Introduction to Image Processing

Prof. Alexandre Zaghetto
http://alexandre.zaghetto.com
zaghetto@unb.br

University of Brasília
Department of Computer Science
LISA: Laboratory of Imagens, Signals and Acoustics

Topic 06
Image Segmentation

1. Fundamentals

• Let *R* represent the **entire spatial region** occupied by an image. We may view image **segmentation** as a process that **partitions** *R* into *n* **subregions**, R_1, R_2,…,R_n, such that,

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

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

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

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

(e) $Q(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j.

1. Fundamentals

• Let R represent the **entire spatial region** occupied by an image. We may view image **segmentation** as a process that **partitions** R into n **subregions**, R_1, R_2,…,R_n, such that,

a. Every pixel must be in a region.

b. Points in a region must be connected in some predefined sense (e.g., 4- or 8-connected).

c. The regions must be disjoint.

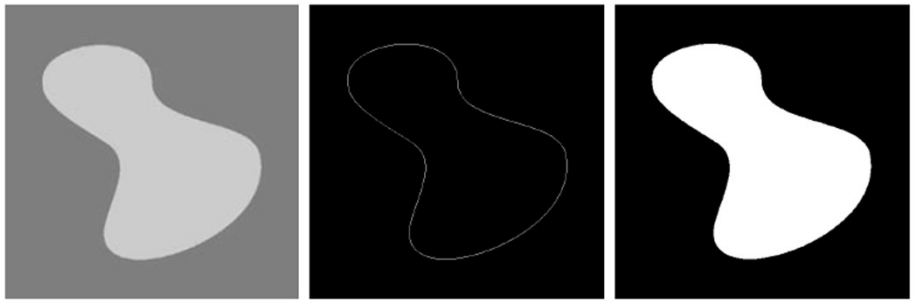
d. Pixels in a segmented region must satisfy an specific property (e.g., all pixels must have the same intensity level).

e. Two adjacent regions must be different in the sense of predicate Q.

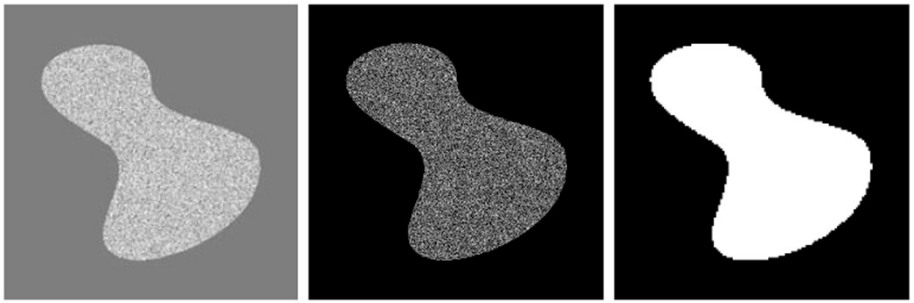
1. Fundamentals

• Segmentation algorithms for monochrome images generally are based on one of two basic categories:

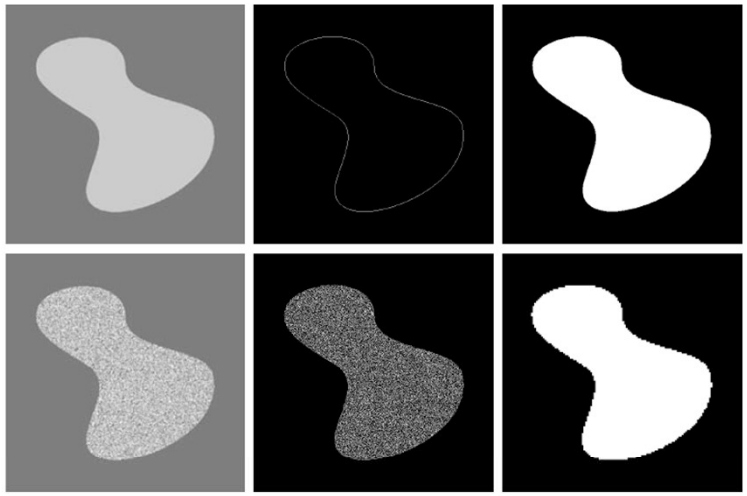
- **Edge-based** segmentation (based on discontinuity)



- **Region-based** segmentation (based on similarity)



1. Fundamentals

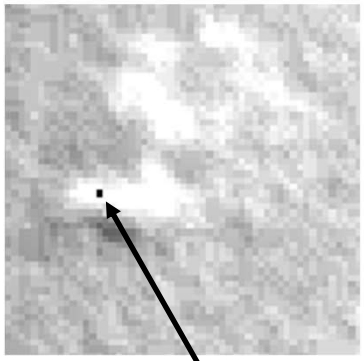


a b c
d e f

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.

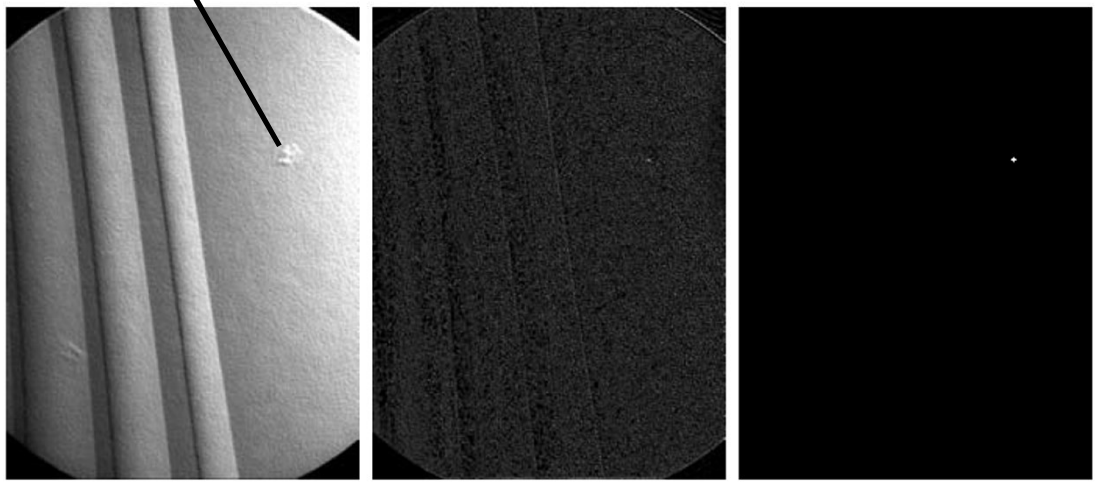
2. Point, Line, and Edge Detection

- Detection of Isolated Points



1	1	1
1	−8	1
1	1	1

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



a
b c d

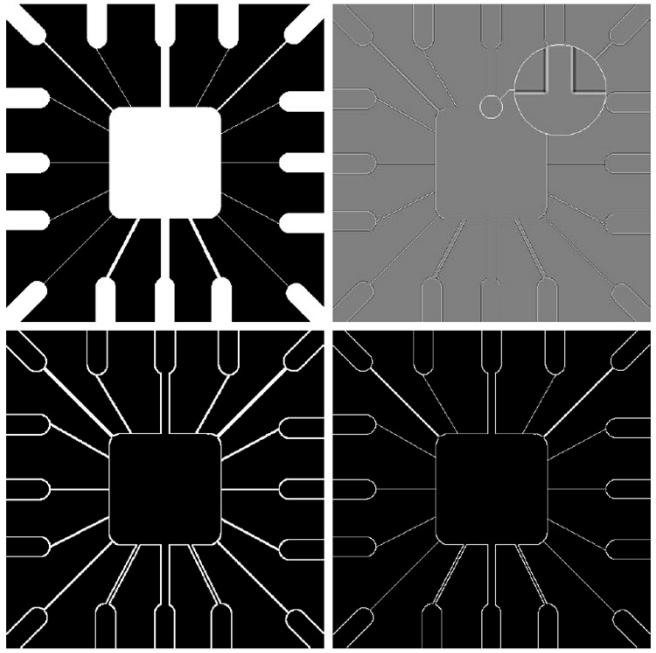
FIGURE 10.4
(a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

2. Point, Line, and Edge Detection

- Line Detection

  - We can use the Laplacian mask for line detection also, keeping in mind that the double-line effect of the second derivative must be handled properly.

1	1	1
1	−8	1
1	1	1



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.

2. Point, Line, and Edge Detection

- Line Detection

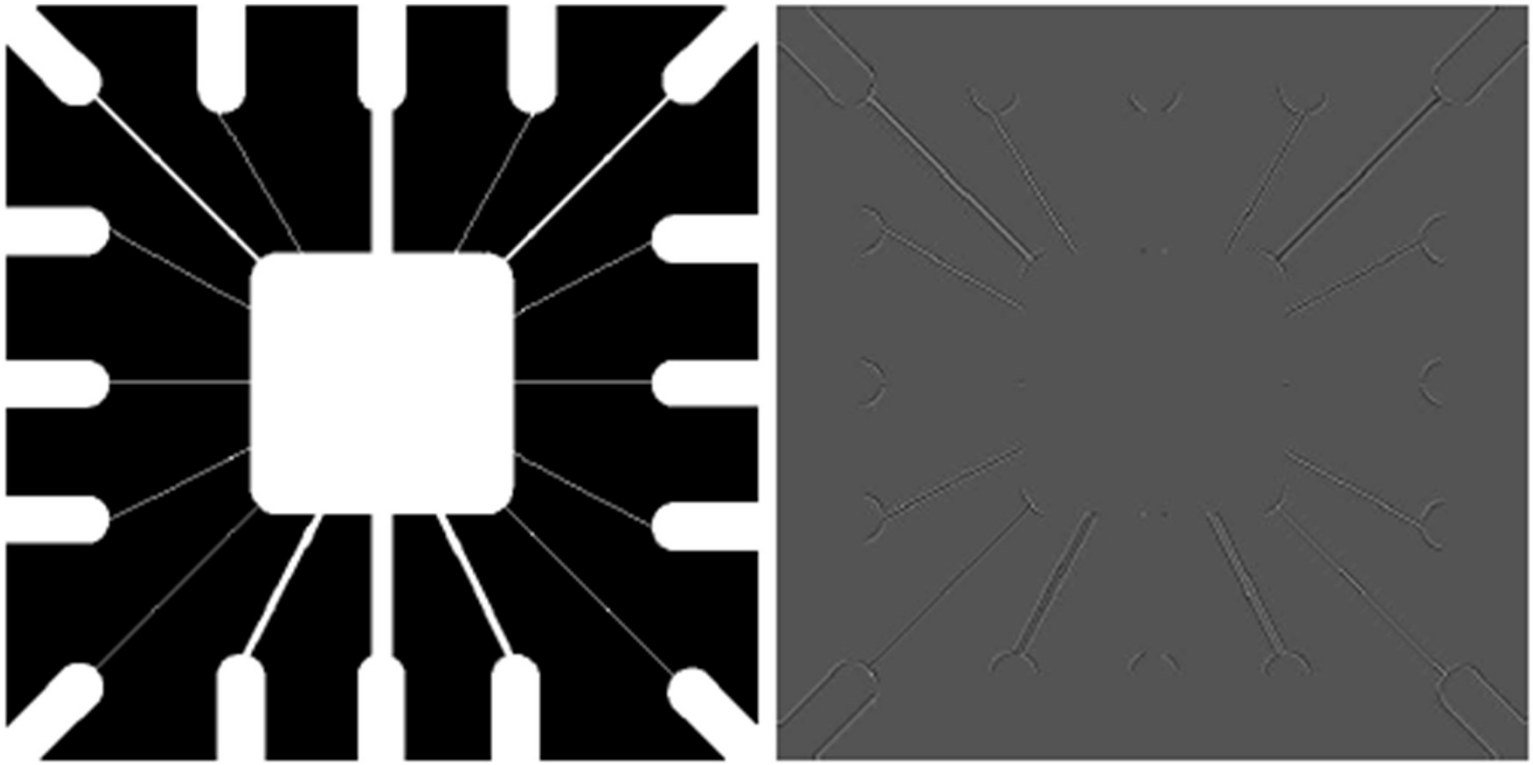
  - The Laplacian detector used before is **isotropic**, so its response is **independent of direction**.

  - Often, interest lies in detecting lines in **specified directions**.

−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

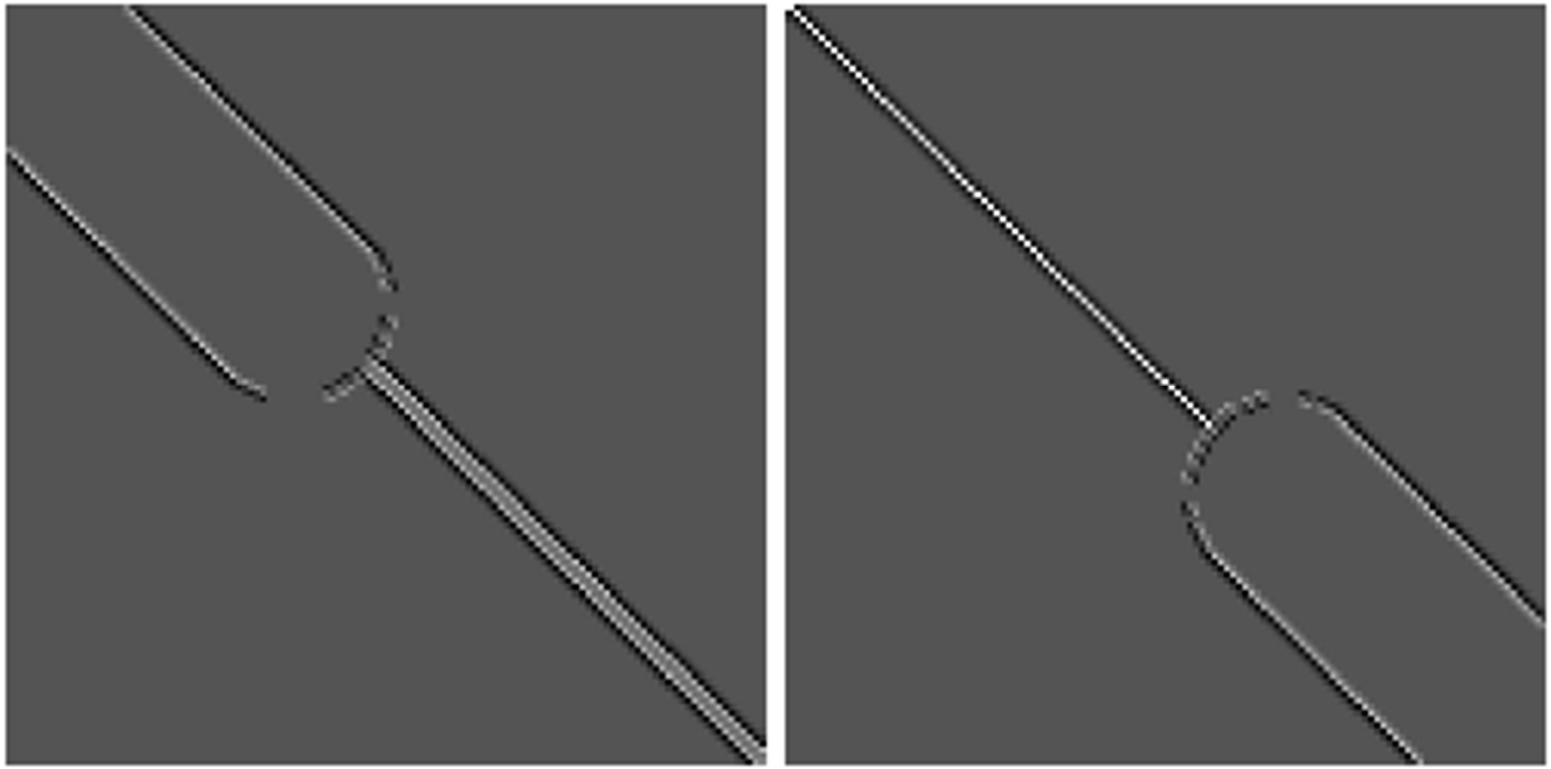
2. Point, Line, and Edge Detection

- Line Detection



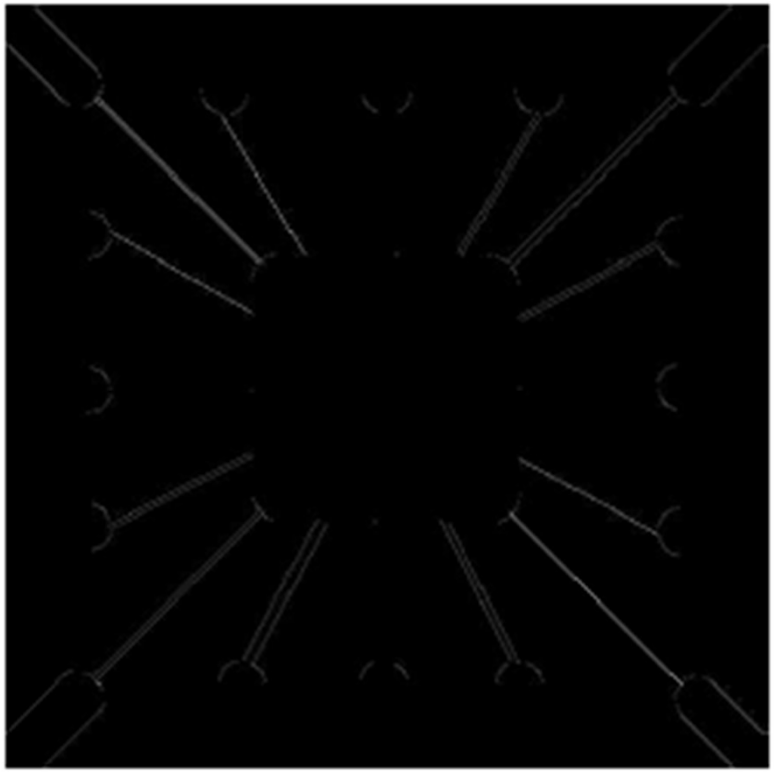
2. Point, Line, and Edge Detection

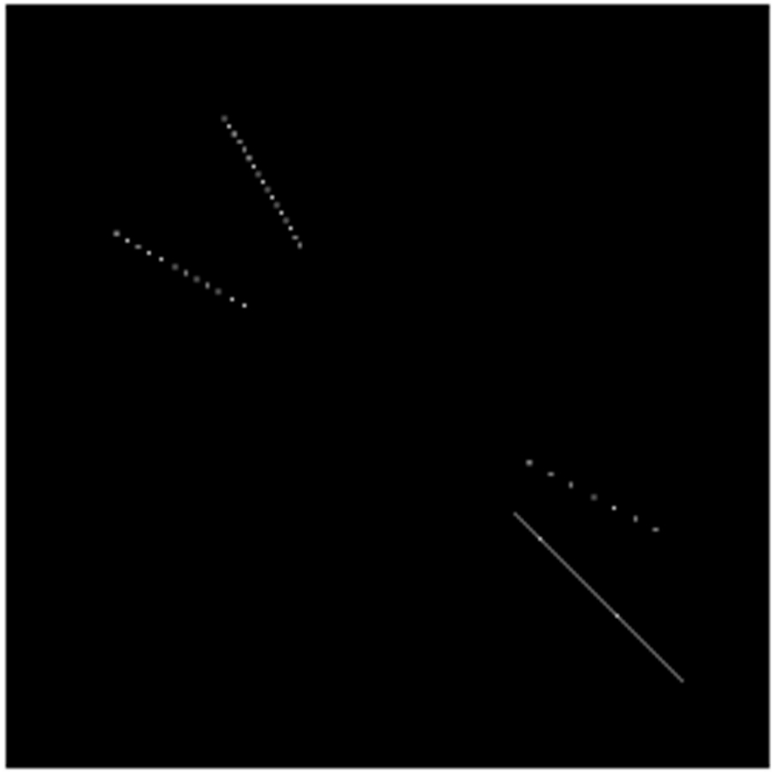
- Line Detection



2. Point, Line, and Edge Detection

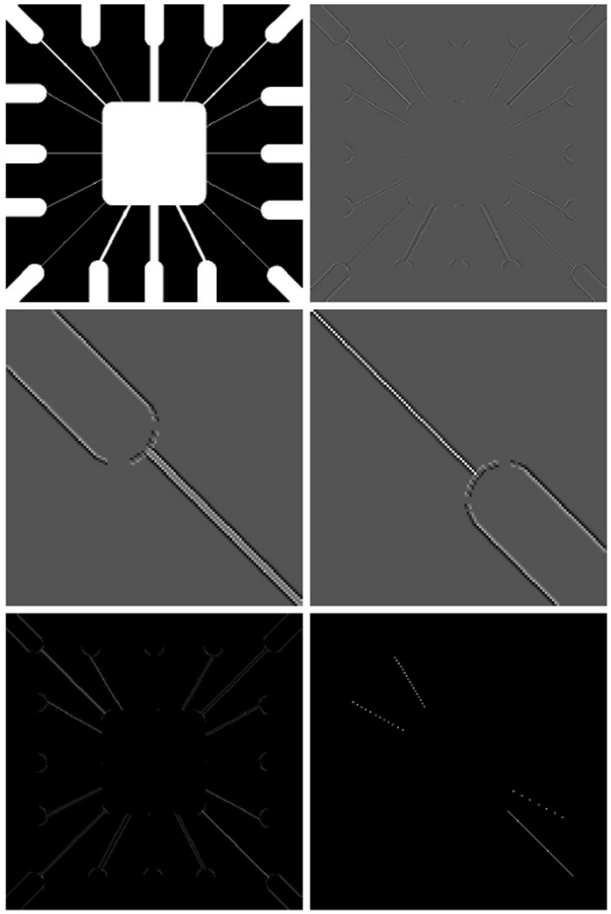
- Line Detection





2. Point, Line, and Edge Detection

- Line Detection



a b
c d
e f

FIGURE 10.7
(a) Image of a wire-bond template. (b) Result of processing with the +45° 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.)

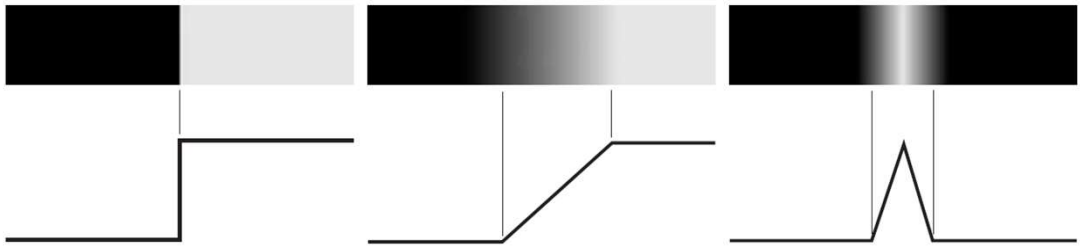
2. Point, Line, and Edge Detection

- Edge Models

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

  - In practice, digital images have edges that are blurred and noisy.

  - In such situations, edges are more closely modeled as having a smooth intensity profile.



a b c

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

2. Point, Line, and Edge Detection

- Edge Models

  - The models allow us to write mathematical expressions for edges in the development of image processing algorithms.

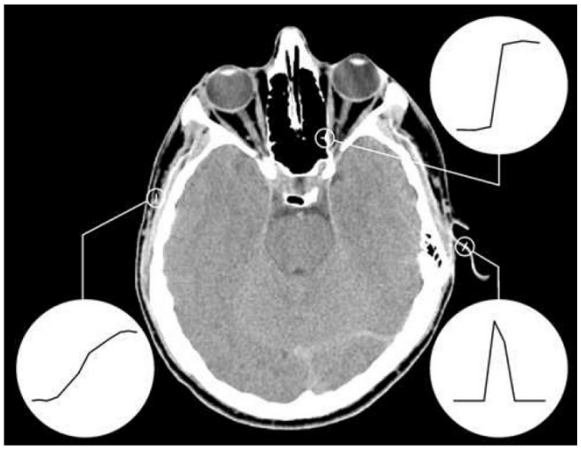
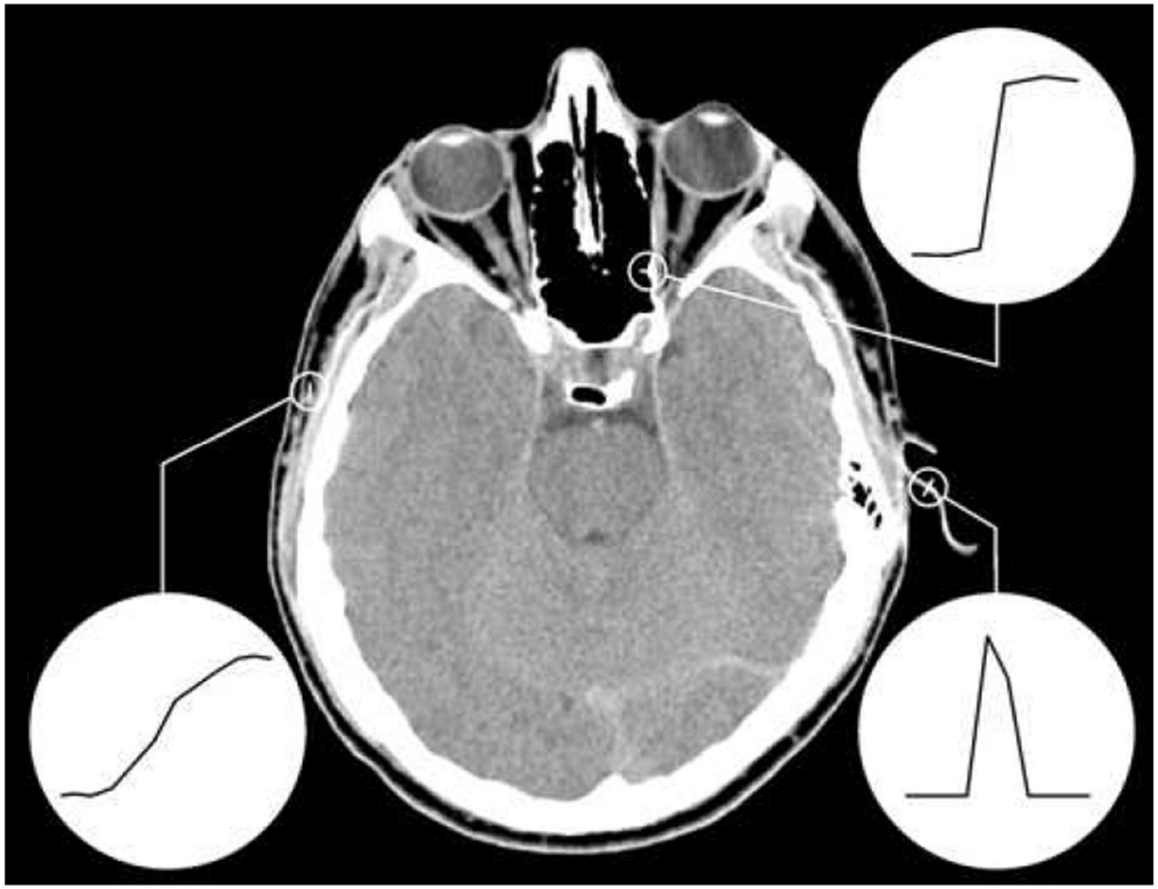


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

2. Point, Line, and Edge Detection

- Edge Models

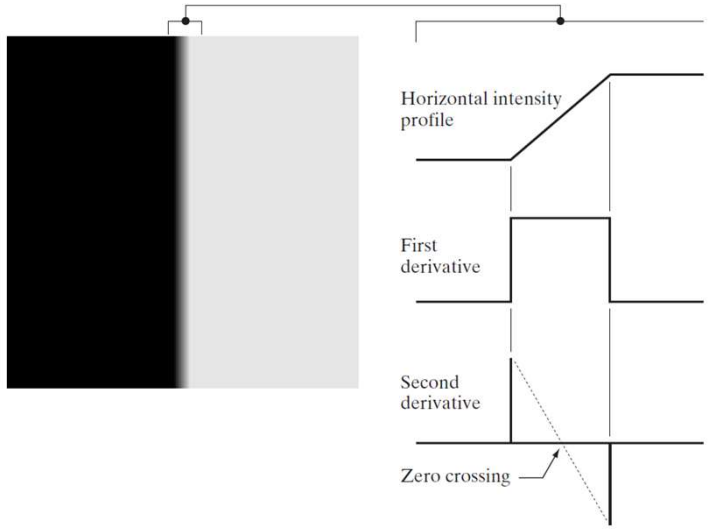


2. Point, Line, and Edge Detection

- Edge Models

  - The **magnitude** of the **first derivative** can be used to detect an edge at a point in an image.

  - 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



2. Point, Line, and Edge Detection

- Edge Models
  - The effect of noise

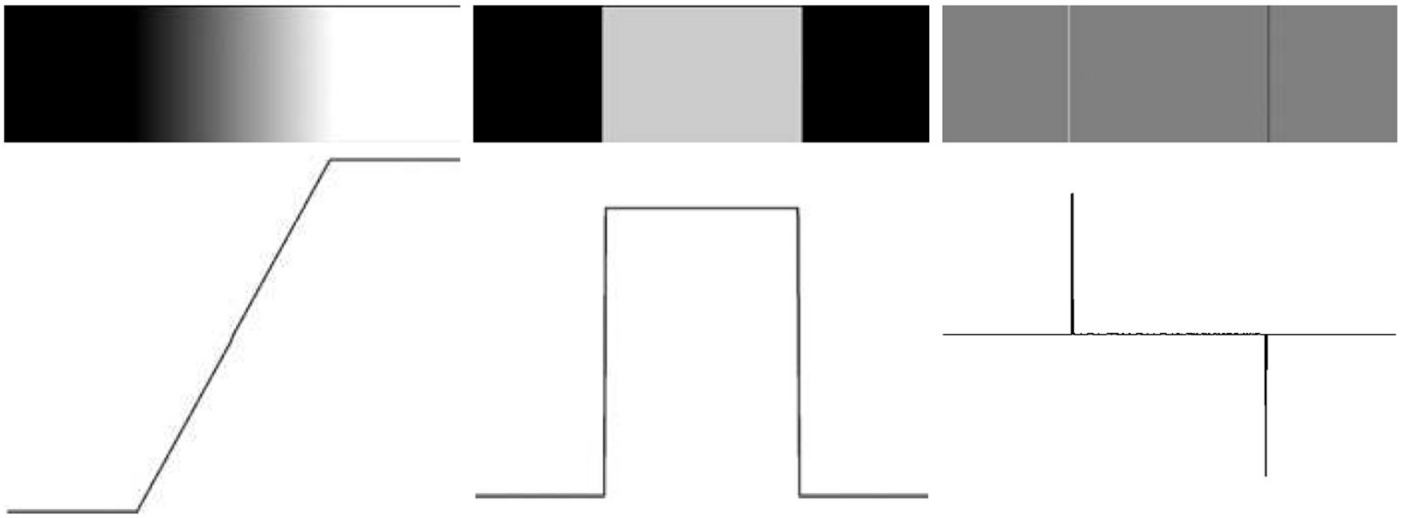


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
  - The effect of noise

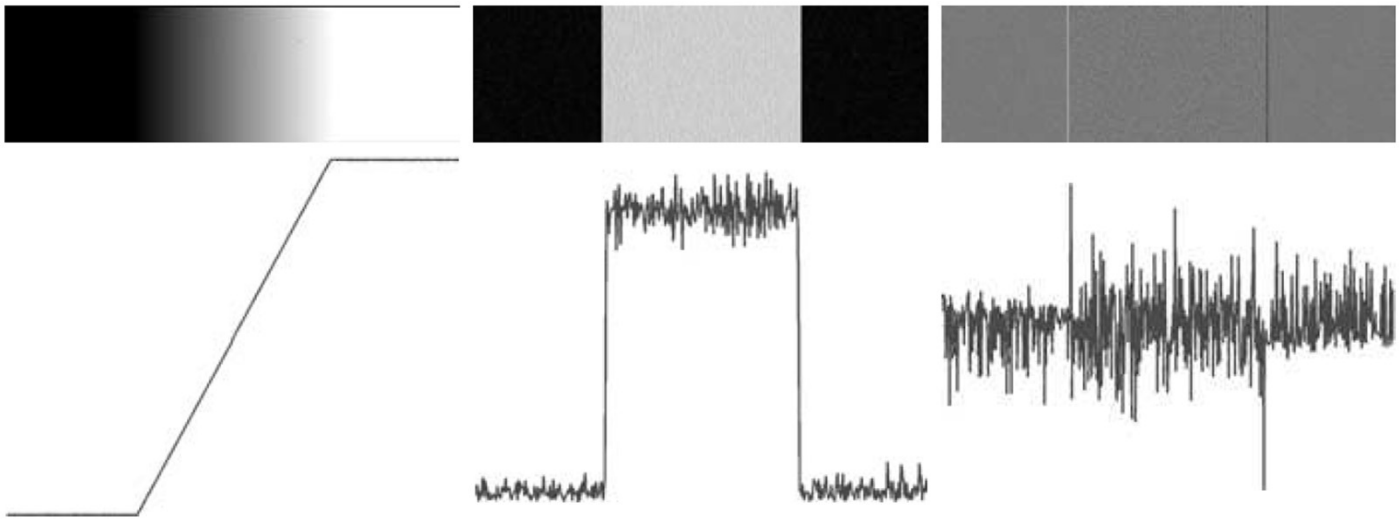


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
  - The effect of noise

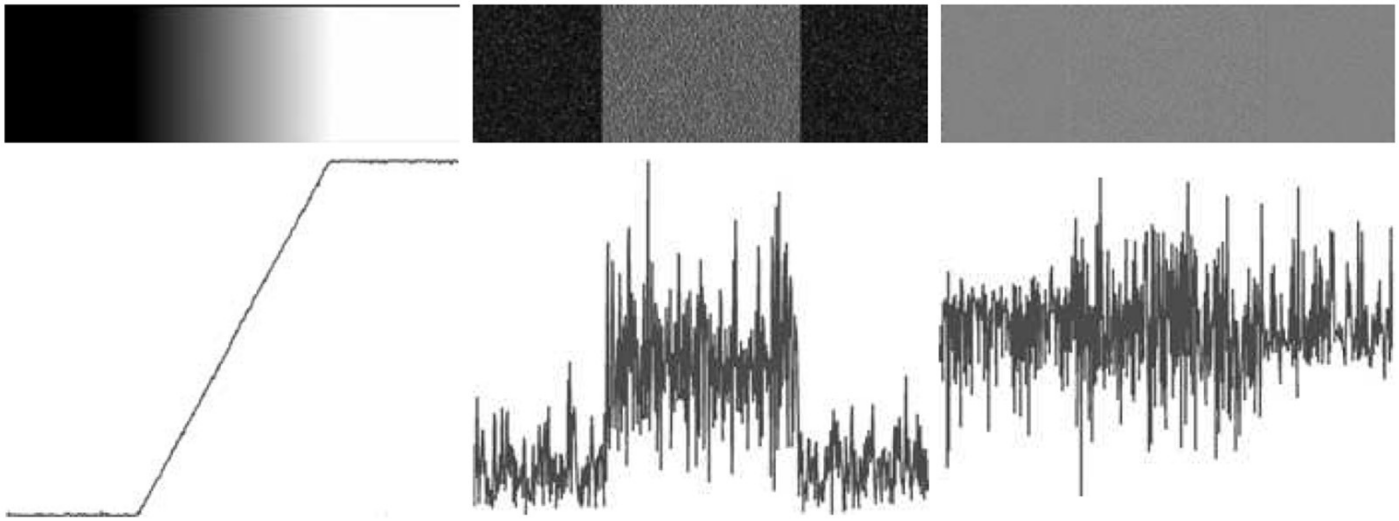


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
  - The effect of noise

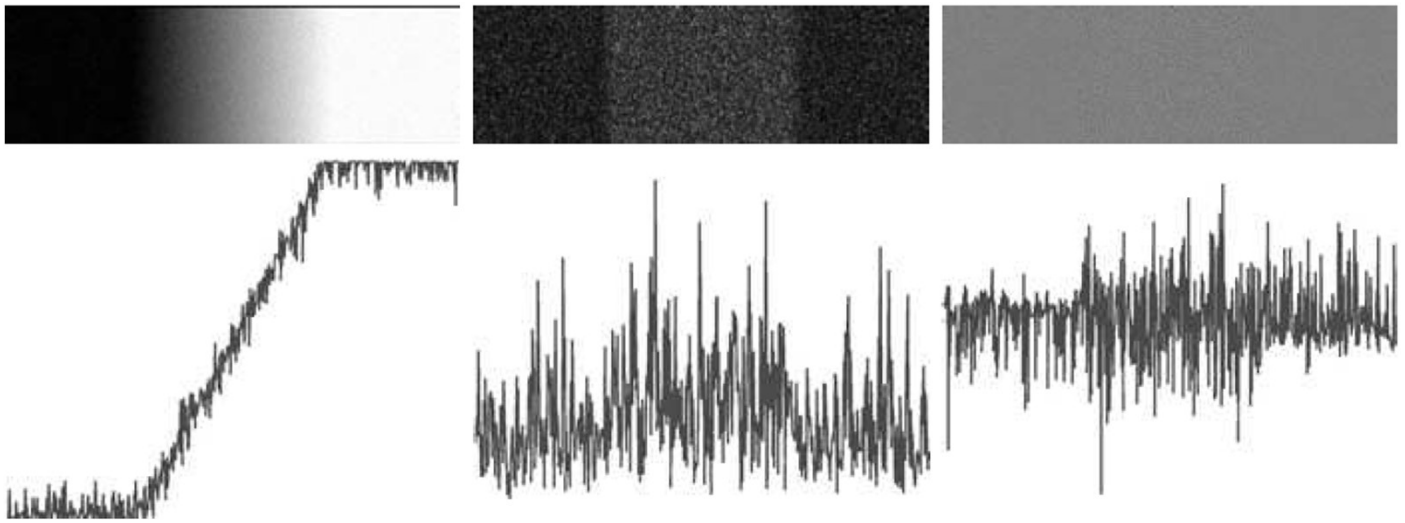


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models

  - We conclude this section by noting that there are three fundamental steps performed in edge detection:

    1. Image smoothing for noise reduction.

    2. Detection of edge points. As mentioned earlier, this is a local operation that extracts from an image all points that are potential candidates to become edge points.

    3. Edge localization. Select from the candidate edge points only the points that are true members of the set of points comprising an edge.

2. Point, Line, and Edge Detection

- Basic Edge Detection

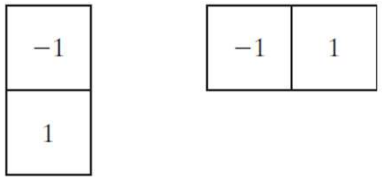
  - Gradient operators

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

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

a b

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



2. Point, Line, and Edge Detection

- Basic Edge Detection
  - Gradient operators

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

−1	0	0	−1
0	1	1	0

Roberts

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection

  - Gradient operators

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

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

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection
  - Gradient operators

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

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

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection
  - Gradient operators

0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

a b
c d

FIGURE 10.15 Prewitt and Sobel masks for detecting diagonal edges.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection

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, $|g_x| + |g_y|$.

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

2. Point, Line, and Edge Detection

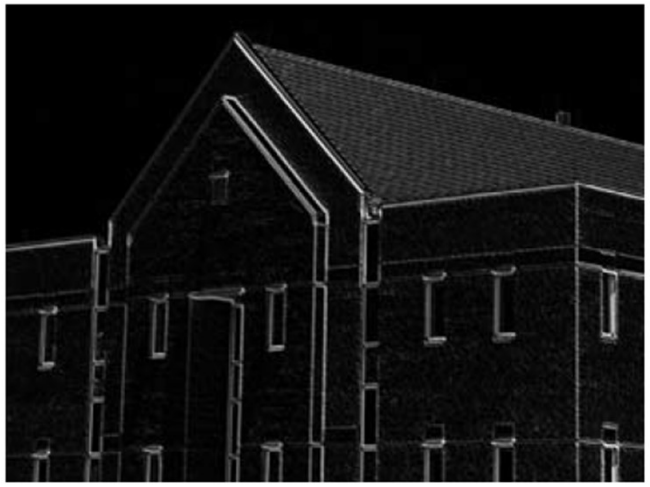
- Basic Edge Detection

a b
c d

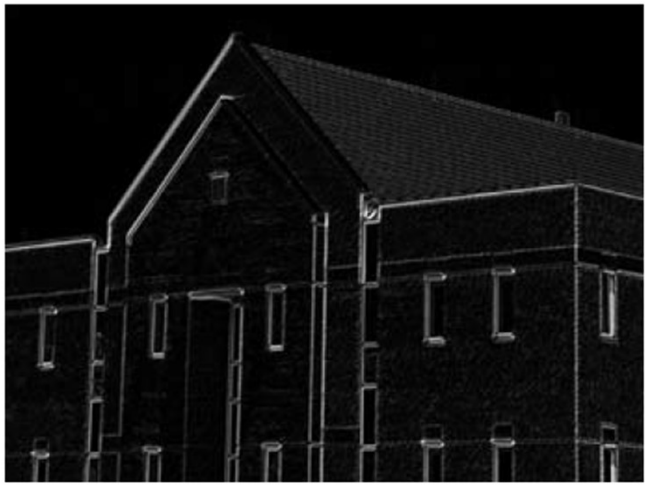
FIGURE 10.18
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5 × 5 averaging filter prior to edge detection.

2. Point, Line, and Edge Detection

- Basic Edge Detection



0	1	2
−1	0	1
−2	−1	0



−2	−1	0
−1	0	1
0	1	2

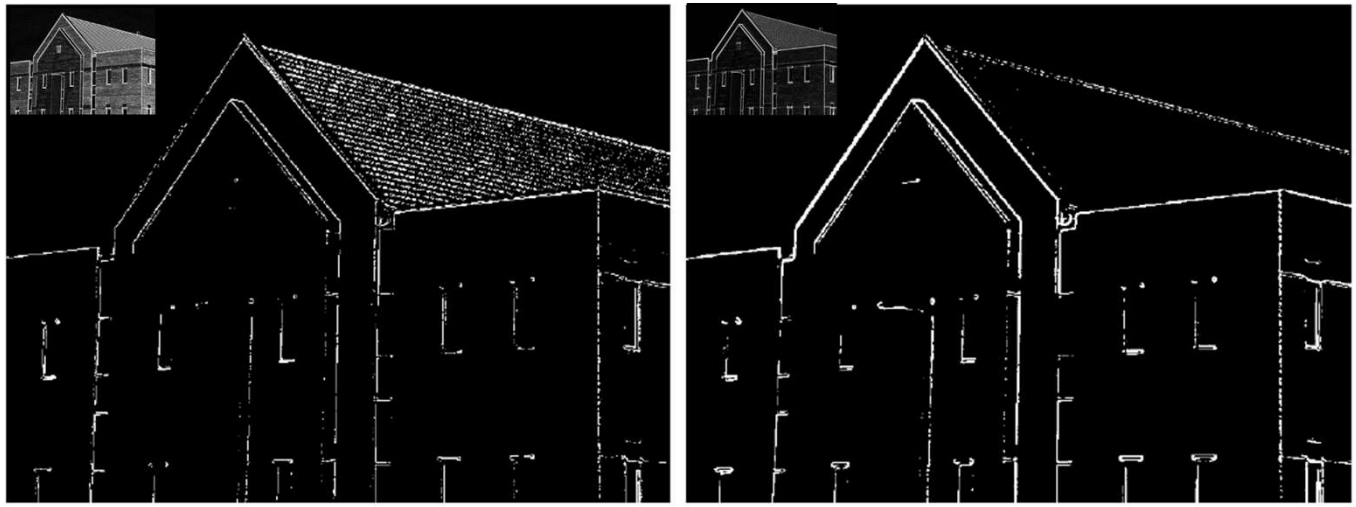
a b

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).

2. Point, Line, and Edge Detection

- Basic Edge Detection

  - Combining the gradient with 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.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

    - ✓ One of the earliest successful attempts at incorporating **more sophisticated analysis** into the edge-finding process is attributed to Marr and Hildreth.

    - ✓ They argued that:

      a. Intensity changes are not independent of image scale;

      b. 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.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

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

      a. It should be an operator capable of computing a digital approximation of the first or second derivative; and

      b. It should be capable of being "tuned": large operators can be used to detect blurry edges and small operators to detect fine details.

    - ✓ Marr and Hildreth argued that the most satisfactory operator fulfilling these conditions is the **Laplacian of the Gaussian** filter $\nabla^2 G$ (LoG).

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

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

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

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2}$$

$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

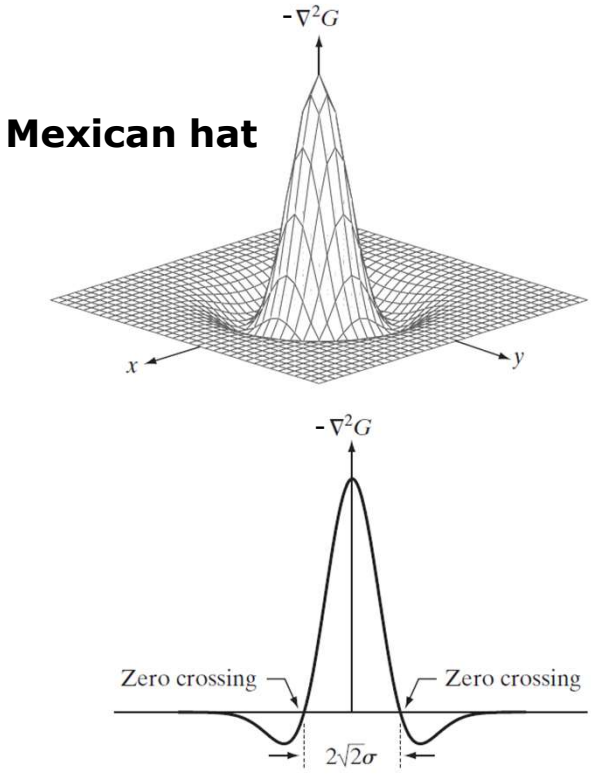
$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

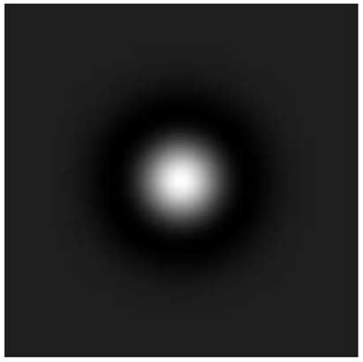
$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector





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.

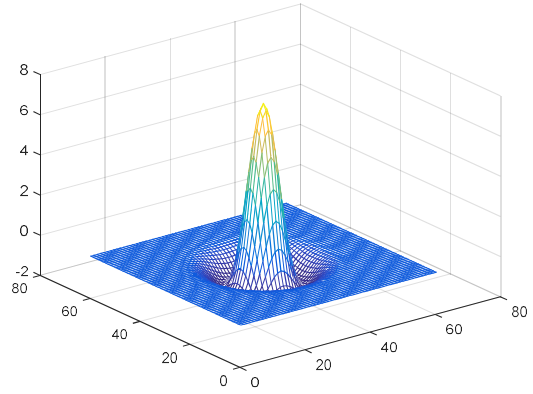
2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - ➢ The Marr-Hildreth edge detector

    - ✓ A more effective approach for generating a LoG filter is to **sample the Gaussian** to the desired *n* x *n* size and then **convolve** the resulting array with a **Laplacian mask**.

    - ✓ MATLAB: s29MarrHildFilters.m



2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

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

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

and then finding the zero crossings of *g(x, y)*.

- ✓ Because these are linear processes,

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

indicating that we can **smooth** the image **first** with a **Gaussian filter** and then compute the **Laplacian** of the result.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

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

      1. Filter the input image with an Gaussian lowpass filter obtained by sampling $G(x, y)$.

      2. Compute the Laplacian of the image resulting from Step 1.

      3. Find the zero crossings of the image from Step 2.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Marr-Hildreth edge detector

    - ✓ Recall that about 99.7% of the volume under a 2-D Gaussian lies between $\pm 3\sigma$ about the mean.

    - ✓ Thus, as a rule of thumb, the size of an *n x n* LoG discrete filter should be such that *n* is the smallest odd integer greater than or equal to 6σ.

    - ✓ One approach for finding the **zero crossings** at any pixel, *p*, of the filtered image, *g(x, y)*, is based on using a **3 x 3 neighborhood** centered at ***p***.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - The Marr-Hildreth edge detector
    - ✓ A zero crossing at *p* implies that the signs of at least two of its opposing neighboring pixels must differ.
      - There are four cases to test: left/right, up/down, and the two diagonals.
    - ✓ If the values of are being **compared** against a **threshold**, the absolute value of their numerical difference must also exceed the threshold before we can call a zero-crossing pixel.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - ➢ The Marr-Hildreth edge detector
    - ✓ MATLAB: s33MarrHildEdges.m

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - ➢ The Canny edge detector
    - ✓ The algorithm is more complex.
    - ✓ The performance is superior in general to the edge detectors discussed so far.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - ➢ The Canny edge detector
    - ✓ Canny's approach is based on three basic objectives:
      a. Low error rate.
        - ○ All edges should be found, and there should be no spurious responses.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - ➢ The Canny edge detector
    - ✓ Canny's approach is based on three basic objectives:
      - b. Edge points should be well localized.
        - ○ The edges located must be as close as possible to the true edges.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - ➢ The Canny edge detector
    - ✓ Canny's approach is based on three basic objectives:
      - c. Single edge point response.
        - ○ The detector should return only one point for each true edge point.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    1. Let *f(x, y)* denote the input image, $f_s(x, y)$ the **smoothed** image and *G(x,y)* the Gaussian function

$$f_s(x, y) = G(x, y) \star f(x, y)$$

       - ✓ Gaussian filter, with $\sigma = 1.4$

$$\frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    2. This operation is followed by computing the **gradient magnitude** and **direction** (angle),

$$g_x = \partial f_s / \partial x \quad g_y = \partial f_s / \partial y$$

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

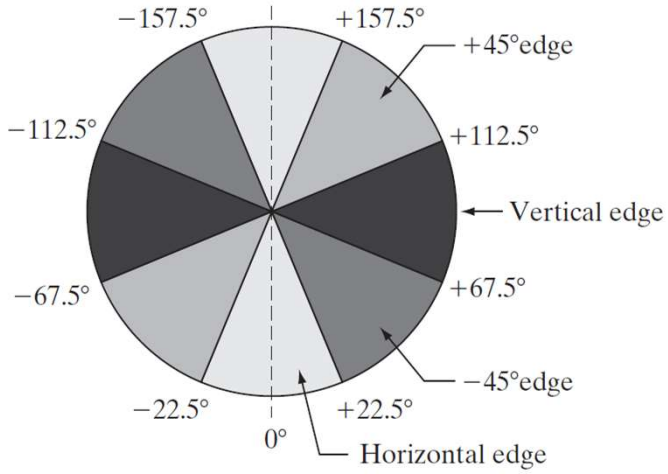
$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    - ✓ In a 3 x 3 region we can define **four discrete orientations** for an edge passing through the center point of the region: horizontal, vertical, +45º, -45º.

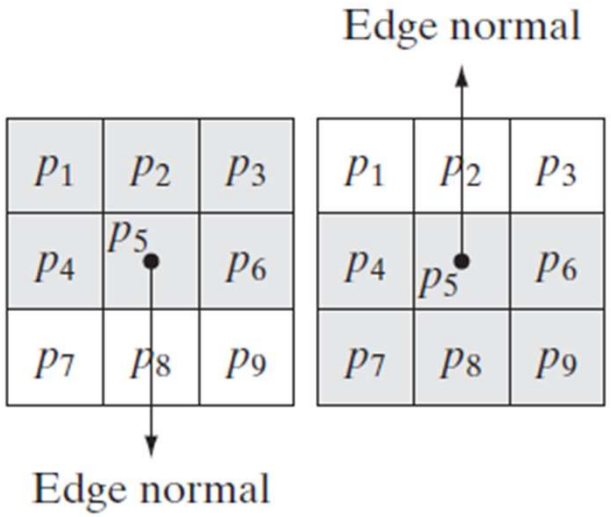


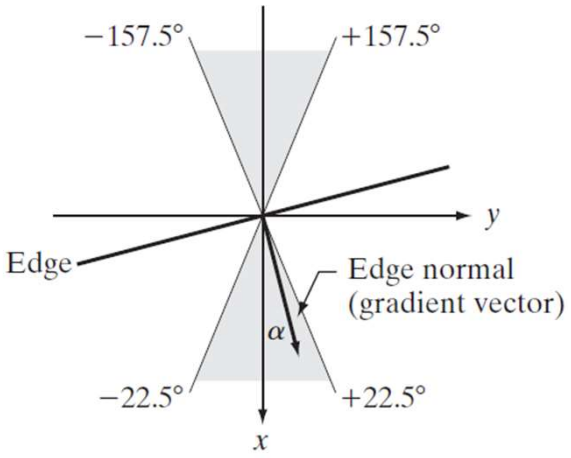
2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    - ✓ The **edge direction** can be determined from the direction of the **edge normal** (gradient direction) $\alpha(x, y)$.





2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    3. Suppression of **nonmaxima**

       a. Let d_1, d_2, d_3 and d_4 denote the four basic gradient directions.

       b. For a region centered at every point (x, y), find the direction, $\boldsymbol{d_k}$, that is closest to $\boldsymbol{\alpha(x, y)}$.

       c. If the value of M(x, y) **is less** than at **least one** of its **two neighbors along $\boldsymbol{d_k}$,** let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$.

          ✓ g_N (x, y) is equal to M(x, y) with the nonmaxima edge points suppressed.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    4. The final operation is to **threshold** to reduce false edge points.

       a. Canny's algorithm uses **two thresholds**, a **low** threshold, T_L and a **high** threshold, T_H.

$$g_{NH}(x, y) = g_N(x, y) \geq T_H$$

$$g_{NL}(x, y) = g_N(x, y) \geq T_L$$

       b. Canny suggested that the **ratio** of the **high** to **low** threshold should be two or three to one.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    c. We eliminate from g_{NL} all the nonzero pixels from g_{NH} by letting,

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

    d. The nonzero pixels in $\boldsymbol{g_{NH}}$ and $\boldsymbol{g_{NL}}$ may be viewed as being "**strong**" and "**weak**" edge pixels, respectively.

    e. Pixels in $\boldsymbol{g_{NH}}$ are assumed to be **valid edge** pixels and are so marked immediately.

    f. The edges in $\boldsymbol{g_{NH}}$ typically have **gaps**.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    g. Longer edges are formed using the following procedure:

    1) Locate the next unvisited pixel, *p,* in $\mathbf{g_{NH}}$.

    2) Mark as **valid edge pixels** all the **weak** pixels in $\mathbf{g_{NL}}$ that are **connected to *p***.

    3) If all **nonzero** pixels in $\mathbf{g_{NH}}$ have been **visited** go to Step 4. Else, return to Step 1.

    4) Set to **zero** all pixels in $\mathbf{g_{NL}}$ that were **not marked** as **valid** edge pixels (step 2).

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    h. The **final image** is formed by **appending** to $\mathbf{g_{NH}(x, y)}$ the **nonzero** pixels from $\mathbf{g_{NL}(x, y)}$.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

  - The Canny edge detector

    - ✓ Summarizing, the Canny edge detection algorithm consists of the following basic steps:

      1. **Smooth** the input image with a Gaussian filter.

      2. Compute the **gradient** magnitude and angle images.

      3. Apply **nonmaxima suppression** to the gradient magnitude image.

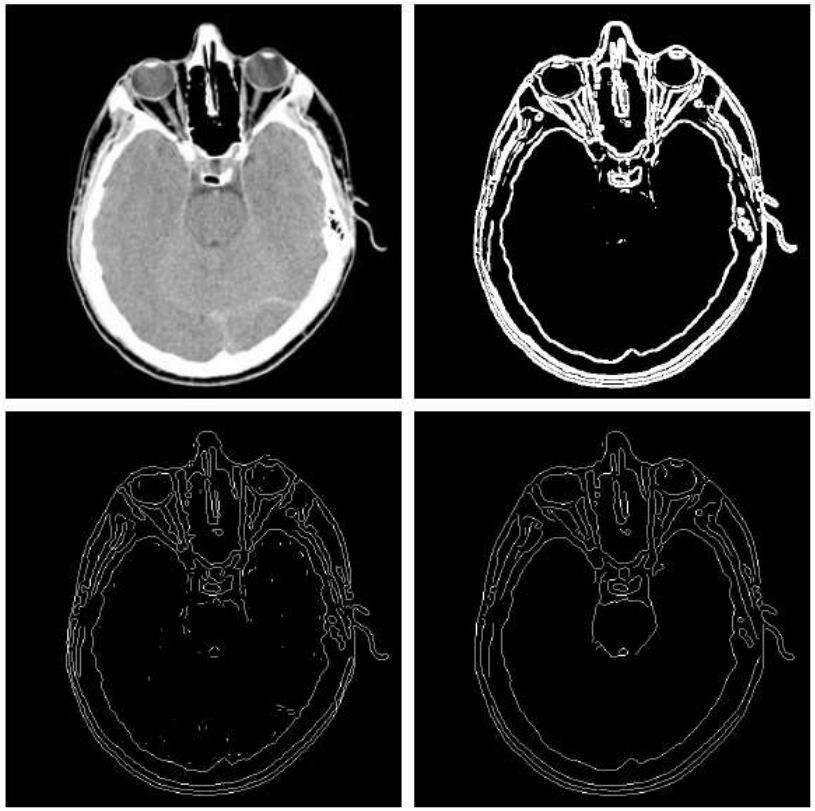
      4. Use **double thresholding** and **connectivity analysis** to detect and link edges.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
  - MATLAB: s44Canny.m

2. Point, Line, and Edge Detection

- Illustration of the three principal edge detection methods previously discussed.



a b
c d

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

3. Threshold

- The basics of intensity thresholding

  - In the previous section, regions were identified by first finding **edge segments** and then attempting to link the segments into boundaries.

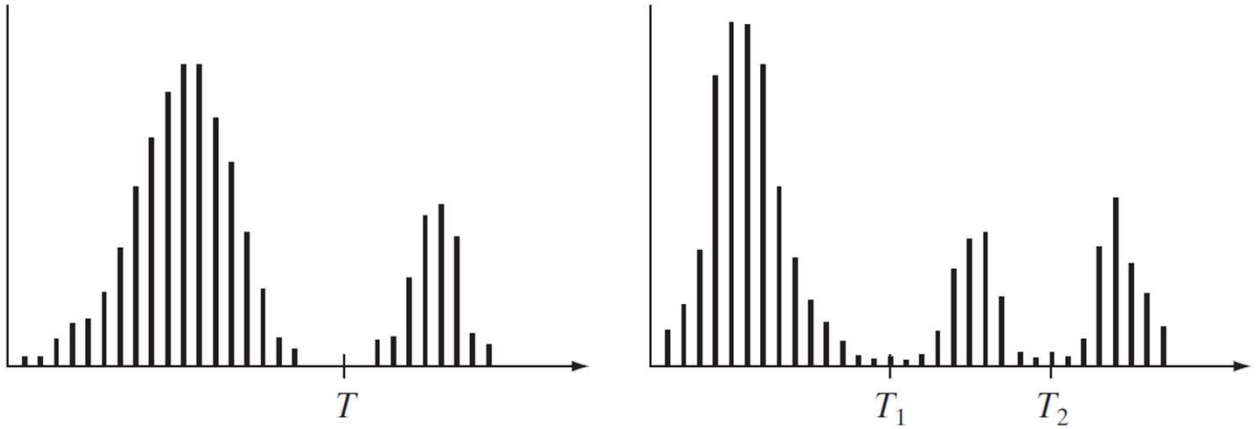
  - In this section, we discuss techniques for partitioning images directly into regions based on **intensity values** and/or properties of these values.

3. Threshold

- The basics of intensity thresholding

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

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

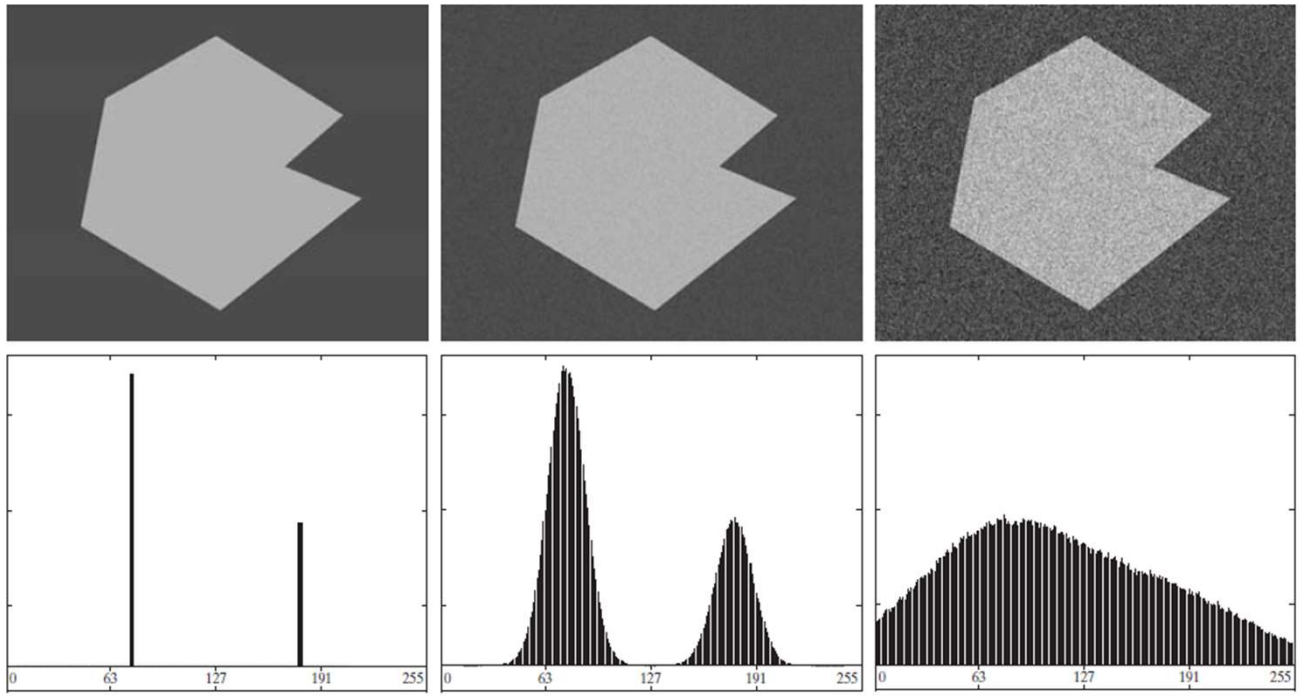


a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

3. Threshold

- The role of noise in image thresholding

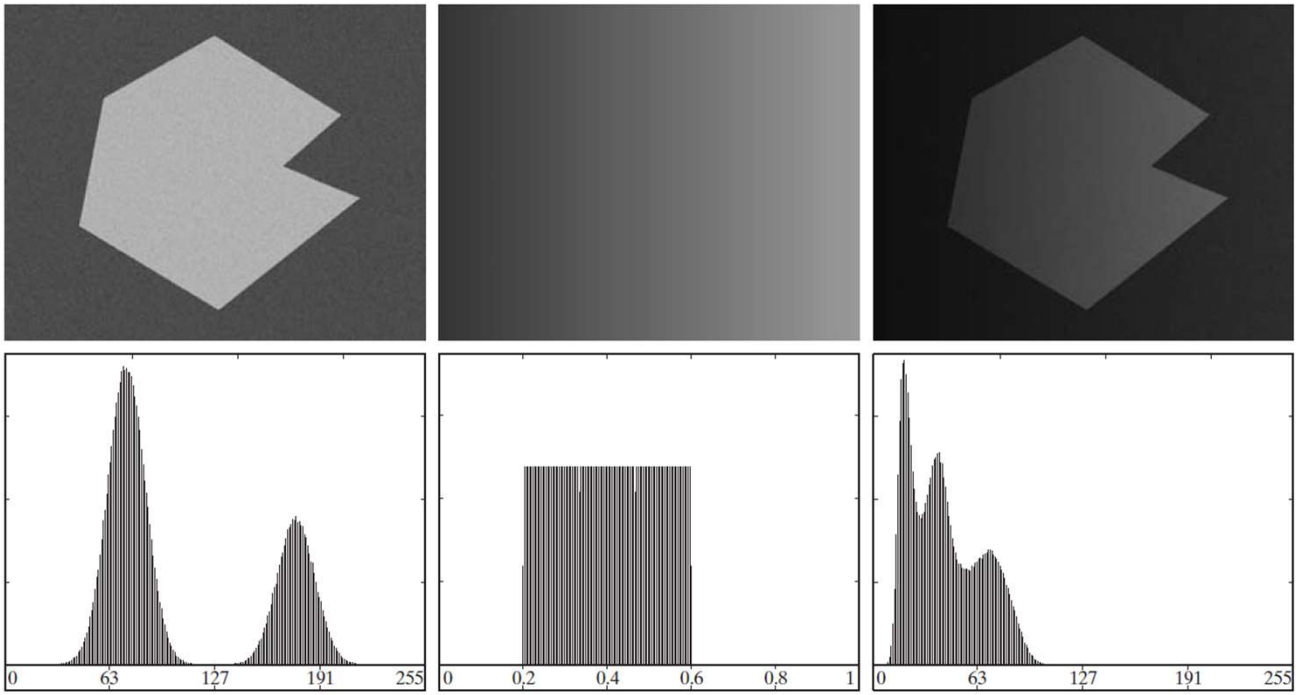


a b c
d e f

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.

3. Threshold

- The role of illumination and reflectance



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.

3. Threshold

- Basic Global Thresholding

  - Even if global thresholding is a suitable approach, an algorithm capable of estimating automatically the threshold value for each image is required.

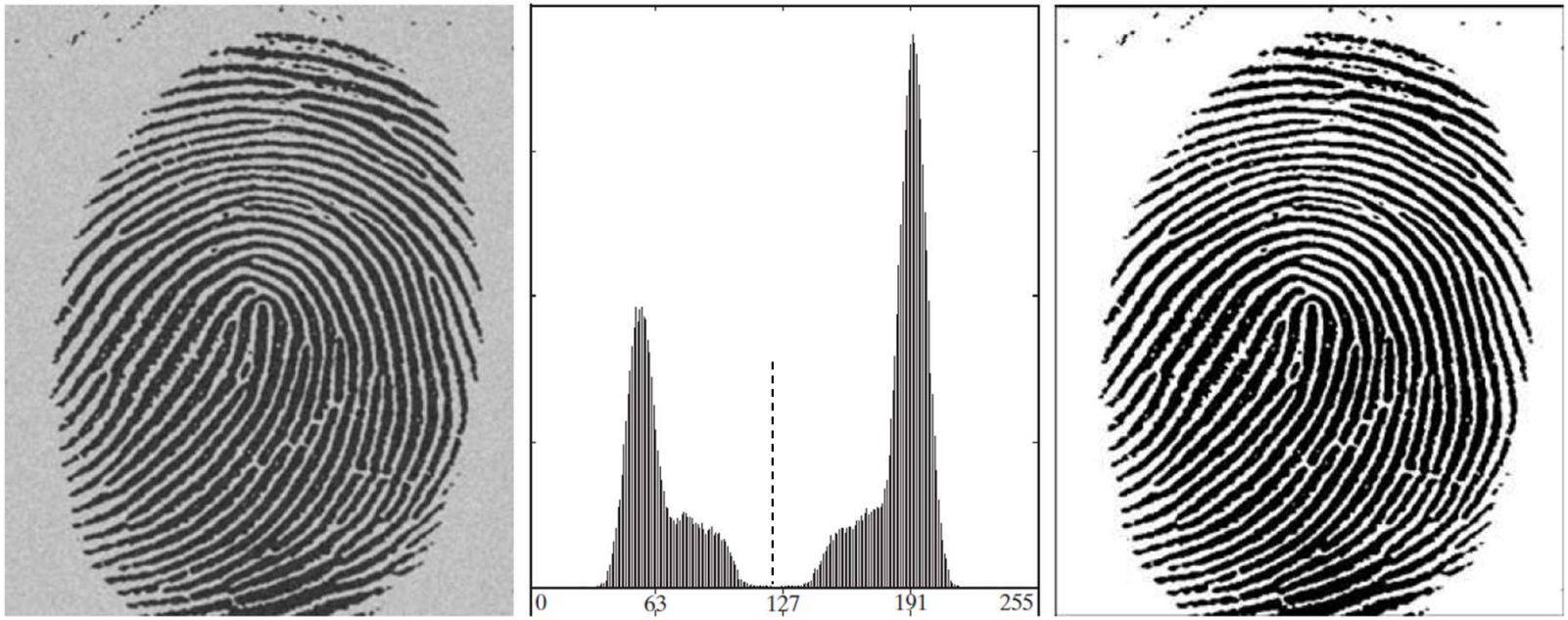
1. Select an initial estimate for the global threshold, T.
2. Segment the image using T in Eq. (10.3-1). This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2, respectively.
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 iterations is smaller than a predefined parameter ΔT.

3. Threshold

- Basic Global Thresholding



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.)

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

  - Thresholding may be viewed as a **statistical-decision theory problem**.

    - ✓ The 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 rule**.

  - The solution is **based on the PDF of the intensity levels** of **each class** and the probability that each class occurs in a given application.

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

  - Unfortunately, **estimating PDFs is not a trivial** 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**, implementing solutions using these assumptions can be complex and **not always well-suited** for practical applications.

  - The approach discussed in this section, called **Otsu's method** is an attractive alternative.

  - It is optimum in the sense that it **maximizes the between-class variance**, a well-known measure used in statistical discriminant analysis.

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

  - Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image. Denote the components of the histogram by $p_i, i = 0, 1, 2, \ldots, L-1$.

2. Compute the cumulative sums, $P_1(k)$, for $k = 0, 1, 2, \ldots, L-1$, using Eq. (10.3-4).

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

3. Compute the cumulative means, $m(k)$, for $k = 0, 1, 2, \ldots, L-1$, using Eq. (10.3-8).

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

4. Compute the global intensity mean, m_G, using (10.3-9).

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

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

  - Otsu's algorithm may be summarized as follows:

5. Compute the between-class variance, $\sigma_B^2(k)$, for $k = 0, 1, 2, \ldots, L-1$, using Eq. (10.3-17).

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

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, η^*, by evaluating Eq. (10.3-16) at $k = k^*$.

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \qquad \sigma_G^2 = \sum_{i=0}^{L-1}(i - m_G)^2 p_i$$

3. Threshold

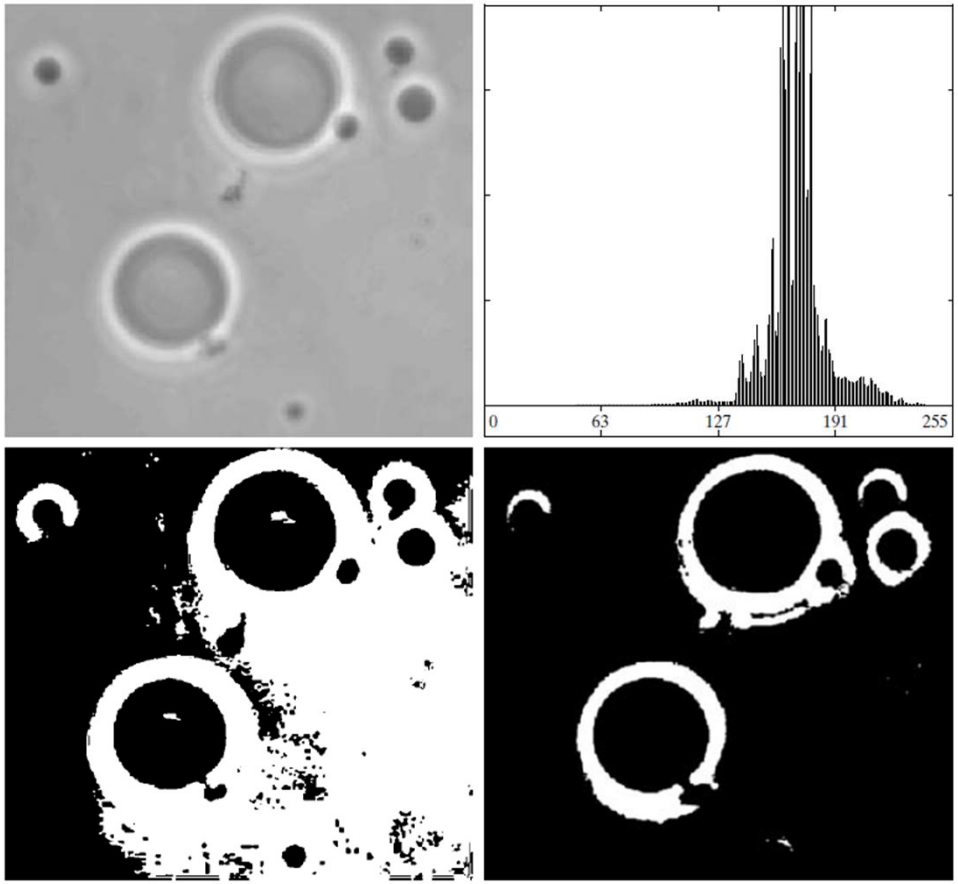
- Optimum Global Thresholding Using Otsu's Method

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

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

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

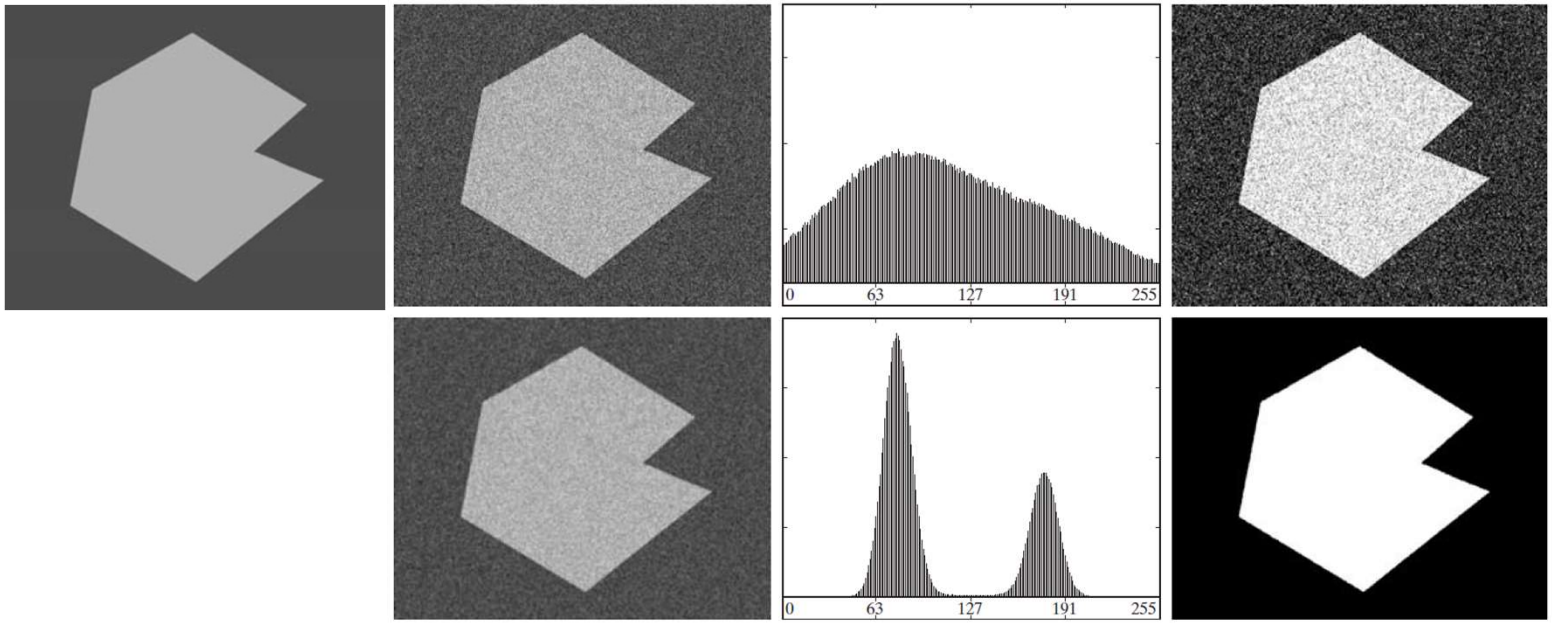


a b
c d

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.)

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

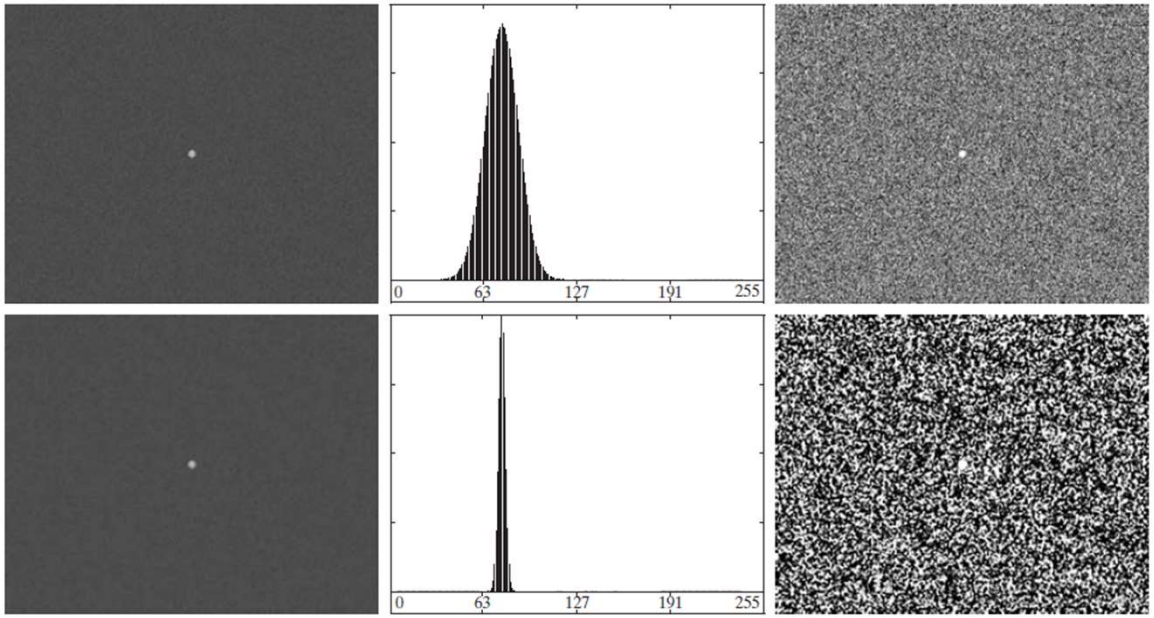


a b c
d e f

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.

3. Threshold

- Using Edges to Improve Global Thresholding

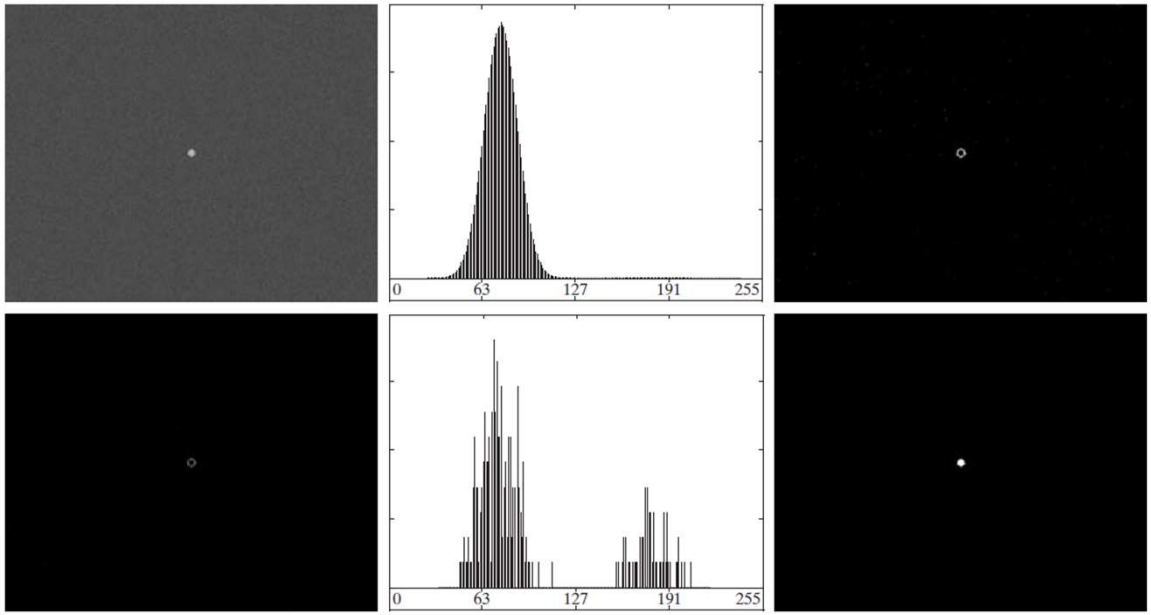


a b c
d e f

FIGURE 10.41 (a) Noisy image 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. Thresholding failed in both cases.

3. Threshold

- Using Edges to Improve Global Thresholding



a b c
d e f

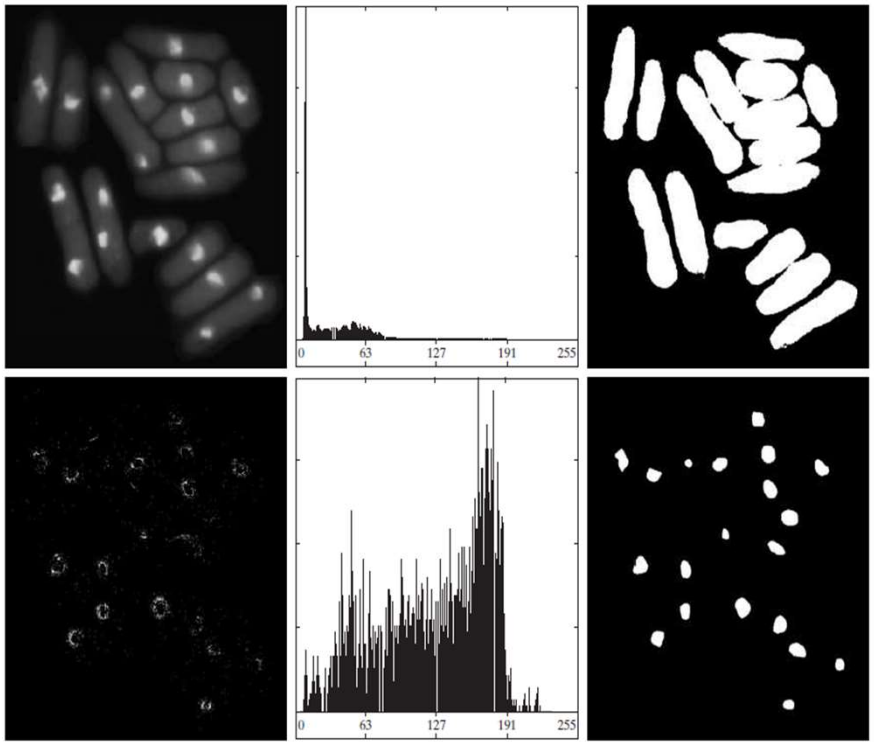
FIGURE 10.42 (a) Noisy image from Fig. 10.41(a) and (b) its histogram. (c) Gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (e). The threshold was 134, which is approximately midway between the peaks in this histogram.

3. Threshold

- Using Edges to Improve Global Thresholding

The objective is to detect the bright spots.

This value of T corresponds approximately to the 99.5 percentile.



a b c
d e f

FIGURE 10.43 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Thresholded absolute Laplacian. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e). (Original image courtesy of Professor Susan L. Forsburg, University of Southern California.)

3. Threshold

- Using Edges to Improve Global Thresholding

  - The preceding discussion is summarized in the following algorithm:

1. Compute an edge image as either the magnitude of the gradient, or absolute value of the Laplacian, of $f(x, y)$ using any of the methods discussed in Section 10.2.
2. Specify a threshold value, T.
3. Threshold the image from Step 1 using the threshold from Step 2 to produce a binary image, $g_T(x, y)$. This image is used as a *mask image* in the following step to select pixels from $f(x, y)$ corresponding to "strong" edge pixels.
4. Compute a histogram using only the pixels in $f(x, y)$ that correspond to the locations of the 1-valued pixels in $g_T(x, y)$.
5. Use the histogram from Step 4 to segment $f(x, y)$ globally using, for example, Otsu's method.

3. Threshold

- Multiple Thresholds
  - In the case of *K* classes, C_1, C_2,…,C_K:

$$m_k = \frac{1}{P_k} \sum_{i \in C_k} i p_i \qquad m_G = \sum_{i=0}^{L-1} i p_i \qquad P_k = \sum_{i \in C_k} p_i$$

$$\sigma_B^2 = \sum_{k=1}^{K} P_k (m_k - m_G)^2$$

$$\sigma_B^2(k_1^*, k_2^*, \ldots, k_{K-1}^*) = \max_{0 < k_1 < k_2 < \ldots k_{n-1} < L-1} \sigma_B^2(k_1, k_2, \ldots k_{K-1})$$

3. Threshold

- Multiple Thresholds
  - In the case of 3 classes, C_1, C_2 and C_3

$$m_1 = \frac{1}{P_1}\sum_{i=0}^{k_1} i p_i \qquad P_1 = \sum_{i=0}^{k_1} p_i$$

$$m_2 = \frac{1}{P_2}\sum_{i=k_1+1}^{k_2} i p_i \qquad P_2 = \sum_{i=k_1+1}^{k_2} p_i$$

$$m_3 = \frac{1}{P_3}\sum_{i=k_2+1}^{L-1} i p_i \qquad P_3 = \sum_{i=k_2+1}^{L-1} p_i$$

3. Threshold

- Multiple Thresholds
  - In the case of 3 classes, C_1, C_2 and C_3

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2$$

$$\sigma_B^2(k_1^*, k_2^*) = \max_{0<k_1<k_2<L-1} \sigma_B^2(k_1, k_2)$$

3. Threshold

- Multiple Thresholds

  - In the case of 3 classes, C_1, C_2 and C_3

    1. The procedure starts by selecting the first value of k_1 (that value is 1).

    2. Next, k_2 is incremented through all its values greater than k_1 and less then L-1.

    3. This procedure is repeated until k_1 = L-3.

    4. The last step is to look for the maximum value in this array.

3. Threshold

- Multiple Thresholds
  - ➢ In the case of 3 classes, C_1, C_2 and C_3
    - ✓ The thresholded image is then given by

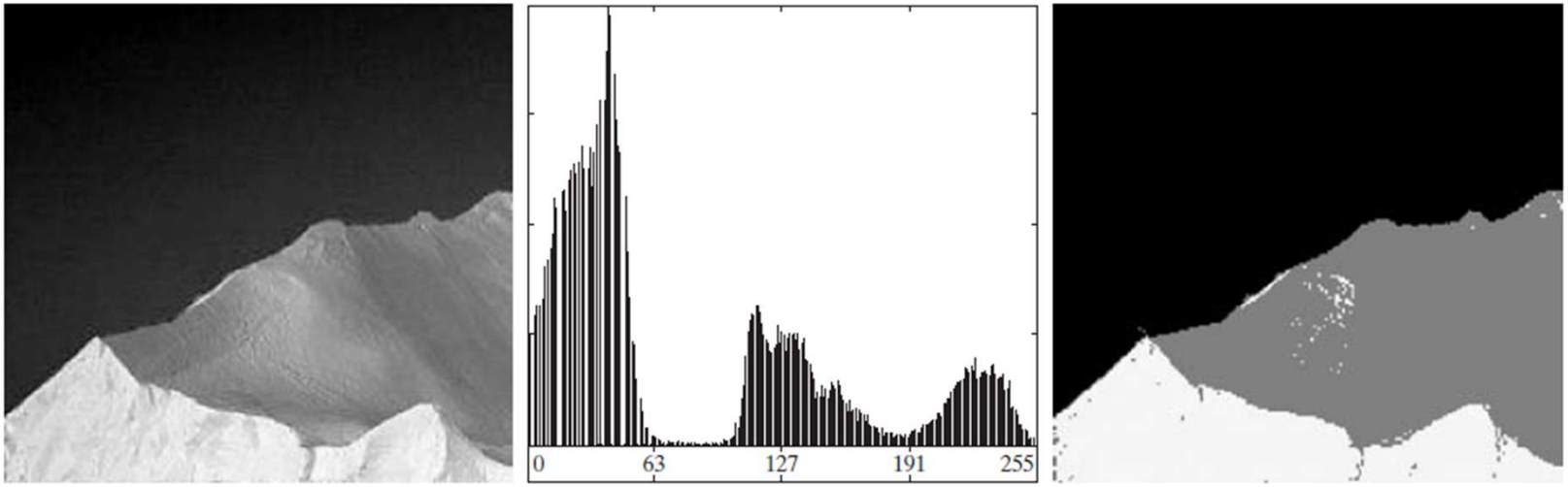
$$g(x,y) = \begin{cases} a & \text{if } f(x,y) \leq k_1^* \\ b & \text{if } k_1^* < f(x,y) \leq k_2^* \\ c & \text{if } f(x,y) > k_2^* \end{cases}$$

    - ✓ Separability measure

$$\eta(k_1^*, k_2^*) = \frac{\sigma_B^2(k_1^*, k_2^*)}{\sigma_G^2}$$

3. Threshold

- Multiple Thresholds

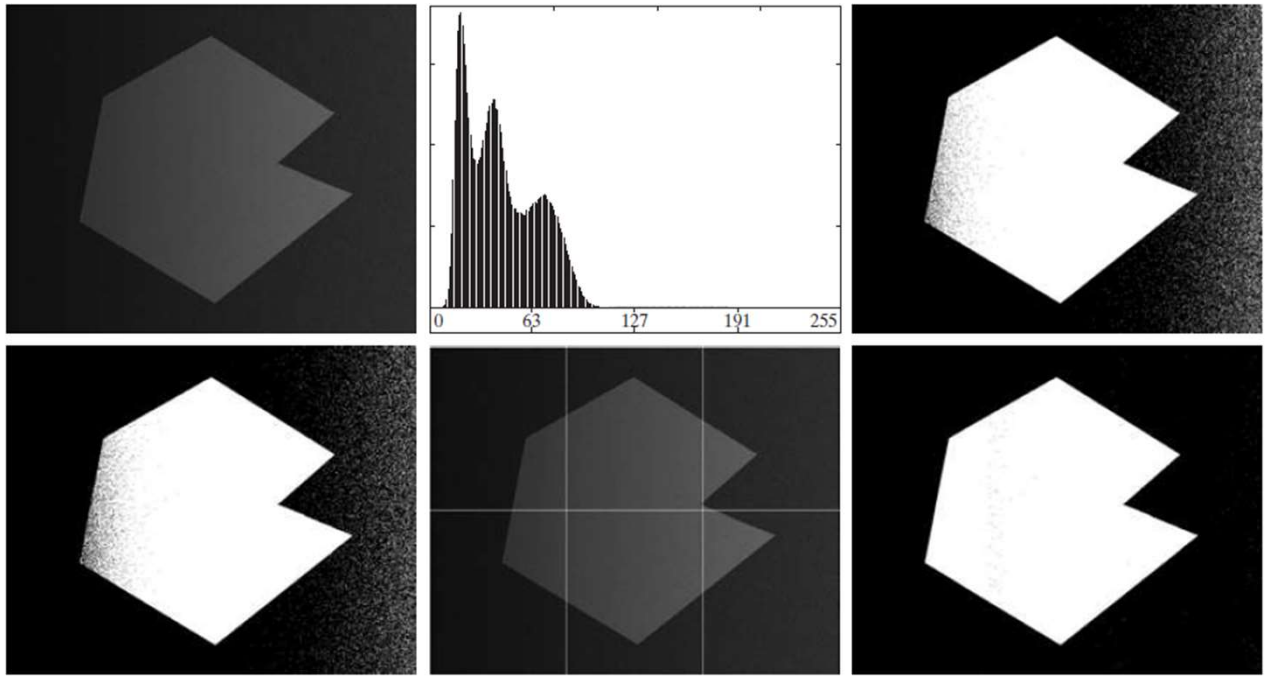


a b c

FIGURE 10.45 (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

3. Threshold

- Variable Thresholding



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.