) - f(n-1)], \end{aligned}$$

which result in the following 1D convolutional filters:

$$\begin{aligned} &\text{First difference:} && h(n) = \delta(n) - \delta(n-1) = [\mathbf{1} \quad -1], \\ &\text{Central difference:} && h(n) = \delta(n+1) - \delta(n-1) = \tfrac{1}{2}[1 \ \mathbf{0} \ -1]. \end{aligned} \tag{9}$$

It is common practice to incorporate the dimensional reversal into the filter kernel mask and then perform correlation instead of convolution. This exploits the close relationship between the discrete convolution and correlation operators, thereby saving a bit of computation. As a practical matter, the convolution and correlation responses to a given gradient-based edge detection mask will differ only in sign. However, the magnitude of the response—not its sign—is usually the important thing. Henceforth, the filters are expressed as correlation masks and the dimensional reversal will be omitted. For instance, the convolutional filters of Eq. (9) are expressed in correlational form, with $h(-n)$ listed as $h(n)$ for simplicity:

$$\begin{aligned} &\text{First difference:} && h(n) = [-\mathbf{1} \quad 1], \\ &\text{Central difference:} && h(n) = \tfrac{1}{2}[-1 \quad \mathbf{0} \quad 1]. \end{aligned} \tag{10}$$

The scaling factor of $\frac{1}{2}$ for the central difference is caused by the two-pixel distance between the nonzero samples. The origin positions for both filters are usually set at (n_1, n_2), shown in boldface. The gradient magnitude threshold value can be easily adjusted to compensate for any scaling, so the scale factor will be omitted from here on.

Let us now extend Eq. (10) to the 2D case. These derivative approximations can be expressed as filter kernels, whose impulse responses, $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$, are shown dimensionally reversed as the following correlation masks:

$$\begin{aligned} &\text{First difference:} && h_1(n_1, n_2) = \begin{bmatrix} 0 & 0 \\ -\mathbf{1} & 1 \end{bmatrix}, && h_2(n_1, n_2) = \begin{bmatrix} 1 & 0 \\ -\mathbf{1} & 0 \end{bmatrix}, \\ &\text{Central difference:} && h_1(n_1, n_2) = \begin{bmatrix} 0 & 0 & 0 \\ -1 & \mathbf{0} & 1 \\ 0 & 0 & 0 \end{bmatrix}, && h_2(n_1, n_2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \mathbf{0} & 0 \\ 0 & -1 & 0 \end{bmatrix}. \end{aligned}$$

Again, the origin position is indicated by boldface. Filters $h_1(n_1, n_2)$ respond most strongly to vertical edges and do not respond to horizontal edges. The $h_2(n_1, n_2)$ counterparts respond to horizontal edges and not to vertical ones.

If used to detect edges, the pair of first difference filters above presents the problem that the zero crossings of its two [−1 1] derivative kernels lie at different positions. This prevents the two filters from measuring horizontal and vertical edge characteristics at the same location, causing error in the estimated gradient. The central difference, caused by the common center of its horizontal and vertical differencing kernels, avoids this position mismatch problem. This benefit comes at the costs of larger filter size and the fact that the measured gradient at a pixel (n_1, n_2) does not actually consider the value of that pixel.

Rotating the first difference kernels by an angle of $\frac{\pi}{4}$ and stretching the grid a bit produces the $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$ correlation masks for the Roberts operator:

$$\begin{bmatrix} 0 & 1 \\ -\mathbf{1} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \mathbf{0} & -1 \end{bmatrix}.$$

The Roberts operator's component filters are tuned for diagonal edges rather than vertical and horizontal ones. For use in an edge detector based on the gradient magnitude, it is important only that the two filters be orthogonal. They need not be aligned with the n_1 and n_2 axes. The pair of Roberts filters have a common zero-crossing point for their differencing kernels. This common center eliminates the position mismatch error exhibited by the horizontal–vertical first difference pair, as described earlier. If the origins of the Roberts kernels are positioned on the +1 samples, as is sometimes found in the literature, then no common center point exists for their first differences.

The Roberts operator, like any simple first-difference gradient operator, has two undesirable characteristics. First, the zero crossing of its [−1 1] diagonal kernel lies off grid, but the edge location must be assigned to an actual pixel location, namely the one at the filter's origin. This can create edge location bias that may lead to location errors approaching the interpixel distance. If we could use the central difference instead of the first difference, this problem would be reduced because the central difference operator inherently constrains its zero crossing to an exact pixel location.

The other difficulty caused by the first difference is its noise sensitivity. In fact, both the first- and central-difference derivative estimators are quite sensitive to noise. The noise problem can be reduced somewhat by incorporating smoothing into each filter in the direction normal to that of the difference. Consider an example based on the central difference in one direction for which we wish to smooth along the orthogonal direction with a simple three-sample

average. To that end, let us define the impulse responses of two filters:

$$h_a(n_1) = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix}, \quad h_b(n_2) = \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}.$$

Since h_a is a function only of n_1 and h_b depends only on n_2, one can simply multiply them as an outer product to form a separable derivative filter that incorporates smoothing:

$$h_a(n_1)h_b(n_2) = h_1(n_1, n_2),$$

$$\begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Repeating this process for the orthogonal case produces the Prewitt operator mask:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & \mathbf{0} & 0 \\ -1 & -1 & -1 \end{bmatrix}.$$

The Prewitt edge gradient operator simultaneously accomplishes differentiation in one coordinate direction, using the central difference, and noise reduction in the orthogonal direction, by means of local averaging. Because it uses the central difference instead of the first difference, there is less edge-location bias.

In general, the smoothing characteristics can be adjusted by choosing an appropriate low-pass filter kernel in place of the Prewitt's three-sample average. One such variation is the Sobel operator, one of the most widely used gradient edge detectors:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & \mathbf{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

Sobel's operator is often a better choice than Prewitt's because the low-pass filter produced by the [1 2 1] kernel results in a smoother frequency response compared to that of [1 1 1].

The Prewitt and Sobel operators respond differently to diagonal edges than to horizontal or vertical ones. This behavior is a consequence of the fact that their filter coefficients do not compensate for the different grid spacings in the diagonal and the horizontal directions. The Prewitt operator is less sensitive to diagonal edges than to vertical or horizontal ones. The opposite is true for the Sobel operator [23]. A variation designed for equal gradient magnitude response to diagonal, horizontal, and vertical edges is the Frei–Chen operator:

$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \mathbf{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & \mathbf{0} & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}.$$

However, even the Frei–Chen operator retains some directional sensitivity in gradient magnitude, so it is not truly isotropic. The residual anisotropy is caused by the fact that the difference operators used to approximate Eq. (1) are not rotationally symmetric. Merron and Brady [21] describe a simple method for greatly reducing the residual directional bias by using a set of four difference operators instead of two. Their operators are oriented in increments of $\frac{\pi}{4}$ radians, adding a pair of diagonal ones to the original horizontal and vertical pair. Averaging the gradients produced by the diagonal operators with those of the nondiagonal ones allows their complementary directional biases to reduce the overall anisotropy. However, Ziou and Wang [31] have described how an isotropic gradient applied to a discrete grid tends to introduce some anisotropy. They have also analyzed the errors of gradient magnitude and direction as a function of edge translation and orientation for several detectors. Figure 6 shows the results of performing edge detection on an example image by applying the discrete gradient operators discussed so far.

Haralick's facet model [10, 11] provides another way of calculating the gradient in order to perform edge detection. In the sloped facet model, a small neighborhood is parameterized by $\alpha n_2 + \beta n_1 + \gamma$, describing the plane that best fits the gray levels in that neighborhood. The plane parameters α and β can be used to compute the gradient magnitude:

$$|\nabla f_c(n_1, n_2)| = \sqrt{\alpha^2 + \beta^2}$$

The facet model also provides means for computing directional derivatives, zero crossings, and a variety of other useful operations.

Improved noise suppression is possible with increased kernel size. The additional coefficients can be used to better approximate the desired continuous-space noise-suppression filter. Greater filter extent can also be used to reduce directional sensitivity by more accurately modeling an ideal isotropic filter. However, increasing the kernel size will exacerbate edge localization problems and create interference between nearby edges. Noise suppression can be improved by other methods as well. Papers by Bovik [3] and Hardie and Boncelet [12] are just two that describe the use of edge-enhancing prefilters, which simultaneously suppress noise and steepen edges prior to gradient edge detection.

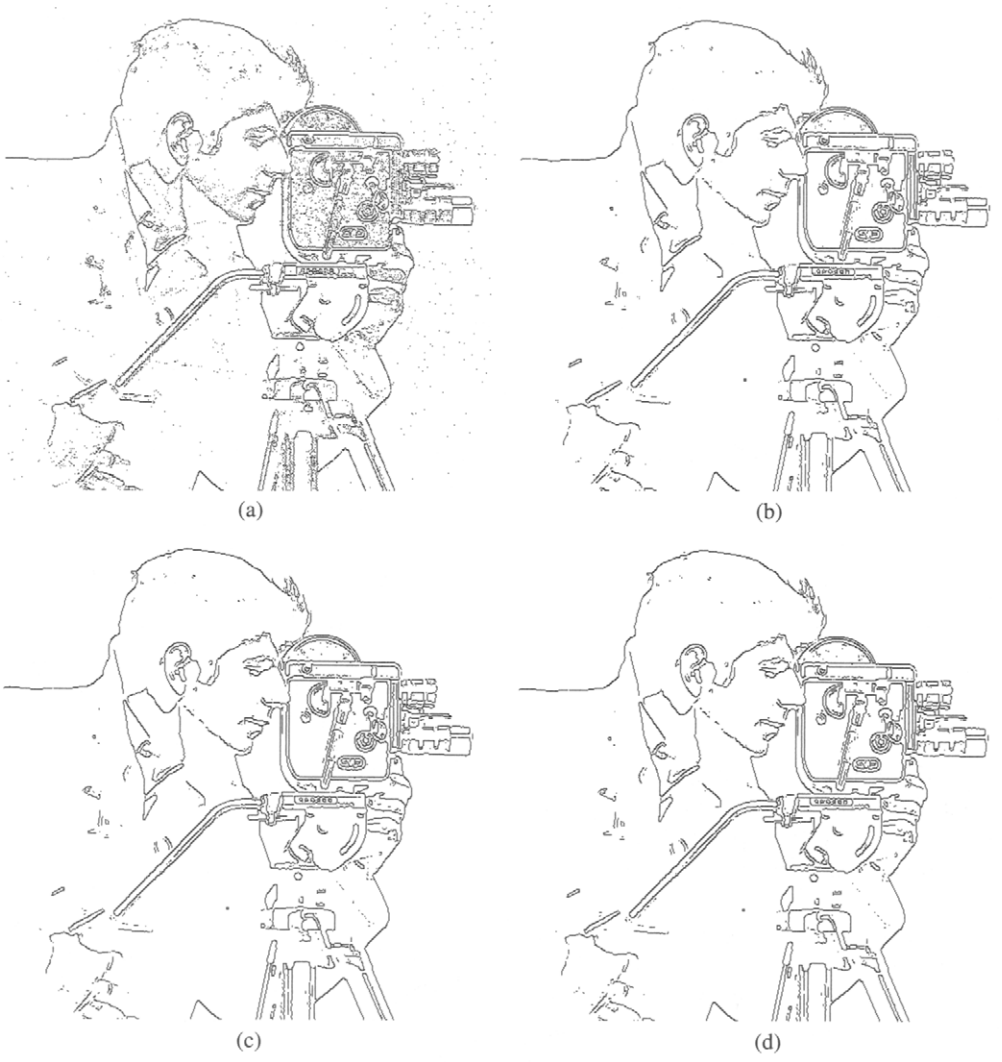

FIGURE 6 Comparison of edge detection using various gradient operators: (a) Roberts, (b) 3 × 3 Prewitt, (c) 3 × 3 Sobel, (d) 3 × 3 Frei-Chen. In each case, the threshold has been set to allow a fair comparison.

3 Laplacian-based Methods

3.1 Continuous Laplacian

The Laplacian is defined as

$$\nabla^2 f_c(x, y) = \boldsymbol{\nabla} \cdot \boldsymbol{\nabla} f_c(x, y) = \frac{\partial^2 f_c(x, y)}{\partial x^2} + \frac{\partial^2 f_c(x, y)}{\partial y^2}. \qquad (11)$$

The zero crossings of $\nabla^2 f_c(x, y)$ occur at the edge points of $f_c(x, y)$ because of the second derivative action (see Fig. 1). Laplacian-based edge detection has the nice property that it produces edges of zero thickness, making edge-thinning steps unnecessary. This is because the zero crossings themselves define the edge locations.

The continuous Laplacian is isotropic, favoring no particular edge orientation. Consequently, its second partial terms in Eq. (11) can be oriented in any direction as long as they remain perpendicular to each other. Consider an ideal, straight, and noise-free edge oriented in an arbitrary direction. Let us realign the first term of Eq. (11) parallel to that edge and the second term perpendicular to it. The first term then generates no response at all because it acts only along the edge. The second term produces a zero crossing at the edge position along its edge-crossing profile.

An edge detector based solely on the zero crossings of the continuous Laplacian produces closed edge contours if the image, $f(x, y)$, meets certain smoothness constraints [28]. The contours are closed because edge strength is not considered, so even the slightest, most gradual intensity transition produces a zero crossing. In effect, the zero-crossing contours define the boundaries that separate regions of nearly constant intensity in the original image. The second derivative zero crossings occur at the local extrema of the first derivative (see Fig. 1), but many zero crossings are not

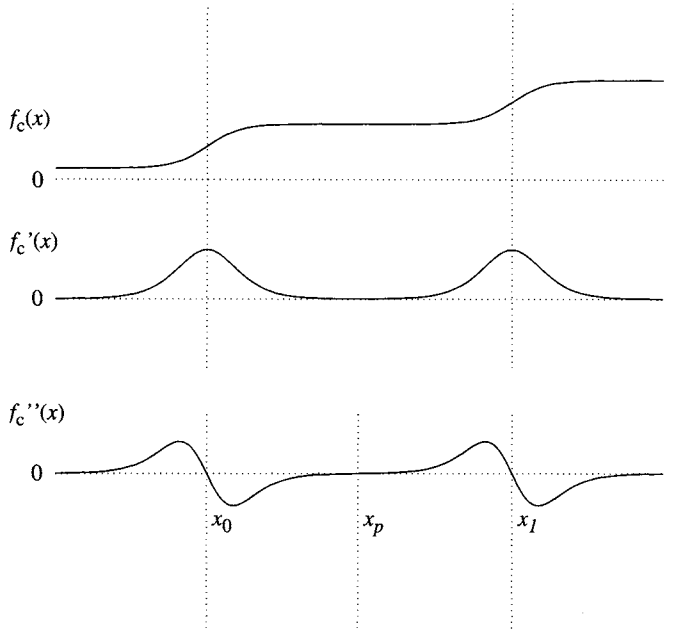

FIGURE 7 The zero-crossing of $f_c''(x)$ at x_p creates a phantom edge.

local maxima of the gradient magnitude. Some local minima of the gradient magnitude give rise to phantom edges, which can be largely eliminated by appropriately thresholding the edge strength. Figure 7 illustrates a 1D example of a phantom edge.

Noise presents a problem for the Laplacian edge detector in several ways. First, the second-derivative action of Eq. (11) makes the Laplacian even more sensitive to noise than the first-derivative-based gradient. Second, noise produces many false edge contours because it introduces variation to the constant-intensity regions in the noise-free image. Third, noise alters the locations of the zero-crossing points, producing location errors along the edge contours. The problem of noise-induced false edges can be addressed by applying an additional test to the zero-crossing points. Only the zero crossings that satisfy this new criterion are considered edge points. One commonly-used technique classifies a zero crossing as an edge point if the local gray-level variance exceeds a threshold amount. Another method is to select the strong edges by thresholding the gradient magnitude or the slope of the Laplacian output at the zero crossing. Both criteria serve to reject zero crossing points that are more likely caused by noise than by a real edge in the original scene. Of course, thresholding the zero crossings in this manner tends to break up the closed contours.

Like any derivative filter, the continuous-space Laplacian filter, $h_c(x, y)$, has this important zero-mean property [15]:

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h_c(x, y)\,dx\,dy = 0. \tag{12}$$

In other words, $h_c(x, y)$ is a surface bounding equal volumes above and below zero. Consequently, $\nabla^2 f_c(x, y)$ will also have equal volumes above and below zero. This property eliminates any response that is due to the constant or DC bias contained in $f_c(x, y)$. Without DC bias rejection, the filter's edge detection performance would be compromised. Gunn [15] has described the effect of truncation of the Laplacian and the subsequent bias introduced when Eq. (12) is violated.

3.2 Discrete Laplacian Operators

It is useful to construct a filter to serve as the Laplacian operator when applied to a discrete-space image. Recall that the gradient, which is a vector, required a pair of orthogonal filters. The Laplacian is a scalar. Therefore, a single filter, $h(n_1, n_2)$, is sufficient for realizing a Laplacian operator. The Laplacian estimate for an image, $f(n_1, n_2)$, is then

$$\widehat{\nabla}^2 f(n_1, n_2) = f(n_1, n_2) * h(n_1, n_2).$$

One of the simplest Laplacian operators can be derived as follows. First needed is an approximation to the derivative in x, so let us use a simple first difference.

$$\frac{\partial f_c(x, y)}{\partial x} \rightarrow f_x(n_1, n_2) = f(n_1 + 1, n_2) - f(n_1, n_2). \tag{13}$$

The second derivative in x can be built by applying the first difference to Eq. (13). However, we discussed earlier how the first difference produces location errors because its zero crossing lies off grid. This second application of a first difference can be shifted to counteract the error introduced by the previous one:

$$\frac{\partial^2 f_c(x, y)}{\partial x^2} \rightarrow f_{xx}(n_1, n_2) = f_x(n_1, n_2) - f_x(n_1 - 1, n_2). \tag{14}$$

Combining the two derivative-approximation stages from Eqs. (13) and (14) produces

$$\frac{\partial^2 f_c(x, y)}{\partial x^2} \rightarrow f_{xx}(n_1, n_2) = f(n_1 + 1, n_2) - 2f(n_1, n_2) + f(n_1 - 1, n_2) = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}. \tag{15}$$

Proceeding in an identical manner for y yields

$$\frac{\partial^2 f_c(x,y)}{\partial y^2} \rightarrow f_{yy}(n_1,n_2) = f(n_1,n_2+1) - 2f(n_1,n_2) + f(n_1,n_2-1) = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}. \tag{16}$$

Combining the x and y second partials of Eqs. (15) and (16) produces a filter, $h(n_1, n_2)$, which estimates the Laplacian:

$$\begin{aligned} \nabla^2 f_c(x,y) &\rightarrow \hat{\nabla}^2 f(n_1,n_2) \\ &= f_{xx}(n_1,n_2) + f_{yy}(n_1,n_2) \\ &= f(n_1+1,n_2) + f(n_1-1,n_2) + f(n_1,n_2+1) \\ &\quad + f(n_1,n_2-1) - 4f(n_1,n_2) \\ &= \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

Other Laplacian estimation filters can be constructed using this method of designing a pair of appropriate 1D second derivative filters and combining them into a single 2D filter. The results depend on the choice of derivative approximator, the size of the desired filter kernel, and the characteristics of any noise-reduction filtering applied. Two other 3×3 examples are

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}.$$

In general, a discrete-space smoothed Laplacian filter can be easily constructed by sampling an appropriate continuous-space function, such as the Laplacian of Gaussian. When constructing a Laplacian filter, make sure that the kernel's coefficients sum to zero in order to satisfy the discrete form of Eq. (12). Truncation effects may upset this property and create bias. If so, the filter coefficients should be adjusted in a way that restores proper balance.

Locating zero crossings in the discrete-space image, $\nabla^2 f(n_1, n_2)$, is fairly straightforward. Each pixel should be compared to its eight immediate neighbors; a four-way neighborhood comparison, while faster, may yield broken contours. If a pixel, **p**, differs in sign with its neighbor, **q**, an edge lies between them. The pixel, **p**, is classified as a zero crossing if

$$|\nabla^2 f(\mathbf{p})| \leq |\nabla^2 f(\mathbf{q})|. \tag{17}$$

3.3 The Laplacian of Gaussian (Marr–Hildreth Operator)

It is common for a single image to contain edges having widely different sharpnesses and scales, from blurry and gradual to crisp and abrupt. Edge scale information is often useful as an aid toward image understanding. For instance, edges at low resolution tend to indicate gross shapes while texture tends to become important at higher resolutions. An edge detected over a wide range of scale is more likely to be physically significant in the scene than an edge found only within a narrow range of scale. Furthermore, the effects of noise are usually most deleterious at the finer scales.

Marr and Hildreth [19] advocated the need for an operator that can be tuned to detect edges at a particular scale. Their method is based on filtering the image with a Gaussian kernel selected for a particular edge scale. The Gaussian smoothing operation serves to band-limit the image to a small range of frequencies, reducing the noise sensitivity problem when detecting zero crossings. The image is filtered over a variety of scales and the Laplacian zero crossings are computed at each. This produces a set of edge maps as a function of edge scale. Each edge point can be considered to reside in a region of scale space, for which edge point location is a function of x, y, and σ. Scale space has been successfully used to refine and analyze edge maps [30].

The Gaussian has some very desirable properties that facilitate this edge detection procedure. First, the Gaussian function is smooth and localized in both the spatial and frequency domains, providing a good compromise between the need for avoiding false edges and for minimizing errors in edge position. In fact, Torre and Poggio [28] describe the Gaussian as the only real-valued function that minimizes the product of spatial- and frequency-domain spreads. The Laplacian of Gaussian essentially acts as a bandpass filter because of its differential and smoothing behavior. Second, the Gaussian is separable, which helps make computation very efficient.

Omitting the scaling factor, the Gaussian filter can be written as

$$g_c(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right). \tag{18}$$

Its frequency response, $G(\Omega_x, \Omega_y)$, is also Gaussian:

$$G(\Omega_x, \Omega_y) = 2\pi\sigma^2 \exp\left(-\frac{\sigma^2}{2}\left(\Omega_x^2 + \Omega_y^2\right)\right).$$

The σ parameter is inversely related to the cutoff frequency.

Because the convolution and Laplacian operations are both linear and shift invariant, their computation order can be interchanged:

$$\nabla^2\left[f_c(x, y) * g_c(x, y)\right] = \left[\nabla^2 g_c(x, y)\right] * f_c(x, y). \quad (19)$$

Here we take advantage of the fact that the derivative is a linear operator. Therefore, Gaussian filtering followed by differentiation is the same as filtering with the derivative of a Gaussian. The right-hand side of Eq. (19) usually provides for more efficient computation since $\nabla^2 g_c(x, y)$ can be prepared in advance due to its image independence. The Laplacian of Gaussian (LoG) filter, $h_c(x, y)$, therefore has the following impulse response:

$$\begin{aligned} h_c(x, y) &= \nabla^2 g_c(x, y) \\ &= \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \end{aligned} \quad (20)$$

To implement the LoG in discrete form, one may construct a filter, $h(n_1, n_2)$, by sampling Eq. (20) after choosing a value of σ, then convolving with the image. If the filter extent is not small, it is usually more efficient to work in the frequency domain by multiplying the discrete Fourier transforms of the filter and the image, then inverse transforming the result. The fast Fourier transform, or FFT, is the method of choice for computing these transforms.

Although the discrete form of Eq. (20) is a 2D filter, Chen, et al. [7], have shown that it is actually the sum of two separable filters because the Gaussian itself is a separable function. By constructing and applying the appropriate 1D filters successively to the rows and columns of the image, the computational expense of 2D convolution becomes unnecessary. Separable convolution to implement the LoG is roughly 1–2 orders of magnitude more efficient than 2D convolution. If an image is $M \times M$ in size, the number of operations at each pixel is M^2 for 2D convolution and only $2M$ if done in a separable, 1D manner.

Figure 8 shows an example of applying the LoG using various σ values. Figure 8(d) includes a gradient magnitude threshold, which suppresses noise and breaks contours. Lim [17] describes an adaptive thresholding scheme that produces better results.

Equation (20) has the shape of a *sombrero* or "Mexican hat". Figure 9 shows a perspective plot of $\nabla^2 g_c(x, y)$ and its frequency response, $F\{\nabla^2 g_c(x, y)\}$. This profile closely mimics the response of the spatial receptive field found in biologic vision. Biologic receptive fields have been shown to have a circularly symmetric impulse response, with a central excitory region surrounded by an inhibitory band.

When sampling the LoG to produce a discrete version, it is important to size the filter large enough to avoid significant truncation effects. A good rule of thumb is to make the filter at least three times the width of the LoG's central excitory lobe [23]. Siohan et al. [27] describes two approaches for the practical design of LoG filters. The errors in edge location produced by the LoG have been analyzed in some detail by Berzins [2]. Gunn [15] has analyzed and described the relationship between the choice of LoG mask size and the resulting probability of edge detection and localization errors.

Sarkar and Boyer [25] have developed an optimal recursive filter for use as a zero-crossing edge detector. They observed that the optimal zero-crossing detector cannot generally be obtained by simply taking the derivative of the optimal gradient detector. Their filter has been optimized for step edges and to simultaneously achieve low error rate, high edge localization precision, and low rate of spurious responses. These goals are similar to those of Canny, whose method is described later in this chapter. The Sarkar–Boyer filter is claimed to perform somewhat better than the LoG filter [25].

3.4 Difference of Gaussian

The Laplacian of Gaussian of Eq. (20) can be closely approximated by the difference of two Gaussians having properly-chosen scales. The difference of Gaussian (DoG) filter is

$$h_c(x, y) = g_{c1}(x, y) - g_{c2}(x, y),$$

where

$$\frac{\sigma_2}{\sigma_1} \approx 1.6$$

and g_{c1}, g_{c2} are evaluated using Eq. (18). However, the LoG is usually preferred because it is theoretically optimal and its separability allows for efficient computation [19]. For the same accuracy of results, the DoG requires a slightly larger filter size [14].

The technique of unsharp masking, used in photography, is basically a difference of Gaussians operation done with light and negatives. Unsharp masking involves making a somewhat blurry exposure of an original negative onto a new piece of film. When the film is developed, it contains a blurred and inverted-brightness version of the original negative. Finally, a print is made from these two negatives sandwiched together, producing a sharpened image with the edges showing increased contrast.

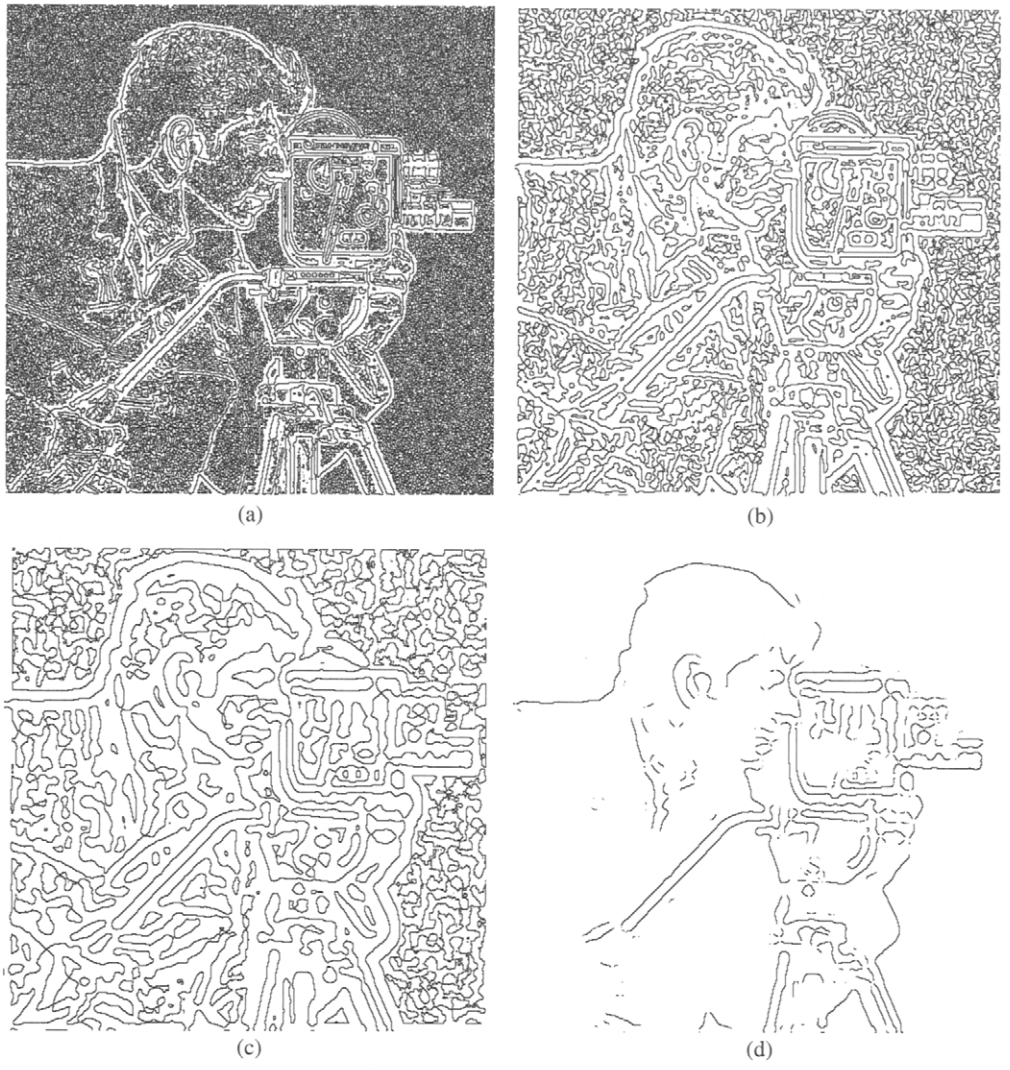

FIGURE 8 Zero crossings of $f * \nabla^2 g$ for several values of σ, with (d) also thresholded: (a) $\sigma = 1.0$, (b) $\sigma = 1.5$, (c) $\sigma = 2.0$, (d) $\sigma = 2.0$ and $T = 20$.

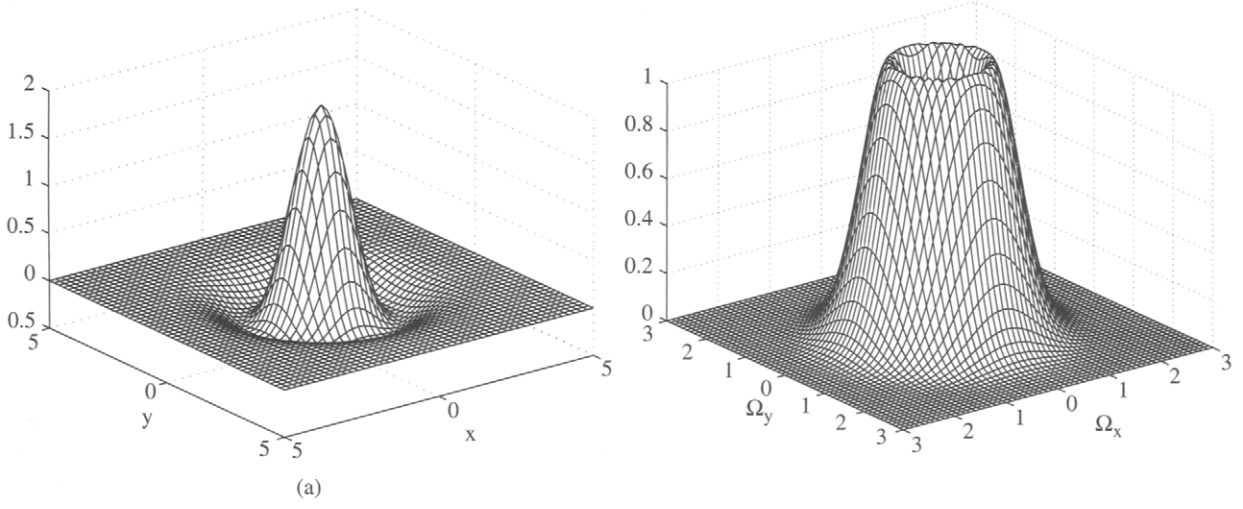

FIGURE 9 Plots of the LoG and its frequency response for $\sigma = 1$: (a) $-\nabla^2 g_c(x, y)$, the negative of Eq. (20); (b) $F\{\nabla^2 g_c(x, y)\}$, the bandpass-shaped frequency response of Eq. (20).

Nature uses the difference of Gaussians as a basis for the architecture of the retina's visual receptive field. The spatial-domain impulse response of a photoreceptor cell in the mammalian retina has a roughly Gaussian shape. The photoreceptor output feeds into horizontal cells in the adjacent layer of neurons. Each horizontal cell averages the responses of the receptors in its immediate neighborhood, producing a Gaussian-shaped impulse response with a higher σ than that of a single photoreceptor. Both layers send their outputs to the third layer, where bipolar neurons subtract the high-σ neighborhood averages from the central photoreceptors' low-σ responses. This produces a biologic realization of the difference-of-Gaussian filter, approximating the behavior of the Laplacian of Gaussian. The retina actually implements DoG bandpass filters at several spatial frequencies [18].

4 Canny's Method

Canny's method [5] uses the concepts of both the first and second derivatives in a very effective manner. His is a classic application of the gradient approach to edge detection in the presence of additive white Gaussian noise, but it also incorporates elements of the Laplacian approach. The method has three simultaneous goals: low rate of detection errors, good edge localization, and only a single detection response per edge. Canny assumed that false-positive and false-negative detection errors are equally undesirable and so gave them equal weight. He further assumed that each edge has nearly constant cross section and orientation, but his general method includes a way to effectively deal with the cases of curved edges and corners. With these constraints, Canny determined the optimal 1D edge detector for the step edge and showed that its impulse response can be approximated fairly well by the derivative of a Gaussian.

An important action of Canny's edge detector is to prevent multiple responses per true edge. Without this criterion, the optimal step-edge detector would have an impulse response in the form of a truncated signum function. (The signum function produces $+1$ for any positive argument and -1 for any negative argument.) But this type of filter has high bandwidth, allowing noise or texture to produce several local maxima in the vicinity of the actual edge. The effect of the derivative of Gaussian is to prevent multiple responses by smoothing the truncated signum in order to permit only one response peak in the edge neighborhood. The choice of variance for the Gaussian kernel controls the filter width and the amount of smoothing. This defines the width of the neighborhood in which only a single peak is to be allowed. The variance selected should be proportional to the amount of noise present. If the variance is chosen too low, the filter can produce multiple detections for a single edge; if too high, edge localization suffers needlessly. Because the edges in a given image are likely to differ in signal-to-noise ratio, a single-filter implementation is usually not best for detecting them. Hence, a thorough edge detection procedure should operate at different scales.

Canny's approach begins by smoothing the image with a Gaussian filter:

$$g_c(x, y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{21}$$

One may sample and truncate Eq. (21) to produce a finite-extent filter, $g(n_1, n_2)$. At each pixel, Eq. (8) is used to estimate the gradient direction. From a set of prepared edge detection filter masks having various orientations, the one oriented nearest to the gradient direction for the targeted pixel is then chosen. When applied to the Gaussian-smoothed image, this filter produces an estimate of gradient magnitude at that pixel. One may instead apply simpler methods, such as the central difference operator, to estimate the partial derivatives. The initial Gaussian smoothing step makes additional smoothing along the edge, as with the Prewitt or Sobel operators, completely unnecessary. Next, the goal is to suppress non-maxima of the gradient magnitude by testing a 3×3 neighborhood, comparing the magnitude at the center pixel with those at interpolated positions to either side along the gradient direction.

The pixels that survive to this point are candidates for the edge map. To produce an edge map from these candidate pixels, Canny applies thresholding by gradient magnitude in an adaptive manner with hysteresis. An estimate of the noise in the image determines the values of a pair of thresholds, with the upper threshold typically two or three times that of the lower. A candidate edge segment is included in the output edge map if at least one of its pixels has a gradient magnitude exceeding the upper threshold, but pixels not meeting the lower threshold are excluded. This hysteresis action helps reduce the problem of broken edge contours while improving the ability to reject noise.

A set of edge maps over a range of scales can be produced by varying the σ values used to Gaussian-filter the image. Since smoothing at different scales produces different errors in edge location, an edge segment that appears in multiple edge maps at different scales may exhibit some position shift. Canny proposed unifying the set of edge maps into a single result by a technique he called "feature synthesis," which proceeds in a fine-to-coarse manner while tracking the edge segments within their possible displacements.

The preoriented edge detection filters, mentioned previously, have some interesting properties. Each mask includes a derivative of Gaussian function to perform the nearly optimal directional derivative across the intended edge. A smooth, averaging profile appears in the mask along the intended edge direction in order to reduce noise without compromising the sharpness of the edge profile. In the

smoothing direction, the filter extent is usually several times that in the derivative direction when the filter is intended for straight edges. Canny's method includes a "goodness of fit" test to determine if the selected filter is appropriate before it is applied. The test examines the gray-level variance of the strip of pixels along the smoothing direction of the filter. If the variance is small, then the edge must be close to linear, and the filter is a good choice. A large variance indicates the presence of curvature or a corner, in which case a better choice of filter would have smaller extent in the smoothing direction. There were six oriented filters used in Canny's work, thus the greatest directional mismatch between the actual gradient and the nearest filter is 15 degrees.

As discussed previously, edges can be detected from either the maxima of the gradient magnitude or the zero crossings of the second derivative. Another way to realize the essence of Canny's method is to look for zero crossings of the second directional derivative taken along the gradient direction. Let us examine the mathematic basis for this. If $\mathbf{n}$ is a unit vector in the gradient direction, and f is the Gaussian-smoothed image, then we wish to find

$$\begin{aligned}\frac{\partial^2 f}{\partial \mathbf{n}^2} &= \nabla\left(\frac{\partial f}{\partial \mathbf{n}}\right)\cdot \mathbf{n} \\ &= \nabla\left(\nabla f \cdot \mathbf{n}\right)\cdot \mathbf{n},\end{aligned}$$

which can be expanded to the following form:

$$\frac{\partial^2 f}{\partial \mathbf{n}^2} = \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{\sqrt{f_x^2 + f_y^2}} \tag{22}$$

In Eq. (22), a concise notation has been used for the partial derivatives.

Like the Laplacian approach, Canny's method looks for zero crossings of the second derivative. The Laplacian's second derivative is nondirectional; it includes a component taken parallel to the edge and another taken across it. Canny's is evaluated only in the gradient direction, directly across the local edge. A derivative taken along an edge is counterproductive because it introduces noise without improving edge detection capability. By being selective about the direction in which its derivatives are evaluated, Canny's approach avoids this source of noise and tends to produce better results.

Figures 10 and 11 illustrate the results of applying the Canny edge detector of Eq. (22) after Gaussian smoothing, then looking for zero crossings. Figure 10 demonstrates the effect of using the same upper and lower thresholds, T_U and T_L, over a range of σ values. The behavior of hysteresis thresholding is shown in Fig. 11. The partial derivatives were approximated using central differences. Thresholding was performed with hysteresis, but using fixed threshold values for each image instead of Canny's noise-adaptive threshold values. Zero-crossing detection was implemented in an eight-way manner, as described by Eq. (17) in the earlier discussion of discrete Laplacian operators. Also, Canny's preoriented edge detection filters were not used in preparing these examples, so it was not possible to adapt the edge detection filters according to the "goodness of fit" of the local edge profile as Canny did.

Ding and Goshtasby [9] have developed refinements to Canny's method. Canny selects edge pixels by comparing gradient magnitudes of neighboring pixels along the gradient direction. Ding does this also, and classifies these as major edge pixels. He further selects minor edge pixels as those having locally maximum gradient magnitude in any direction. Next, the major–minor edge branch points are located. Branches that do not contain a major edge are eliminated. The result is a decrease in the number of missed and broken edges.

5 Approaches for Color and Multispectral Images

Edge detection for color images presents additional challenges because of the three color components used. The most straightforward technique is to perform edge detection on the luminance component image while ignoring the chrominance information. The only computational cost beyond that for gray-scale images is incurred in obtaining the luminance component image, if necessary. In many color spaces, such as *YIQ*, *HSL*, *CIELUV*, and *CIELAB*, the luminance image is simply one of the components in that representation. For others, such as *RGB*, computing the luminance image is usually easy and efficient. The main drawback to luminance-only processing is that important edges are often not confined to the luminance component. Therefore, a gray-level difference in the luminance component is often not the most appropriate criterion for edge detection in color images.

Another rather obvious approach is to apply a desired edge detection method separately to each color component and construct a cumulative edge map. One possibility for overall gradient magnitude, shown here for the *RGB* color space, combines the component gradient magnitudes [24]:

$$\left|\nabla f_c(x,y)\right| = \left|\nabla f_R(x,y)\right| + \left|\nabla f_G(x,y)\right| + \left|\nabla f_B(x,y)\right|.$$

The results, however, are biased according to the properties of the particular color space used. It is often important to employ a color space that is appropriate for the target application. For example, edge detection that is intended to approximate the human visual system's behavior should utilize a color space having a perceptual basis, such as *CIELUV* or perhaps *HSL*. Another complication is the fact that the components' gradient vectors may not always be similarly oriented,

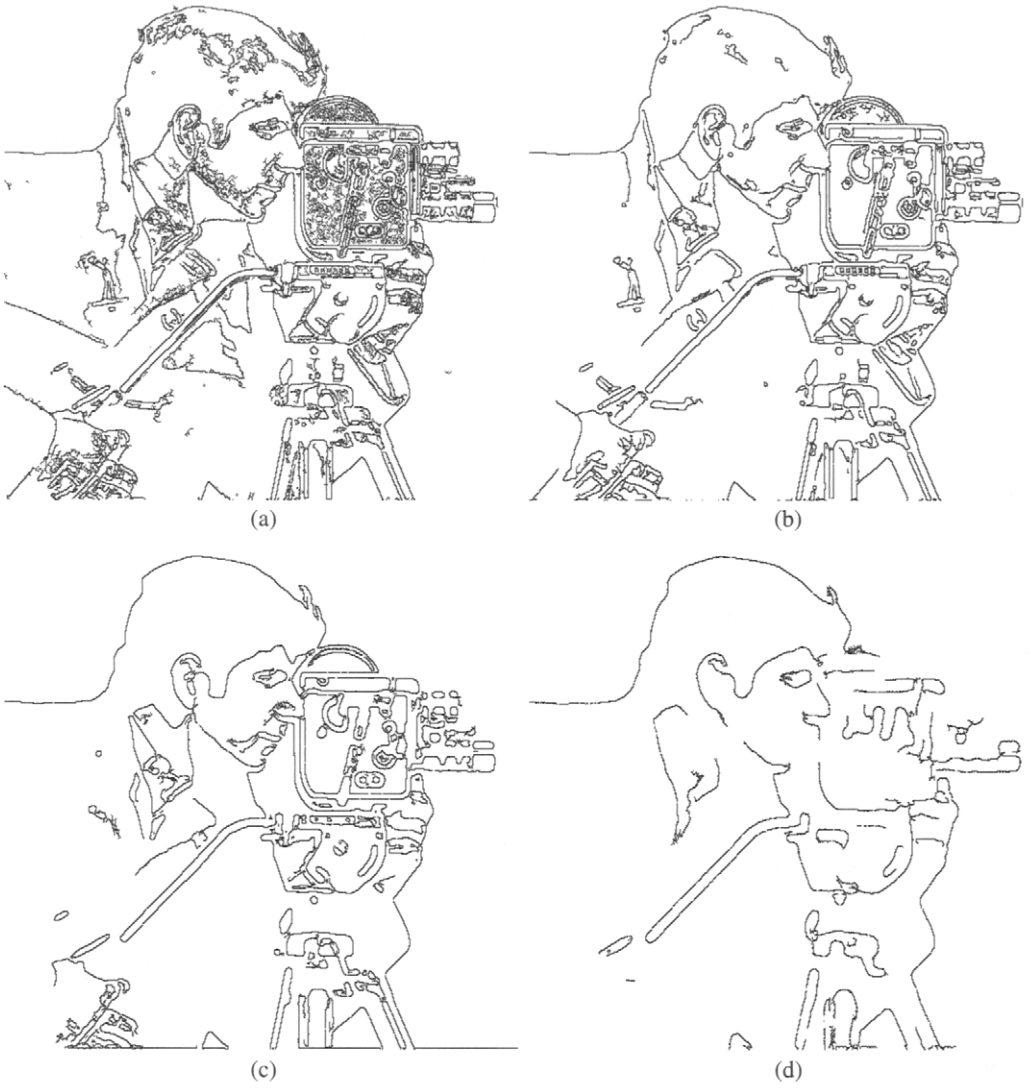

FIGURE 10 Canny edge detector of Eq. (22) applied after Gaussian smoothing over a range of σ: (a) $\sigma = 0.5$, (b) $\sigma = 1$, (c) $\sigma = 2$, (d) $\sigma = 4$. The thresholds are fixed in each case at $T_U = 10$ and $T_L = 4$.

making the search for local maxima of $|\nabla f_c|$ along the gradient direction more difficult. If a total gradient image were to be computed by summing the color component gradient vectors, not just their magnitudes, then inconsistent orientations of the component gradients could destructively interfere and nullify some edges.

Vector approaches to color edge detection, while generally less computationally efficient, tend to have better theoretical justification. Euclidean distance in color space between the color vectors of a given pixel and its neighbors can be a good basis for an edge detector [24]. For the *RGB* case, the magnitude of the vector gradient is

$$|\nabla f_c(x,y)| = \sqrt{|\nabla f_R(x,y)|^2 + |\nabla f_G(x,y)|^2 + |\nabla f_B(x,y)|^2}.$$

Trahanias and Venetsanopoulos [29] described the use of vector order statistics as the basis for color edge detection. A later paper by Scharcanski and Venetsanopoulos [26] furthered the concept. While not strictly founded on the gradient or Laplacian, their techniques are effective and worth mention here because of their vector bases. The basic idea is to look for changes in local vector statistics, particularly vector dispersion, to indicate the presence of edges.

Multispectral images can have many components, complicating the edge detection problem even further. Cebrián et al. [6] describes several methods that are useful for multispectral images having any number of components. His description uses the second directional derivative in the gradient direction as the basis for the edge detector, but other types of detectors can be used instead. The components-average method forms a gray-scale image by averaging all components, which have first

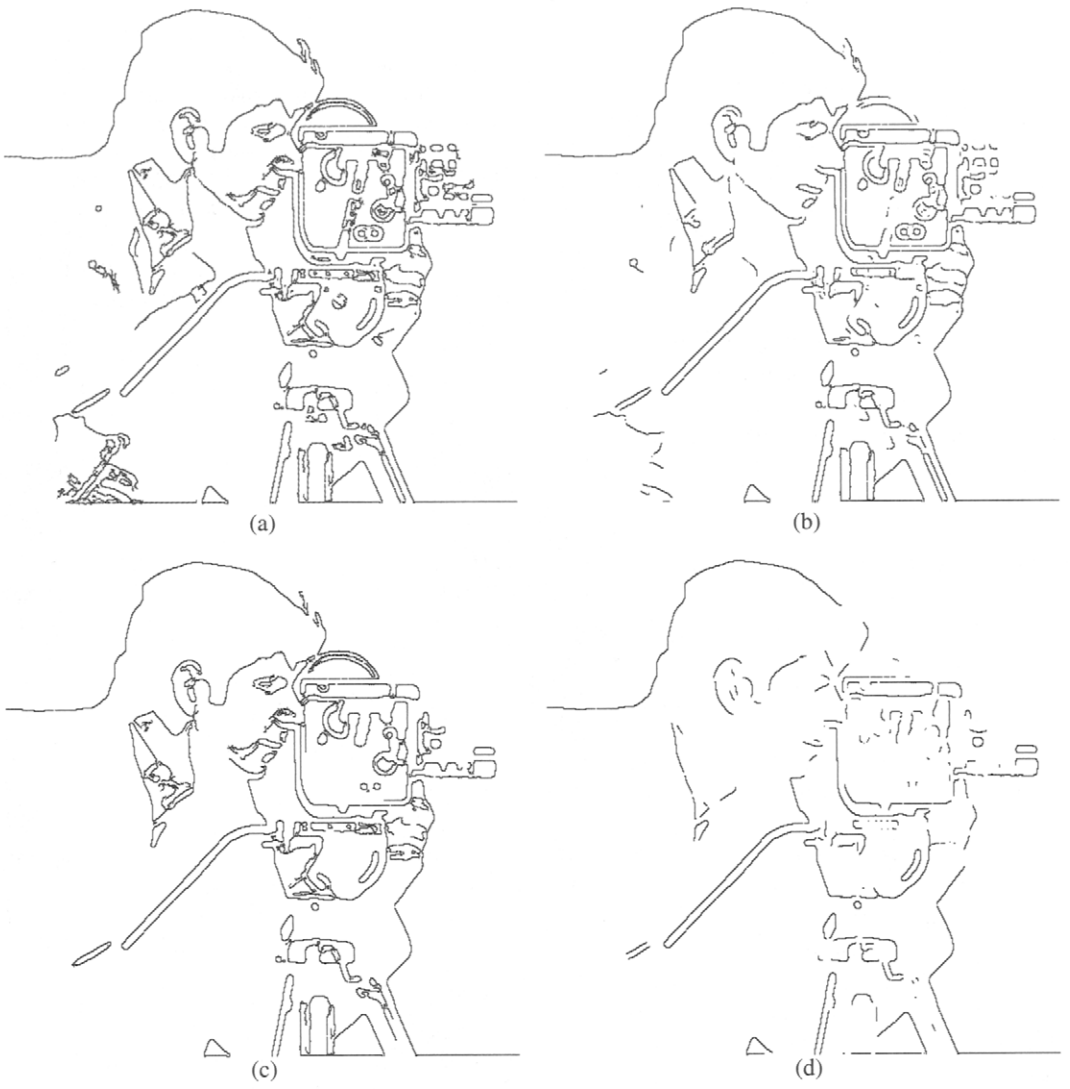

FIGURE 11 Canny edge detector of Eq. (22) applied after Gaussian smoothing with $\sigma = 2$: (a) $T_U = 10$, $T_L = 1$; (b) $T_U = T_L = 10$; (c) $T_U = 20$, $T_L = 1$; (d) $T_U = T_L = 20$. As T_L is changed, notice the effect on the results of hysteresis thresholding.

been Gaussian-smoothed, and then finds the edges in that image. The method generally works well because multispectral images tend to have high correlation between components. However, it is possible for edge information to diminish or vanish if the components destructively interfere.

Cumani [8] explored operators for computing the vector gradient and created an edge detection approach based on combining the component gradients. A multispectral contrast function is defined, and the image is searched for pixels having maximal directional contrast. Cumani's method does not always detect edges present in the component bands, but it better avoids the problem of destructive interference between bands.

The maximal gradient method constructs a single gradient image from the component images [6]. The overall gradient image's magnitude and direction values at a given pixel are those of the component having the greatest gradient magnitude at that pixel. Some edges can be missed by the maximal gradient technique because they may be swamped by differently oriented, stronger edges present in another band.

The method of combining component edge maps is the least efficient because an edge map must first be computed for every band. On the positive side, this method is capable of detecting any edge that is detectable in at least one component image. Combination of component edge maps into a single result is made more difficult by the edge location errors induced by Gaussian smoothing done in advance. The superimposed edges can become smeared in width because of the accumulated uncertainty in edge localization. A thinning step applied during the combination procedure can greatly reduce this edge blurring problem.

6 Summary

Gray-level edge detection is most commonly performed by convolving an image, f, with a filter that is somehow based on the idea of the derivative. Conceptually, edges can be revealed by locating either the local extrema of the first derivative of f or zero crossings of its second derivative. The gradient and the Laplacian are the primary derivative-based functions used to construct such edge-detection filters. The gradient, ∇, is a 2D extension of the first derivative while the Laplacian, ∇^2, acts as a 2D second derivative. A variety of edge detection algorithms and techniques have been developed that are based on the gradient or Laplacian in some way. Like any type of derivative-based filter, ones based on these two functions tend to be very sensitive to noise. Edge location errors, false edges, and broken or missing edge segments are often problems with edge detection applied to noisy images. For gradient techniques, thresholding is a common way to suppress noise and can be done adaptively for better results. Gaussian smoothing is also very helpful for noise suppression, especially when second-derivative methods such as the Laplacian are used. The Laplacian of Gaussian approach can also provide edge information over a range of scales, helping to further improve detection accuracy and noise suppression as well as providing clues that may be useful during subsequent processing.

Recent comparisons of various edge detectors have been made by Heath et al. [13] and Bowyeret al. [4]. They have concluded that the subjective quality of the results of various edge detectors applied to real images is quite dependent on the images themselves. Thus, there is no single edge detector that produces a consistently best overall result. Furthermore, they found it difficult to predict the best choice of edge detector for a given situation.

References

[1] D. H. Ballard and C. M. Brown, *Computer Vision* (Prentice-Hall, Englewood Cliffs, NJ, 1982).

[2] V. Berzins, "Accuracy of Laplacian edge detectors," *Comput. Vis. Graph. Image Proc.*, 27, 195–210 (1984).

[3] A. C. Bovik, T. S. Huang, and D. C. Munson Jr., "The effect of median filtering on edge estimation and detection," *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-9, 181–194 (Mar. 1987).

[4] K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical ROC curves," *Computer Vision and Image Understanding*, 84, 77–103 (2001).

[5] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intel.*, PAMI-8, 679–698 (Nov. 1986).

[6] M. Cebrián, M. Perez-Luque, and G. Cisneros, "Edge detection alternatives for multispectral remote sensing images," in *Proceedings of the 8th Scandinavian Conference on Image Analysis* (NOBIM-Norwegian Soc. Image Pross & Pattern Recognition, Tromso, Norway, 1993) 2, 1047–1054.

[7] J. S. Chen, A. Huertas, and G. Medioni, "Very fast convolution with Laplacian-of-Gaussian masks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 1986), 293–298.

[8] A. Cumani, "Edge detection in multispectral images," *Comput. Vis. Graph. Image Proc. Graph. Models Image Process.* 53, 40–51, (Jan. 1991).

[9] L. Ding and A. Goshtasby, "On the Canny edge detector," *Pattern Recognition*, 34, 721–725 (2001).

[10] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision* (Addison-Wesley, Reading, MA, 1992), Vol 1.

[11] Q. Ji and R. M. Haralick, "Efficient facet edge detection and quantitative performance evaluation," *Pattern Recognition*, 35, 689–700 (2002).

[12] R. C. Hardie and C. G. Boncelet, "Gradient-based edge detection using nonlinear edge enhancing prefilters," *IEEE Trans. Image Proc.*, 4, 1572–1577 (Nov. 1995).

[13] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of edge detectors, a methodology and initial study," *Computer Vision and Image Understanding* 69:1, 38–54 (Jan. 1998).

[14] A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks," *IEEE Trans. Patt. Anal. Machine Intel.*, PAMI-8:5, 651–664 (1986).

[15] S. R. Gunn, "On the discrete representation of the Laplacian of Gaussian," *Pattern Recognition*, 32, 1463–1472 (1999).

[16] A. K. Jain, *Fundamentals of Digital Image Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[17] J. S. Lim, *Two-Dimensional Signal and Image Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1990).

[18] D. Marr, *Vision* (W. H. Freeman, New York, 1982).

[19] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Roy. Soc. London B* 270, 187–217 (1980).

[20] B. Mathieu, P. Melchior, A. Oustaloup, and Ch. Ceyral, "Fractional differentiation for edge detection," *Signal Processing*, 83, 2421–2432 (2003).

[21] J. Merron and M. Brady, "Isotropic gradient estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 1996) 652–659.

[22] V. S. Nalwa and T. O. Binford, "On detecting edges," *IEEE Trans. on Pattern Anal. Machine Intel.*, PAMI-8:6, 699–714 (Nov. 1986).

[23] W. K. Pratt, *Digital Image Processing*, 2nd ed. (Wiley, New York, 1991).

[24] S. J. Sangwine and R. E. N. Horne, eds., *The Colour Image Processing Handbook*, (Chapman and Hall, London, 1998).

[25] S. Sarkar and K. L. Boyer, "Optimal infinite impulse response zero crossing based edge detectors," *Comput. Vis. Graph. Image Proc.: Image Understanding*, 54:2, 224–243 (Sept. 1991).

[26] J. Scharcanski and A. N. Venetsanopoulos, "Edge detection of color images using directional operators," *IEEE Trans. on Circuits and Systems for Video Technology*, 7:2, 397–401 (April 1997).

[27] P. Siohan, D. Pele, and V. Ouvrard, "Two design techniques for 2D FIR LoG filters," in *Visual Communications and Image Processing*, M. Kunt, ed., *Proc. SPIE*, 1360, 970–981 (1990).

[28] V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Pattern Anal. Machine Intel.*, PAMI-8:2, 147–163 (March 1986).

[29] P. E. Trahanias and A. N. Venetsanopoulos, "Color edge detection using vector order statistics," *IEEE Trans. Image Proc.*, 2:2, 259–264 (April 1993).

[30] A. P. Witkin, "Scale-space filtering," in *Proceedings of the International Joint Conference on Artificial Intelligence* (William Kaufmann Inc., Karlsruhe, Germany, 1983), 1019–1022.

[31] D. Ziou, and S. Wang, "Isotropic processing for gradient estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 1996), 660–665.