UNIT 4
IMAGE SEGMENTATION

Course: Digital Image Processing(EC662)

Course Instructor: Prof. Shashidhar R

Dept.: Electronics and Communication Engg.

CONTENTS

- Point, Line, and Edge Detection(Robert canny and Prewitt techniques)
- Thresholding
- Basic global thresholding
- Optimum global thresholding using Ostu's method
- Region Based segmentation

Introduction to Image segmentation

- The Purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application.
- The segmentation is based on measurements taken from the image and might be gray level, color, texture, depth or motion.
- Usually image segmentation is an initial and vital step in a series of processes aimed at overall image understanding.

Application of image segmentation:

- Identifying objects in a scene for object-based measurements such as size and shape.
- Identifying objects in a moving scene for object-based video compression(MPEG4)
- Identifying objects which are at different distances from a sensor using depth measurements from a laser range finder enabling path planning for a mobile robots.

Basic Methods:

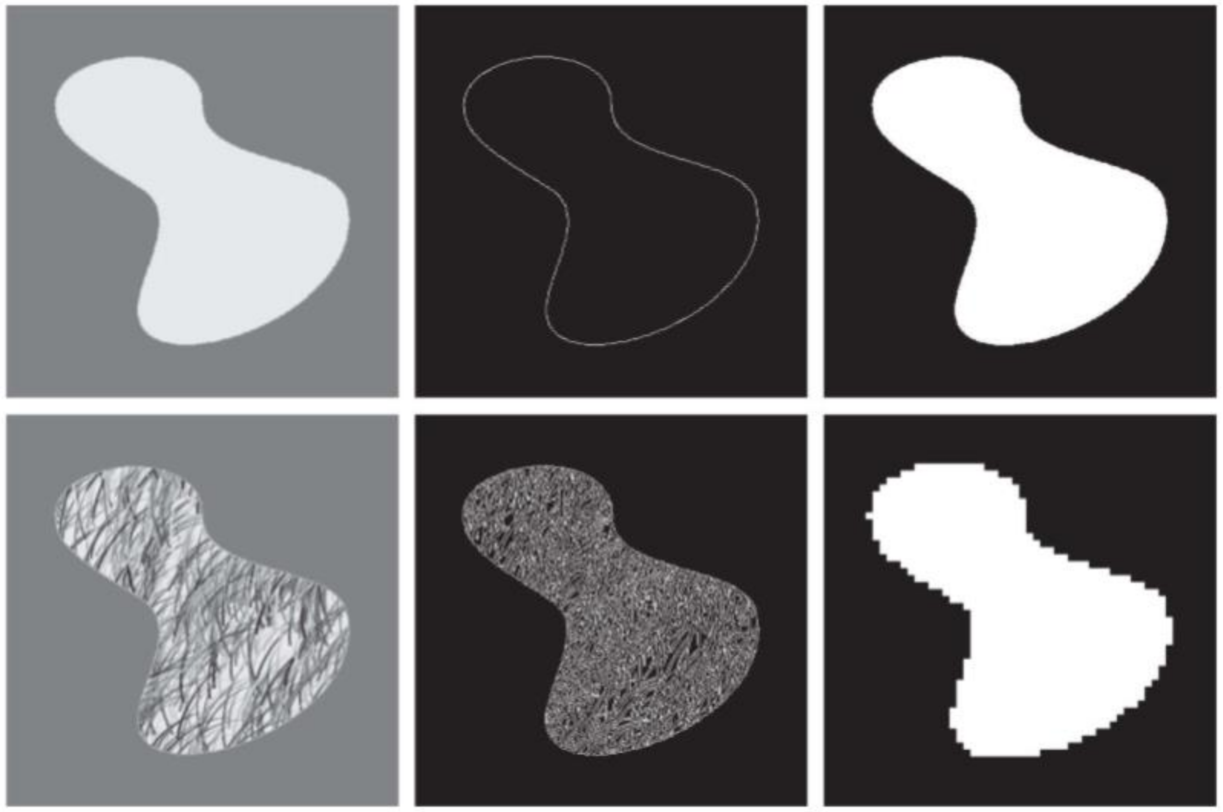
- Point, line, edge detection
- Thresholding
- Region growing
- Morphological water sheets.

Advanced Methods:

- Clustering
- Model fitting
- Probabilistic methods

Constant Intensity (edge-based Segmentation)

Textured region (region-based Segmentation)



Point, line and edge detection

- The focus of this section is on segmentation methods that are based on detecting sharp, local changes in intensity.
- The three types of image characteristics in which we are interested are isolated points, lines, and edges.
- Edge pixels are pixels at which the intensity of an image changes abruptly, and edges (or edge segments) are sets of connected edge pixels.
- Edge detectors are local image processing tools designed to detect edge pixels.
- A line may be viewed as a (typically) thin 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.

- lines give rise to so-called “roof edges.”
- Finally, an isolated point may be viewed as a foreground (background) pixel surrounded by background (foreground) pixels.
- Derivatives of a digital function are defined in terms of finite differences.
- There are various ways to compute these differences but we require that any approximation used for first derivatives

(1) must be zero in areas of constant intensity;

(2) must be nonzero at the onset of an intensity step or ramp;

(3) and must be nonzero at points along an intensity ramp.

Similarly, we require that an approximation used for second derivatives

(1) must be zero in areas of constant intensity;

(2) must be nonzero at the onset and end of an intensity step or ramp; and

(3) must be zero along intensity ramps.

Because we are dealing with digital quantities whose values are finite, the maximum possible intensity change is also finite, and the shortest distance over which a change can occur is between adjacent pixels.

Detection of Isolated Points

- First order derivatives produce thick edges at ramps.
- Second order derivatives are non zero at the onset and at the end of a ramp or step edge (sign change).
- Second order derivatives respond stronger at isolated points and thin lines than first order derivatives.
- First order derivative

$$\frac{\partial f(x)}{\partial x} = f'(x) = f(x+1) - f(x) \tag{10-4}$$

- Second order derivative

$$\frac{\partial^2 f(x)}{\partial x^2} = f''(x) = f(x+1) - 2f(x) + f(x-1) \tag{10-7}$$

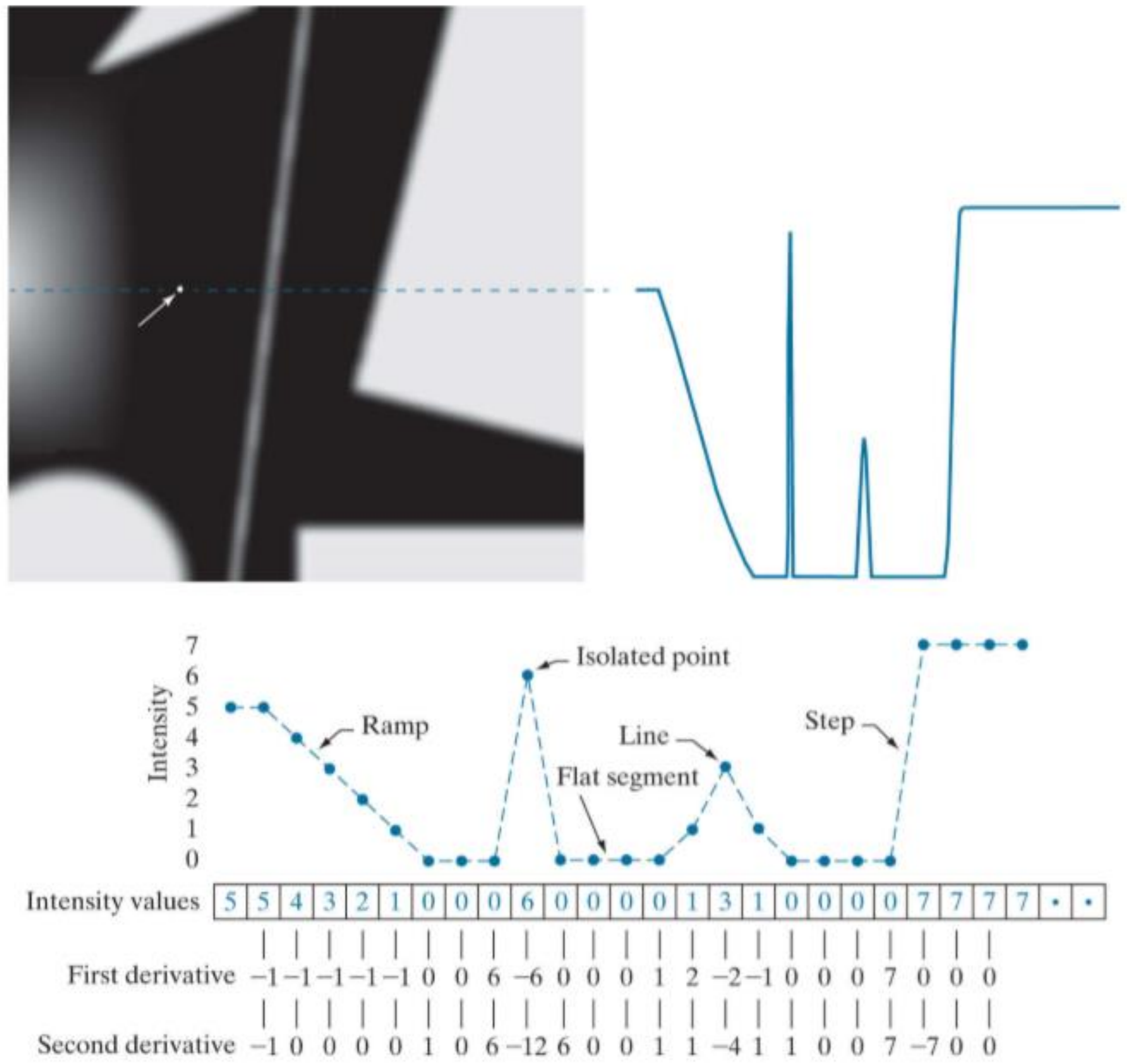

Intensity
7 6 5 4 3 2 1 0
Isolated point
Ramp
Line
Flat segment
Step
Intensity values 5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7 • •
First derivative −1 −1 −1 −1 −1 0 0 6 −6 0 0 0 1 2 −2 −1 0 0 0 7 0 0 0
Second derivative −1 0 0 0 0 1 0 6 −12 6 0 0 1 1 −4 1 1 0 0 7 −7 0 0

FIGURE 10.2

(a) Image. (b) Horizontal intensity profile that includes the isolated point indicated by the arrow. (c) Subsampled profile; the dashes were added for clarity

- using the Laplacian:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (10\text{-}13)$$

- where the partial derivatives are computed using the second-order finite differences The Laplacian is then

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \qquad (10\text{-}14)$$

- This expression can be implemented using the Laplacian kernel in Fig. 10.4(a)

- A point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold.

- Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image. In other words, we use the expression:

$$g(x,y) = \begin{cases} 1 & \text{i } |Z(x,y)| > T \\ 0 & \text{o t h e r} \end{cases} \qquad (10\text{-}15)$$

- This formulation simply measures the weighted differences between a pixel and its 8-neighbors. Intuitively, 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 kernel.
- Differences in intensity that are considered of interest are those large enough (as determined by T) to be considered isolated points

EXAMPLE 10.1: Detection of isolated points in an image

1	1	1
1	−8	1
1	1	1

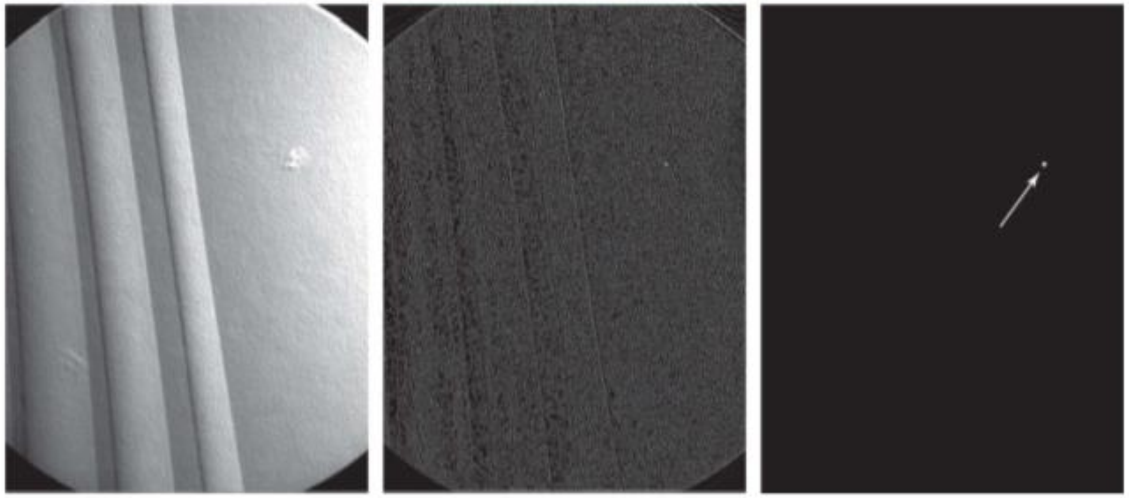


FIGURE 10.4

(a) Laplacian kernel used for point detection. (b) X-ray image of a turbine blade with a porosity manifested by a single black pixel. (c) Result of convolving the kernel with the image. (d) Result of using Eq. (10-15) was a single point (shown enlarged at the tip of the arrow).

- Figure 10.4(b) is an X-ray image of a turbine blade from a jet engine. The blade has a porosity manifested by a single black pixel in the upper-right quadrant of the image.
- Figure 10.4(c) is the result of filtering the image with the Laplacian kernel, and Fig.10.4(d) shows the result of Eq.(10-15) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.4(c) .
- The single pixel is clearly visible in this image at the tip of the arrow (the pixel was enlarged to enhance its visibility).
- This type of detection process is specialized because it is based on abrupt intensity changes at single-pixel locations that are surrounded by a homogeneous background in the area of the detector kernel.
- When this condition is not satisfied, other methods discussed in this chapter are more suitable for detecting intensity changes.

Line Detection

- The next level of complexity is line detection.
- Based on the discussion earlier in this section, we know that for line detection we can expect second derivatives to result in a stronger filter response, and to produce thinner lines than first derivatives.
- Thus, we can use the Laplacian kernel in Fig. 10.4(a) for line detection also, keeping in mind that the double-line effect of the second derivative must be handled properly.

EXAMPLE 10.2: Using the Laplacian for line detection

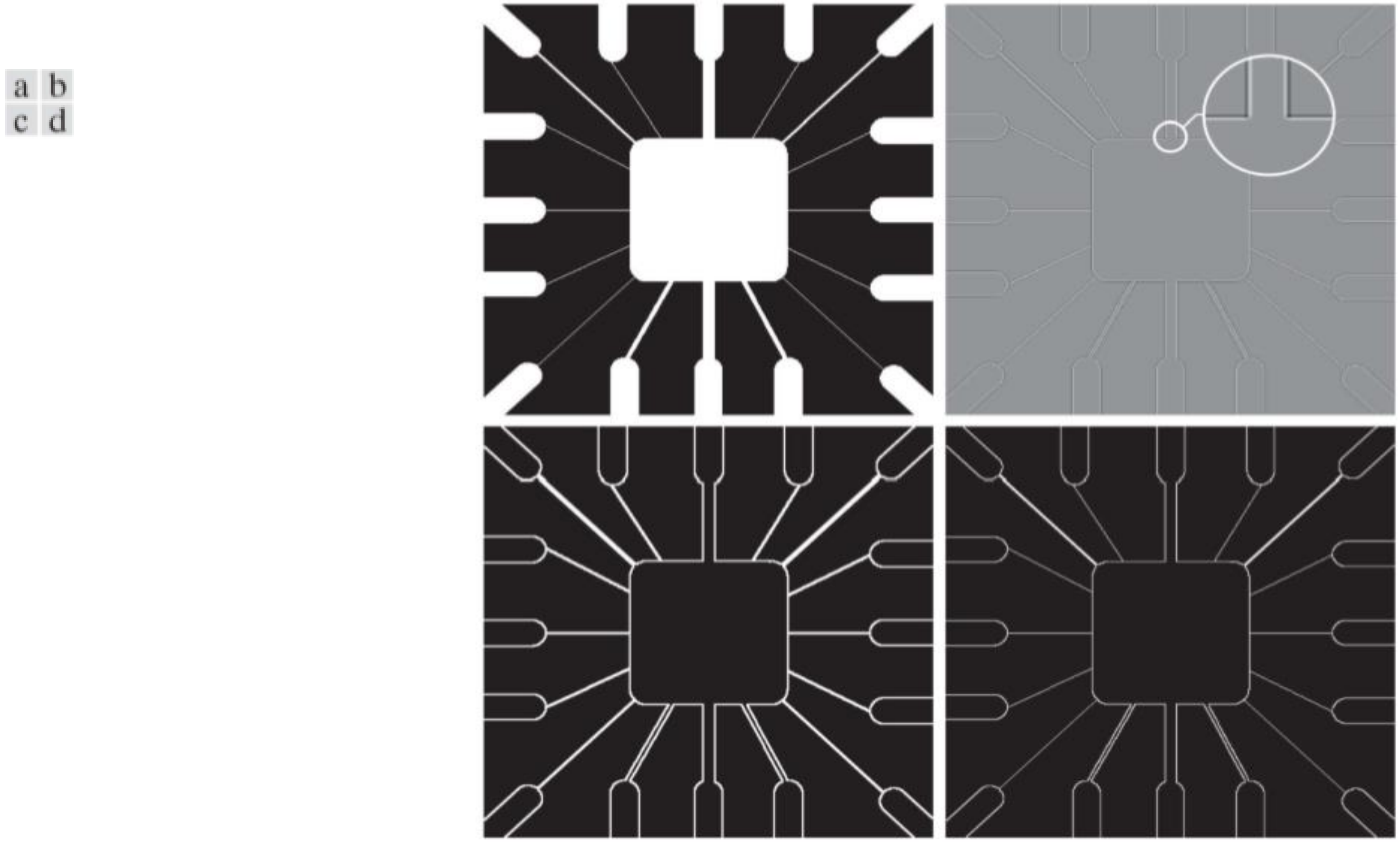
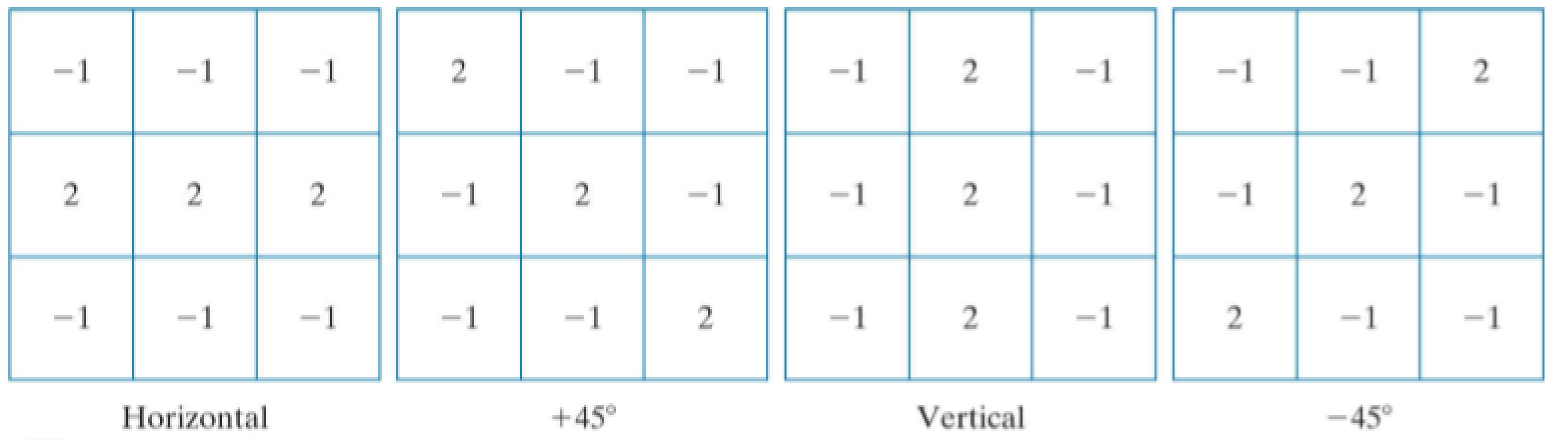


FIGURE10.5 (a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

- Figure 10.5(a) shows a 486x486 (binary) portion of a wire-bond mask for an electronic circuit, and Fig.10.5(b) shows its Laplacian image.
- Because the Laplacian image contains negative values, scaling is necessary for display.
- Mid gray represents zero, darker shades of gray represent negative values, and lighter shades are positive. The double-line effect is clearly visible in the magnified region.
- At first, it might appear that the negative values can be handled simply by taking the absolute value of the Laplacian image.
- However, as Fig.10.5(c) shows, this approach doubles the thickness of the lines.
- A more suitable approach is to use only the positive values of the Laplacian.

- As Fig.10.5(d) shows, this approach results in thinner lines that generally are more useful.
- Note in Figs.10.5(b) through (d) that when the lines are wide with respect to the size of the Laplacian kernel, the lines are separated by a zero "valley."
- This is not unexpected. For example, when the kernel is centered on a line of constant intensity 5 pixels wide, the response will be zero, thus producing the effect just mentioned.
- When we talk about line detection, the assumption is that lines are thin with respect to the size of the detector.
- Lines that do not satisfy this assumption are best treated as regions and handled by the edge detection methods discussed in the following section.

a b c d



−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

2	−1	−1
−1	2	−1
−1	−1	2

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

−1	−1	2
−1	2	−1
2	−1	−1

−45°

FIGURE 10.6

Line detection kernels. Detection angles are with respect to the axis system in with respect to the (vertical) x-axis.

EXAMPLE 10.3: Detecting lines in specified directions.

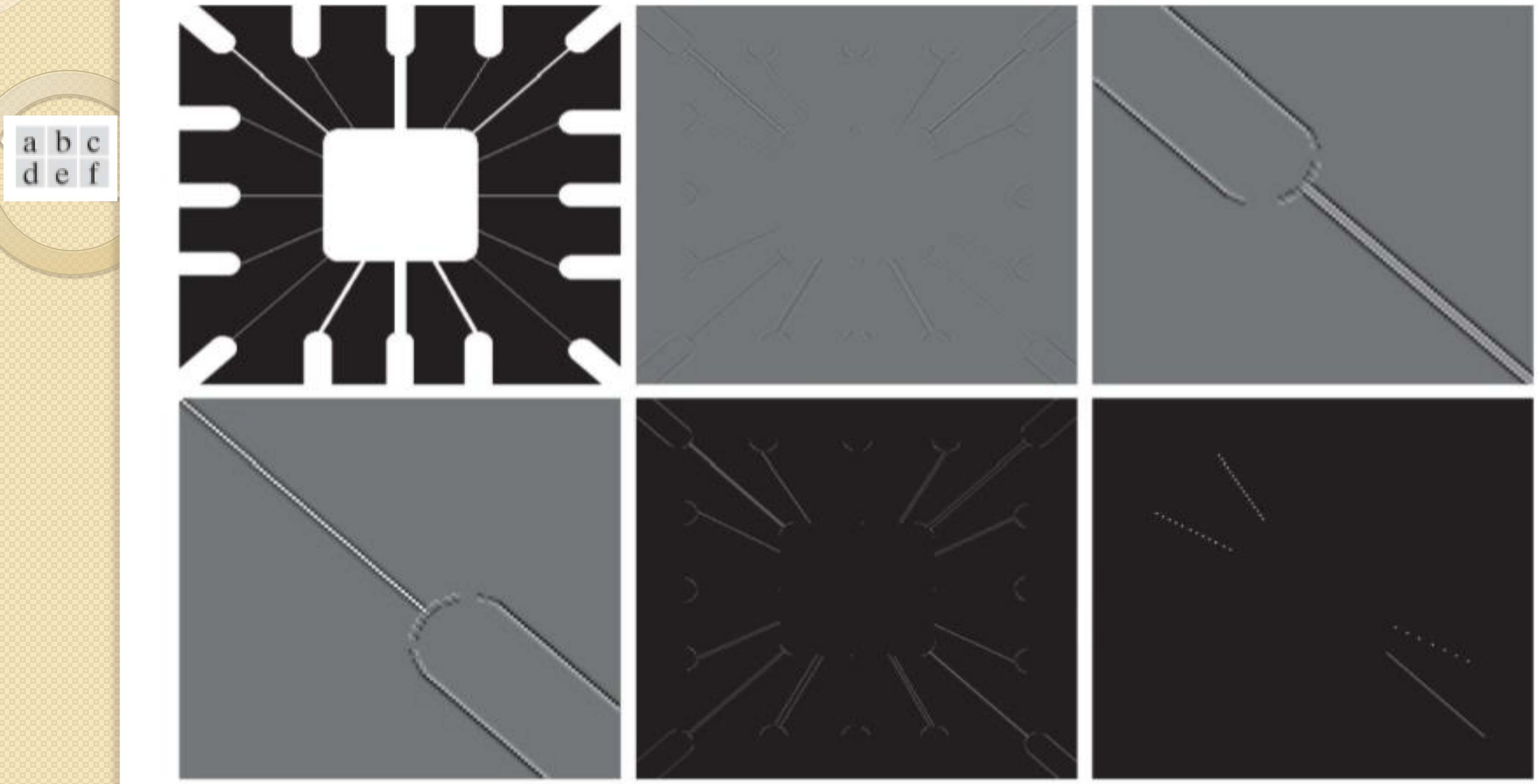


FIGURE10.7 (a) Image of a wire-bond template. (b) Result of processing with the+45^0 line detector kernel in Fig.10.6 . (c) Zoomed view of the top left region of (b). (d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the g>T condition where g is the image in (e) and T=25 (the maximum pixel value in the image minus 1).

Edge Models

- Edge detection is an approach used frequently for segmenting images based on abrupt (local) changes in intensity.
- We begin by introducing several ways to model edges and then discuss a number of approaches for edge detection.
- Edge models are classified according to their intensity profiles.
- A step edge is characterized by a transition between two intensity levels occurring ideally over the distance of one pixel.

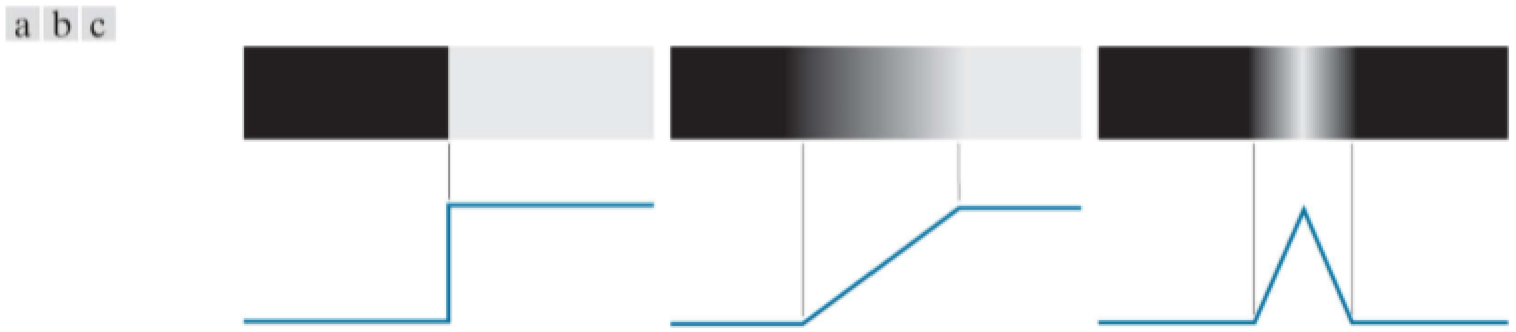


FIGURE 10.8 From left to right, models (ideal representations) of a step(a), a ramp,(b) and a roof edge(c) , and their corresponding intensity profiles.

- Figure10.8(a) shows a section of a vertical step edge and a horizontal intensity profile through the edge.
- Step edges occur, for example, in images generated by a computer for use in areas such as solid modeling and animation.
- These clean, ideal edges can occur over the distance of one pixel, provided that no additional processing (such as smoothing) is used to make them look "real."

- Digital step edges are used frequently as edge models in algorithm development.
- For example, the Canny edge detection algorithm discussed later in this section was derived originally using a step-edge model.
- In practice, digital images have edges that are blurred and noisy, with the degree of blurring determined principally by limitations in the focusing mechanism (e.g., lenses in the case of optical images), and the noise level determined principally by the electronic components of the imaging system.

- In such situations, edges are more closely modeled as having an intensity ramp profile, such as the edge in Fig. 10.8(b)
- The slope of the ramp is inversely proportional to the degree to which the edge is blurred. In this model, we no longer have a single "edge point" along the profile.
- Instead, 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 type of edge is the so-called roof edge, having the characteristics illustrated in Fig.10.8(c).
- Roof edges are models of lines through a region, with the base (width) of the edge being determined by the thickness and sharpness of the line.
- In the limit, when its base is one pixel wide, a roof edge is nothing more than a one-pixel-thick line running through a region in an image.

- Roof edges arise, for example, in range imaging, when thin objects (such as pipes) are closer to the sensor than the background (such as walls).
- The pipes appear brighter and thus create an image similar to the model in Fig.10.8(c)
- Other areas in which roof edges appear routinely are in the digitization of line drawings and also in satellite images, where thin features, such as roads, can be modeled by this type of edge.

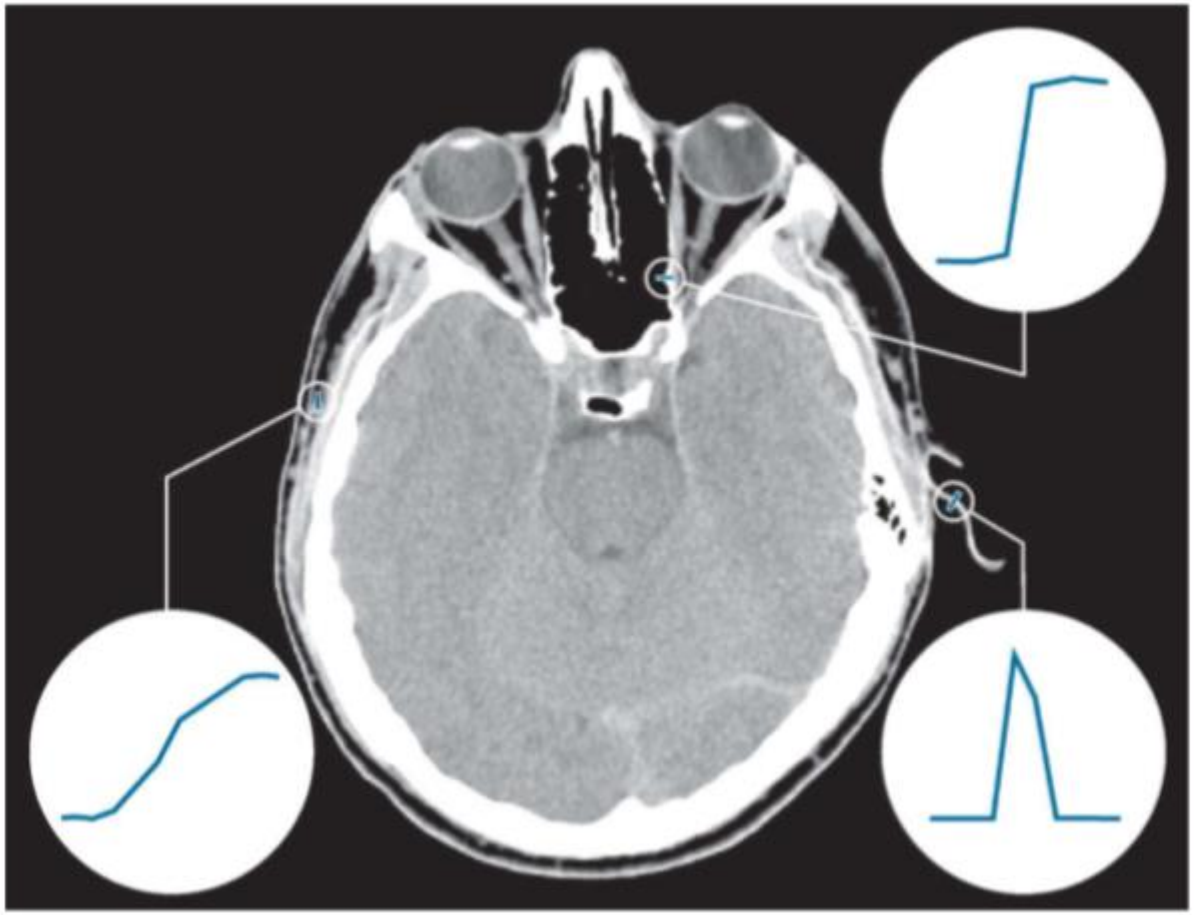


FIGURE 10.9 A image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas enclosed by the small circles. The ramp and step profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels

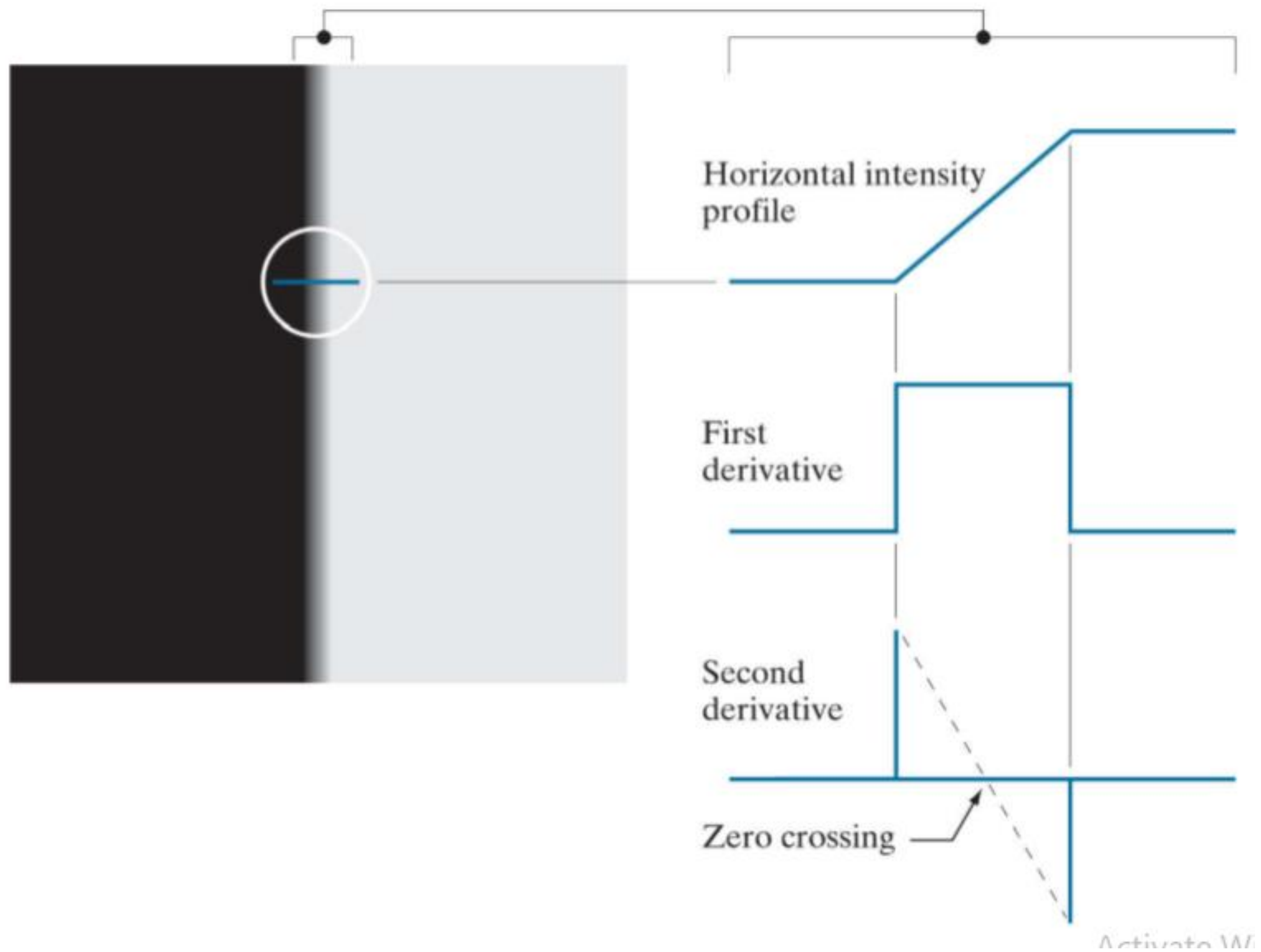


FIGURE 10.10 (a) Two regions of constant intensity separated by an ideal ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, and its first and second derivatives.

- Figure 10.10(a) shows the image from which the segment in Fig. 10.8(b) was extracted.Figure10.10(b) shows a horizontal intensity profile.
- This figure shows also the first and second derivatives of the intensity profile.
- Moving from left to right along the intensity profile, we note that the first derivative is positive at the onset of the ramp and at points on the ramp, and it is zero in areas of constant intensity.
- The second derivative is positive at the beginning of the ramp, negative at the end of the ramp, zero at points on the ramp, and zero at points of constant intensity.
- The signs of the derivatives just discussed would be reversed for an edge that transitions from light to dark.
- The intersection between the zero intensity axis and a line extending between the extrema of the second derivative marks a point called the zero crossing of the second derivative.

- We conclude from these observations that 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.
- Two additional properties of the second derivative around an edge are:

(1) it produces two values for every edge in an image;

(2) its zero crossings can be used for locating the centers of thick edges

EXAMPLE 10.4: Behavior of the first and second derivatives in the region of a noisy edge

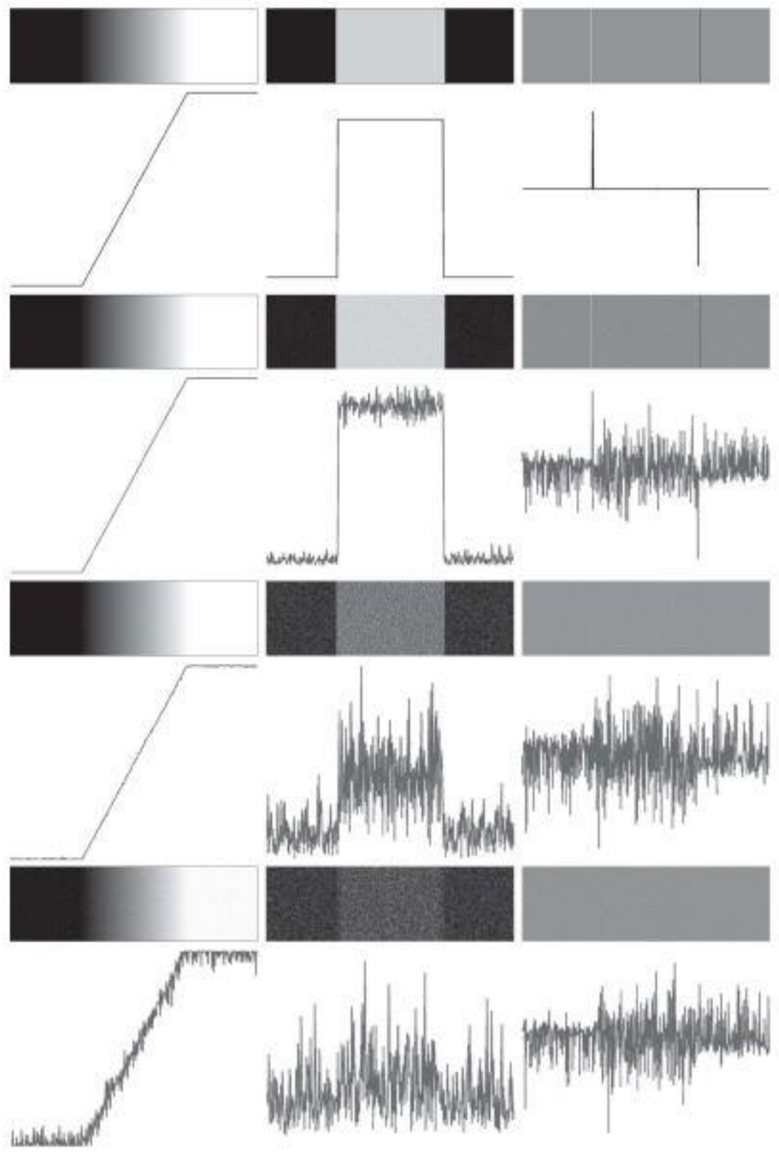


FIGURE 10.11 First column: 8-bit images with values in the range [0, 255], and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

- The edge models in Fig. 10.8 are free of noise. The image segments in the first column in Fig. 10.11 show close-ups of four ramp edges that transition from a black region on the left to a white region on the right (keep in mind that the entire transition from black to white is a single edge).
- The image segment at the top left is free of noise.
- The other three images in the first column are corrupted by additive Gaussian noise with zero mean and standard deviation of 0.1, 1.0, and 10.0 intensity levels, respectively.
- The graph below each image is a horizontal intensity profile passing through the center of the image.
- . All images have 8 bits of intensity resolution, with 0 and 255 representing black and white, respectively.

- Consider the image at the top of the center column. As discussed in connection with Fig. 10.10(b) , the derivative of the scan line on the left is zero in the constant areas.
- These are the two black bands shown in the derivative image.
- The derivatives at points on the ramp are constant and equal to the slope of the ramp.
- These constant values in the derivative image are shown in gray.
- As we move down the center column, the derivatives become increasingly different from the noiseless case.
- In fact, it would be difficult to associate the last profile in the center column with the first derivative of a ramp edge.
- What makes these results interesting is that the noise is almost visually undetectable in the images on the left column.
- These examples are good illustrations of the sensitivity of derivatives to noise.

- As expected, the second derivative is even more sensitive to noise.
- The second derivative of the noiseless image is shown at the top of the right column.
- . The thin white and black vertical lines are the positive and negative components of the second derivative, as explained in Fig. 10.10.
- The gray in these images represents zero (as discussed earlier, scaling causes zero to show as gray).
- The only noisy second derivative image that barely resembles the noiseless case corresponds to noise with a standard deviation of 0.1.
- The remaining second-derivative images and profiles clearly illustrate that it would be difficult indeed to detect their positive and negative components, which are the truly useful features of the second derivative in terms of edge detection

- The fact that such little visual noise can have such a significant impact on the two key derivatives used for detecting edges is an important issue to keep in mind.
- In particular, image smoothing should be a serious consideration prior to the use of derivatives in applications where noise with levels similar to those we have just discussed is likely to be present.

summary, the three steps performed typically for edge detection are:

1. Image smoothing for noise reduction. The need for this step is illustrated by the results in the second and third columns of Fig. 10.11.
2. Detection of edge points. As mentioned earlier, this is a local operation that extracts from an image all points that are potential edge-point candidates.
3. Edge localization. The objective of this step is to select from the candidate points only the points that are members of the set of points comprising an edge.

Basic Edge Detection

- As illustrated in the preceding discussion, detecting changes in intensity for the purpose of finding edges can be accomplished using first- or second-order derivatives.
- We begin with first-order derivatives, and work with second-order derivatives in the following subsection.

The Image Gradient and Its Properties

- The tool of choice for finding edge strength and direction at an arbitrary location (x, y) of an image, f, is the gradient, denoted by and defined as the vector.

$$\nabla f(x,y) \equiv \text{grad}[f(x,y)] \equiv \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \qquad (10\text{-}16)$$

This vector has the well-known property that it points in the direction of maximum rate of change of f at (x, y)

- Equation (10-16) is valid at an arbitrary (but single) point (x, y).
- When evaluated for all applicable values of x and y, ∇f(x,y) becomes a vector image, each element of which is a vector given by Eq. (10-16).
- The magnitude, M(x, y), of this gradient vector at a point (x, y) is given by its Euclidean vector norm:

$$M(x,y) = \|\nabla f(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)} \qquad (10\text{-}17)$$

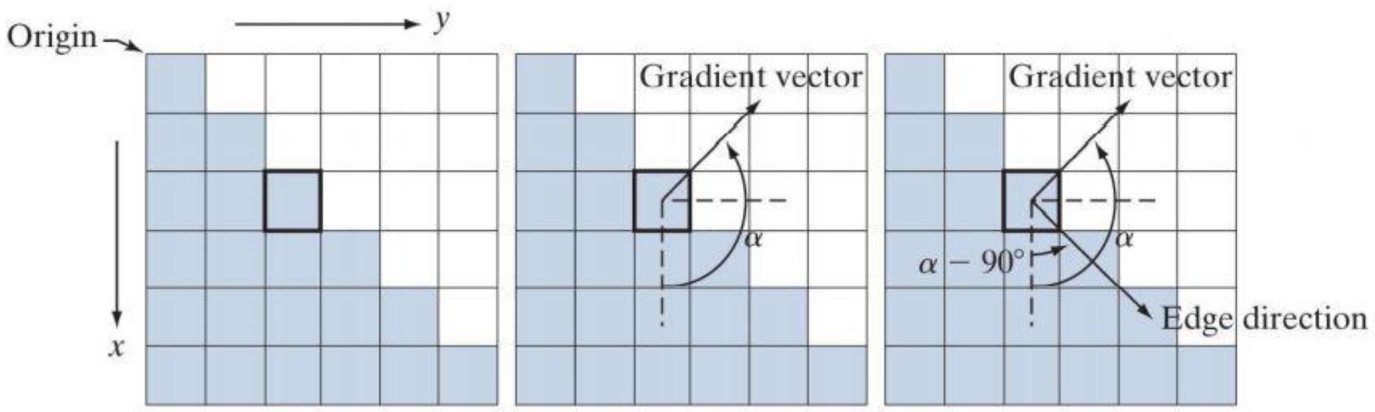
This is the *value* of the rate of change in the direction of the gradient vector at point (*x*, *y*). Note that *M*(*x*, *y*), $\|\nabla f(x,y)\|$, $g_x(x,y)$, and $g_y(x,y)$ are arrays of the same size as *f*, created when *x* and *y* are allowed to vary over all pixel locations in *f*. It is common practice to refer to *M*(*x*, *y*) and $\|\nabla f(x,y)\|$ as the *gradient image*, or simply as the *gradient* when the meaning is clear. The summation, square, and square root operations are elementwise operations, as defined in Section 2.6 .

- The direction of the gradient vector at a point (x, y) is given by

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right] \qquad (10\text{-}18)$$

- Angles are measured in the counterclockwise direction with respect to the x-axis.
- This is also an image of the same size as f, created by the element wise division of g_x and g_y over all applicable values of x and y.
- The following example illustrates, the direction of an edge at a point (x, y) is orthogonal to the direction, of the gradient vector at the point.

EXAMPLE 10.5: Computing the gradient.



a b c

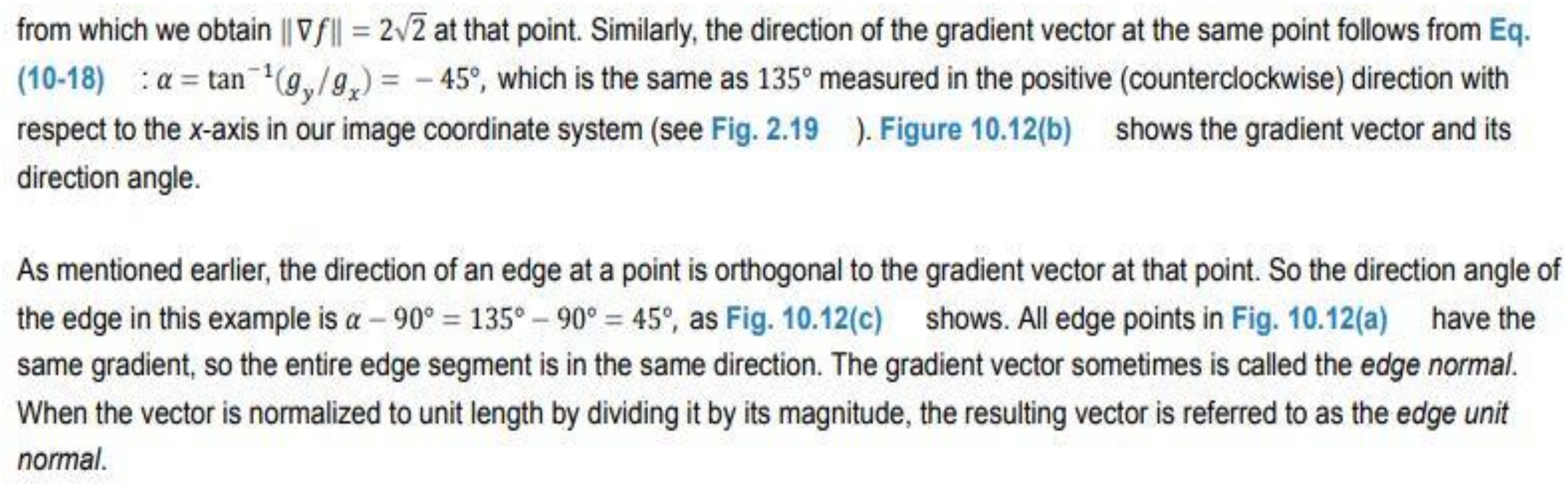
FIGURE 10.12
Using the gradient to determine edge strength and direction at a point. Note that the edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square represents one pixel. (Recall from Fig. 2.19 that the origin of our coordinate system is at the top, left.)

- Figure 10.12(a) shows a zoomed section of an image containing a straight edge segment.
- Each square corresponds to a pixel, and we are interested in obtaining the strength and direction of the edge at the point highlighted with a box.

- The shaded pixels in this figure are assumed to have value 0, and the white pixels have value 1.
- We discuss after this example an approach for computing the derivatives in the x- and y-directions using a neighborhood centered at a point.
- The method consists of subtracting the pixels in the top row of the neighborhood from the pixels in the bottom row to obtain the partial derivative in the x-direction.
- Similarly, we subtract the pixels in the left column from the pixels in the right column of the neighborhood to obtain the partial derivative in the y-direction.

- It then follows, using these differences as our estimates of the partials, that ∂f / ∂x = -2 and ∂f / ∂y = 2 at the point in question. Then,

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$



from which we obtain $\|\nabla f\| = 2\sqrt{2}$ at that point. Similarly, the direction of the gradient vector at the same point follows from Eq. (10-18) : $\alpha = \tan^{-1}(g_y/g_x) = -45°$, which is the same as 135° measured in the positive (counterclockwise) direction with respect to the *x*-axis in our image coordinate system (see Fig. 2.19). Figure 10.12(b) shows the gradient vector and its direction angle.

As mentioned earlier, the direction of an edge at a point is orthogonal to the gradient vector at that point. So the direction angle of the edge in this example is $\alpha - 90° = 135° - 90° = 45°$, as Fig. 10.12(c) shows. All edge points in Fig. 10.12(a) have the same gradient, so the entire edge segment is in the same direction. 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 referred to as the *edge unit normal*.

Gradient Operators

- Obtaining the gradient of an image requires computing the partial derivatives $\partial f / \partial x$ and $\partial f / \partial y$ at every pixel location in the image. For the gradient, we typically use a forward or centered finite difference. Using forward differences we obtain

$$g_x(x,y) = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \qquad (10\text{-}19)$$

and

$$g_y(x,y) = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \qquad (10\text{-}20)$$

- These two equations can be implemented for all values of x and y by filtering f(x, y) with the 1-D kernels in Fig. 10.13

a b

$$\begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$$

FIGURE 10.13
1-D kernels used to implement Eqs. (10-19) and (10-20) .

- When diagonal edge direction is of interest, we need 2-D kernels. The Roberts cross-gradient operators (Roberts [1965]) are one of the earliest attempts to use 2-D kernels with a diagonal preference. Consider the 3x3 region in Fig. 10.14(a) . The Roberts operators are based on implementing the diagonal differences.
- Filter kernels used to compute the derivatives needed for the gradient are often called gradient operators, difference operators, edge operators, or edge detectors.

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5) \qquad (10\text{-}21)$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6) \qquad (10\text{-}22)$$

a
b c
d e
f g

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
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

FIGURE 10.14
A 3×3 region of an image (the z's are intensity values), and various kernels used to compute the gradient at the point labeled z_5.

- Kernels of size are simple conceptually, but they are not as useful for computing edge direction as kernels that are symmetric about their centers, the smallest of which are of size These kernels take into account the nature of the data on opposite sides of the center point, and thus carry more information regarding the direction of an edge. The simplest digital approximations to the partial derivatives using kernels of size are given by

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad (10\text{-}23)$$

and

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

- In this formulation, the difference between the third and first rows of the region approximates the derivative in the x-direction, and the difference between the third and first columns approximate the derivative in the y-direction.

- Intuitively, we would expect these approximations to be more accurate than the approximations obtained using the Roberts operators. Equations (10-22) and (10-23) can be implemented over an entire image by filtering it with the two kernels in Figs. 10.14(d) and (e) . These kernels are called the Prewitt operators.
- A slight variation of the preceding two equations uses a weight of 2 in the center coefficient:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad (10\text{-}24)$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \qquad (10\text{-}25)$$

- It can be demonstrated that using a 2 in the center location provides image smoothing.
- Figures 10.14(f) and (g) show the kernels used to implement Eqs. (10-24) and (10-25) . These kernels are called the Sobel operators

- The Prewitt kernels are simpler to implement than the Sobel kernels, but the slight computational difference between them typically is not an issue.
- The fact that the Sobel kernels have better noise-suppression (smoothing) characteristics makes them preferable because, as mentioned earlier in the discussion of Fig. 10.11 , noise suppression is an important issue when dealing with derivatives.
- Note that the coefficients of all the kernels in Fig. 10.14 sum to zero, thus giving a response of zero in areas of constant intensity, as expected of derivative operators.
- Any of the pairs of kernels from Fig. 10.14 are convolved with an image to obtain the gradient components and at every pixel location.

- Obtaining the magnitude of the gradient requires the computations in Eq. (10-17).
- This implementation is not always desirable because of the computational burden required by squares and square roots, and an approach used frequently is to approximate the magnitude of the gradient by absolute values

$$M(x,y) \approx |g_x| + |g_y| \qquad (10\text{-}26)$$

- This equation is more attractive computationally, and it still preserves relative changes in intensity levels.
- The price paid for this advantage is that the resulting filters will not be isotropic (invariant to rotation) in general.
- However, this is not an issue when kernels such as the Prewitt and Sobel kernels are used to compute g_x and g_y because these kernels give isotropic results only for vertical and horizontal edges.

- This means that results would be isotropic only for edges in those two directions anyway, regardless of which of the two equations is used.
- That is, Eqs. (10-17) and (10-26) give identical results for vertical and horizontal edges when either the Sobel or Prewitt kernels are used

The Canny Edge Detector

- The algorithm is more complex, the performance of the Canny edge detector.

Canny's approach is based on three basic objectives:

- Low error rate. All edges should be found, and there should be no spurious responses.
- Edge points should be well localized. The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
- Single edge point response. The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum.

- This means that the detector should not identify multiple edge pixels where only a single edge point exists.
- white noise is noise having a frequency spectrum that is continuous and uniform over a specified frequency band.
- White Gaussian noise is white noise in which the distribution of amplitude values is Gaussian.
- Gaussian white noise is a good approximation of many real-world situations and generates mathematically tractable models.
- It has the useful property that its values are statistically independent.

$$\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \qquad (10\text{-}34)$$

- where the approximation was only about 20% worse that using the optimized numerical solution

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

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (10\text{-}35)$$

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

$$f_s(x,y) = G(x,y) \star f(x,y) \qquad (10\text{-}36)$$

This operation is followed by computing the gradient magnitude and direction (angle), as discussed earlier.

$$M_s(x,y) = \|\nabla f_s(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)} \qquad (10\text{-}37)$$

And

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right] \qquad (10\text{-}38)$$

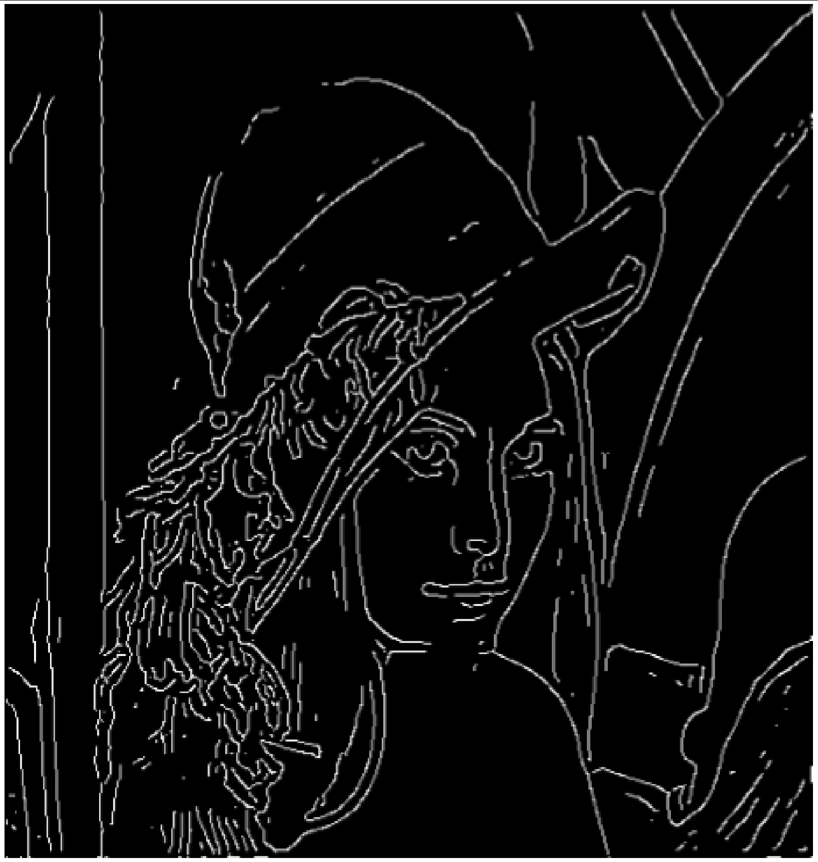
with $g_x(x,y) = \partial f_s(x,y)/\partial x$ and $g_y(x,y) = \partial f_s(x,y)/\partial y$.

with $g_x(x,y) = \partial f_s(x,y)/\partial x$ and $g_y(x,y) = \partial f_s(x,y)/\partial y$. Any of the derivative filter kernel pairs in Fig. 10.14 can be used to obtain $g_x(x,y)$ and $g_y(x,y)$. Equation (10-36) is implemented using an $n \times n$ Gaussian kernel whose size is discussed below. Keep in mind that $\|\nabla f_s(x,y)\|$ and $\alpha(x,y)$ are arrays of the same size as the image from which they are computed.

The Canny edge detection algorithm consists of the following steps:

- Smooth the input image with a Gaussian filter.
- Compute the gradient magnitude and angle images.
- Apply nonmaxima suppression to the gradient magnitude image.
- ➢ Thin multi-pixel wide "ridges" down to single pixel width
- Use double thresholding and connectivity analysis to detect and link edges.
- ➢ Define two thresholds: low and high.
- ➢ Use the high threshold to start edge curves and the low threshold to continue them.

original image



Canny edge Output Image

- The Roberts edge detector

$$\frac{\partial f}{\partial x} = f(i,j) - f(i+1,j+1)$$

$$\frac{\partial f}{\partial y} = f(i+1,j) - f(i,j+1)$$

- This approximation can be implemented by the following masks:

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

(*note*: M_x and M_y are is approximations at $(i+1/2, j+1/2)$)

• The Prewitt edge detector

- Consider the arrangement of pixels about the pixel (i, j):

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i, j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

- The partial derivatives can be computed by:

$$\begin{aligned} M_x &= (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \\ M_y &= (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2) \end{aligned}$$

- The constant c implies the emphasis given to pixels closer to the center of the mask.

- Setting $c = 1$, we get the Prewitt operator:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(*note*: M_x and M_y are approximations at (i, j))

THRESHOLDING

- **Thresholding** is a type of image segmentation, where we change the pixels of an image to make the easier to analyze.
- Suppose that the intensity histogram in Fig.10.32(a) corresponds to an image, f(x, y), composed of light objects on a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes.
- One obvious way to extract the objects from the background is to select a threshold, T, that separates these modes.
- Then, any point (x, y) in the image at which f(x,y) > T is called an object point.
- Otherwise, the point is called a background point. In other words, the segmented image, denoted by g(x, y), is given by

- Remember, f(x, y) denotes the intensity of f at coordinates (x, y).

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \le T \end{cases} \qquad (10\text{-}46)$$

- Although we follow convention in using 0 intensity for the background and 1 for object pixels, any two distinct values can be used in Eq. (10-46).
- When 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 terms local or regional thresholding are 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. Use of these terms is not universal.

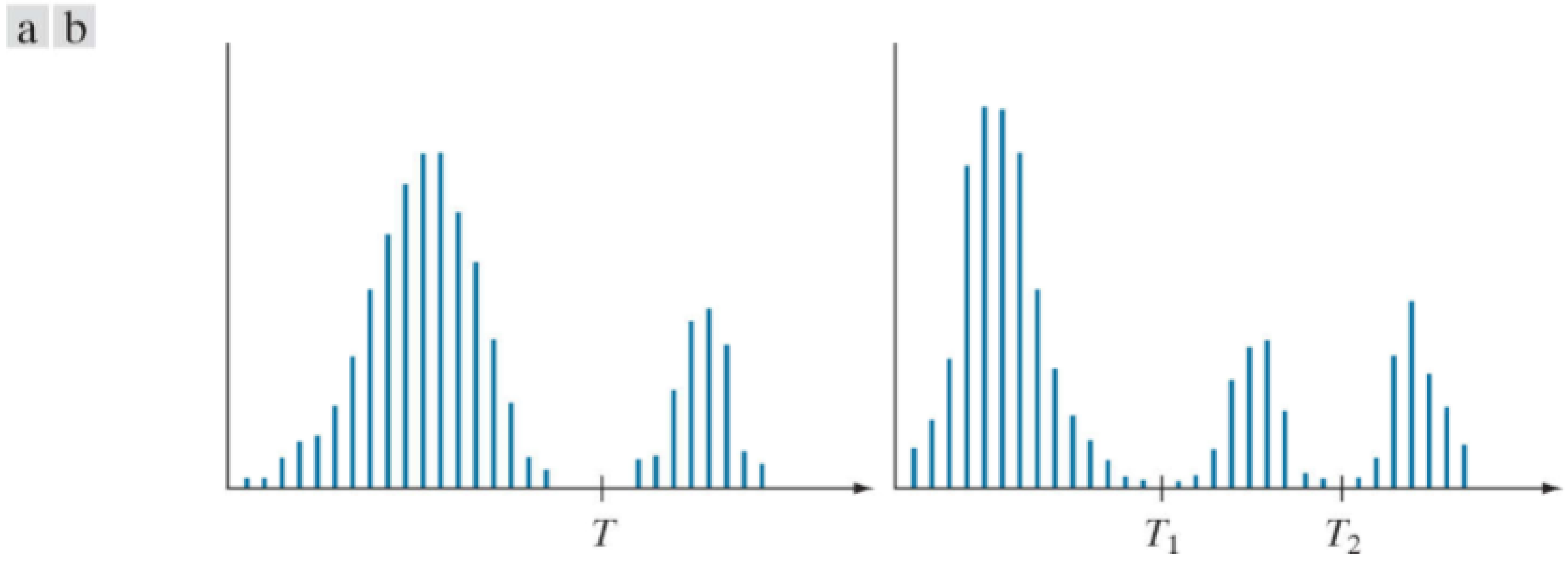


FIGURE 10.32
Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

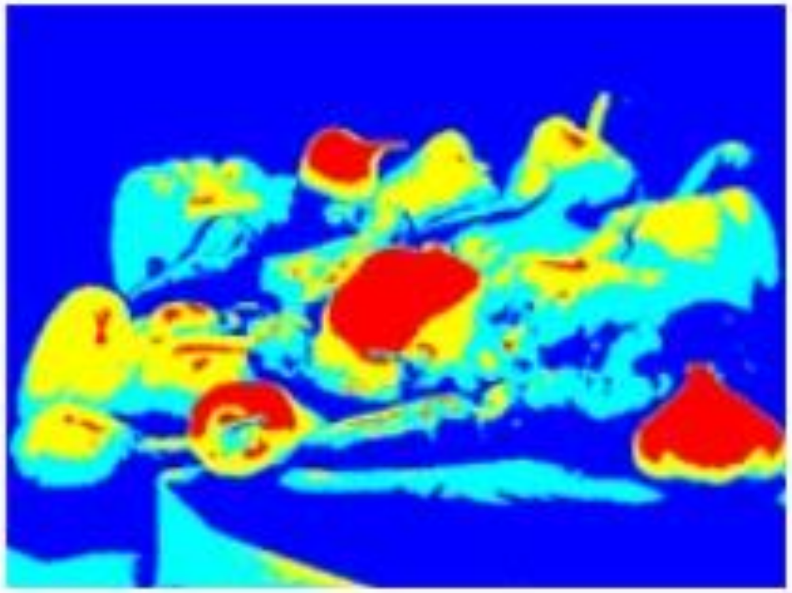
- Figure 10.32(b) shows a more difficult thresholding problem involving a histogram with three dominant modes corresponding, for example, to two types of light objects on a dark background.
- Here, multiple thresholding classifies a point (x, y) as belonging to the background if f(x,y) ≤ T1 to one object class if T1 < f(x,y) ≤ T2 and to the other object class if f(x,y) > T2.

  That is , the segmented image is given by

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \le T_2 \\ c & \text{if } f(x,y) \le T_1 \end{cases} \tag{10-47}$$

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

Segmentation problems requiring more than two thresholds are difficult (or often impossible) to solve, and better results usually are obtained using other methods, such as variable thresholding.



- Based on the preceding discussion, we may infer intuitively that the success of intensity thresholding is related directly to the width and depth of the valley(s) separating the histogram modes.
- In turn, 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 sizes 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

- The simple synthetic image in Fig. 10.33(a) is free of noise, so its histogram consists of two “spike” modes, as Fig.10.33(d) shows.
- Segmenting this image into two regions is a trivial task: we just select a threshold anywhere between the two modes.
- Figure 10.33(b) shows the original image corrupted by Gaussian noise of zero mean and a standard deviation of 10 intensity levels.
- The modes are broader now [see Fig. 10.33(e)], but their separation is enough so that the depth of the valley between them is sufficient to make the modes easy to separate.

- A threshold placed midway between the two peaks would do the job. Figure 10.33(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 Fig. 10.33(f) shows, the situation is much more serious now, as there is no way to differentiate between the two modes.
- Without additional processing (such as the methods discussed later in this section) we have little hope of finding a suitable threshold for segmenting this image.

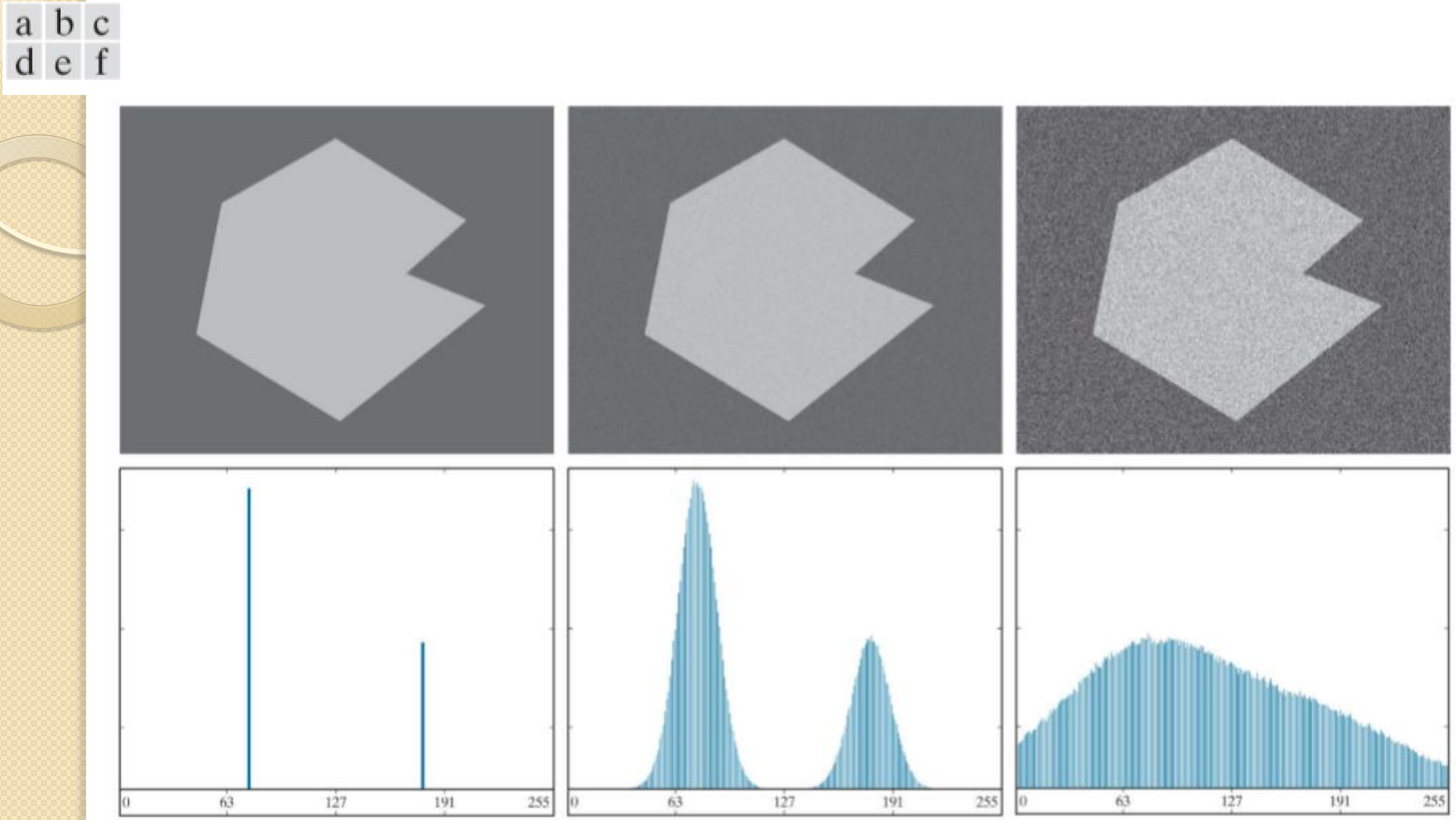


FIGURE 10.33

(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) through (f) Corresponding histograms

The Role of Illumination and Reflectance in Image Thresholding

- Figure 10.34 illustrates the effect that illumination can have on the histogram of an image. Figure 10.34(a) is the noisy image from Fig. 10.33(b) , and Fig.10.34(d) shows its histogram.
- As before, this image is easily segmentable with a single threshold. With reference to the image formation model
- suppose that we multiply the image in Fig.10.34(a) by a nonuniform intensity function, such as the intensity ramp in Fig. 10.37(b) , whose histogram is shown in Fig.10.34(e).
- Figure 10.34(c) shows the product of these two images, and Fig.10.34(f) is the resulting histogram.
- The deep valley between peaks was corrupted to the point where separation of the modes without additional processing (to be discussed later in this section) is no longer possible.

- Similar results would be obtained if the illumination was perfectly uniform, but the reflectance of the image was not, as a results, for example, of natural reflectivity variations in the surface of objects and/or background.
- In theory, the histogram of a ramp image is uniform. In practice, the degree of uniformity depends on the size of the image and number of intensity levels.
- The important point is that illumination and reflectance play a central role in the success of image segmentation using thresholding or other segmentation techniques.
- Therefore, controlling these factors when possible should be the first step considered in the solution of a segmentation problem.
- There are three basic approaches to the problem when control over these factors is not possible.

- The first is to correct the shading pattern directly.
- For example, nonuniform (but fixed) illumination can be corrected by multiplying the image by the inverse of the pattern, which can be obtained by imaging a flat surface of constant intensity.
- The second is to attempt to correct the global shading pattern via processing using, for example, the top-hat transformation.
- The third approach is to "work around" non uniformities using variable thresholding,

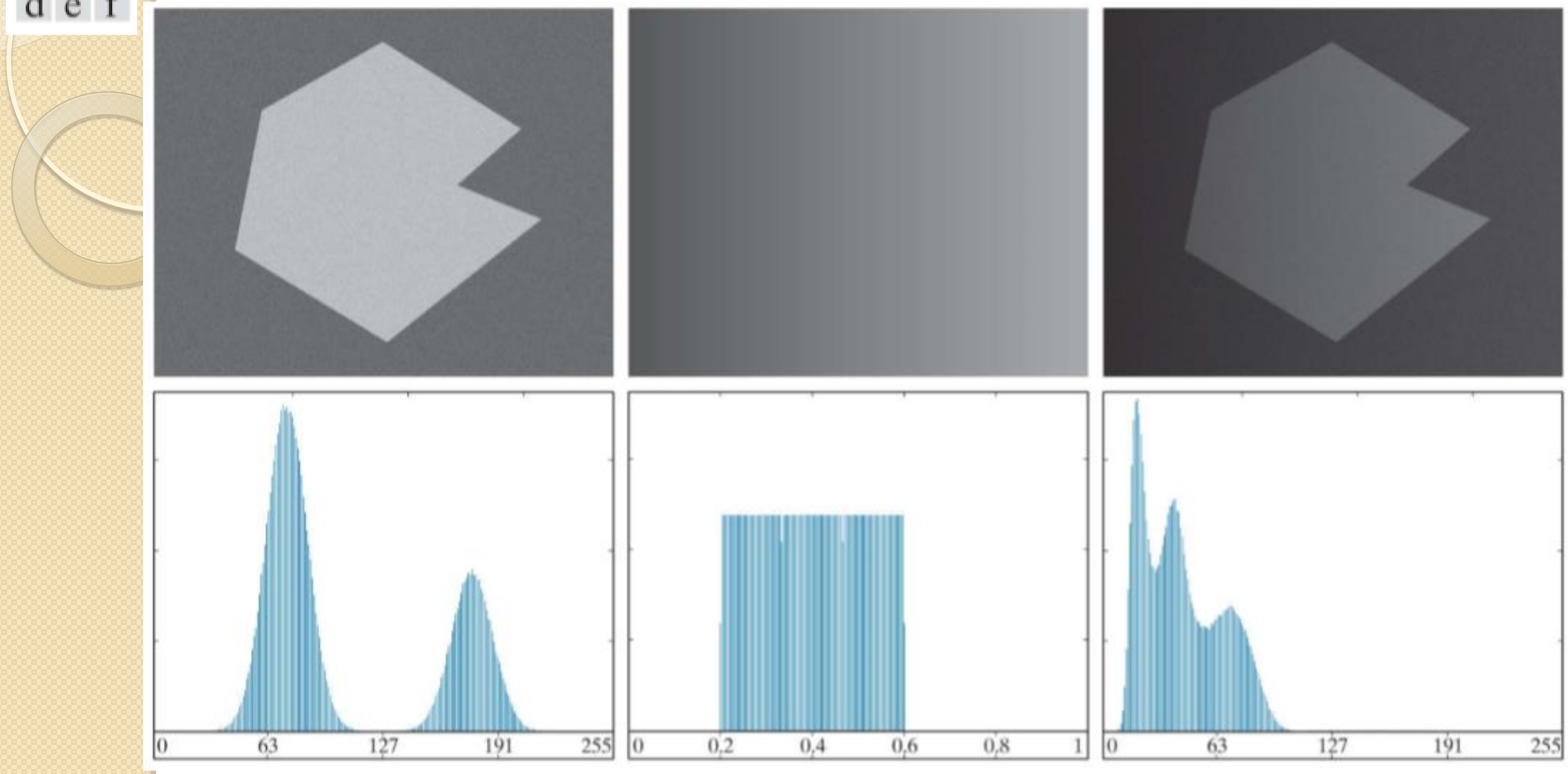


FIGURE 10.34

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

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.
- In most applications, there is usually enough variability between images that, even if global thresholding is a suitable approach, an algorithm capable of estimating 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 in Eq. (10-46) . This will produce two groups of pixels: G1,consisting of pixels with intensity values > T; and G2, consisting of pixels with values ≤ T.
3. Compute the average (mean) intensity values m1 and m2 for the pixels in G1 and G2 respectively.
4. Compute a new threshold value midway between m1 and m2 : $T = \frac{1}{2}(m_1 + m_2)$
5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined value, ΔT.

- The algorithm is stated here in terms of successively thresholding the input image and calculating the means at each step, because it is more intuitive to introduce it in this manner.
- However, it is possible to develop an equivalent (and more efficient) procedure by expressing all computations in the terms of the image histogram.
- The preceding 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 stop iterating when the changes in threshold values is small.
- The initial threshold must be chosen greater than the minimum and less than the maximum intensity level in the image (the average intensity of the image is a good initial choice for T).

EXAMPLE 10.13: Global thresholding

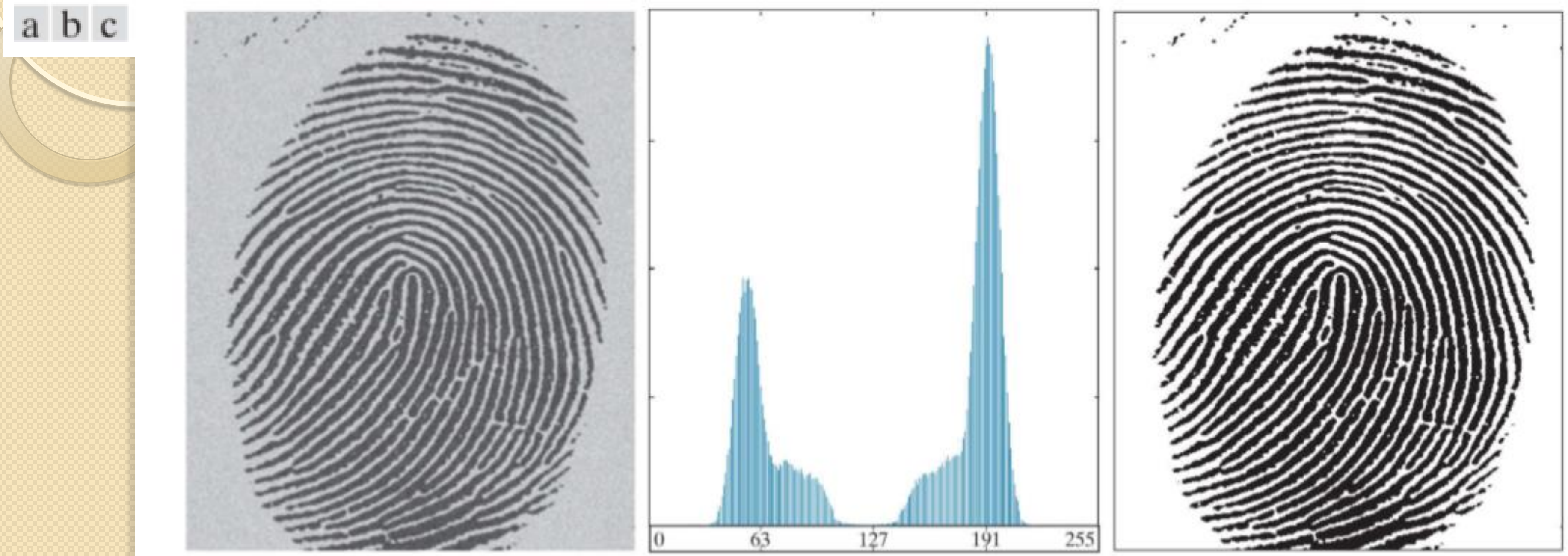


FIGURE10.35
(a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (thin image border added for clarity).

- Figure 10.35 shows an example of segmentation using the preceding iterative algorithm.
- Figure 10.35(a) is the original image and Fig.10.35(b) is the image histogram, showing a distinct valley.
- Application of the basic global algorithm resulted in the threshold after three iterations, starting with T=125 equal to the average intensity of the image, and using ΔT=0
- Figure 10.35(c) shows the result obtained using T=125 to segment the original image.
- As expected from the clear separation of modes in the histogram, the segmentation between object and background was perfect.

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 incurred in assigning pixels to two or more groups (also called classes).
- This problem is known to have an elegant closed-form solution known as the Bayes decision function.
- The solution is based on only two parameters:
  - The probability density function (PDF) of the intensity levels of each class.
  - The probability that each class occurs in a given application.
- Unfortunately, estimating PDFs is not a trivial matter, so the problem usually is simplified by making workable assumptions about the form of the PDFs, such as assuming that they are Gaussian functions.

- Even with simplifications, the process of implementing solutions using these assumptions can be complex and not always well-suited for real-time applications.
- The approach in the following discussion, called Otsu’s method, is an attractive alternative.
- The method is optimum in the sense that it maximizes the between-class variance, a well-known measure used in statistical discriminant analysis.
- The basic idea is that properly 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.
- In addition to its optimality, Otsu’s method has the important property that it is based entirely on computations performed on the histogram of an image.

- Let {0,1,2…L-1} denote the set of L distinct integer intensity levels in a digital image of size MxN pixels and let ni denote the number of pixels with intensity i.
- The total number MN1 of pixels in the image is MN= n0 +n1+n2+…..+nL-1.
- The normalized histogram has components Pi = ni/MN, from which it follows that

$$\sum_{i=0}^{L-1} p_i = 1 \qquad p_i \geq 0 \tag{10-48}$$

- Now, suppose that we select a threshold T(k) = k, 0<k<L-1 and use it to threshold the input image into two classes, C1 and C2
- Where C1 consists of all the pixels in the image with intensity values in the range [0, k] and C2 consists of the pixels with values in the range [K+1,L-1].

- Using this threshold, the probability, P1(k) that a pixel is assigned to (i.e., threshold into) class C1 is given by the cumulative sum

$$P_1(k) = \sum_{i=0}^{k} p_i \qquad (10\text{-}49)$$

- Viewed another way, this is the probability of class C1 occurring. For example, if we set k=0, the probability of class C1 having any pixels assigned to it is zero. Similarly, the probability of class C2 occurring is

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k) \qquad (10\text{-}50)$$

- the mean intensity value of the pixels in C1 is

$$m_1(k) = \sum_{i=0}^{k} iP(i/c_1) = \sum_{i=0}^{k} iP(c_1/i)P(i)/P(c_1) \qquad (10\text{-}51)$$

$$= \frac{1}{P_1(k)} \sum_{i=0}^{k} ip_i$$

- where p1(k) is given by Eq. (10-49) . The term P(i/c1) in Eq. (10-51) is the probability of intensity value i, given that i comes from class C1.
- The rightmost term in the first line of the equation follows from Bayes' formula

$$P(A/B) = P(B/A)P(A)/P(B)$$

- The second line follows from the fact that P(C1/I) the probability of C1 given i, is 1 because we are dealing only with values of i from class C1.
- Also, P(i) is the probability of the ith value, which is the ith component of the histogram Pi, Finally, P(Ci) is the probability of Class C1 which, from Eq. (10-49) , is equal to P1 (k)

- Similarly, the mean intensity value of the pixels assigned to class C2 is

$$m_2(k) = \sum_{i=k+1}^{L-1} iP(i/c_2) \tag{10-52}$$

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

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

$$m(k) = \sum_{i=0}^{k} ip_i \tag{10-53}$$

- and the average intensity of the entire image (i.e., the global mean) is given by

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{10-54}$$

- The validity of the following two equations can be verified by direct substitution of the preceding results:

$$P_1 m_1 + P_2 m_2 = m_G \tag{10-55}$$

and

$$P_1 + P_2 = 1 \tag{10-56}$$

- where we have omitted the ks temporarily in favor of notational clarity.
- In order to evaluate the effectiveness of the threshold at level k, we use the normalized, dimensionless measure

$$\eta = \frac{\sigma_B^2}{\sigma_G^2} \tag{10-57}$$

Where σ_G^2 is the global variance

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i \tag{10-58}$$

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

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \qquad (10\text{-}59)$$

This expression can also be written as

$$\begin{aligned}\sigma_B^2 &= P_1 P_2 (m_1 - m_2)^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)}\end{aligned} \qquad (10\text{-}60)$$

- The first line of this equation follows from Eqs.(10-55) , (10-56) , and (10-59).
- The second line follows from Eqs. (10-50) through (10-54)
- This form is slightly more efficient computationally because the global mean, mG, is computed only once so only two parameters, m1 and P1 need to be computed for any value of k .

- The first line in Eq. (10-60) indicates that the farther the two means m1 and m2 are from each other, the larger σ_B^2 will be, implying that the between-class variance is a measure of separability between classes.
- Because σ_G^2 is a constant, it follows that n also is a measure of separability, 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, as stated earlier. Note that Eq.(10-57) assumes implicitly that σ_G^2 >0.
- This variance can be zero only when all the intensity levels in the image are the same, which implies the existence of only one class of pixels.
- This in turn means that n=0 for a constant image because the separability of a single class from itself is zero.

- Reintroducing k, we have the final results:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \tag{10-61}$$

and

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \tag{10-62}$$

- Then, the optimum threshold is the value, K* that maximizes $\sigma_B^2(k)$:

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k) \tag{10-63}$$

- To find K* we simply evaluate this equation for all integer values of k and select the value of K that yielded the maximum $\sigma_B^2(k)$:
-

- 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.

- It can be shown that a maximum always exists, subject to the condition 0<p1(k)<1 Evaluating Eqs.(10-62) and (10-63) for all values of k is a relatively inexpensive computational procedure, because the maximum number of integer values that k can have is L, which is only 256 for 8-bit images,.
- Once k* has been obtained, input image f(x, y) is segmented as before:

$$g(x,y) = \begin{cases} 1 & \text{i } f(x,y) > k^* \\ 0 & \text{i } f(x,y) \le k^* \end{cases} \qquad (10\text{-}64)$$

- for x=0,1,2..M-1 and y=0,1,2..N-1 Note that all the quantities needed to evaluate Eq.(10-62) are obtained using only the histogram of f(x, y).

- In addition to the optimum threshold, other information regarding the segmented image can be extracted from the histogram.
- For example, P1(k*) and P2(K*) the class probabilities evaluated at the optimum threshold, indicate the portions of the areas occupied by the classes (groups of pixels) in the thresholded image.
- Similarly, the means m1(K*) and m2(k*) are estimates of the average intensity of the classes in the original image.
- In general, the measure in Eq.(10-61) has values in the range

$$0 \leq \eta(k) \leq 1 \qquad (10\text{-}65)$$

for values of k in the range [0,L-1].

- When evaluated at the optimum threshold K*, this measure is a quantitative estimate of the separability of classes, which in turn gives us an idea of the accuracy of thresholding a given image with K*
- The lower bound in Eq. (10-65) is attainable only by images with a single, constant intensity level.
- The upper bound is attainable only by two-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. Denote the components of the histogram by Pi, i=0,1,2…L-1.
2. Compute the cumulative sums, P1(k), for k=0,1,2…L-1 using Eq.(10-49).

3. Compute the cumulative means, m(k), for k=0,1,2,..L-1 using Eq. (10-53).

4. Compute the global mean, m_G, using Eq. (10-54) .

5. Compute the between-class variance term $\sigma_B^2(k)$, for k=0,1,2…L-1, using Eq. (10-62).

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.

1. Compute the global variance σ_G^2, using Eq. (10-58), and and then obtain the separability measure n*.by evaluating eq(10.41) with k=k*

EXAMPLE 10.14: Optimum global thresholding using Otsu's method

a	b
c	d

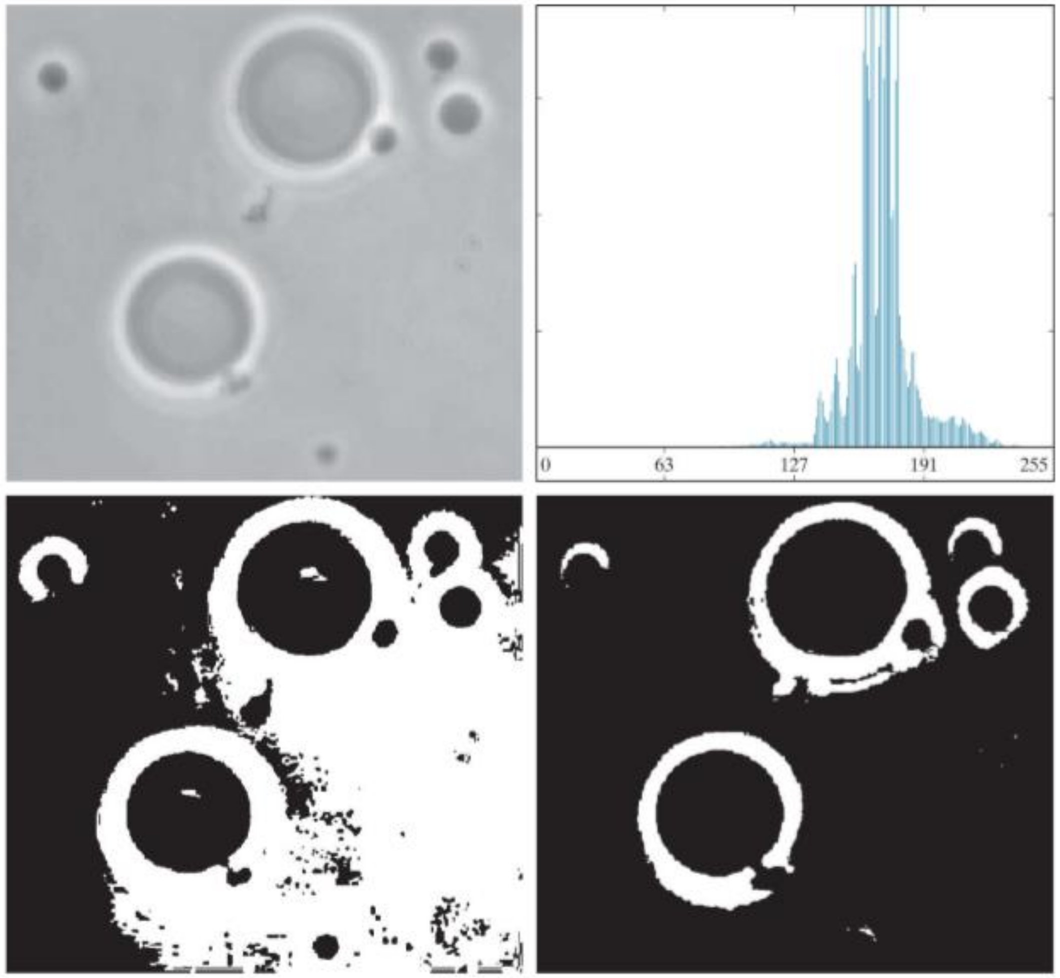


FIGURE10.36 (a) Original image. (b) Histogram (high peaks were clipped to highlight details in the lower values). (c) Segmentation result using the basic global algorithm. (d) Result using Otsu's method

- Figure 10.36(a) shows an optical microscope image of polymersome cells.
- These are cells artificially engineered using polymers.
- They are invisible to the human immune system and can be used, for example, to deliver medication to targeted regions of the body.
- Figure 10.36(b) shows the image histogram. The objective of this example is to segment the molecules from the background.
- Figure 10.36(c) is the result of using the basic global thresholding algorithm discussed earlier.
- Because the histogram has no distinct valleys and the intensity difference between the background and objects is small, the algorithm failed to achieve the desired segmentation.
- Figure 10.36(d) shows the result obtained using Otsu's method.

- This result obviously is superior to Fig. 10.36(c).
- The threshold value computed by the basic algorithm was 169, while the threshold computed by Otsu’s method was 182, which is closer to the lighter areas in the image defining the cells.

Region Based Segmentation

- In this section, we discuss segmentation techniques that are based on finding the regions directly.

1. **Region Growing :**

  - Region growing is a procedure that groups pixels or sub region into larger regions.
  - The simplest of these approaches is pixel aggregation, which starts with a set of "seed" points and from these grows regions by appending to each seed points those neighboring pixels that have similar properties (such as graylevel, texture, color, shape).
  - Region growing based techniques are better than the edge-based techniques in noisy images where edges are difficult to detect.

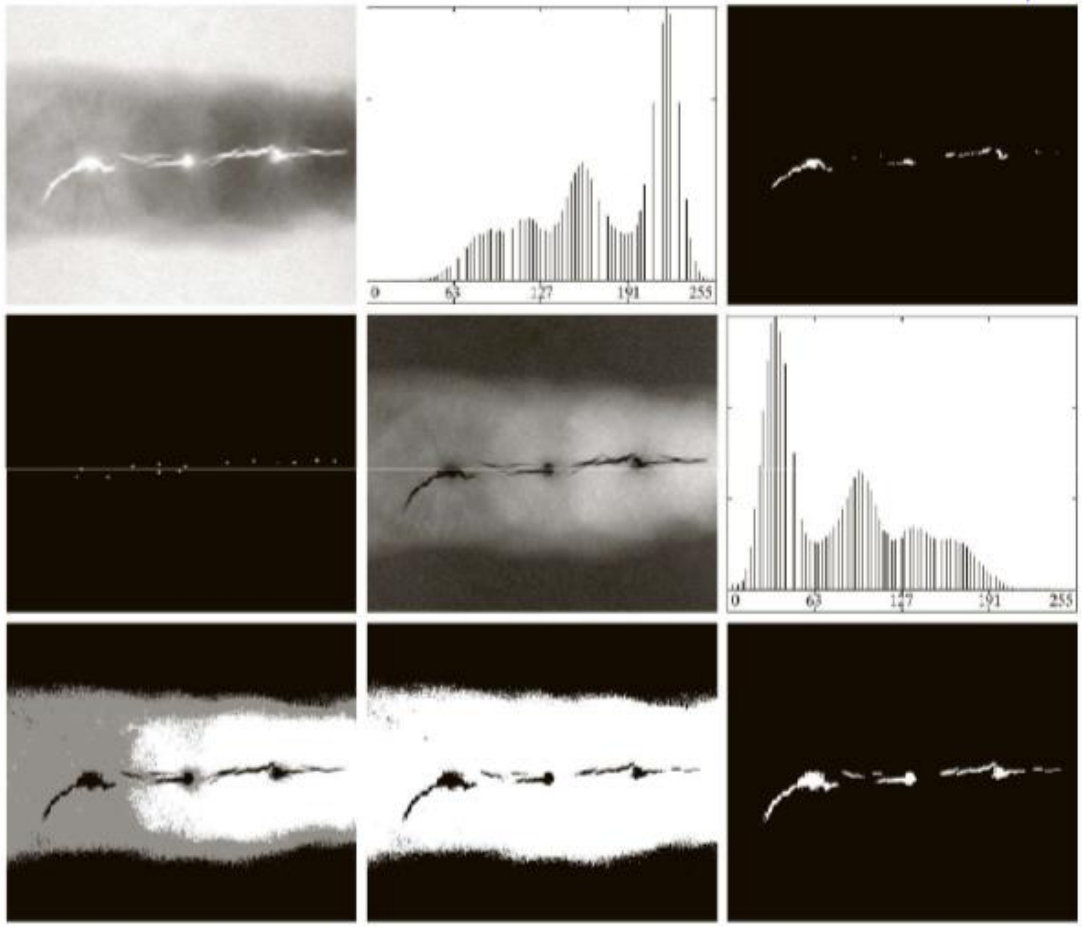
- Let: f(x ,y) denote an input image array;
- S(x ,y) denote a seed array containing 1s at the locations of seed points and 0s elsewhere; and Q denote a predicate to be applied at each location(x,y).
- Arrays f and S are assumed to be of the same size. A basic region-growing algorithm based on 8-connectivity 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. From an image fQ such that , at a pair of coordinates (x,y), let fQ(x,y)=1 if the input image satisfies the given predicate, Q, at those coordinates; otherwise, let fQ(x,y) = 0.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in fQ that are 8-connected to the seed point.

4. Label each connected component in g with a different region label(e.g.,1.2.3….). This is the segmented image obtained by region growing.

criteria:

1. the absolute gray-level difference between any pixel and the seed has to be less than 65
2. the pixel has to be 8-connected to at least one pixel in that region (if more, the regions are merged)



a	b	c
d	e	f
g	h	i

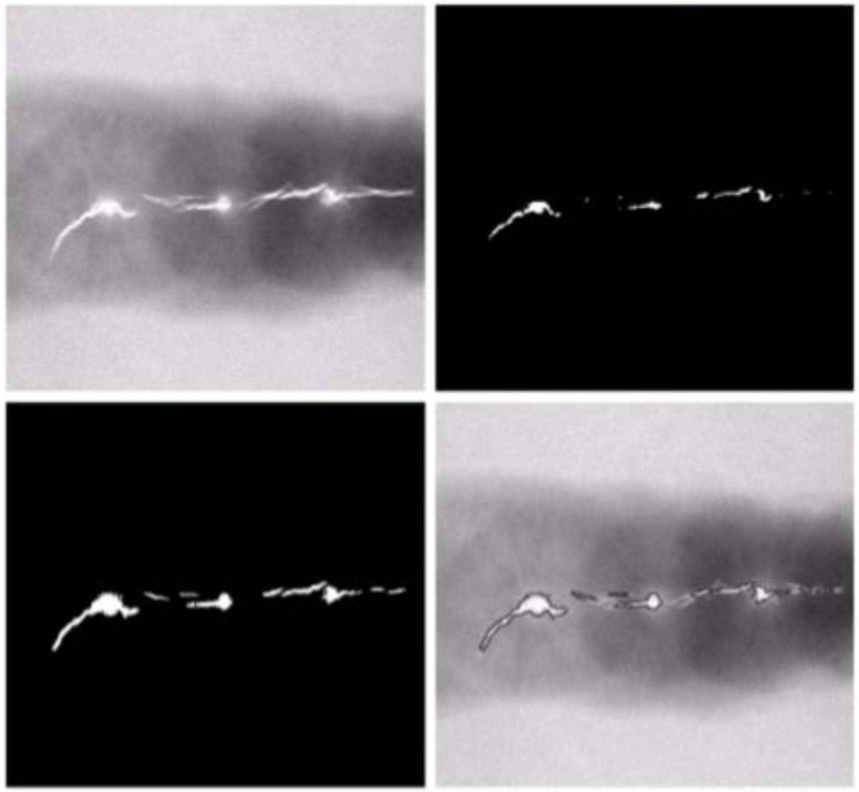
FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference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

select all seed points with gray level 255

Region Growing

criteria:

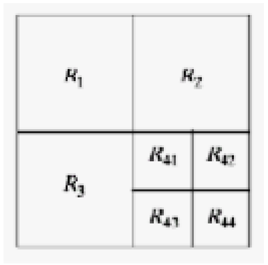
1. the absolute gray-level difference between any pixel and the seed has to be less than 65
2. the pixel has to be 8-connected to at least one pixel in that region (if more, the regions are merged)



a b
c d

FIGURE 10.40 (a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

Region Splitting and Merging

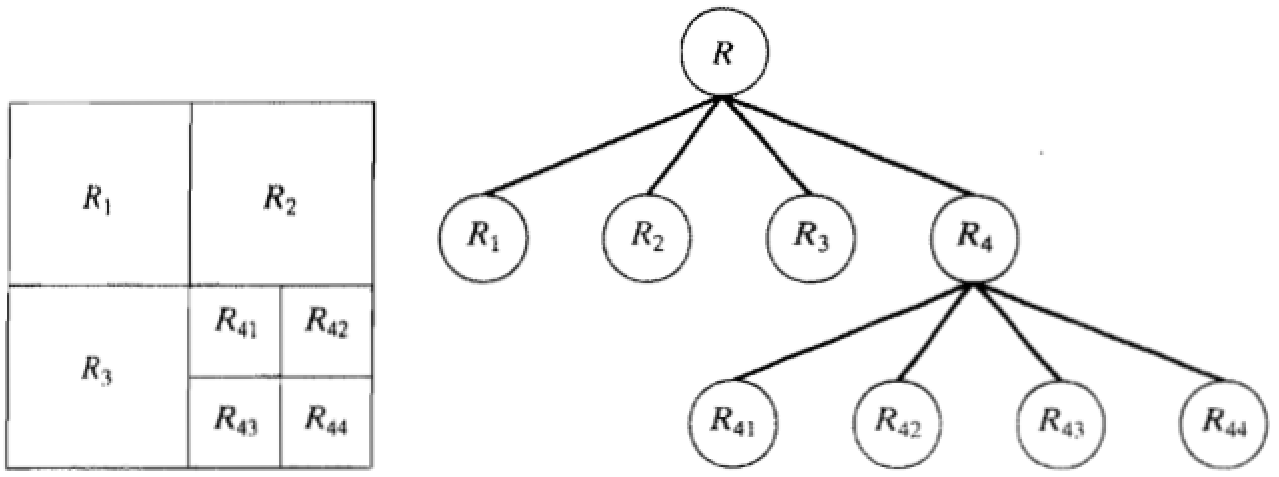


Region Splitting

- Region growing starts from a set of seed points.
- An alternative is to start with the whole image as a single region and subdivide the regions that do not satisfy a condition of homogeneity.

Region Merging

- Region merging is the opposite of region splitting.
- Start with small regions (e.g. 2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).
- Typically, splitting and merging approaches are used iteratively.



a b

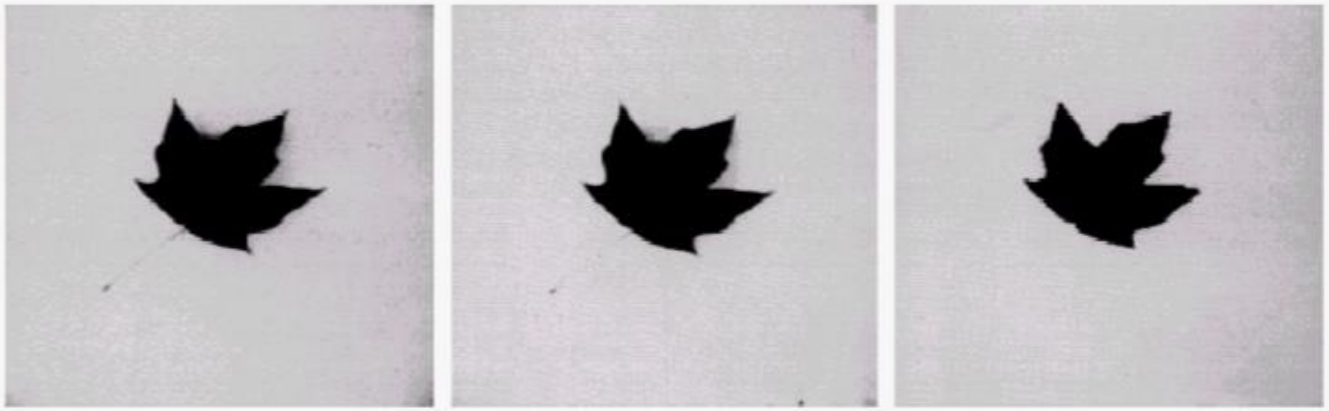
FIGURE 10.52 (a) Partitioned image. (b) Corresponding quadtree. *R* represents the entire image region.

Quadtree

1. Split into 4 disjoint quadrants any region R_i for which $P(R_i) = \text{FALSE}$
2. Merge any adjacent region R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$
3. Stop when no further merging or splitting is possible.

a b c

FIGURE 10.43 (a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).



$P(R_i)$ = TRUE if at least 80% of the pixels in R_i have the property $|z_j - m_i| \leq 2\sigma_i$, where

- z_j is the gray level of the jth pixel in R_i
- m_i is the mean gray level of that region
- σ_i is the standard deviation of the gray levels in R_i