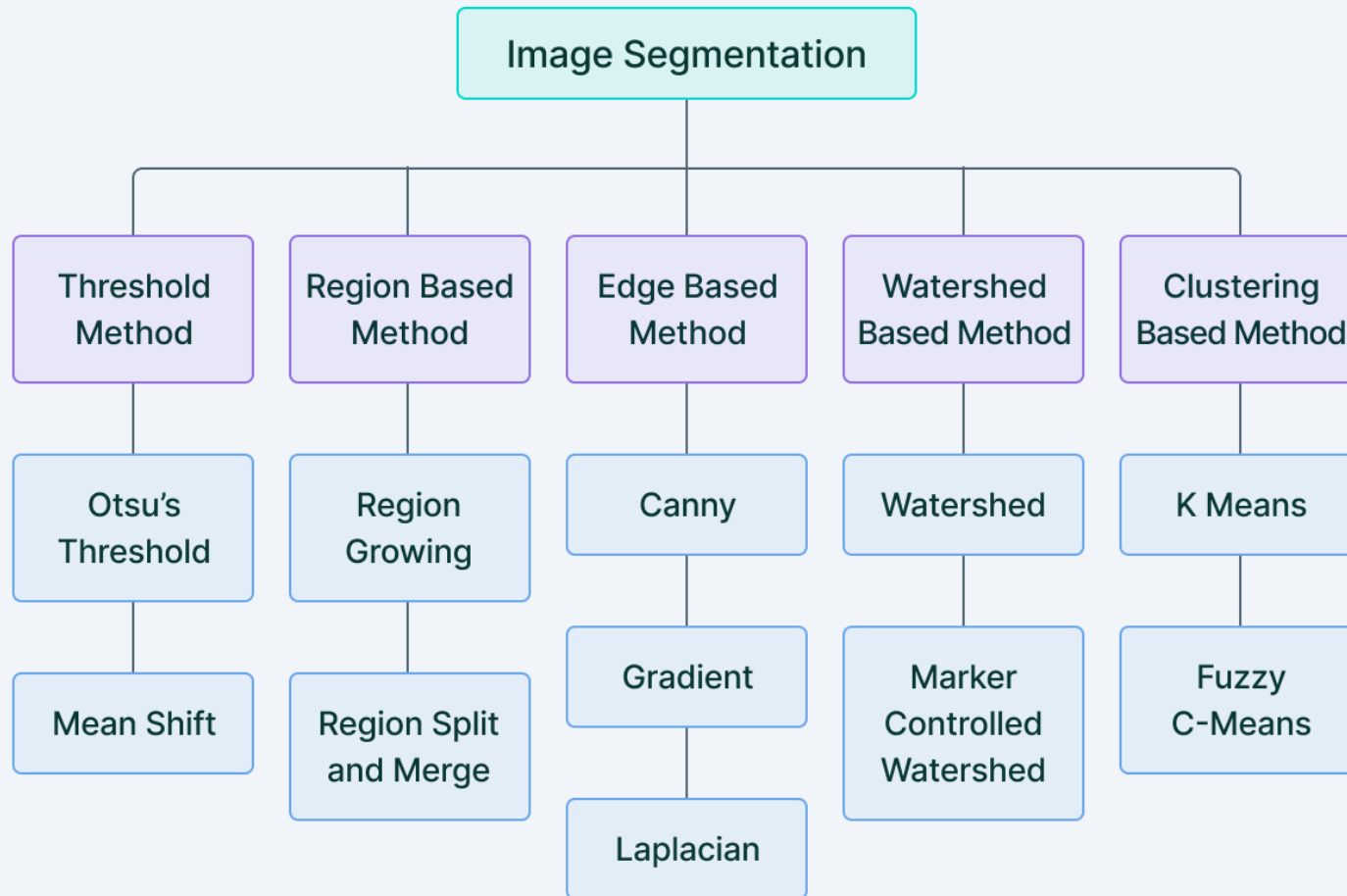


Computer Vision

Course 9



Fundamentals

Let R represent the entire spatial region occupied by an image. Image segmentation can be viewed as a process that partitions R into n subregions R_1, R_2, \dots, R_n such that:

(a)
$$\bigcup_{i=1}^n R_i = R$$

(b) R_i is a connected set, $i = 1, 2, \dots, n$

(c) $R_i \cap R_j = \emptyset$ for all i and j , $i \neq j$

(d) $Q(R_i) = \text{TRUE}$ for $i = 1, 2, \dots, n$

(e) $Q(R_i \cup R_j) = FALSE$ for any adjacent regions R_i , R_j

$Q(R)$ is a logical predicate defined over the points in set R .

Condition (a) indicates that the segmentation must be complete; that every pixel must be in a region. Condition (b) requires that points in a region be connected in some predefined sense (e.g. the points must be **4-** or **8-**connected).

Condition (c) indicates that the regions must be disjoint.

Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region (for example $Q(R_i)$ is true

if all pixels have the same intensity level). Finally, condition (e) indicates that two adjacent regions R_i and R_j must be different in the sense of predicate Q .

The fundamental problem in segmentation is to partition an image into regions that satisfy the preceding conditions. Segmentation algorithms for monochrome images generally are based on one of two basic categories dealing with properties of intensity values: discontinuity and similarity. In the first category, the assumption is that boundaries of regions

are sufficiently different from each other and from the background to allow boundary detection based on local discontinuities in intensity.

Edge-based segmentation is the principal approach used in this category. *Region-based segmentation* approaches from the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria.

Computer Vision

Course 9

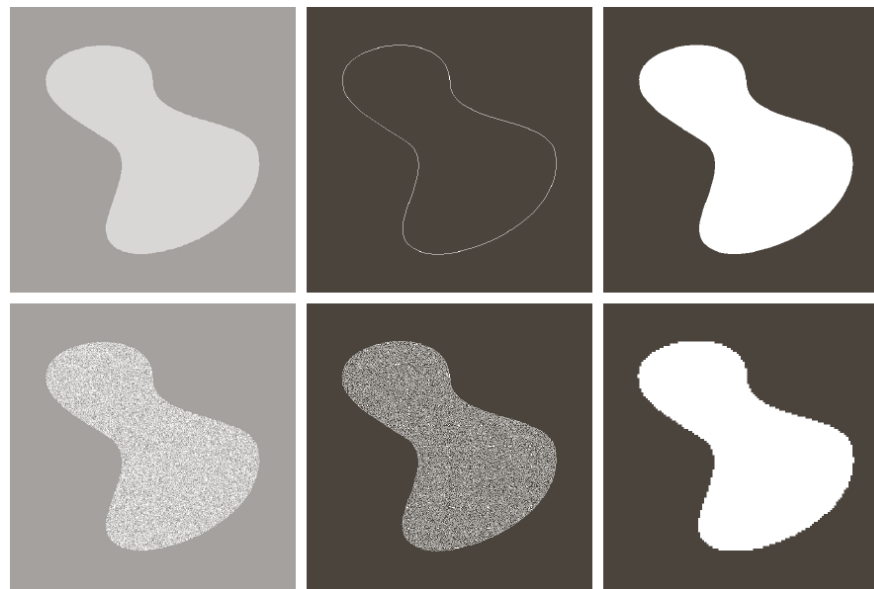


FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

Point, Line, and Edge Detection

Edge pixels are pixels at which the intensity of an image function changes abruptly, and *edges* (or *edge segments*) are sets of connected edge pixels. *Edge detectors* are local image processing methods designed to detect edge pixels. A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.

Background

Abrupt, local changes in intensity can be detected using derivatives, usually first- and second-order derivatives which are defined in terms of differences.

Any approximation for a first derivative must be:

- (1) zero in areas of constant intensity
- (2) non-zero at the onset of an intensity step or ramp
- (3) non-zero at points along an intensity ramp.

An approximation for a second derivative must be:

- (1) zero in areas of constant intensity
- (2) nonzero at the onset and end of an intensity step or ramp
- (3) zero along intensity ramps.

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x, y)$$

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{\partial}{\partial x} \left(\frac{\partial f}{\partial x}(x+1, y) - \frac{\partial f}{\partial x}(x, y) \right) \approx f(x+2) - 2f(x+1) + f(x)$$

Computer Vision

Course 9

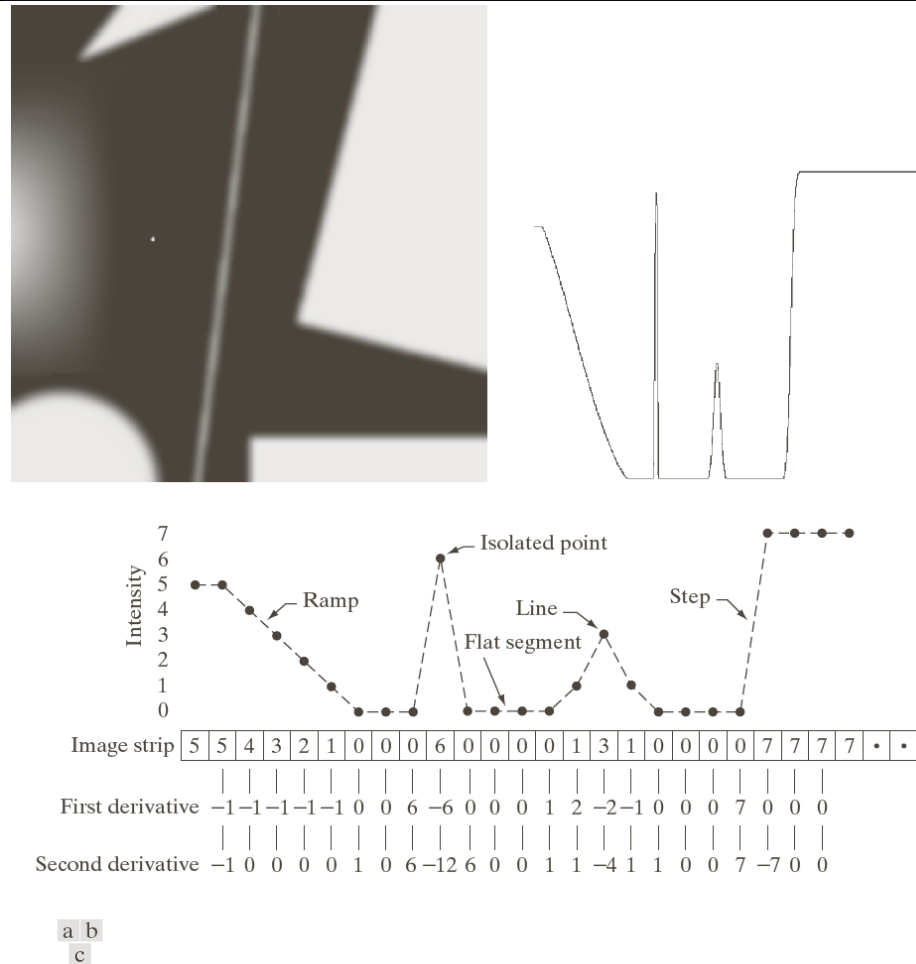


FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

- (a) First-order derivatives generally produce thicker edges in an image
- (b) Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise
- (c) Second-order derivatives produce a double-edge response at ramp and step transitions in intensity
- (d) The sign of the second-order derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

Computing first and second derivatives at every pixel location is done using spatial filters.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.3
A general 3×3
spatial filter mask.

The *response* of the mask at the center point of the region is:

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^9 w_k z_k \quad (1)$$

where z_k is the intensity of the pixel whose spatial location corresponds to the location of the k -th coefficient in the mask.

Detection of isolated points

Point detection is based on computation of the second derivative of the image.

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \text{the Laplacian}$$

$$\frac{\partial^2 f}{\partial x^2} \approx f(x+1, y) + f(x-1, y) - 2f(x, y)$$

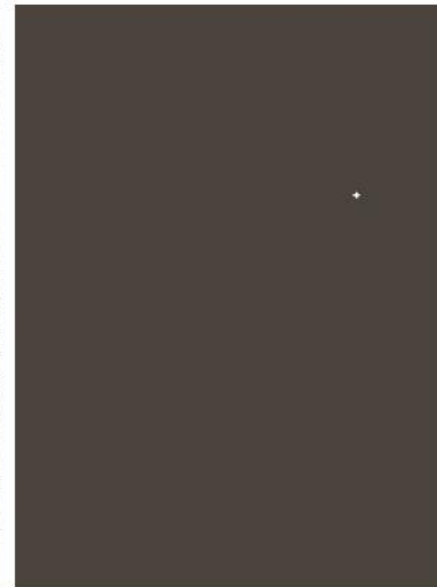
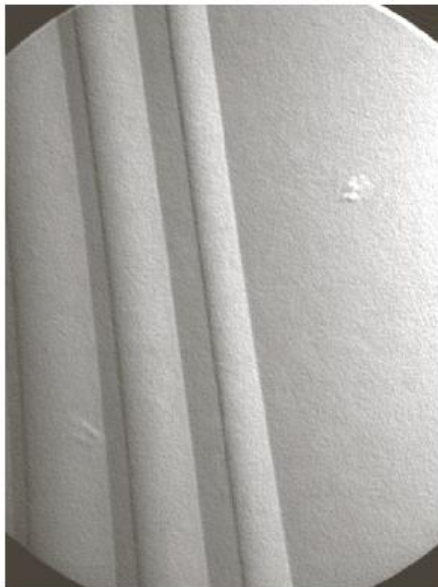
$$\frac{\partial^2 f}{\partial y^2} \approx f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\begin{aligned} \nabla^2 f(x, y) \approx & f(x+1, y) + f(x-1, y) + f(x, y+1) \\ & + f(x, y-1) - 4f(x, y) \end{aligned}$$

Computer Vision

Course 9

1	1	1
1	-8	1
1	1	1



a
b c d

FIGURE 10.4

(a) Point detection (Laplacian) mask.
(b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel.
(c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

Using the Laplacian mask in Figure 10.4(a) we say that a point has been detected at the location (x, y) on which the mask is centred if the absolute value of the response of the mask at that point exceeds a specified threshold. Such points are labelled 1 in the output image and all others are labelled 0 , thus producing a binary image. The output is obtained using the following expression:

$$g(x, y) = \begin{cases} \mathbf{1} & \text{if } |R(x, y)| \geq T \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where g is the output image, $T > 0$ is the threshold, and R is given by (1). This formulation measures the weighted difference between a pixel and its 8-neighbors. The idea is that the intensity of an isolated point will be quite different from its surroundings and thus will be easily detectable by this type of mask. The only differences in intensity that are considered of interest are those large enough (as determined

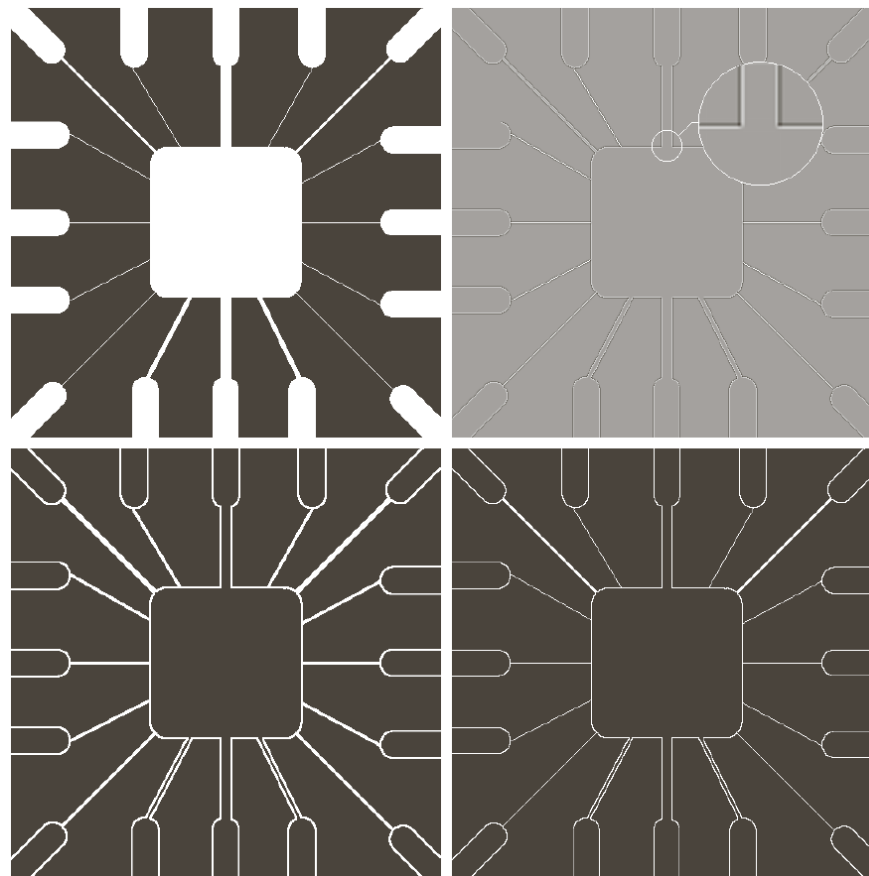
by T) to be considered isolated points. The sum of the coefficients of the mask is zero, indicating that the mask response will be zero in areas of constant intensity.

Line Detection

For line detection, we can expect second derivatives to result in a stronger response and to produce thinner lines than first derivatives. We can use the Laplacian mask in Figure 10.4(a) for line detection also, taking care of the double-line effect of the second order derivative.

Figure 10.5(a) shows a 486×486 (binary) portion of a wire-bond mask for an electronic circuit and Figure 10.5(b) shows its Laplacian. Scaling is necessary in this case (the Laplacian image contains negative values). Mid grey represents 0, darker shades of grey represent negative values, and lighter shades are positive. It might appear that negative values can be handled simply by taking the absolute value of the Laplacian image. Figure 10.5(c) shows that this approach doubles the thickness of the lines. A more suitable approach

is to use only the positive values of the Laplacian (Figure 10.5(d)).



a b
c d

FIGURE 10.5
(a) Original image.
(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.
(c) Absolute value of the Laplacian.
(d) Positive values of the Laplacian.

The Laplacian detector in Figure 10.4(a) is isotropic, so its response is independent of the direction (with respect to the four directions of the 3×3 Laplacian mask: vertical, horizontal, and two diagonals). Often, interest lies in detecting lines in *specified* directions.

Consider the masks in Figure 10.6. Suppose that an image with a constant background and containing various lines (oriented at 0° , $\pm 45^\circ$ and 90°) is filtered with the first mask. The maximum responses would occur at image locations in

which horizontal lines passed through the middle row of the mask.

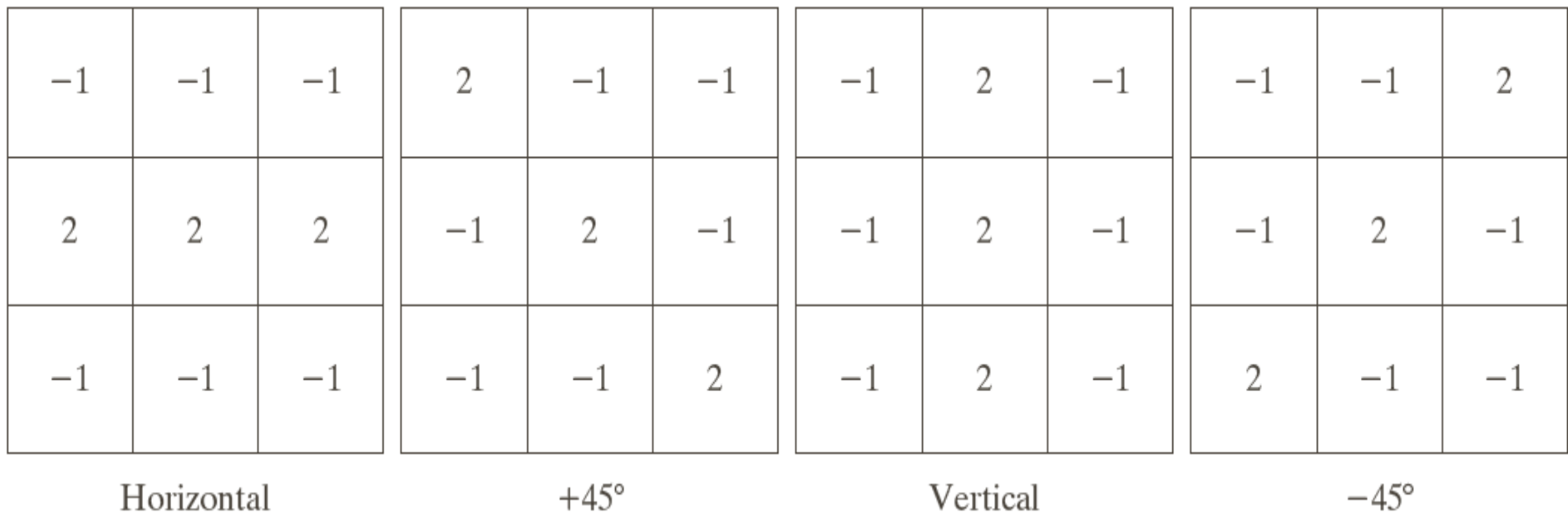


FIGURE 10.6 Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

A similar experiment would reveal that the second mask in Figure 10.6 responds best to lines oriented $+45^\circ$; the third mask to vertical lines; and the fourth mask to lines in the -45° direction.

Let \mathbf{R}_1 , \mathbf{R}_2 , \mathbf{R}_3 and \mathbf{R}_4 denote the response of the masks in Figure 10.6 from left to right, where the \mathbf{R} s are given by (1). Suppose that an image is filtered (individually) with the four masks. If at a given point in the image $|\mathbf{R}_k| > |\mathbf{R}_j|$, for all $j \neq k$,

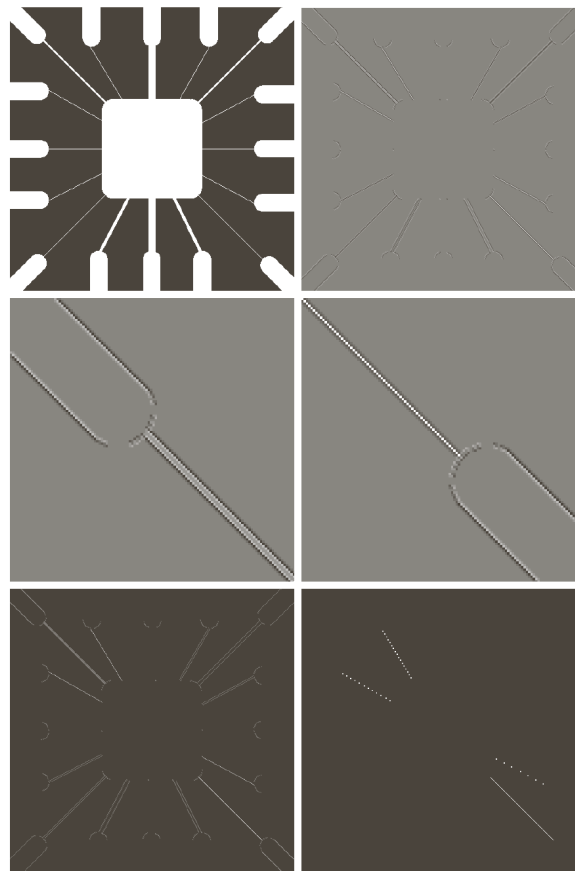
that point is said to be more likely associated with a line in the direction of mask k .

If we are interested in detecting all the lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result. The points that are left are the strongest responses which, for line l pixels thick, correspond closest to the direction defined by the mask.

Computer Vision

Course 9

In Figure 10.7(a) image, we are interested in lines oriented at $+45^\circ$. We use the second mask, the result is in Figure 10.7(b).



a b
c d
e f

FIGURE 10.7
(a) Image of a wire-bond template.
(b) Result of processing with the $+45^\circ$ line detector mask in Fig. 10.6.
(c) Zoomed view of the top left region of (b).
(d) Zoomed view of the bottom right region of (b).
(e) The image in (b) with all negative values set to zero.
(f) All points (in white) whose values satisfied the condition $g \geq T$, where g is the image in (e). (The points in (f) were enlarged to make them easier to see.)

Edge Models

Edge detection is the approach used most frequently for segmenting images based on abrupt (local) changes in intensity.

Edge models are classified according to their intensity profiles. A *step edge* involves a transition between two intensity levels occurring ideally over the distance of 1 pixel. Figure 10.8(a) shows a section of a vertical step edge and a horizontal intensity profile through the edge.

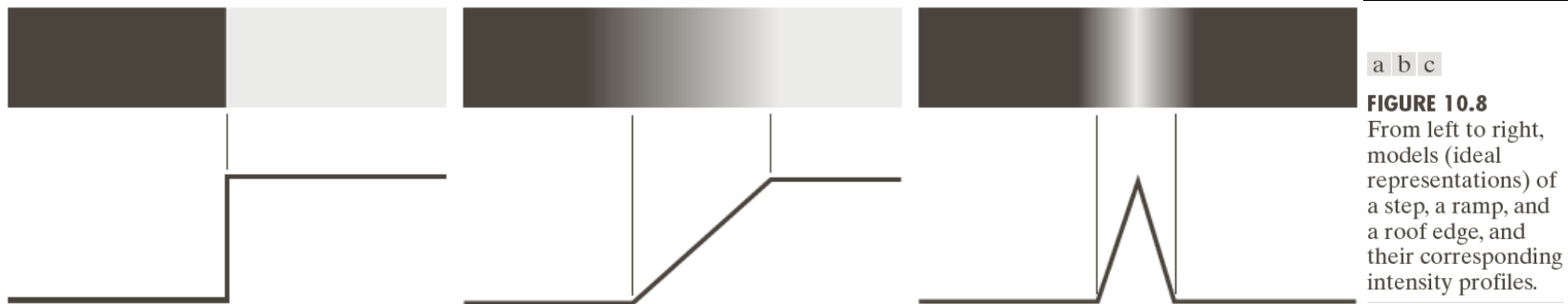


FIGURE 10.8

From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

In practice, digital images have edges that are blurred and noisy, with the degree of blurring determined principally by limitations in the focusing mechanism, and the noise level determined principally by the electronic components of the imaging system. In such situations, edges are more closely

modelled as having an intensity *ramp* profile, such as the edge in Figure 10.8(b). The slope of the ramp is inversely proportional to the degree of blurring in the edge. In this model, we no longer have a thin (*1* pixel thick) path. An edge point now is any point contained in the ramp and an edge segment would then be a set of such points that are connected.

A third model of an edge is the so-called *roof edge*, having the characteristics illustrated in Figure 10.8(c). Roof edges

are models of lines through a region, with the base (width) of a roof edge being determined by the thickness and sharpness of the line.

It is not unusual to find images that contain all three types of edges.

The *magnitude* of the first derivative can be used to detect the presence of an edge at a point in an image. Similarly, the *sign* of the second derivative can be used to determine whether an

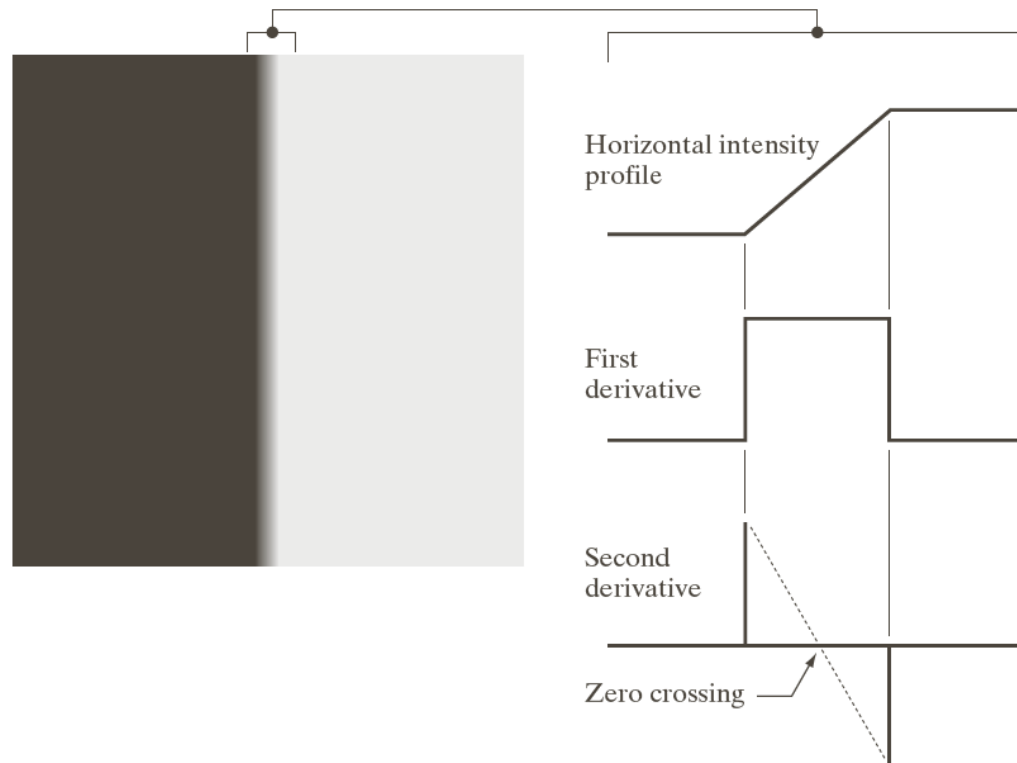
edge pixel lies on the dark or light side of an edge. The second derivative has the following properties:

- (1) it produces two values for every edge in an image (an undesirable feature)
- (2) its zero crossing can be used for locating the centres of thick edges

The *zero crossing* of the second derivative is the intersection between the zero-intensity axis and a line extending between the extrema of the second derivative.

Computer Vision

Course 9



a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge.
(b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

There are three fundamental steps performed in edge detection:

1. *Image smoothing for noise reduction*
2. *Detection of edge points* – this is a local operation that extracts from an image all points that are potential candidates to become edge points
3. *Edge localization* – the objective of this step is to select from the candidate points only the points that are true members of the set of points comprising an edge.

Basic Edge Detection

The image gradient and its properties

The gradient of an image is the tool for finding edge strength and direction at location (x, y) :

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

This vector has the important geometrical property that it points in the direction of the greatest rate of change for f at location (x,y) .

The *magnitude (length)* of vector ∇f

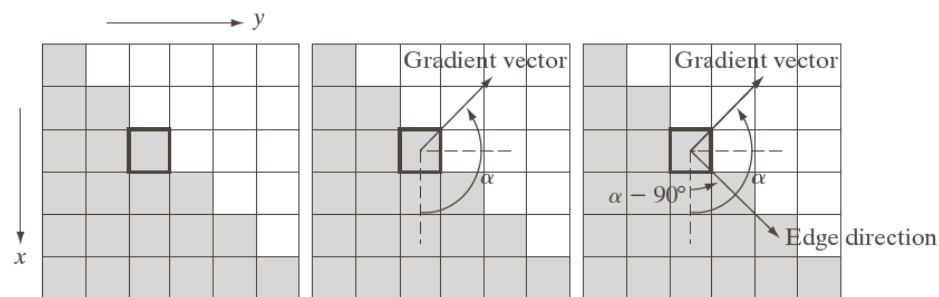
$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

is the value of the rate of change in the direction of the gradient vector.

The *direction* of the gradient vector is given by the angle:

$$\alpha(x, y) = \arctan \left[\frac{g_x}{g_y} \right]$$

measured with respect to the x -axis. The direction of an edge at any arbitrary point (x, y) is *orthogonal* to the direction, $\alpha(x, y)$, of the gradient vector at the point.

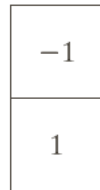


The gradient vector sometimes is called the *edge normal*. When the vector is normalized to unit length (by dividing it by its magnitude) the resulting vector is commonly referred to as the *edge unit normal*.

Gradient operators

$$g_x = \frac{\partial f(x, y)}{\partial x} = f(x + 1, y) - f(x, y)$$

$$g_y = \frac{\partial f(x, y)}{\partial y} = f(x, y + 1) - f(x, y)$$



a b

FIGURE 10.13
One-dimensional masks used to implement Eqs. (10.2-12) and (10.2-13).

When diagonal edge direction is of interest, we need a 2-D mask. The *Roberts cross-gradient operators* are one of the earliest attempts to use 2-D masks with a diagonal preference. Consider the 3×3 region in Figure 10.14(a). The Roberts operators are based on implementing the diagonal differences.

Computer Vision

Course 9

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0	0	-1
0	1	1	0

Roberts

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Prewitt

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Sobel

a
b c
d e
f g

FIGURE 10.14

A 3×3 region of an image (the z 's are intensity values) and various masks used to compute the gradient at the point labeled z_5 .

$$g_x = \frac{\partial f}{\partial x} = z_9 - z_5 = f(x+1, y+1) - f(x, y)$$

$$g_y = \frac{\partial f}{\partial y} = z_8 - z_6 = f(x+1, y) - f(x, y+1)$$

Masks of size 2×2 are simple conceptually, but they are not as useful for computing edge direction as masks that are symmetric about the centre point, the smallest of which are of size 3×3 .

Prewitt operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Sobel operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

The Sobel masks have better noise-suppression (smoothing) effects than the Prewitt masks.

0	1	1	-1	-1	0
-1	0	1	-1	0	1
-1	-1	0	0	1	1

Prewitt

0	1	2	-2	-1	0
-1	0	1	-1	0	1
-2	-1	0	0	1	2

Sobel

a	b
c	d

FIGURE 10.15
Prewitt and Sobel masks for detecting diagonal edges.

Computer Vision

Course 9



a b
c d

FIGURE 10.16

(a) Original image of size 834×1114 pixels, with intensity values scaled to the range $[0, 1]$.
(b) $|g_x|$, the component of the gradient in the x -direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image.
(c) $|g_y|$, obtained using the mask in Fig. 10.14(g).
(d) The gradient image, $|g_x| + |g_y|$.

Computer Vision

Course 9



a b
c d

FIGURE 10.18
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5×5 averaging filter prior to edge detection.



a b

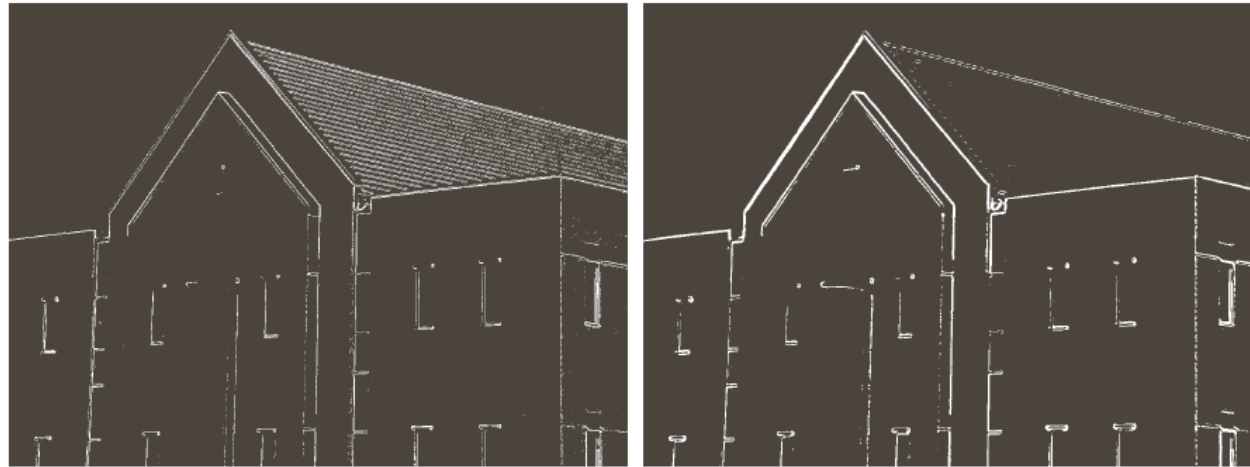
FIGURE 10.19

Diagonal edge detection.

(a) Result of using the mask in Fig. 10.15(c).

(b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

When interest lies both in highlighting the principal edges and on maintaining as much connectivity as possible, it is common practice to use both smoothing and thresholding.



a b

FIGURE 10.20 (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

More Advanced Techniques for Edge Detection

The edge-detection methods described until now are based on filtering an image with one or more masks, without approaching the edge characteristics or the noise content of the image. In this section, the noise and the nature of the edges are considered in more advanced edge-detection techniques.

The Marr-Hildreth edge detector

Marr and Hildreth noticed that:

- (1) intensity changes are dependent of image scale and so their detection requires the use of operators of different sizes;
- (2) a sudden intensity change will give rise to a peak or trough in the first derivative or, equivalently, to a zero crossing in the second derivative.

These ideas suggest that an operator used for edge detection should have two features:

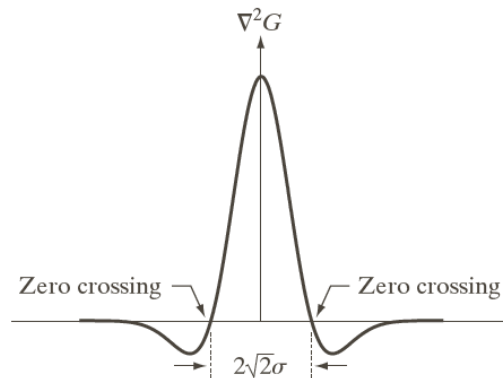
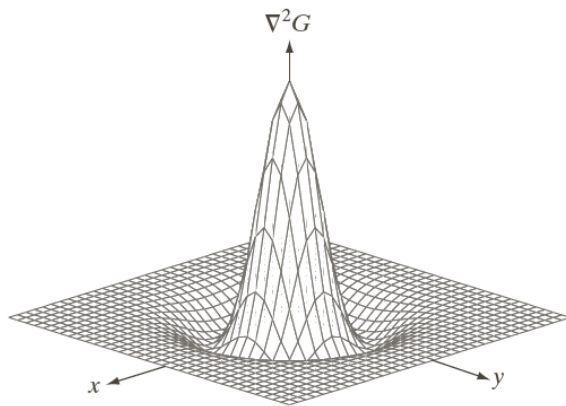
- 1) it should be a differential operator capable of computing a digital approximation of the first or second derivative at every point in the image
- 2) it should be capable of being “tuned” to act at any desired scale, so that large operators can be used to detect blurry edges and small operators to detect sharply focused fine detail.

Marr and Hildreth argued that the most satisfactory operator fulfilling these conditions is the filter $\nabla^2 G$, the Laplacian of G , the 2-D Gaussian function with standard deviation σ :

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

$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3)$$

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



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.

Because of the shape illustrated in Figure 10.21(a), the LoG function sometimes is called the *Mexican hat* operator. Figure 10.21(d) shows a 5×5 mask that approximates the shape in Figure 10.21(a) (in practice, the *negative* of this mask is used). This approximation is not unique. Its purpose is to capture the essential *shape* of the LoG function.

Masks of arbitrary size can be generated by sampling equation (3) and scaling the coefficients so that they sum to zero. A more effective approach for generating LoG filters is

to sample equation (2) to the desired $n \times n$ size and then convolve the resulting array with a Laplacian mask, such as the mask in Figure 10.4 (a).

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

$$g(x, y) = [\nabla^2 G(x, y)] \diamond f(x, y) = \nabla^2 [G(x, y) \diamond f(x, y)]$$

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 equation (2)
2. Compute the Laplacian of the image resulting in Step 1
3. Find the zero crossing of the image from Step 2.

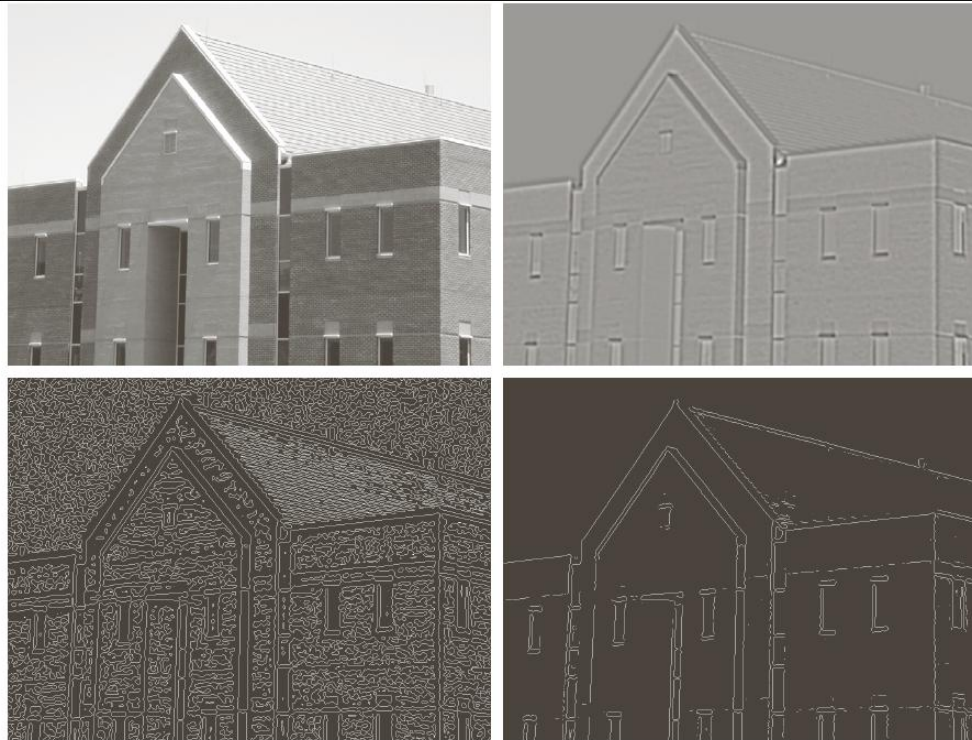
The size of an $n \times n$ LoG discrete filter should be such that n is the smallest odd integer greater than or equal to 6σ . Choosing a filter mask smaller than this will tend to “truncate” the LoG function, with the degree of truncation

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 crossing at any pixel p of the filtered image, $g(x,y)$, is based on using a 3×3 neighbourhood centred at p . A zero crossing at p implies that the *signs* of at least two of its opposing neighbouring pixel must differ. There are 4 cases to test: left/right, up/down, and the two diagonals.

Computer Vision

Course 9



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.

The Canny edge detector

Canny's approach is based on three basic objectives:

1. *Low error rate.* All edges should be found, and there should be no false responses. The detected edges 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 centre of the true edge should be minim.

3. *Single edge 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 minim. This means that the detector should not identify multiple edge pixels where only a single edge point exists.

In general, it is difficult (or impossible) to find a closed form solution that satisfy 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*:

$$\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}.$$

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

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

Form a smoothed, $f_s(x, y)$, by convolving G and f :

$$f_s(x, y) = G(x, y) \diamond f(x, y).$$

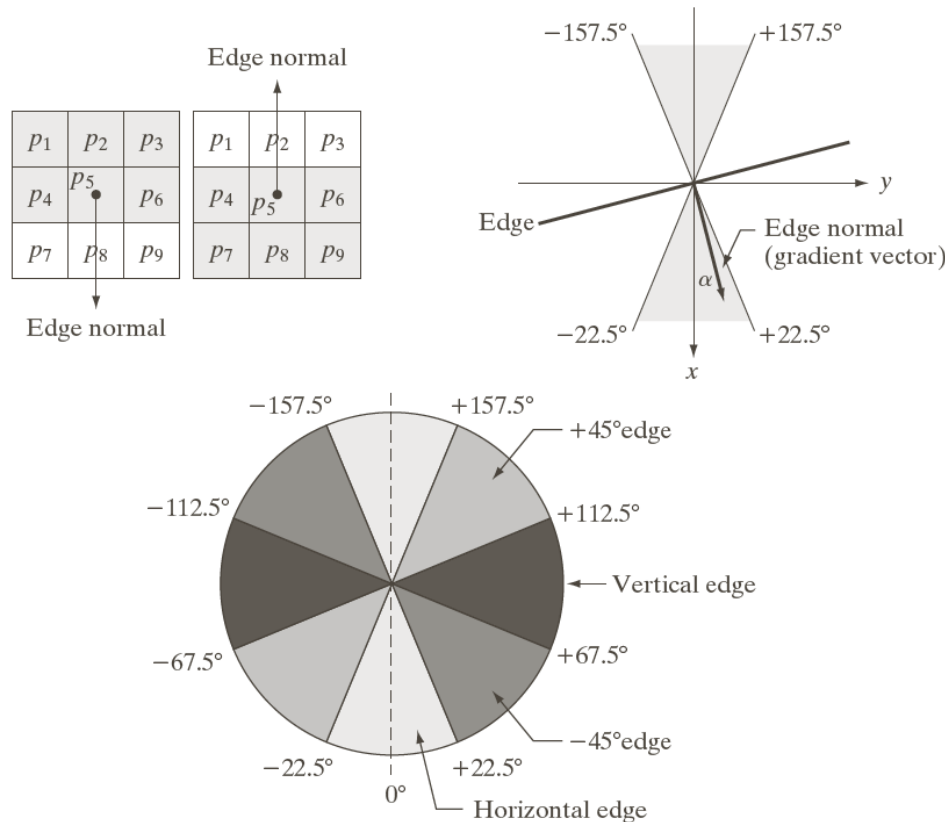
We compute the gradient magnitude and the angle for f_s

$$M(x, y) = \sqrt{g_x^2 + g_y^2}, \quad g_x = \frac{\partial f_s}{\partial x}, \quad g_y = \frac{\partial f_s}{\partial y}$$

$$\alpha(x, y) = \arctan \left[\frac{g_x}{g_y} \right]$$

$M(x, y)$ contains ridges around local maxima. The next step is to thin those ridges. One approach is to use *non-maxima*

suppression. This can be done in several ways, but the essence of this approach is to specify a number of discrete orientations of the edge normal (gradient vector). For example, in a 3×3 region we can define four orientations for an edge passing through the centre point of the image: horizontal, vertical, $+45^\circ$ and -45° .



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.

Let d_1, d_2, d_3 and d_4 denote the four basic edge directions for a 3×3 region: horizontal, -45° , vertical, and $+45^\circ$, respectively.

We can formulate the following non-maxima suppression scheme for a 3×3 region centred 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 neighbours along d_k , let $g_N(x,y)=0$ (suppression); otherwise, let $g_N(x,y)=M(x,y)$, where $g_N(x,y)$ is the non-maxima-suppressed image.

The final operation is to threshold $g_N(x, y)$ to reduce false edge points. If we set the threshold to low, there will still be some false edges (called *false positive*). If the threshold is too high, then actual valid edge points will be eliminated (*false negative*). Canny's algorithm attempts to improve on this situation by using *hysteresis thresholding*, which 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

$$\mathbf{g}_{NH}(\mathbf{x}, \mathbf{y}) = \mathbf{g}_N(\mathbf{x}, \mathbf{y}) \geq T_H$$

$$\mathbf{g}_{NL}(\mathbf{x}, \mathbf{y}) = \mathbf{g}_N(\mathbf{x}, \mathbf{y}) \geq T_L$$

$$\mathbf{g}_{NH} = \mathbf{g}_{NL} = \mathbf{0} \text{ -- initially}$$

After thresholding $\mathbf{g}_{NH}(\mathbf{x}, \mathbf{y})$ will have fewer nonzero pixels than $\mathbf{g}_{NL}(\mathbf{x}, \mathbf{y})$ in general, but all the nonzero pixels in $\mathbf{g}_{NH}(\mathbf{x}, \mathbf{y})$ will be contained in $\mathbf{g}_{NL}(\mathbf{x}, \mathbf{y})$ because the later

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:

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

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

After the thresholding operation, 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 pixels in $g_{NL}(x,y)$ that are connected to p (using δ -connectivity, for example)
- (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 $g_{NL}(x,y)$.

In practice, hysteresis thresholding can be implemented directly during non-maxima 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.

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 non-maxima suppression to the gradient magnitude image.
4. Use double thresholding and connectivity analysis to detect and link edges.

Edge Linking and Boundary Detection

Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edge completely because of noise, breaks in the edges due to nonuniform illumination, and other effects that introduce fake 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. We discuss three fundamental approaches to edge

linking that are representative of techniques used in practice. The first requires knowledge about edge points in a local region; the second requires that points on the boundary of a region be known; the third is a global approach that works with an entire edge image.

Local processing

A simple way to link edge points is to analyze the characteristics of pixels in small neighborhoods about every point (x, y) that has been declared an edge point. All points

that are similar according to predefined criteria are linked, forming an edge of pixels that share common properties according to the specified criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are:

- (1) the strength (magnitude)
- (2) the direction

of the gradient vector. Let S_{xy} denote the set of coordinates of a neighborhood centered at (x, y) in an image. An edge pixel

with coordinates (s, t) in S_{xy} is similar in *magnitude* to the pixel at (x, y) if:

$$|M(s, t) - M(x, y)| \leq E, \quad E > 0 \text{ - positive threshold.}$$

An edge pixel with coordinates (s, t) in S_{xy} has an *angle* similar to the pixel at (x, y) if:

$$|\alpha(s, t) - \alpha(x, y)| \leq A, \quad A > 0 \text{ - positive angle threshold.}$$

The direction of the edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

A pixel with coordinates (s, t) in S_{xy} is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple procedure would be to assign a different intensity value to each set of linked edge pixels.

The preceding formulation is computationally expensive because all neighbors of every point have to be examined. A

simplification particularly well suited for real time applications consists of the following steps:

1. Compute the gradient magnitude and the angle arrays $M(x, y)$ and $\alpha(x, y)$ of the input image $f(x, y)$.
2. Form a binary image g :

$$g(x, y) = \begin{cases} 1 & \text{if } M(x, y) > T_M \text{ AND } \alpha(x, y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

where T_M is a threshold, A is a specified angle direction, and $\pm T_A$ defines a “band” of acceptable directions about A

3. Scan the rows of \mathbf{g} and fill (set to $\mathbf{1}$) all gaps (sets of $\mathbf{0}$ s) in each row that do not exceed a specified length \mathbf{K} . Note that a gap is bounded at both ends by one or more $\mathbf{1}$ s. The rows are processed individually, with no memory between them.
4. To detect gaps in any other direction θ , rotate \mathbf{g} by this angle and apply the horizontal scanning procedure in Step 3. Rotate the result back by $-\theta$.

In general, image rotation is an expensive computational process so, when linking in numerous angle directions is required, it is more practical to combine Steps 3 and 4 into a single radial scanning procedure.

Figure 10.27(a) shows an image of the rear of a vehicle. The objective of this example is to illustrate the use of the preceding algorithm for finding rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges.

Computer Vision

Course 9



a	b	c
d	e	f

FIGURE 10.27 (a) A 534×566 image of the rear of a vehicle. (b) Gradient magnitude image. (c) Horizontally connected edge pixels. (d) Vertically connected edge pixels. (e) The logical OR of the two preceding images. (f) Final result obtained using morphological thinning. (Original image courtesy of Perceptics Corporation.)

$T_M = 30\%$ of the maximum gradient value, $A = 90^\circ$, $T_A = 45^\circ$,
 $K = 25$.

Regional processing

Often the location of the regions of interest in an image is known or can be determined. This implies that knowledge is available regarding the regional membership of pixels in the corresponding edge image. We can use techniques for linking pixels on a regional basis, with the desired result being an approximation to the boundary of the region. One approach is to fit a 2-D curve to the known points. Interest lies in fast-executing techniques that yield an approximation to

essential features of the boundary, such as extreme points and concavities. Polygonal approximations are particularly attractive because they capture the essential shape features of a region while keeping the representation of the boundary relatively simple. We present an algorithm suitable for this purpose.

Two important requirements are necessary. First, two starting points must be specified; second, all the points must be ordered (e.g. in a clockwise or counter clockwise direction).

An algorithm for finding a polygonal fit to open and closed curves may be stated as follows:

1. Let P be a sequence of ordered, distinct, I -valued points of a binary image. Specify two starting points, A and B . These are the two starting vertices of the polygon.
2. Specify a threshold T , and two empty stacks OPEN and CLOSED.

3. If the points in P correspond to a closed curve, put A into OPEN and put B into OPEN *and* into CLOSED. If the points correspond to an open curve, put A into OPEN and put B into CLOSED.
4. Compute the parameters of the line passing from the last vertex in CLOSED to the last vertex in OPEN.
5. Compute the distance from the line in Step 4 to all points in P whose sequence places them between the

vertices from Step 4. Select the point V_{max} with the maximum distance D_{max} (ties are resolved arbitrarily)

6. If $D_{max} > T$, place V_{max} at the end of the OPEN stack as a new vertex. Go to step 4.
7. Else, remove the last vertex from OPEN and insert it as the last vertex in CLOSED.
8. If OPEN is not empty go to Step 4.

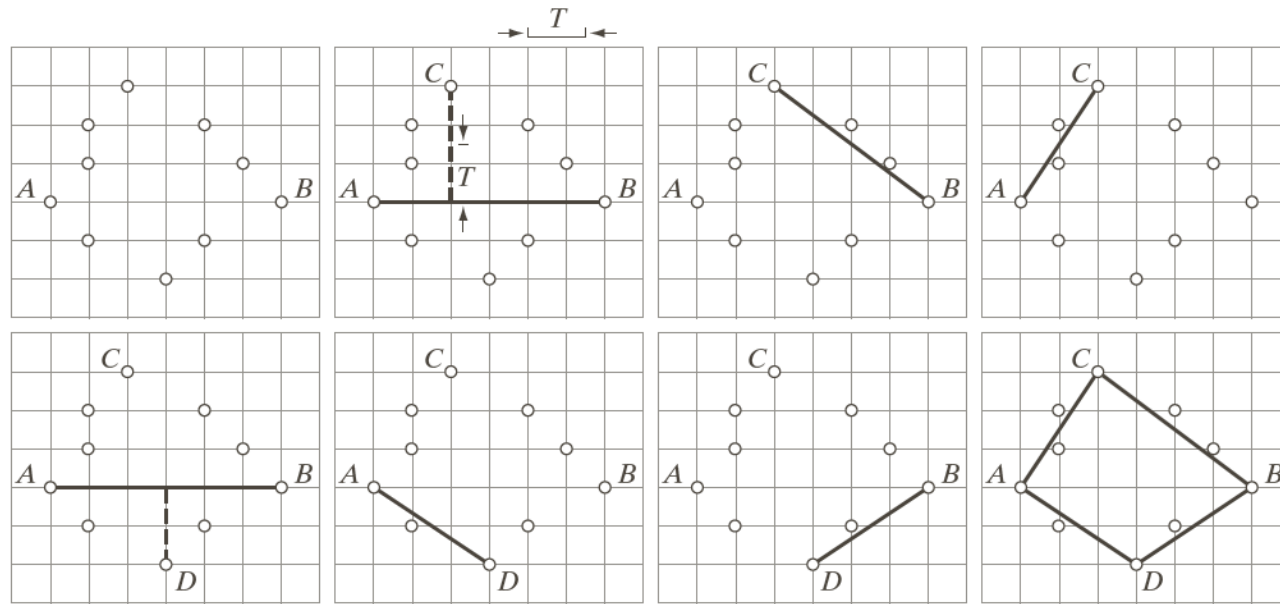
9. Else, exit. The vertices in CLOSED are the vertices of the polygonal fit to the points in P .

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—
B, C, A, D	B	(DB)	—
B, C, A, D, B	Empty	—	—

TABLE 10.1
Step-by-step details of the mechanics in Example 10.11.

Computer Vision

Course 9



a	b	c	d
e	f	g	h

FIGURE 10.29 (a) A set of points in a clockwise path (the points labeled *A* and *B* were chosen as the starting vertices). (b) The distance from point *C* to the line passing through *A* and *B* is the largest of all the points between *A* and *B* and also passed the threshold, so *C* is a new vertex. (d)–(g) Various stages of the algorithm. (h) The final vertices, shown connected with straight lines to form a polygon. Table 10.1 shows step-by-step details.

Global processing using the Hough transform

The previous methods assumed available knowledge about pixels belonging to individual objects. Often, we work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be. In such situations, all pixels are candidates for linking and thus have to be accepted or eliminated based on predefined *global* properties. The technique approach in this section is based on whether set of pixels lie on curves of a

specified shape. Once detected, these curves form the edges or region boundaries of interest.

Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to find first of all lines determined by every pair of points and then find all subsets of points that are close to particular lines. This approach involves finding $\frac{n(n-1)}{2} \sim n^2$

lines and then performing $n \frac{n(n-1)}{2} \sim n^3$ comparisons of

every point to all lines. This is a computationally prohibitive task.

Hough proposed an alternative approach, commonly referred to as *Hough transform*. Consider a point (x_i, y_i) in the xy -plane and the general equation of a line that passes through this point:

$$y_i = a x_i + b$$

Infinitely many lines pass through (x_i, y_i) , but they all satisfy the equation $y_i = a x_i + b$ for varying values of a and b .

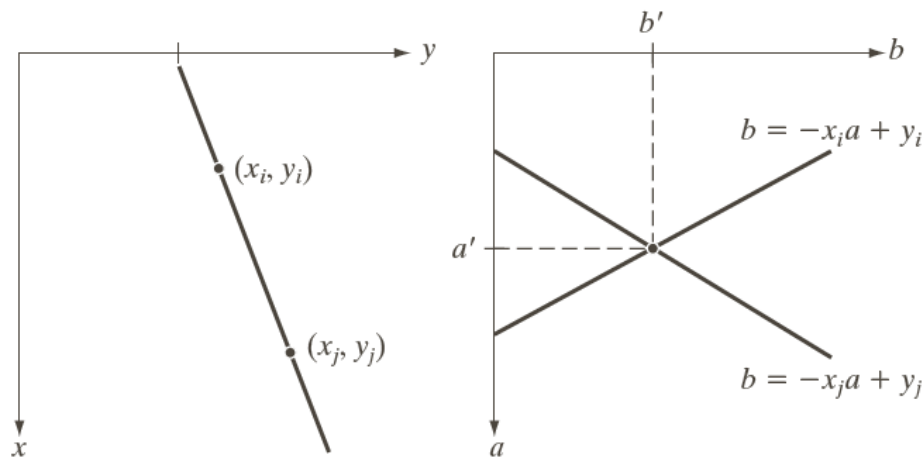
However, writing this equation as:

$$b = -x_i a + y_i$$

And considering the ab -plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i) .

Furthermore, a second point (x_j, y_j) also has a line in the parameter space associated with it, and, unless they are

parallel, this line intersects the line associated with (x_i, y_i) at some point (a', b') . In fact, all the points on this line have lines in parameter space that intersect at (a', b') .



a b

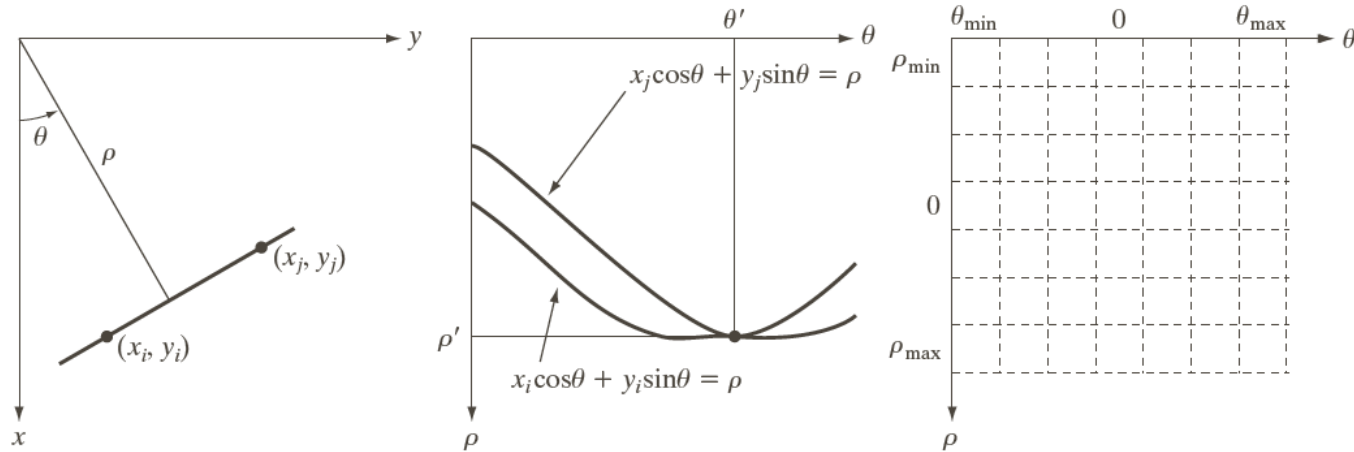
FIGURE 10.31
(a) xy -plane.
(b) Parameter space.

In principle, the parameter-space lines corresponding to all points (x_k, y_k) in the xy -plane could be plotted, and the principal lines in that plane could be found by identifying points in parameter space where large numbers of parameter space lines intersect. A practical difficulty with this approach, however, is that a tends to infinity as the lines approach the vertical direction. To solve this inconvenience, we use the normal representation of a line:

$$x \cos \theta + y \sin \theta = \rho$$

Computer Vision

Course 9



a b c

FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy -plane. (b) Sinusoidal curves in the $\rho\theta$ -plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy -plane. (c) Division of the $\rho\theta$ -plane into accumulator cells.

The computational attractiveness of the Hough transform arises from subdividing the $\rho\theta$ parameter space into so called *accumulator cells*, as Figure 10.32(c) illustrates, where $(\rho_{\min}, \rho_{\max})$ and $(\theta_{\min}, \theta_{\max})$ are the expected ranges of the parameter values: $-90^\circ \leq \theta \leq 90^\circ$ and $-D \leq \rho \leq D$, where D is the maximum distance between opposite corners in an image. The cell at coordinates (i, j) with accumulator value $A(i, j)$, corresponds to the square associated with parameter-space coordinates (ρ_i, θ_j) . In initially, these cells are set to zero.

Then, for every non-background point (x_k, y_k) in the xy -plane, we let θ equal each of the allowed subdivision values on the θ -axis and solve for the corresponding ρ using the equation $\rho = x_k \cos \theta + y_k \sin \theta$. The resulting ρ values are then rounded off to the nearest allowed cell value along the ρ axis. If a choice of θ_p results in solution ρ_q , then we let

$$A(p, q) = A(p, q) + 1.$$

At the end of this procedure, a value of P in $A(i, j)$ means that P points in the xy -plane lie on the line $x \cos \theta_j + y \sin \theta_j = \rho_i$.

The number of subdivisions in the $\rho\theta$ -plane determines the accuracy of the collinearity of these points. It can be shown that the number of computations in the above described method is linear with respect to n , the number of non-background points in the xy -plane.

An approach based on the Hough transform for edge-linking is as follows:

1. Obtain a *binary* edge image.
2. Specify subdivisions in the $\rho\theta$ -plane.

3. Examine the counts of the accumulator cells for high pixel concentration
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

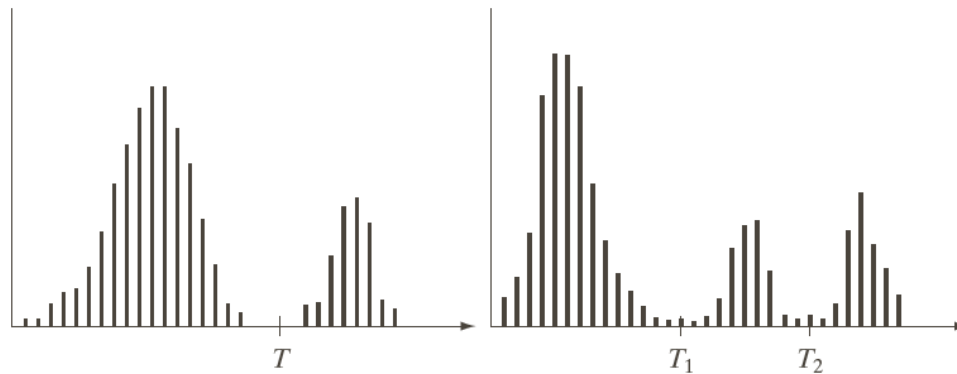
Continuity in this case usually is based on computing the distance between disconnected pixels corresponding to a given accumulator cell. A gap in a line associated with a given cell is bridged if the length of the gap is less than a specified threshold.

Image Segmentation – Thresholding

We discuss techniques for partitioning images directly into regions based on intensity values and/or properties of these values.

Suppose that the intensity histogram in Figure 10.35(a) corresponds to an image, $f(x,y)$, composed of light objects and a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One way to extract the objects

from the background is to select a threshold T that separates these modes.



a b

FIGURE 10.35

Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

Any point (x,y) in the image for which $f(x,y) > T$ is called an *object point*; otherwise, the point is called a

background point. The segmented image, $g(x,y)$, is given by:

$$g(x, y) = \begin{cases} \mathbf{1} & \text{if } f(x, y) > T \\ \mathbf{0} & \text{if } f(x, y) \leq T \end{cases}$$

where T is a constant applicable over an entire image, the process given in this equation is referred to as *global thresholding*. When the value of T changes over an image, we use the term *variable thresholding*. The term *local* or *regional thresholding* is used sometimes to

denote variable thresholding in which the value of T at any point (x, y) in an image depends on properties of a neighborhood of (x, y) (for example, the average intensity of the pixels in the neighborhood). If T depends on the spatial coordinates (x, y) themselves, then variable thresholding is often referred to as *dynamic* or *adaptive* thresholding.

If in an image we have, for example, two types of light objects on a dark background, *multiple thresholding* is used. The segmented image is given by:

$$g(x, y) = \begin{cases} a & \text{if } f(x, y) > T_2 \\ b & \text{if } T_1 < f(x, y) \leq T_2 \\ c & \text{if } f(x, y) \leq T_1 \end{cases}$$

where a , b , and c are any three distinct intensity values.

Segmentation problems requiring more than two

thresholds are difficult to solve, and better results usually are obtained using other methods.

The success of intensity thresholding is directly related to the width and depth of the valley(s) separating the histogram modes. The key factors affecting the properties of the valley(s) are:

- (1) the separation between peaks (the further apart the peaks are, the better the chances of separating the modes)

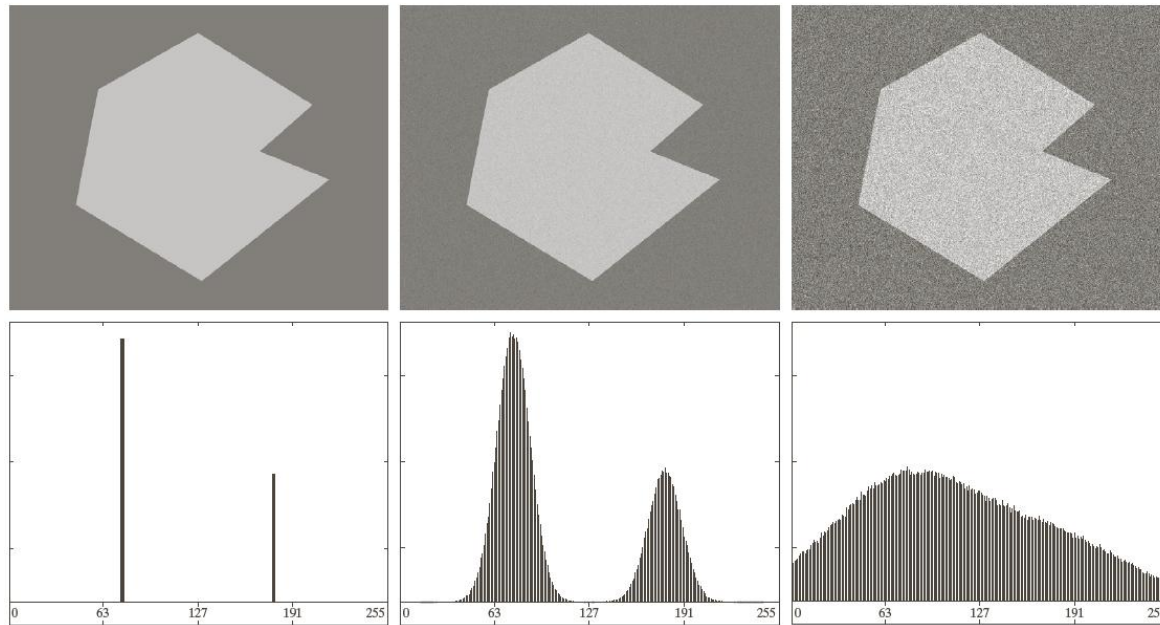
- (2) the noise content in the image (the modes broaden as noise increases)
- (3) the relative size of objects and background
- (4) the uniformity of the illumination source
- (5) the uniformity of the reflectance properties of the image

The role of noise in image thresholding

Consider Figure 10.36(a) – the image is free of noise and its histogram has two “spike” modes. Figure 10.36(b) shows the original image corrupted by Gaussian noise of zero mean and a standard deviation of 10 intensity levels. Although the corresponding histogram modes are now broader (Figure 10.36(e)), their separation is large enough so that the depth of the valley between them is sufficient to make the modes easy to separate.

Computer Vision

Course 9



a	b	c
d	e	f

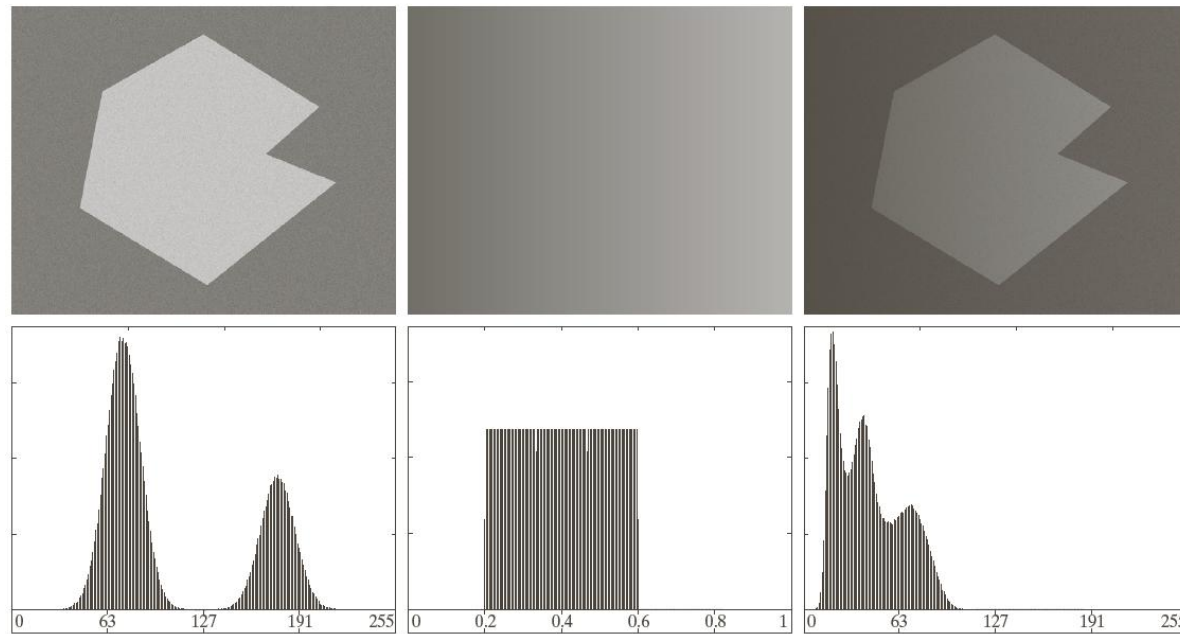
FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Figure 10.36(c) shows the result of corrupting the image with Gaussian noise of zero mean and a standard deviation of 50 intensity levels. As the histogram in Figure 10.36(f) shows, the situation is much more difficult as there is now way to differentiate between two modes.

The role of illumination and reflectance

Figure 10.37 illustrates the effect that illumination can have on the histogram of an image. Figure 10.37(a) is the

noisy image from Figure 10.36(b) and Figure 10.37(d) shows its histogram.



a	b	c
d	e	f

FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range $[0.2, 0.6]$. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

We can illustrate the effects of nonuniform illumination by multiplying the image in Figure 10.37(a) by a variable intensity function, such the intensity ramp in Figure 10.37(b), whose histogram is shown in Figure 10.37(e). Figure 10.37(c) shows the product of the image and this shading pattern. Figure 10.37(f) shows, the deep valley between peaks was corrupted to the point where separation of the modes without additional processing is no longer possible.

Illumination and reflectance play a central role in the success of image segmentation using thresholding or other segmentation techniques. Therefore, controlling these factors when it is possible to do so should be the first step considered in the solution of segmentation problem. There are three basic approaches to the problem when control over these factors is not possible. One is to correct the shading pattern directly. For example, nonuniform (but fixed) illumination can be corrected by

multiplying the image by the inverse pattern, which can be obtained by imaging a flat surface of constant intensity. The second approach is to attempt to correct the global shading pattern via processing it. The third approach is to “work around” nonuniformities using variable thresholding.

Basic Global Thresholding

When the intensity distributions of objects and background pixels are sufficiently distinct, it is possible to use a single (*global*) threshold applicable over the entire image. An algorithm capable of estimating automatically the threshold value for each image is required. The following iterative algorithm can be used for this purpose:

1. Select an initial estimate for the global threshold, T .

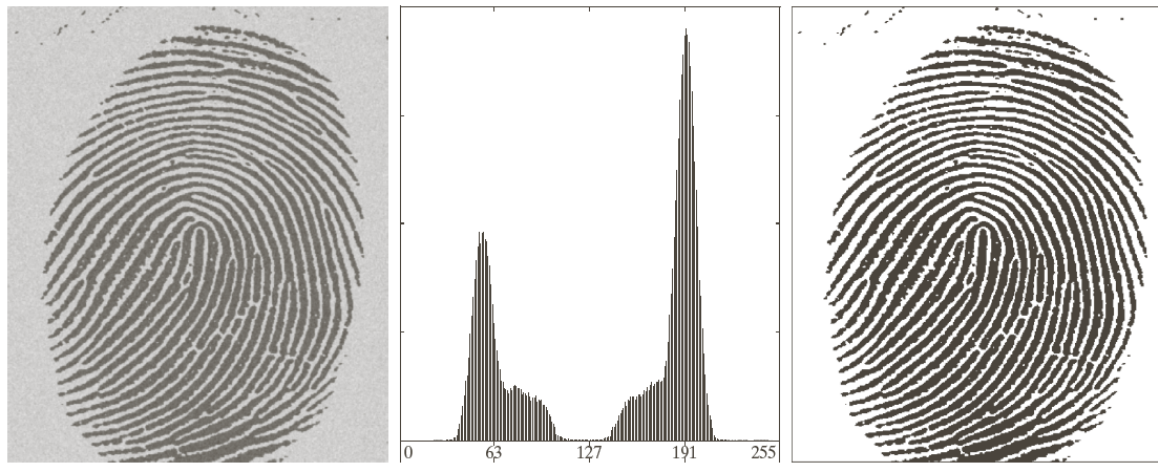
2. Segment the image using T . This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2 .
4. Compute a new threshold value:

$$T = \frac{1}{2}(m_1 + m_2)$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iteration is smaller than a predefined parameter ΔT

This simple algorithm works well in situations where there is a reasonably clear valley between the modes of the histogram related to objects and background. Parameter ΔT is used to control the number of iterations in situations when speed is an important issue. The initial threshold must be chosen greater than the minimum and

less than maximum intensity level in the image. The average intensity of the image is a good initial choice for T .



a b c

FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

Optimum Global Thresholding Using Otsu's Method

Thresholding may be viewed as a statistical-decision theory problem whose objective is to minimize the average error that appears in assigning pixels to two or more groups (also called *classes*). The solution (*Bayes decision rule*) is based on only two parameters: the probability density function (PDF) of the intensity levels of each class and the probability that each class occurs in a given application. Estimating PDFs is not a trivial task,

so the problem usually is simplified by making workable assumption about the form of the PDFs, such as assuming that they are Gaussian function.

Otsu's method offers an alternative solution. The method is optimum in the sense that it maximizes the *between-class variance*. The basic idea is that the well-thresholded classes should be distinct with respect to the intensity values of their pixels and, conversely, that a threshold giving the best separation between classes in

terms of their intensity values would be the best (optimum) threshold. Otsu's method has the important property that it is based entirely on computations performed of the histogram of an image.

Let $\{0, 1, 2, \dots, L-1\}$ denote the L distinct intensity levels in a digital image of size $M \times N$ pixels, and let n_i denote the number of pixels with intensity i .

$$MN \text{ (total number of pixels)} = n_0 + n_1 + n_2 + \dots + n_{L-1}$$

The normalized histogram has components $p_i = \frac{n_i}{MN}$

with:

$$\sum_{i=0}^{L-1} p_i = 1, \quad p_i \geq 0.$$

Suppose that we select a threshold $T(k)=k$, $0 < k < L-1$ and use it to threshold the image into two classes C_1 and C_2 where C_1 consists of all pixels in the image with intensity values in the range $[0,k]$ and C_2 consists of all pixels in the

image with intensity values in the range $[k+1, L-1]$. Using this threshold the probability, $P_1(k)$ that a pixel is assigned to class C_1 is given by the cumulative sum:

$$P_1(k) = \sum_{i=0}^k p_i$$

This is the probability of class C_1 occurring. Similarly, the probability of class C_2 occurring is:

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i .$$

The mean intensity values of the pixels assigned to class C_1 is

$$m_1(k) = \sum_{i=0}^k iP(i / C_1) = \sum_{i=0}^k iP(C_1 / i) \frac{P(i)}{P(C_1)} = \frac{1}{P_1(k)} \sum_{i=0}^k ip_i$$

$P(i/C_1)$ is the probability of value i , given that i comes from class C_1 . We have used the Bayes' formula:

$$P(A / B) = P(B / A) \frac{P(A)}{P(B)}.$$

$P(C_1/i)=1$ – the probability of C_1 given i (i belongs to C_1).

Similarly, the mean intensity value of the pixels assigned to class C_2 is:

$$m_2(k) = \sum_{i=k+1}^{L-1} i P(i / C_2) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i p_i .$$

The cumulative mean (average intensity) up to level k is given by:

$$m(k) = \sum_{i=0}^k ip_i$$

and the average intensity of the entire image (the *global mean*) is given by:

$$m_G = \sum_{i=0}^{L-1} ip_i$$

We have:

$$P_1 m_1 + P_2 m_2 = m_G \quad , \quad P_1 + P_2 = 1.$$

In order to evaluate the “goodness” of the threshold at level k we use the normalized, dimensionless metric:

Computer Vision

Course 9

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

where σ_G^2 is the *global variance*:

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

and σ_B^2 is the *between-class variance*, defined as:

$$\begin{aligned} \sigma_B^2 &= P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1 P_2 (m_1 - m_2)^2 = \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)} \end{aligned}$$

From the above formula, we see that the farther the two means m_1 and m_2 are from each other the larger σ_B^2 will be, indicating that the between-class variance is a measure of *separability* between classes. Because σ_G^2 is a constant, it follows that η *also is a measure* of separability, and maximizing this metric is equivalent to maximizing σ_B^2 .

The objective then is to determine the threshold value k that maximizes the between-class variance.

We have:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]}{P_1(k)[1 - P_1(k)]}$$

The optimum threshold is the value k^* that maximizes $\sigma_B^2(k)$

$$\sigma_B^2(k^*) = \max\{ \sigma_B^2(k) ; 0 \leq k \leq L-1, k - \text{integer} \}.$$

If the maximum exists for more than one value of k , it is customary to average the various values of k for which $\sigma_B^2(k)$ is maximum.

Once k^* has been obtained, the input image is segmented as:

$$g(x, y) = \begin{cases} \mathbf{1} & \text{if } f(x, y) > k^* \\ \mathbf{0} & \text{if } f(x, y) \leq k^* \end{cases}$$

The metric $\eta(k^*)$ can be used to obtain quantitative estimate of the separability of classes.

$$\mathbf{0} \leq \eta(k^*) \leq \mathbf{1}$$

The lower bound is attainable only by images with a single, constant intensity level, the upper bound is

attainable only by 2-valued images with intensities equal to 0 and $L-1$.

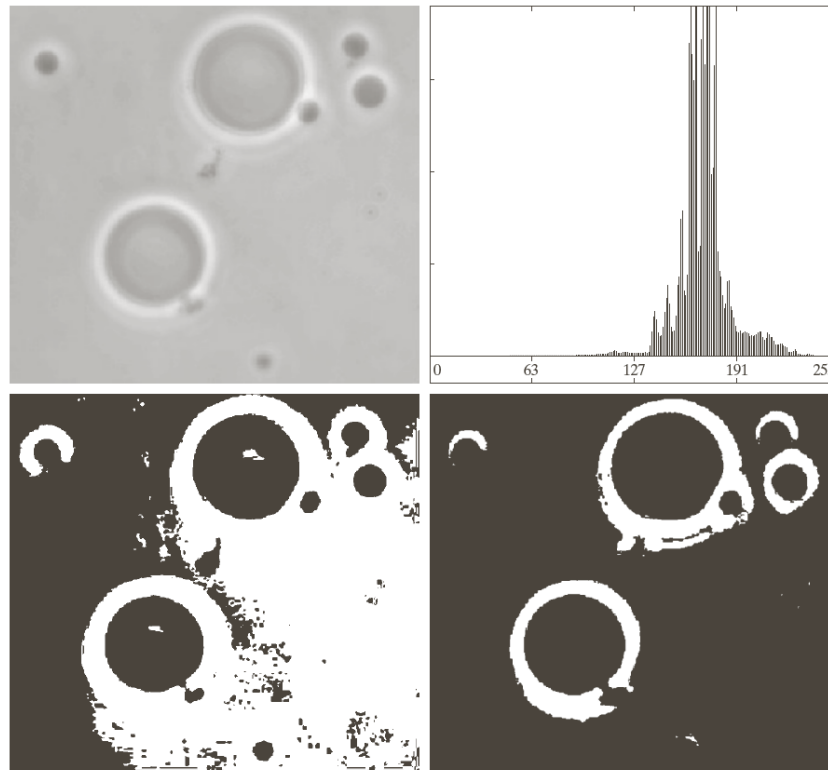
Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image, p_i , $i=0,1,2,\dots,L-1$
2. Compute the cumulative sums, $P_1(k)$, $k=0,1,2,\dots,L-1$
3. Compute the cumulative means, $m(k)$, $k=0,1,\dots,L-1$
4. Compute the global intensity mean, m_G

5. Compute the between-class variance, $\sigma_B^2(k)$,
 $k=0,1,\dots,L-1$
6. Obtain the Otsu threshold, k^* , as the value of k for which $\sigma_B^2(k)$ is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected
7. Obtain the separability measure, $\eta(k^*)$

Computer Vision

Course 9



a b
c d

FIGURE 10.39

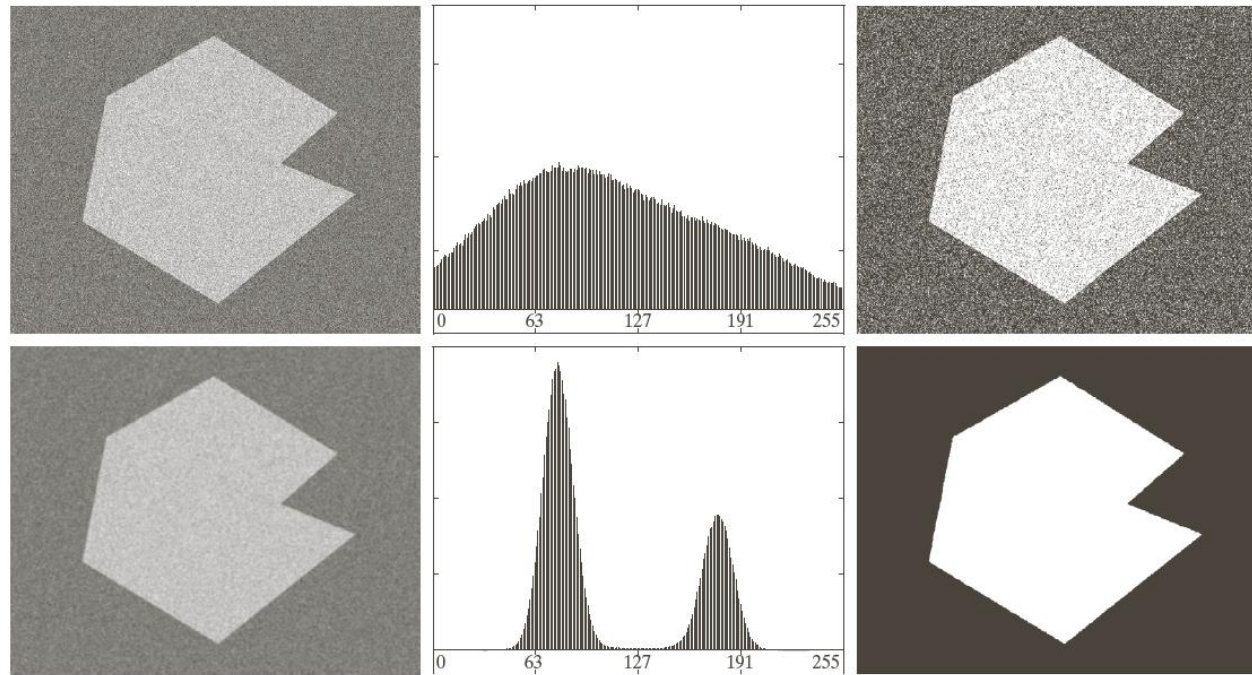
(a) Original image.

(b) Histogram (high peaks were clipped to highlight details in the lower values).

(c) Segmentation result using the basic global algorithm from Section 10.3.2.

(d) Result obtained using Otsu's method. (Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

Noise can turn a simple thresholding problem into an unsolvable one. When noise cannot be reduced at the source, and thresholding is the segmentation method used, a technique that often enhances performances is to smooth the image before thresholding it.



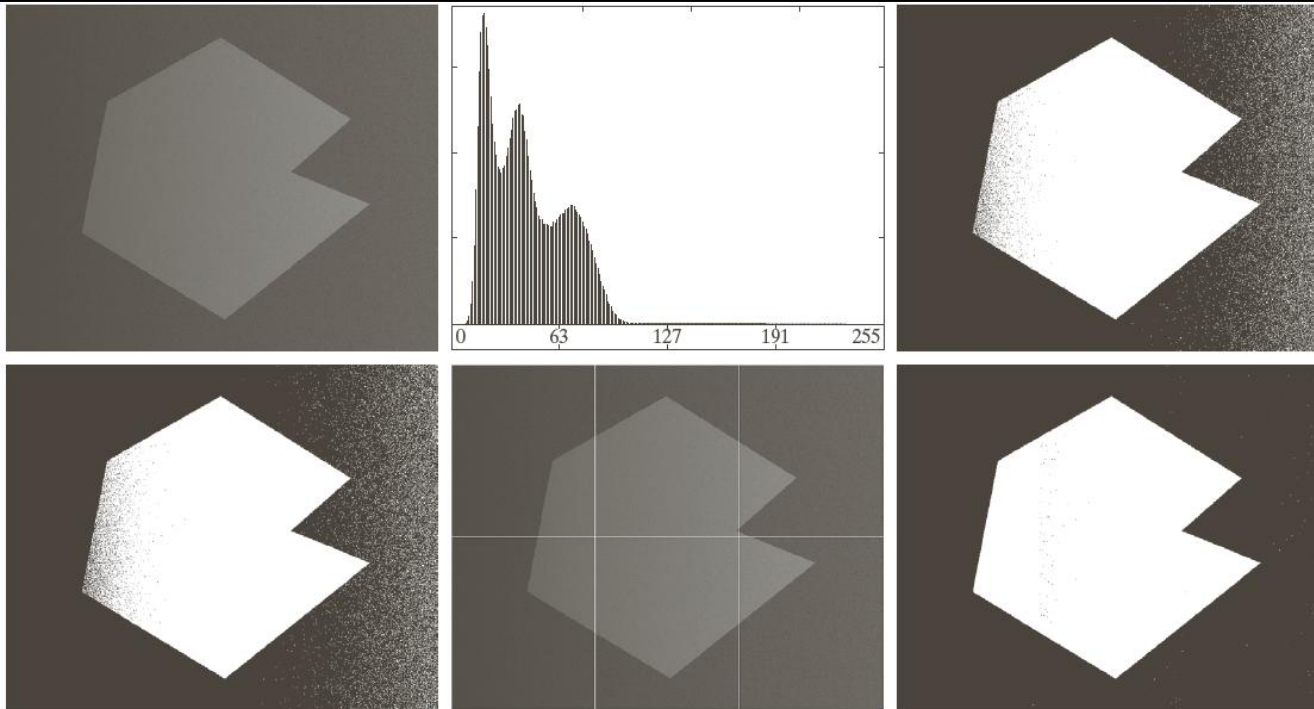
a b c
d e f

FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

Variable Thresholding

Image partitioning

One of the simplest approaches to variable thresholding is to subdivide an image into nonoverlapping rectangles. This approach is used to compensate for non-uniformities in illumination and/or reflectance. The rectangles are chosen small enough so that the illumination of each is approximately uniform.



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

Image subdivision generally works well when the objects of interest and the background occupy regions of reasonably comparable size. When this is not the case, the method fails because of the likelihood of subdivisions containing only object or background pixels.

Variable thresholding based on local image properties

A more general approach than the image subdivision method is to compute a threshold at every point (x,y) in the image based on one or more specified properties computed in a neighborhood of (x, y) .

We illustrate the basic approach to local thresholding by using the standard deviation and mean of the pixels in a neighborhood of every point in an image. Let σ_{xy} and m_{xy} denote the standard deviation and mean value of a set of

pixels contained in a neighborhood S_{xy} centered at coordinates (x, y) in an image.

The following are common forms of variable, local thresholds

$$T_{xy} = a\sigma_{xy} + bm_{xy}, \quad a, b \geq 0$$

$$T_{xy} = a\sigma_{xy} + bm_G, \quad m_G - \text{global image mean.}$$

The segmented image is computed as:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T_{xy} \\ 0 & \text{if } f(x, y) \leq T_{xy} \end{cases}.$$

Significant improvement can be obtained in local thresholding by using predicates based on the parameters computed in the neighborhood of (x, y) :

$$g(x, y) = \begin{cases} 1 & \text{if } Q(\text{local parameters}) \text{ is true} \\ 0 & \text{if } Q(\text{local parameters}) \text{ is false} \end{cases}$$

Where Q is a *predicate* based on parameters computed using the pixels in neighborhood S_{xy} .

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{true} & \text{if } f(x, y) > a\sigma_{xy} \text{ AND } f(x, y) > bm_{xy} \\ \text{false} & \text{otherwise} \end{cases}$$

Multivariable Thresholding

In some cases, a sensor can make available more than one variable to characterize each pixel in an image, and thus allow *multivariable thresholding*. A notable example is color imaging where red (R), green (G), and blue (B) components are used to form a composite color image. In this case, each “pixel” is characterized by three values, and can be represented as a 3-D vector $\mathbf{z} = (z_1, z_2, z_3)^T$ whose components are the RGB colors at a point.

These 3D points often are referred to as *voxels*, to denote *volumetric* elements, as opposed to *image* elements.

Multivariable thresholding may be viewed as a distance computation. Suppose that we want to extract from a color image all regions having a specified color range: say, reddish hues. Let \mathbf{a} denote the average reddish color in which we are interested.

One way to segment a color image based on this parameter is to compute a distance measure, $D(z, a)$ between an arbitrary color point z and the average color a . Then we segment the input image:

$$g = \begin{cases} \mathbf{1} & \text{if } D(z, a) < T \\ \mathbf{0} & \text{otherwise} \end{cases}, \quad T \text{ is a threshold}$$

$D(z, a)$ – *Euclidian distance*

$$D(z, a) = \left[(z - a)^T (z - a) \right]^{\frac{1}{2}}$$

Mahalanobis distance

$$D(\mathbf{z}, \mathbf{a}) = \left[(\mathbf{z} - \mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{a}) \right]^{\frac{1}{2}}$$

where \mathbf{C} is the covariance matrix of the \mathbf{z} s.

Region-Based Segmentation

Region growing

Region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria for growth. The basic approach is to start with a set of “seed” points and from these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as specific ranges of intensity colors).

Selecting a set of one or more starting points often can be based on the nature of the problem. When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available.

Another problem in region growing is the formulation of a stopping rule. Region growth should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as intensity values, texture, and color are local in nature and do not take into account the “history” of region growth.

Additional criteria that increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far, and the shape of the region being grown.

Let $f(x,y)$ denote an input image array, $S(x,y)$ denote a *seed* array containing 1 s at the locations of seed points and 0 s elsewhere, Q denote a predicate to be applied at each pixel location (x, y) .

Arrays f and S are assumed to be of the same size. A basic region-growing algorithm based on 8-connectivity may be stated as follows:

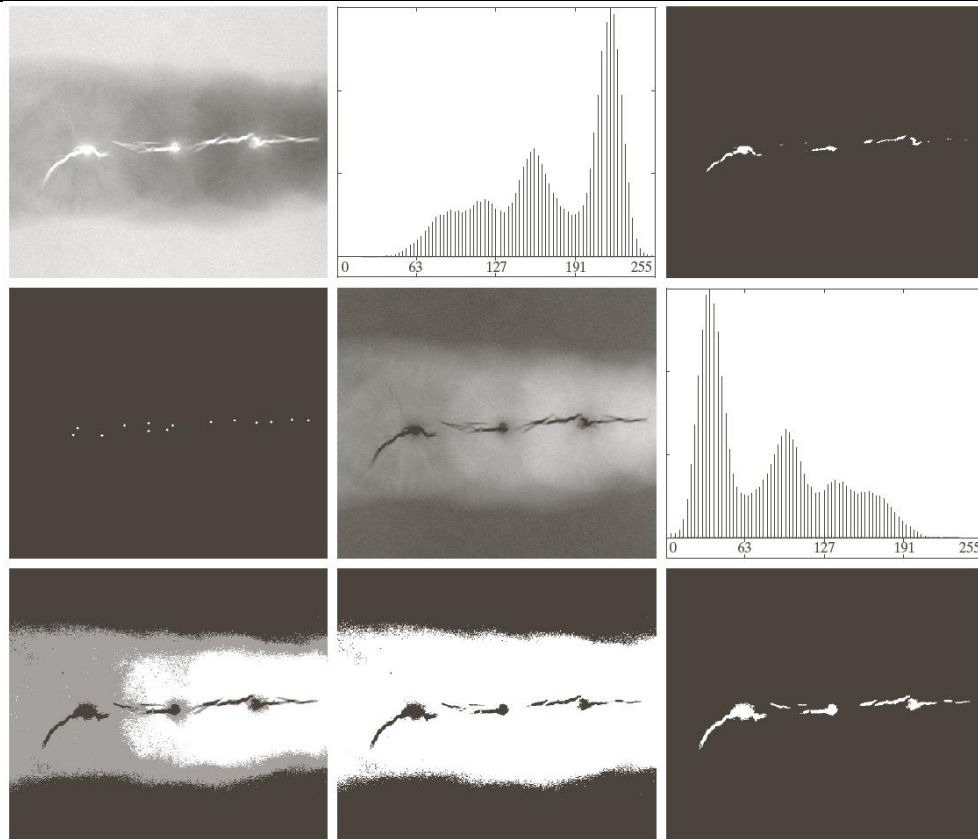
1. Find all connected components in $S(x,y)$ and erode each connected component to one pixel; label all such pixels found as 1 . All other pixels in S are labeled 0 .
2. Form an image f_Q such that, at a pair of coordinates (x,y) , let $f_Q(x,y)=1$ if the input image satisfies the given predicate, Q , at those coordinates, otherwise, let $f_Q(x,y)=0$.

3. Let g be an image formed by appending to each seed point in S all the I -valued points in f_Q that are δ -connected to that seed point.
4. Label each connected component in g with a different region label. This is the segmented region obtained by region growing.

$$Q = \begin{cases} \text{TRUE} & \text{if the absolute difference of the intensities} \\ & \text{between the seed and the pixel at } (x, y) \text{ is } \leq T \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Computer Vision

Course 9



a b c
d e f
g h i

FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

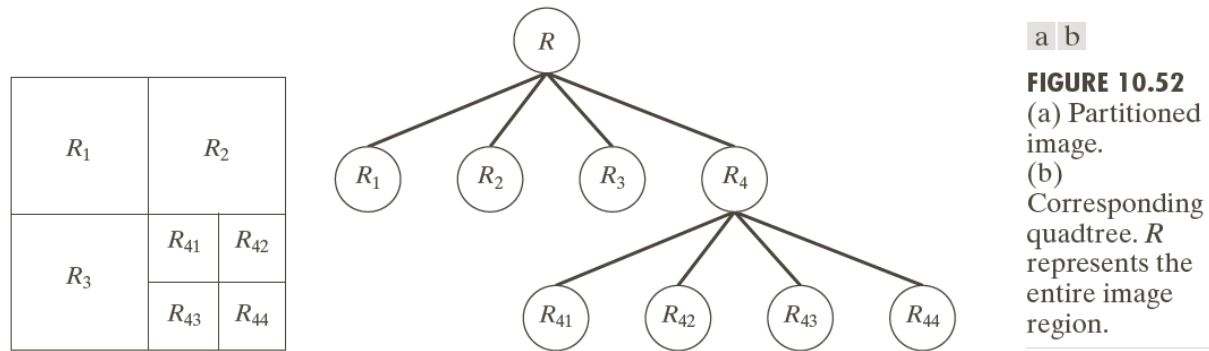
Region Splitting and Merging

The method used in this case is to subdivide an image initially into a set of arbitrary, disjoint regions and then merge and/or split the regions in an attempt to satisfy the condition of segmentation.

Let R represent the entire image region and select a predicate Q . One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i , $Q(R_i)=TRUE$. We start with the entire

region. If $Q(R)=FALSE$, we divide the image into quadrants. If Q is False for any quadrant, we subdivide the quadrant into subquadrant, and so on. This particular splitting technique has a convenient representation in the form of so-called *quadtrees*, that are trees in which each node has exactly four descendants. The images corresponding to the nodes of a quadtree sometimes are called *quadregions* or *quadimages*. Note that the root of the tree corresponds to the entire image

and that each node corresponds to the subdivision of a node into four descendant nodes.



If only splitting is used, the final partition normally contains adjacent region with identical properties. Satisfying the constraints of segmentation requires merging only adjacent regions whose combined pixels satisfy the predicate Q . That

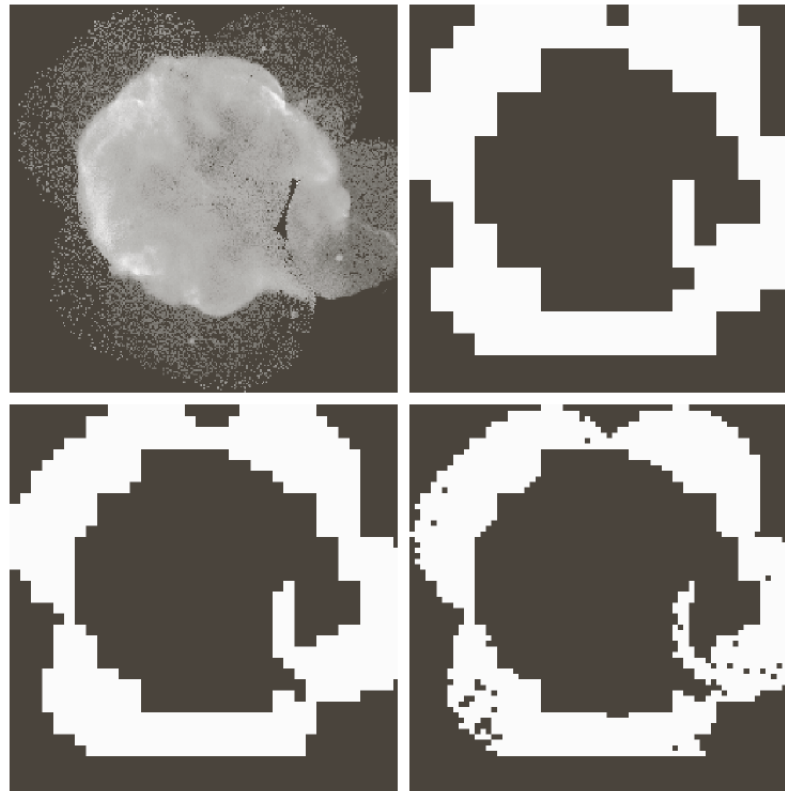
is, two adjacent regions R_j and R_k are merged only if $Q(R_j \cup R_k) = \text{TRUE}$.

The procedure described above can be summarized as follows

1. Split into four quadrants any region R_i for which $Q(R_i) = \text{TRUE}$
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = \text{TRUE}$
3. Stop when no further merging is possible.

It is customary to specify a minimum quadregions size beyond which no further splitting is carried out.

Numerous variations of the preceding basic theme are possible. For example, a significant simplification results if in Step 2 we allow merging for any two adjacent regions if each one satisfies the predicate individually. This results in a much simpler (and faster) algorithm, because testing the predicate is limited to individual quadregions.



a b
c d

FIGURE 10.53
(a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b)–(d) Results of limiting the smallest allowed quadregion to sizes of 32×32 , 16×16 , and 8×8 pixels, respectively. (Original image courtesy of NASA.)

$$Q = \begin{cases} \mathbf{TRUE} & \text{if } \sigma > a \text{ AND } 0 < m < b \\ \mathbf{FALSE} & \text{otherwise} \end{cases}$$

Where m and σ are the mean and the standard deviation of the pixels in a quadregion, and a and b are constants.