Digital Image Processing

Dr. Mubashir Ahmad (Ph.D.)

Color Image Histogram Equalization

- Color image histogram equalization is a technique used to enhance the contrast and improve the visual appearance of an image. It operates on the histogram of the image, which represents the distribution of pixel intensities.
- Histogram equalization works by redistributing the pixel intensities across the entire range of values available in the image. This helps in stretching out the histogram to cover the full dynamic range, resulting in a more balanced distribution of intensities.

Color Image Histogram Equalization

- To apply histogram equalization to a color image, you can follow these steps:
- Convert the image from the RGB color space to another color space, such as the HSI (Hue, Saturation, Intensity) or HSV (Hue, Saturation, Value) color space. The choice of color space depends on your preference and the characteristics of the image.
- Extract the intensity or value component from the color space. This component represents the perceived brightness of the image.
- Compute the histogram of the intensity component. This histogram represents the distribution of pixel intensities in the image.

Color Image Histogram Equalization

- Apply histogram equalization to the intensity histogram. You can achieve this by computing the cumulative distribution function (CDF) of the histogram and then mapping the pixel intensities to new values based on the CDF.
- Map the equalized intensity values back to the color image, while keeping the hue and saturation components unchanged. This can be done by replacing the intensity component in the color space with the equalized values.
- Convert the image back to the RGB color space if necessary.
- It's worth noting that histogram equalization can sometimes result in amplified noise or artifacts, particularly in images with extreme lighting variations or localized details. In such cases, alternative techniques like adaptive histogram equalization or contrast-limited adaptive histogram equalization may be used to achieve better results.

5 by 5 color image and calculate histogram equalization

100				
200				
50				
80				
150				

120				
180				
50				
90				
130				

150				
100				
50				
70				
120				

5 by 1 color image and calculate histogram equalization

$$\begin{array}{cccc}
\text{Red} & \text{Green} & \text{Blue} & \\
[\ 100 & 120 & 150\ ] & \text{Row 1} \\
[\ 200 & 180 & 100\ ] & \text{Row 2} \\
[\ 50 & 50 & 50\ ] & \text{Row 3} \\
[\ 80 & 90 & 70\ ] & \text{Row 4} \\
[\ 150 & 130 & 120\ ] & \text{Row 5}
\end{array}$$

Step 1: Convert the image to the HSI color space.

Calculate the hui, saturation and Intensity

```
R = 150;
G = 130;
B = 120;
R = R/255;
G = G/255;
B = B/255;
s = 3/(R+G+B) * min([R G B]);
S = 1- s;
I = (1/3) * (R+G+B);
I = I * 255;
```

```
a = 1/2 * ((R-G) + (R - B));
b = sqrt((R-G)^2);
d = (R-B);
e = (G-B);
f = b+d+e;
c = a/f;

theta = acosd(c);

if(B>G)
    H = 360 - theta;
else
    H = theta;
end
```

5 by 1 color image and calculate histogram equalization

	Hue	Saturation	Intensity		
[	305	0.20	123	]	Row 1
[	72	0.33	160	]	Row 2
[	0	0	50	]	Row 3
[	90	0.23	80	]	Row 4
[	65	0.17	133	]	Row 5

- Step 2: Extract the intensity component from the HSI image.

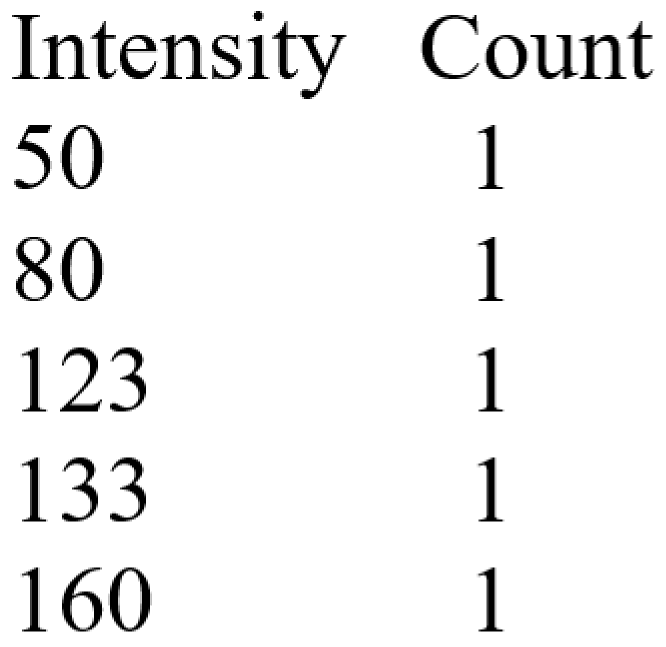
5 by 1 color image and calculate histogram equalization

- Intensity Component:

[123]	Row 1
[160]	Row 2
[50]	Row 3
[80]	Row 4
[133]	Row 5

5 by 1 color image and calculate histogram equalization

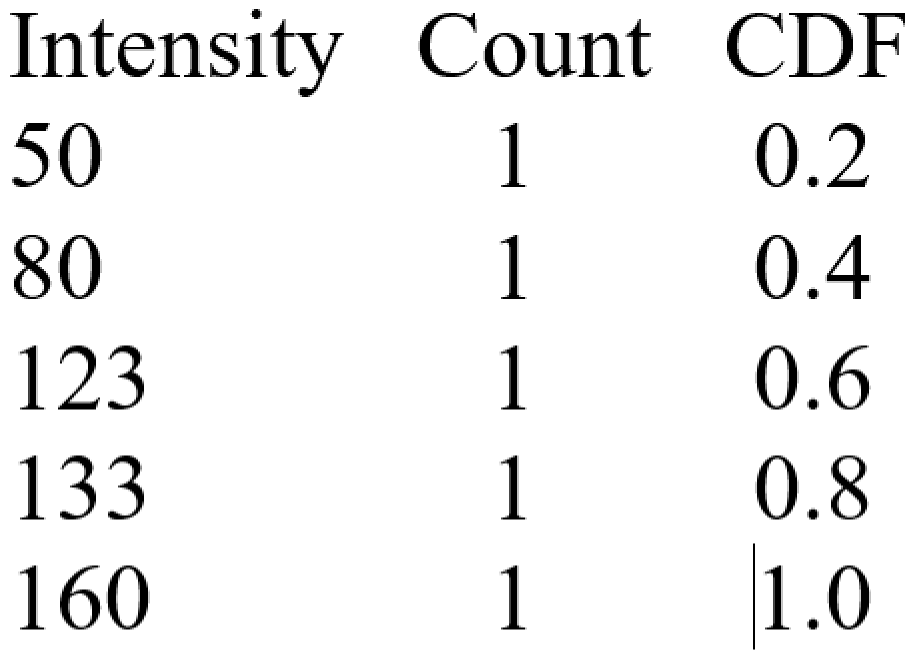
- Step 3: Compute the histogram of the intensity component.



Intensity	Count
50	1
80	1
123	1
133	1
160	1

5 by 1 color image and calculate histogram equalization

- Step 4: Apply histogram equalization to the intensity histogram.
- CDF (Cumulative Distribution Function):



Intensity	Count	CDF
50	1	0.2
80	1	0.4
123	1	0.6
133	1	0.8
160	1	1.0

5 by 1 color image and calculate histogram equalization

- Step 5: Map the equalized intensity values back to the color image.
- CDF * (L - 1) = 0.2 × (256 − 1)

Intensity	Count	CDF	Equalized
50	1	0.2	51
80	1	0.4	102
123	1	0.6	153
133	1	0.8	204
160	1	1.0	255

Image Segmentation

- Image segmentation divides an image into connected regions with some similarity within the region and some difference between adjacent regions.
- The goal is usually to find individual objects in an image.
- For the most part there are fundamentally two kinds of approaches to segmentation: discontinuity and similarity.
  - Similarity may be due to pixel intensity, color, or texture.
  - Differences are sudden changes (discontinuities) in any of these, but especially sudden changes in intensity along a boundary line, which is called an edge.

Image Segmentation

- There are three kinds of discontinuities of intensity: points, lines and edges.
- The most common way to look for discontinuities is to scan a small mask over the image. The mask determines which kind of discontinuity to look for.

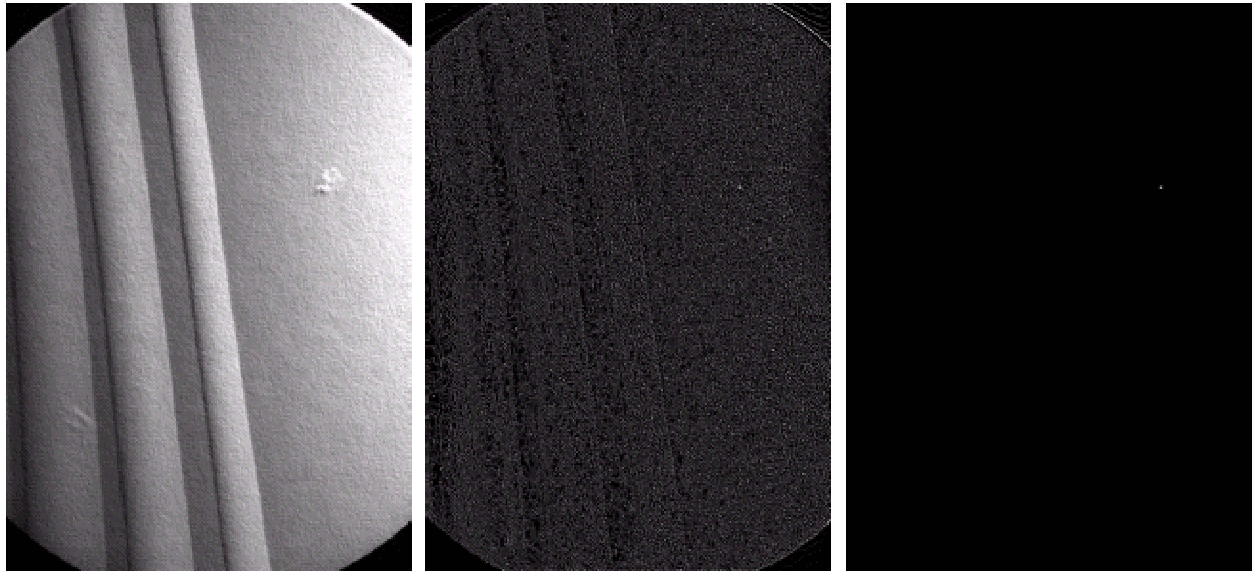
$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

FIGURE 10.1 A general 3 × 3 mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Detection of Discontinuities
P

–1	–1	–1
–1	8	–1
–1	–1	–1



a
b c d

FIGURE 10.2 (a) Point detection mask. (b) X-ray image of a turbine blade with a porosity. (c) Result of point detection. (d) Result of using Eq. (10.1-2). (Original image courtesy of X-TEK Systems Ltd.)

R - pixel intensity

R >= T	1
R <= T	0

$$|R| \geq T$$

where T : a nonnegative threshold

Point Detection

- ✓ *This is concerned with detecting isolated image points in relation to its neighborhood, an area of nearly constant intensity.*
- ✓ *The simplest point detection method works in 2 steps:*

  1. *Filter the image with the mask:*

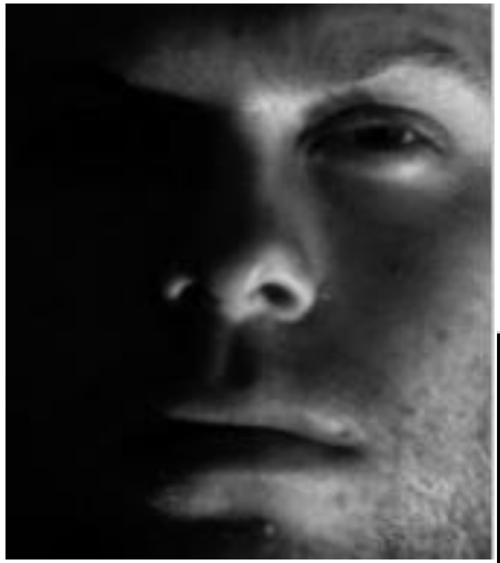
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\left|8f_5 - \left(f_1 + f_2 + f_3 + f_4 + f_6 + f_7 + f_8 + f_9\right)\right| > T$$

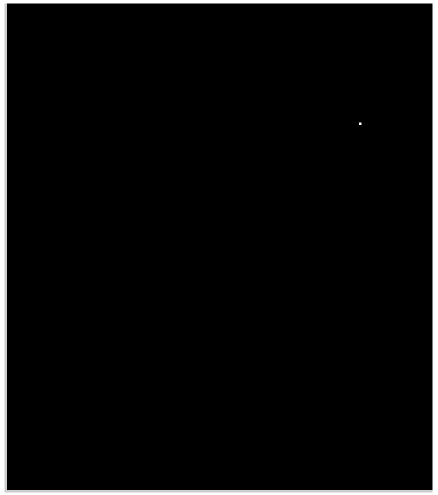
  1. *On the filtered image apply an appropriate threshold (e.g. the maximum pixel value).*

- ✓ *We need to use the abs function in step 1!!*

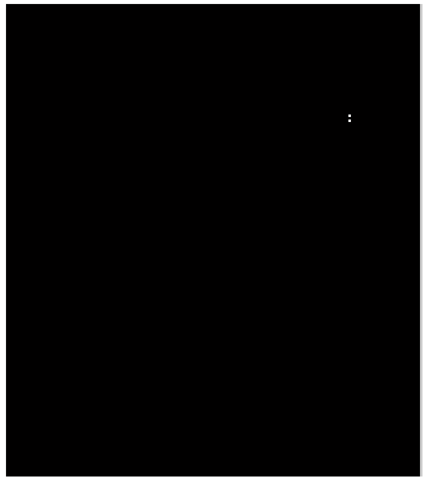
Point Detection – Example with different Thresholds



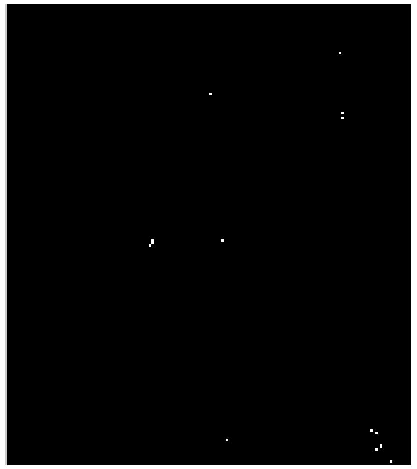
Original Image



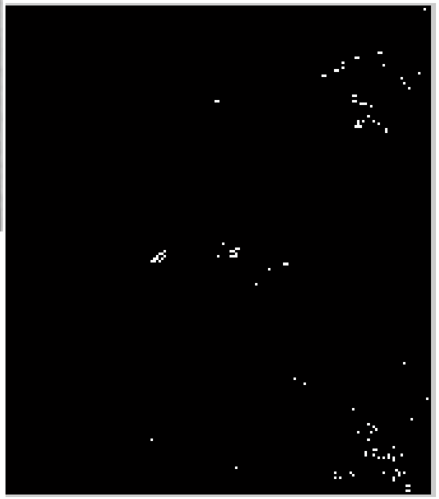
Threshold=max



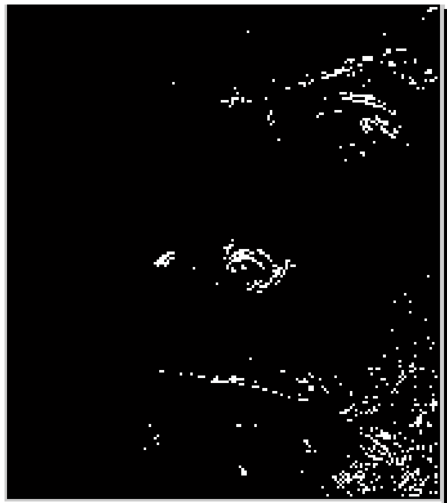
Threshold=168



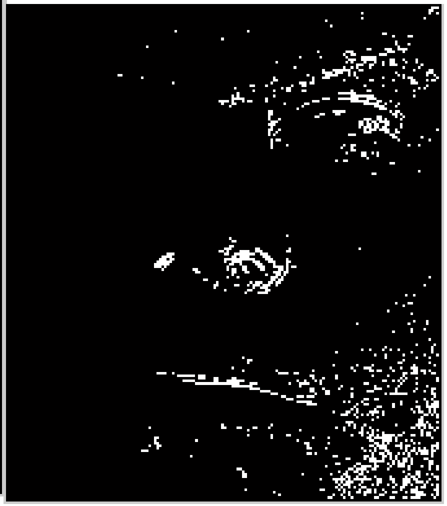
Threshold=118



Threshold=68



Threshold=18

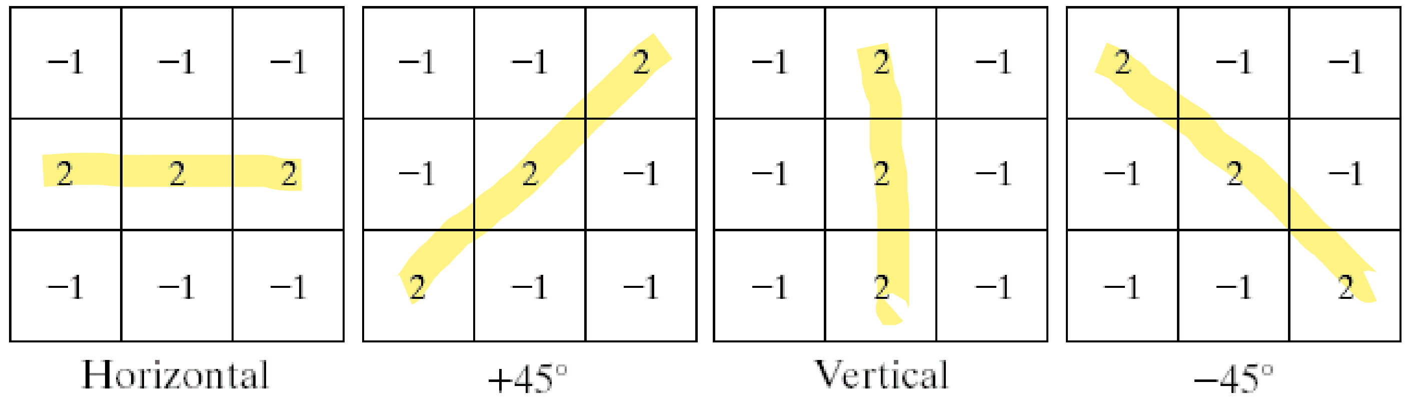


Threshold=0

Detection of Discontinuities
Line Detection

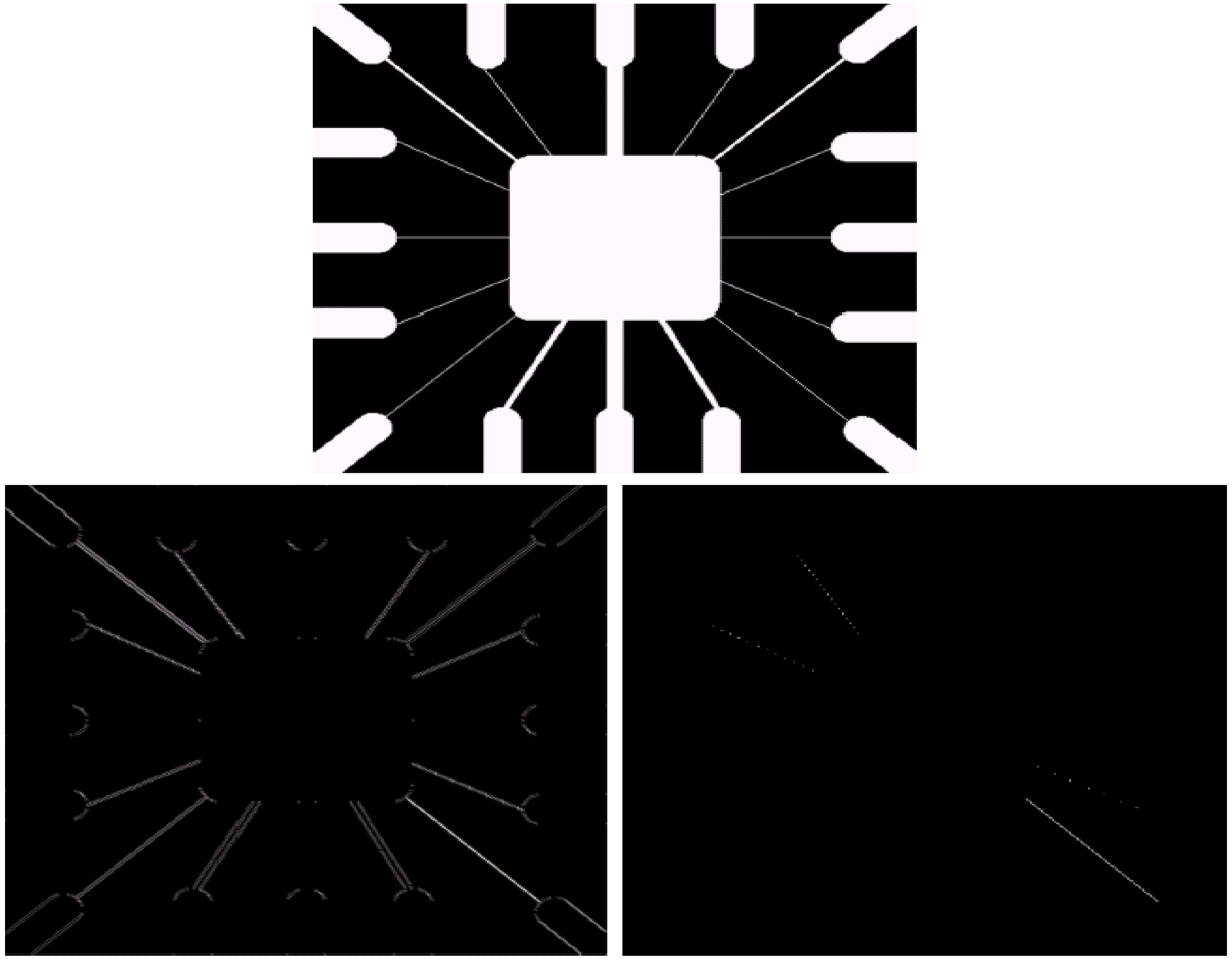
- Only slightly more common than point detection is to find a one pixel wide line in an image.
- For digital images the only three point straight lines are only horizontal, vertical, or diagonal (+ or –45°).

FIGURE 10.3 Line masks.



$$\begin{array}{cccc}
\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} &
\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} &
\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} &
\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \\
\text{Horizontal} & +45^\circ & \text{Vertical} & -45^\circ
\end{array}$$

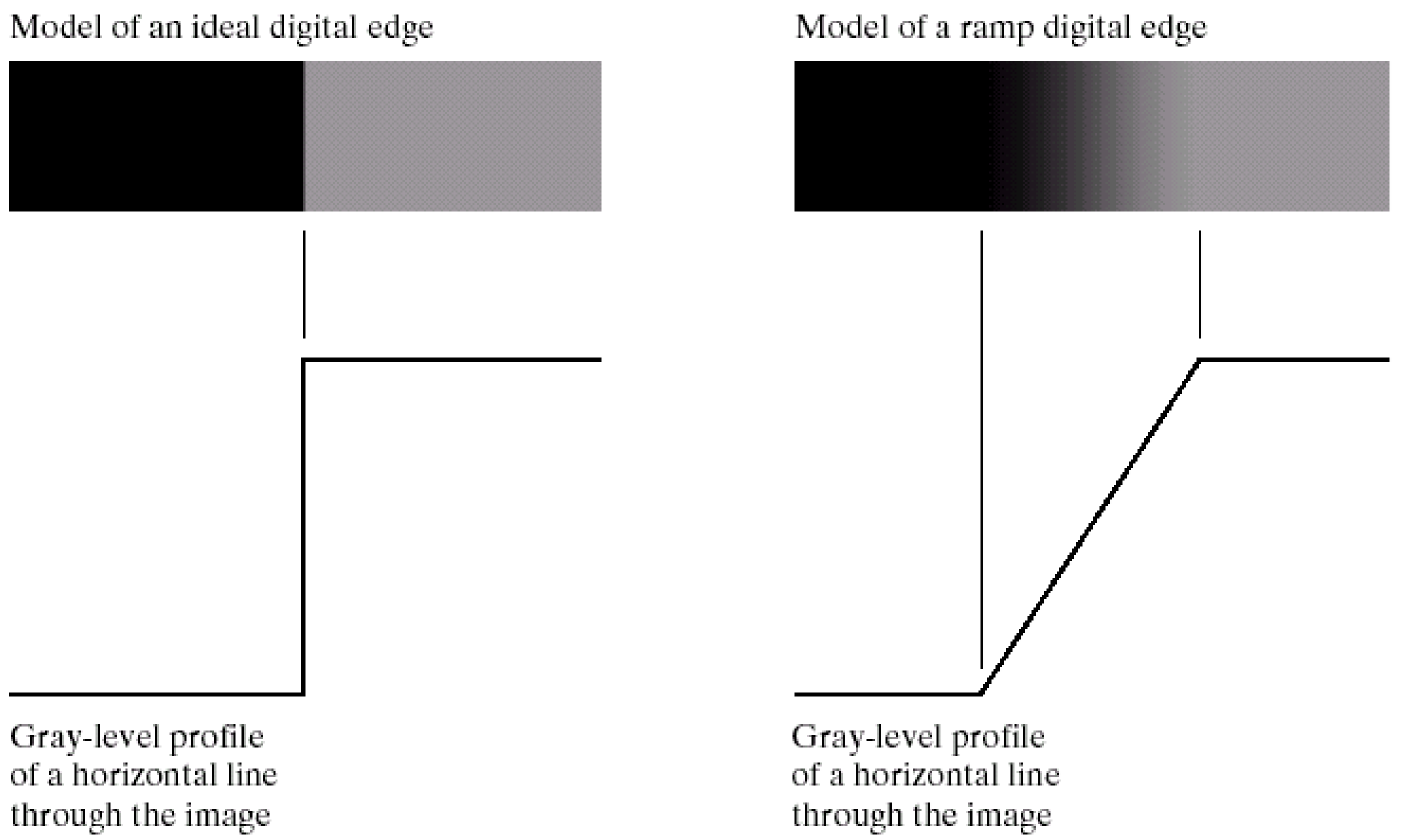
Detection of Discontinuities
Line Detection



a
b c

FIGURE 10.4 Illustration of line detection. (a) Binary wire-bond mask. (b) Absolute value of result after processing with −45° line detector. (c) Result of thresholding image (b).

Detection of Discontinuities
Edge Detection


Model of an ideal digital edge
Model of a ramp digital edge
Gray-level profile of a horizontal line through the image
Gray-level profile of a horizontal line through the image

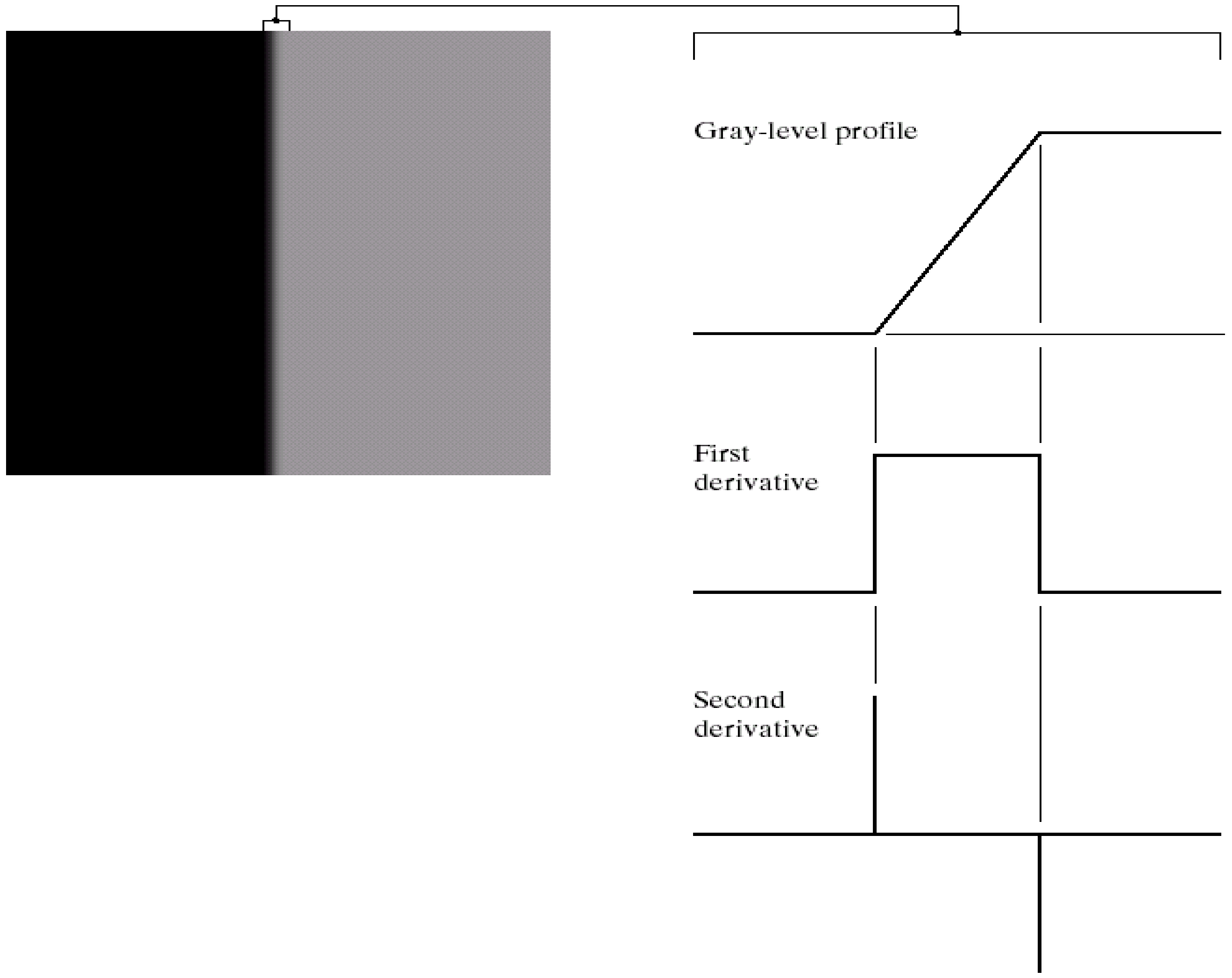
a b

FIGURE 10.5 (a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

Detection of Discontinuities
Edge Detection

a b

FIGURE 10.6
(a) Two regions separated by a vertical edge.
(b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.



Detection of Discontinuities
Edge Detection

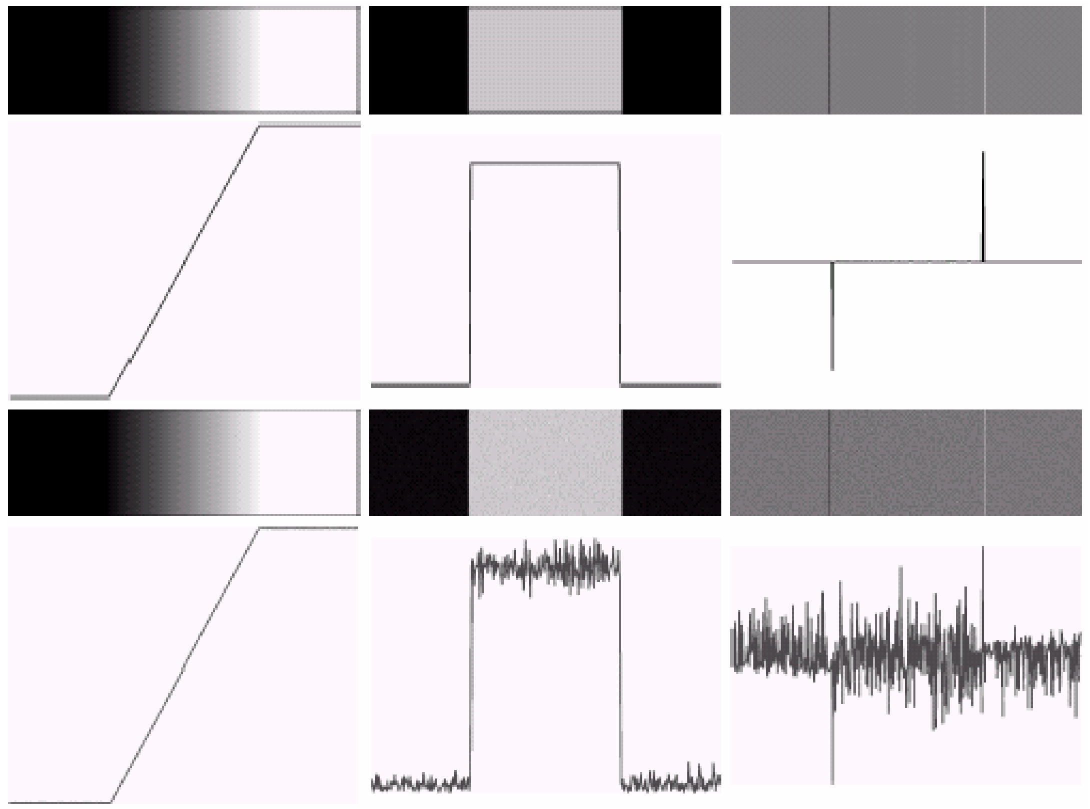


FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and $\sigma = 0.0, 0.1, 1.0$, and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

a
b
c
d

Detection of Discontinuities
Edge Detection

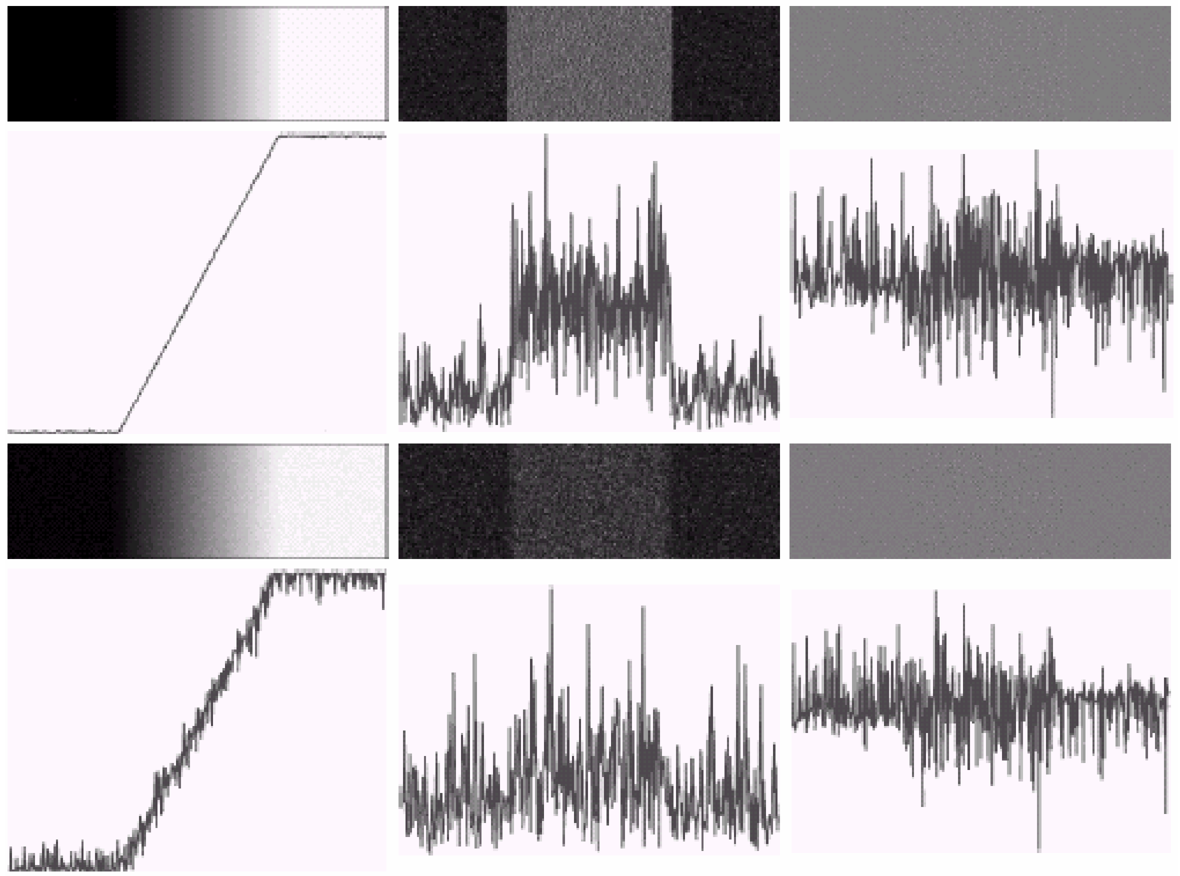


FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and $\sigma = 0.0, 0.1, 1.0,$ and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

a
b
c
d

Detection of Discontinuities
Gradient Operators

- First-order derivatives:
  - The gradient of an image $f(x,y)$ at location (x,y) is defined as the vector:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

  - The magnitude of this vector: $\nabla f = \mathrm{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$

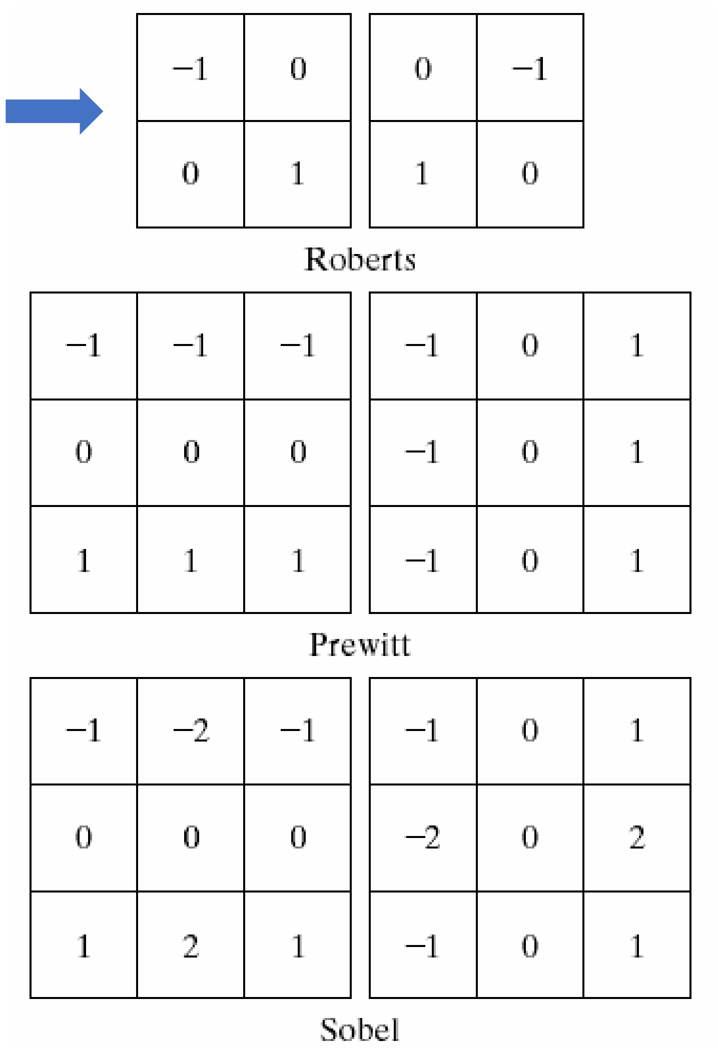
  - The direction of this vector: $\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right)$

Detection of Discontinuities
Gradient Operators

Roberts cross-gradient operators

Prewitt operators

Sobel operators



−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

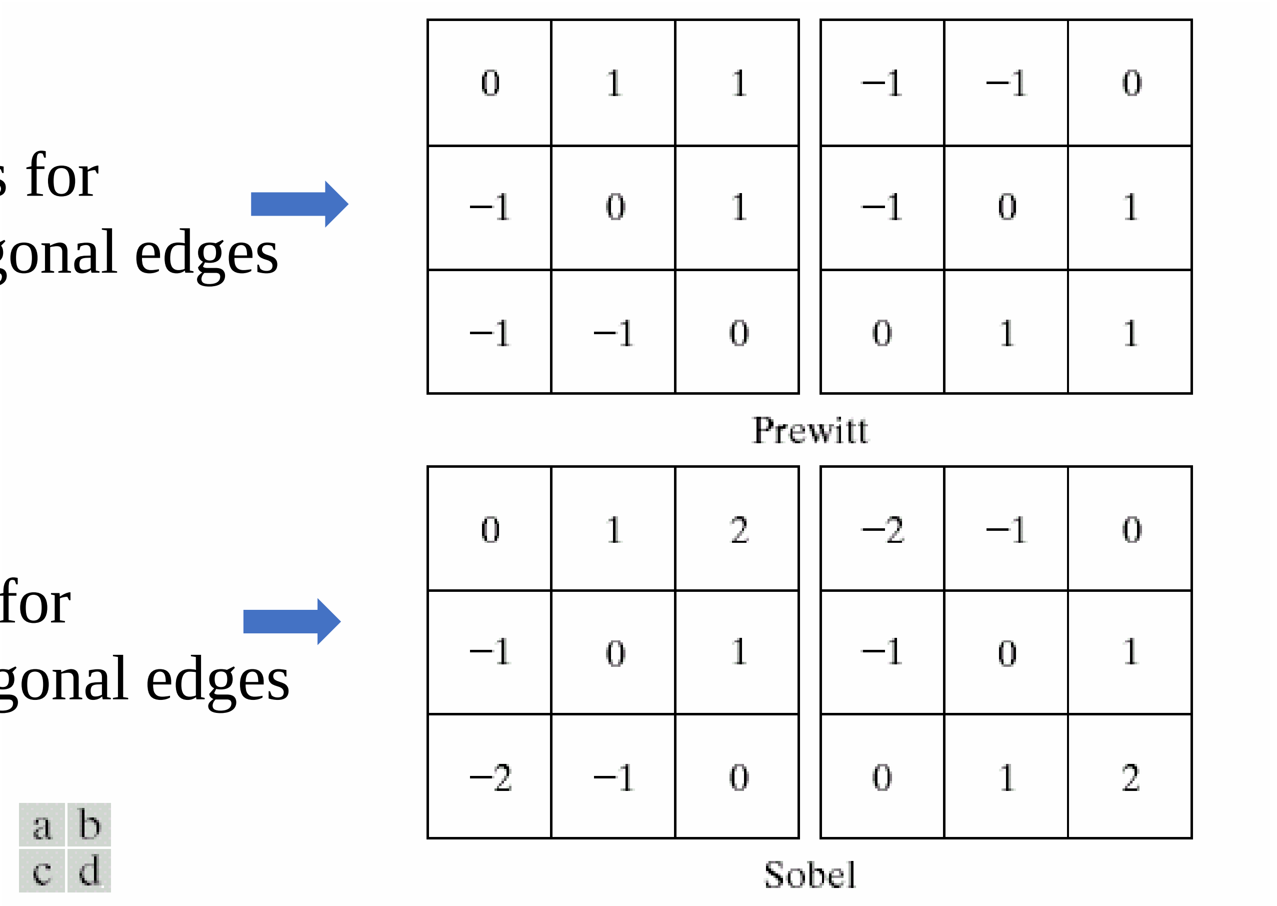
−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

Detection of Discontinuities
Gradient Operators

Prewitt masks for detecting diagonal edges

Sobel masks for detecting diagonal edges



0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

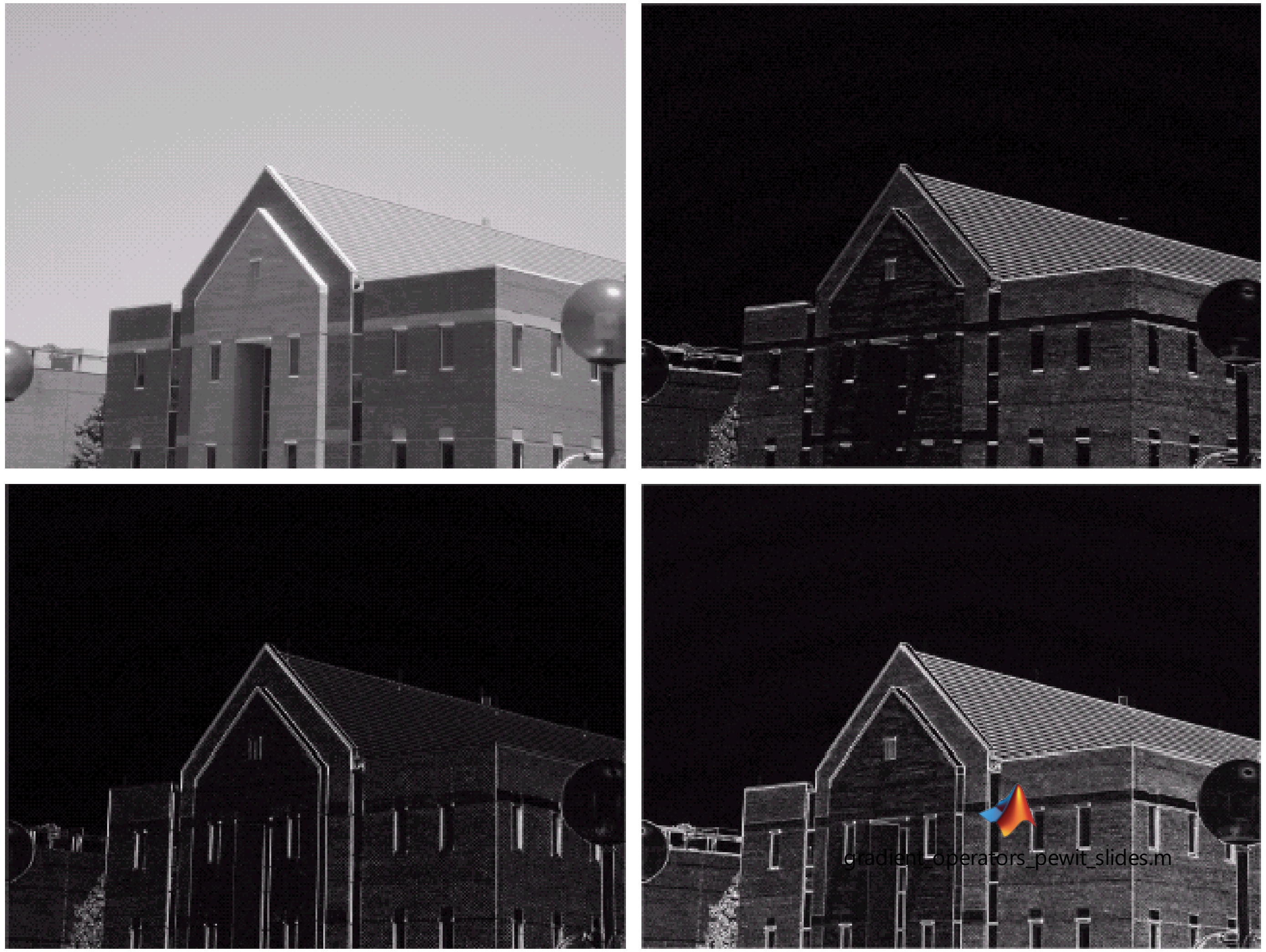
0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

FIGURE 10.9 Prewitt and Sobel masks for detecting diagonal edges.

Detection of Discontinuities
Gradient Operators: Example

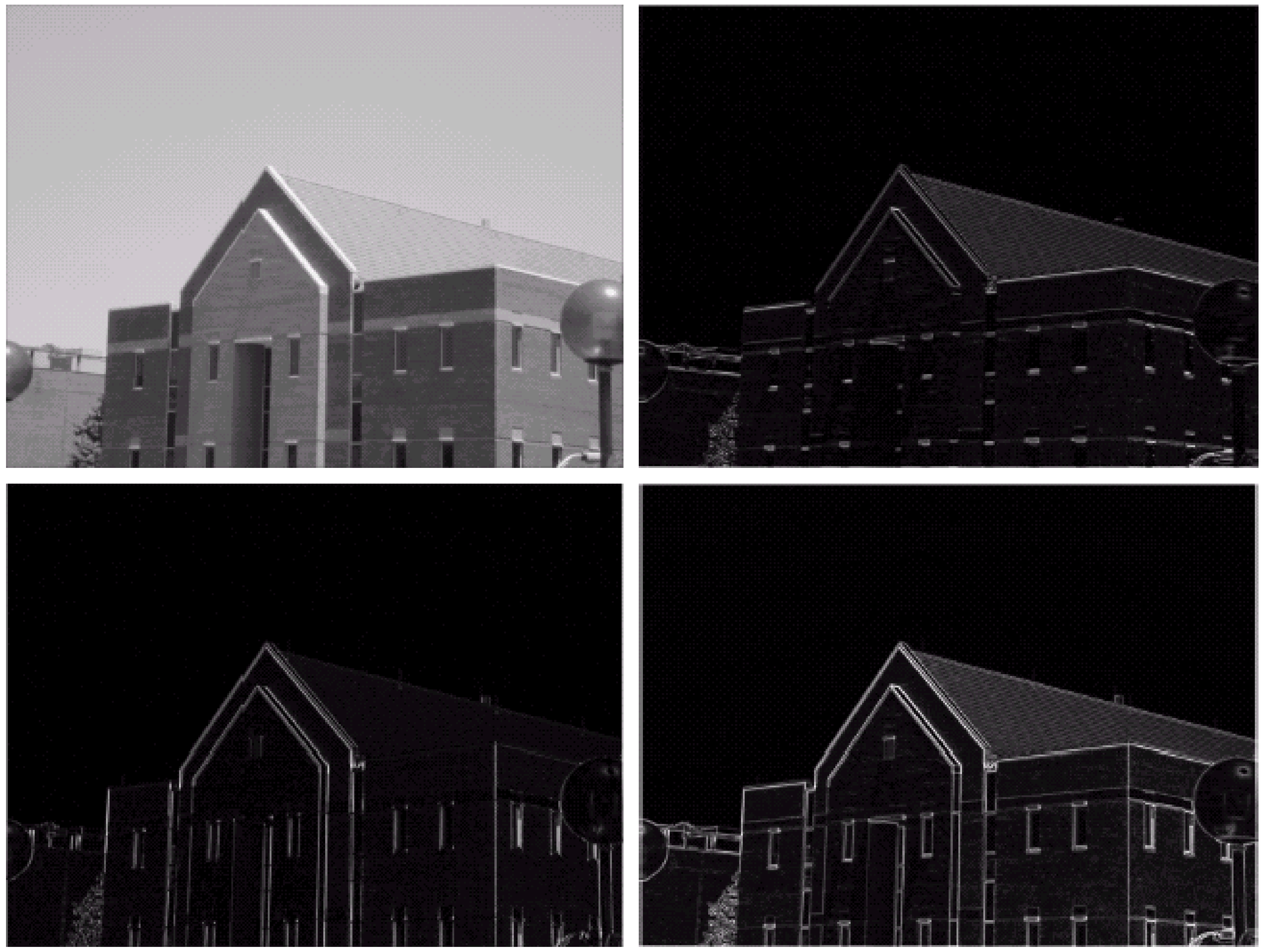


a b
c d

FIGURE 10.10 (a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

$$\nabla f \approx |G_x| + |G_y|$$

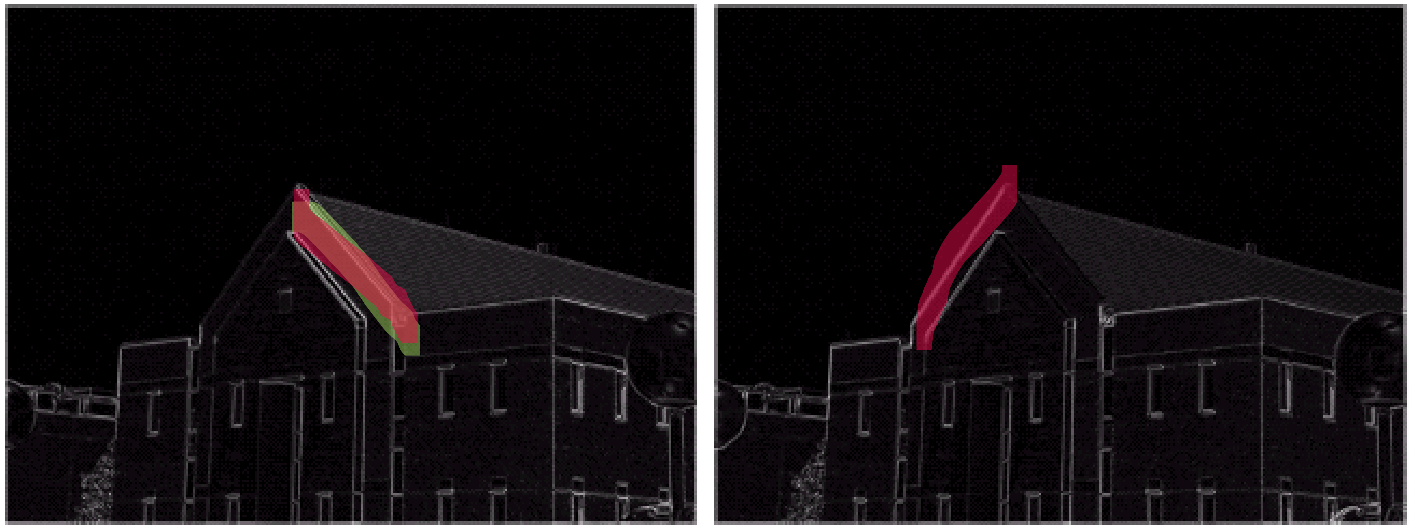
Detection of Discontinuities
Gradient Operators: Example



a	b
c	d

FIGURE 10.11 Same sequence as in Fig. 10.10, but with the original image smoothed with a 5 × 5 averaging filter.

Detection of Discontinuities Gradient Operators: Example



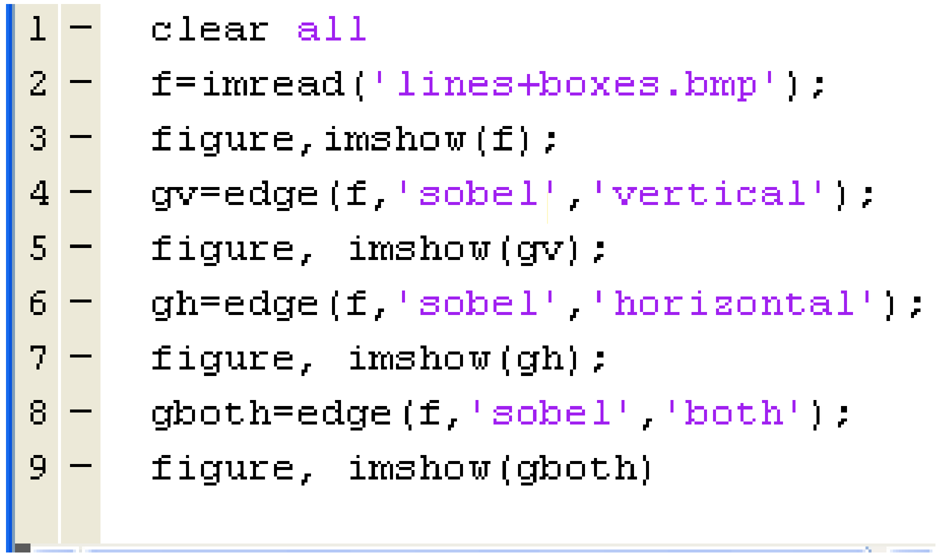
a b

FIGURE 10.12 Diagonal edge detection. (a) Result of using the mask in Fig. 10.9(c). (b) Result of using the mask in Fig. 10.9(d). The input in both cases was Fig. 10.11(a).

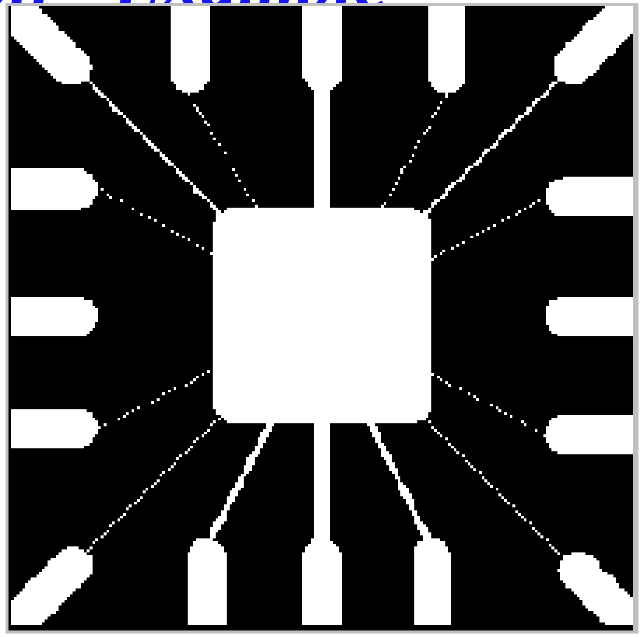
0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

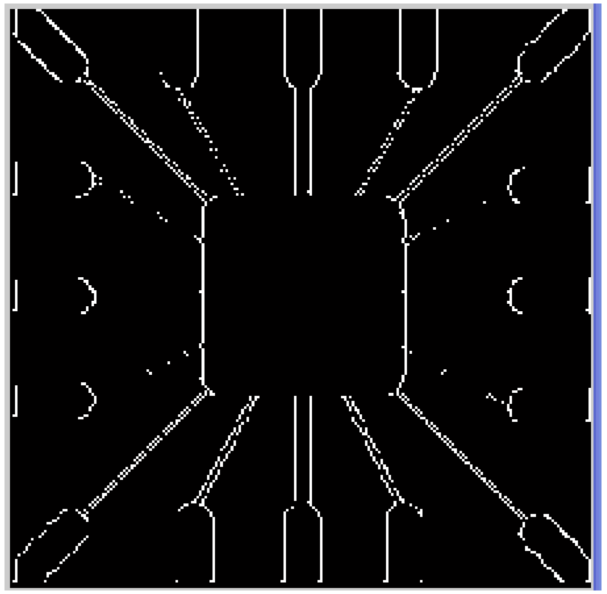
Sobel Edge Detection - Example



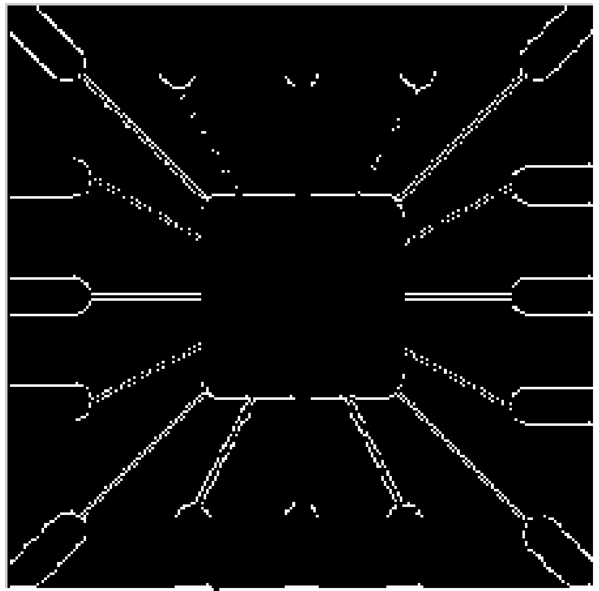
```
clear all
f=imread('lines+boxes.bmp');
figure,imshow(f);
gv=edge(f,'sobel','vertical');
figure, imshow(gv);
gh=edge(f,'sobel','horizontal');
figure, imshow(gh);
gboth=edge(f,'sobel','both');
figure, imshow(gboth)
```



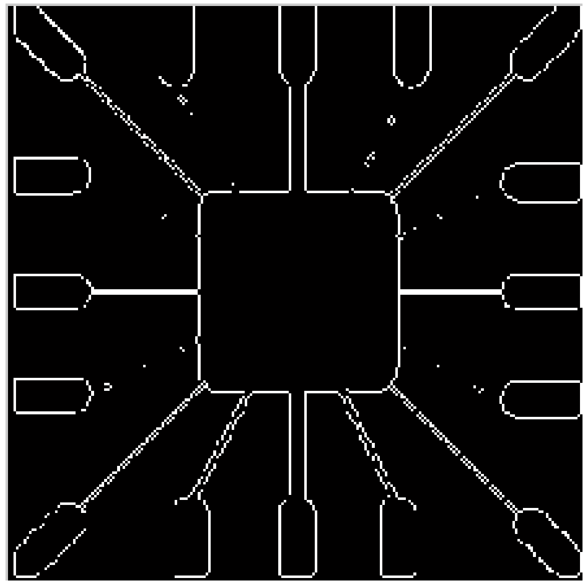
f



gv

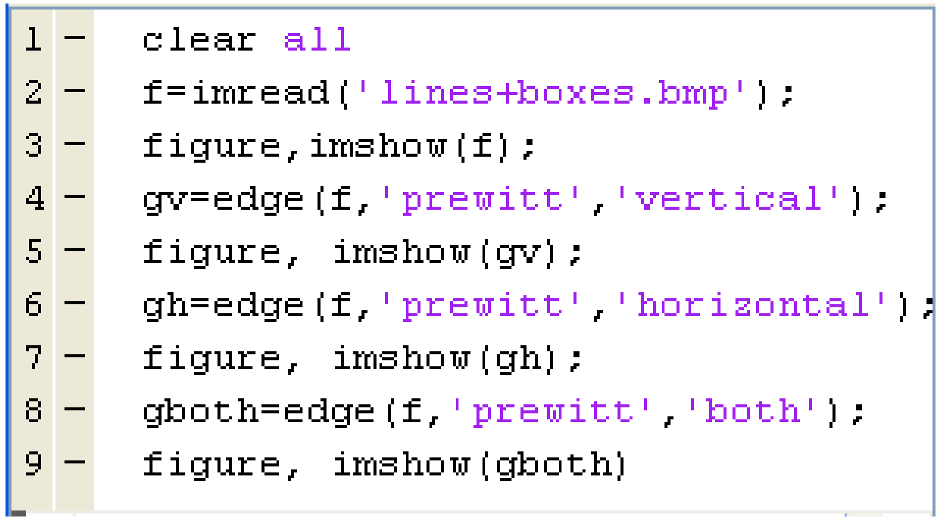


gh

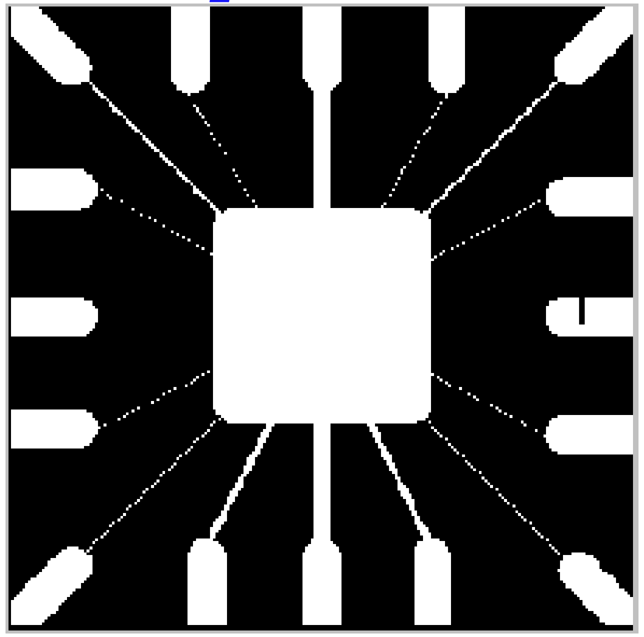


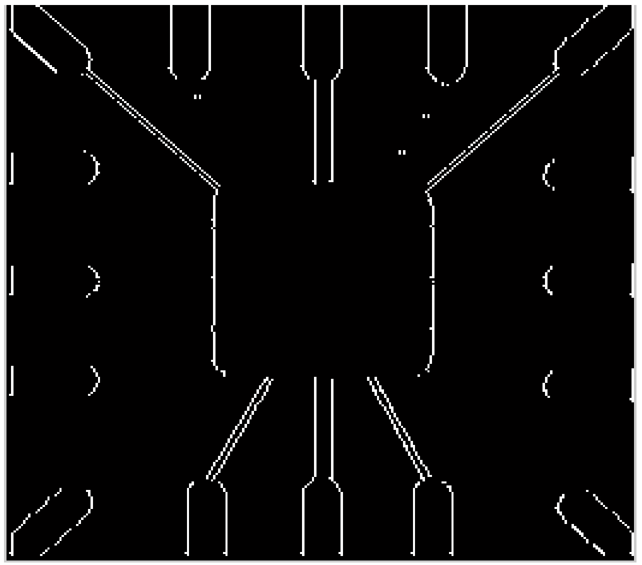
gboth

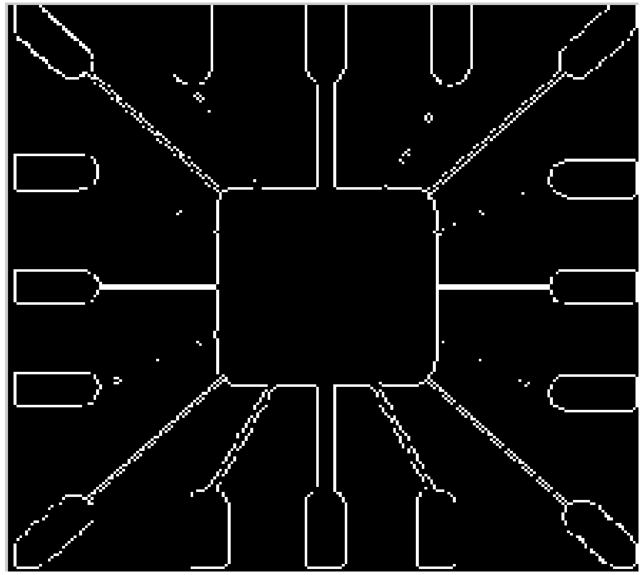
Prewitt Edge Detection - Example



```
clear all
f=imread('lines+boxes.bmp');
figure,imshow(f);
gv=edge(f,'prewitt','vertical');
figure, imshow(gv);
gh=edge(f,'prewitt','horizontal');
figure, imshow(gh);
gboth=edge(f,'prewitt','both');
figure, imshow(gboth)
```

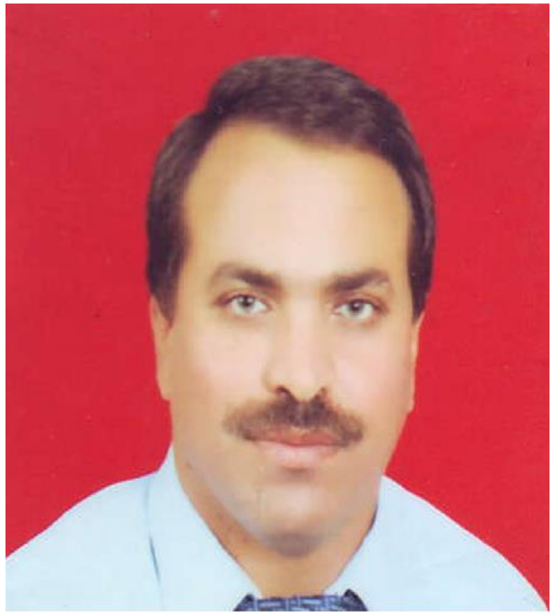


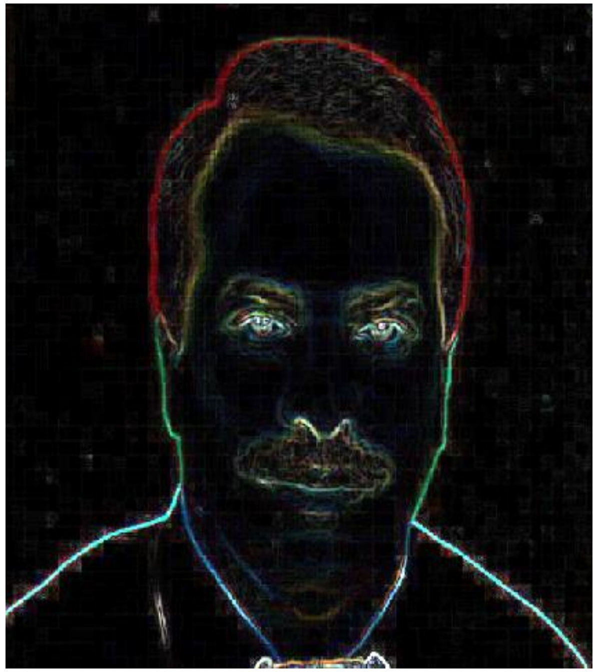


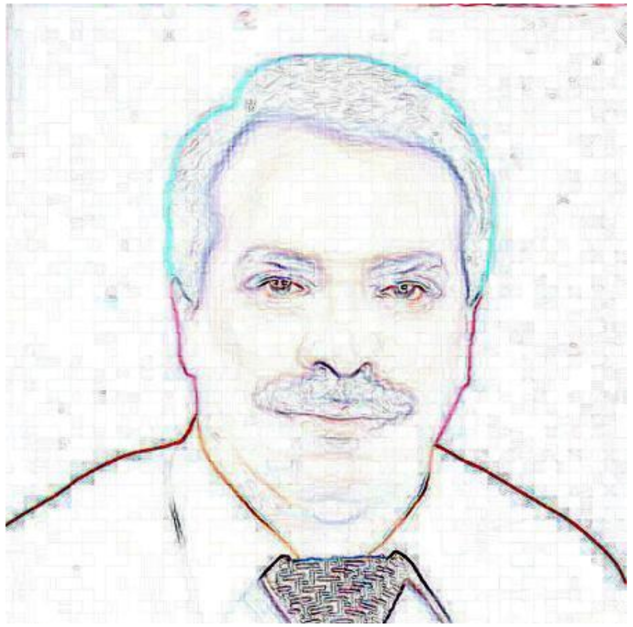


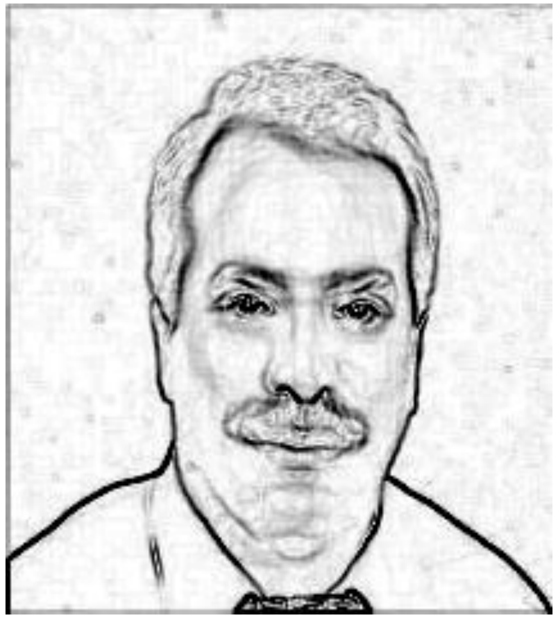
gv gh gboth

Sobel edge detection – example









Detection of Discontinuities
Gradient Operators

- Second-order derivatives: (The Laplacian)
  - The Laplacian of a 2D function $f(x,y)$ is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  - Two forms in practice:

FIGURE 10.13 Laplacian masks used to implement Eqs. (10.1-14) and (10.1-15), respectively.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Segmentation by thresholding

- ✓ ***Thresholding is the simplest segmentation method.***
- ✓ ***The pixels are partitioned depending on their intensity value.***

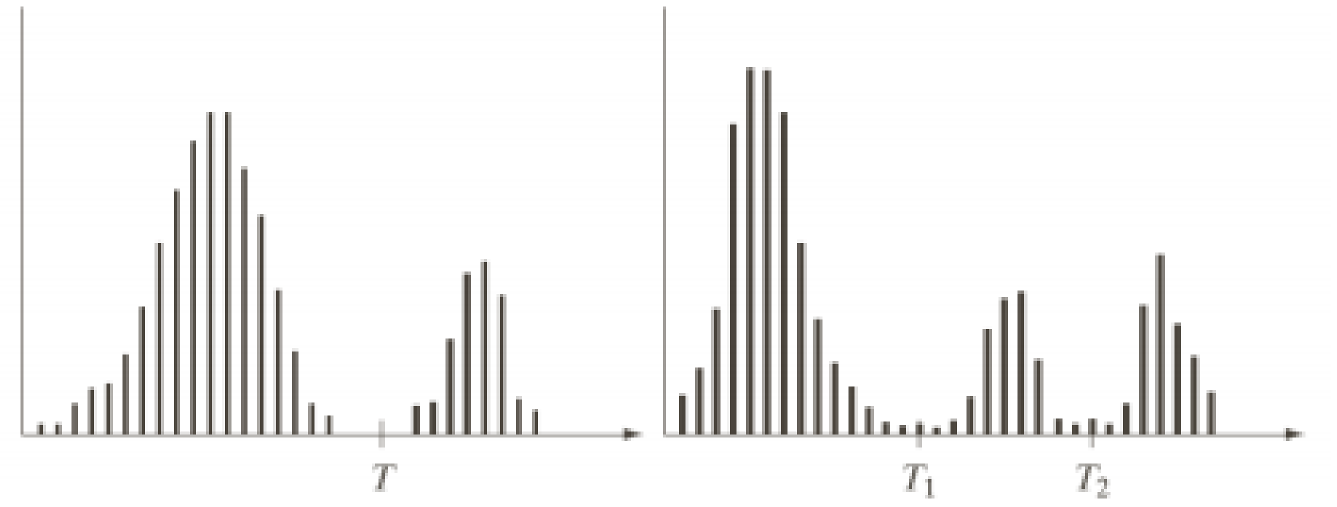
1. ***Global thresholding, using an appropriate threshold T:***

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

1. ***Local or regional thresholding, if T depends on a neighborhood of (x, y).***
2. ***Multiple thresholding:***

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

Choosing the thresholds

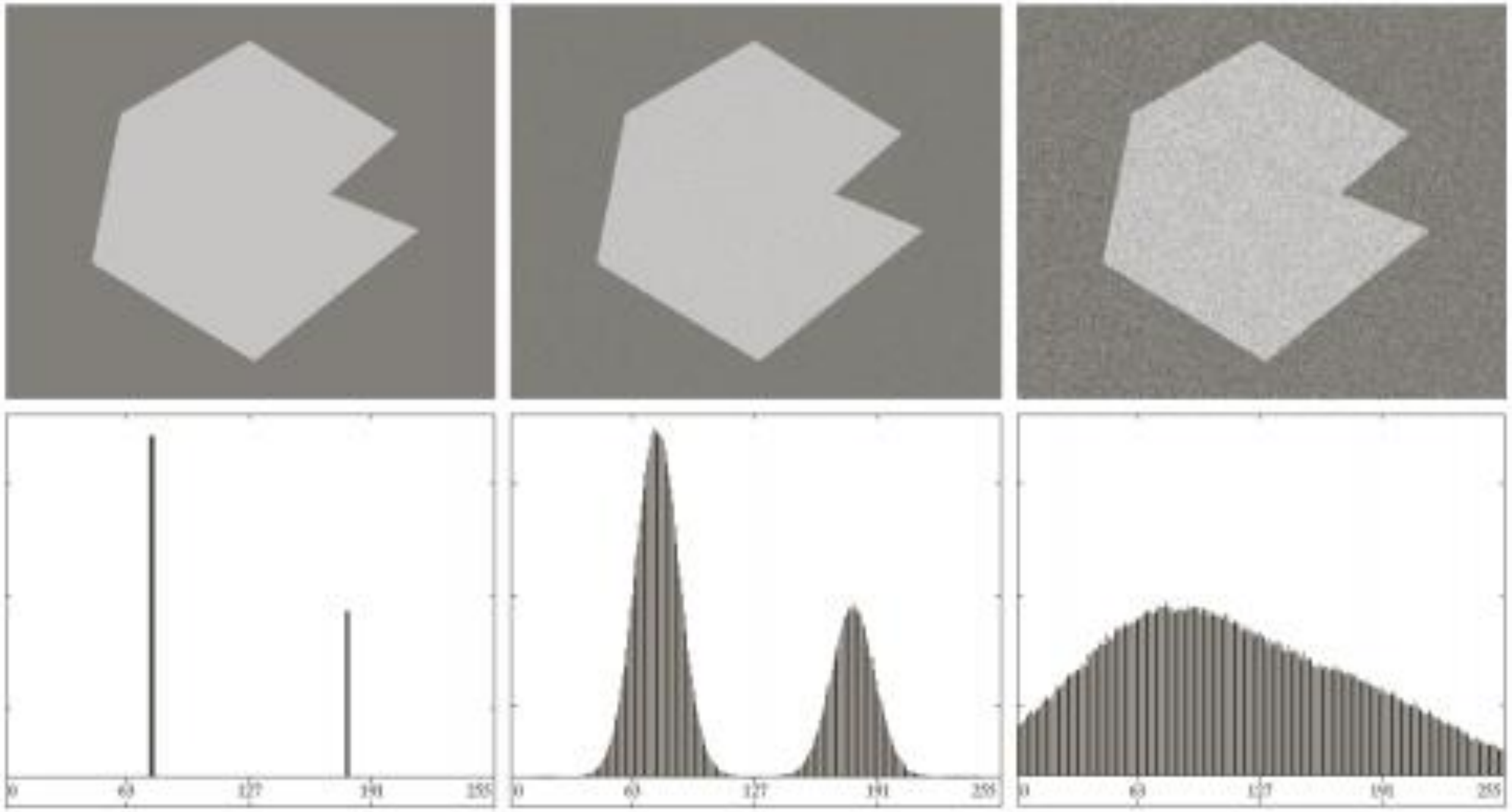


✓ ***Peaks and valleys of the image histogram can help in choosing the appropriate value for the threshold(s).***

✓ ***Some factors affect the suitability of the histogram for guiding the choice of the threshold:***

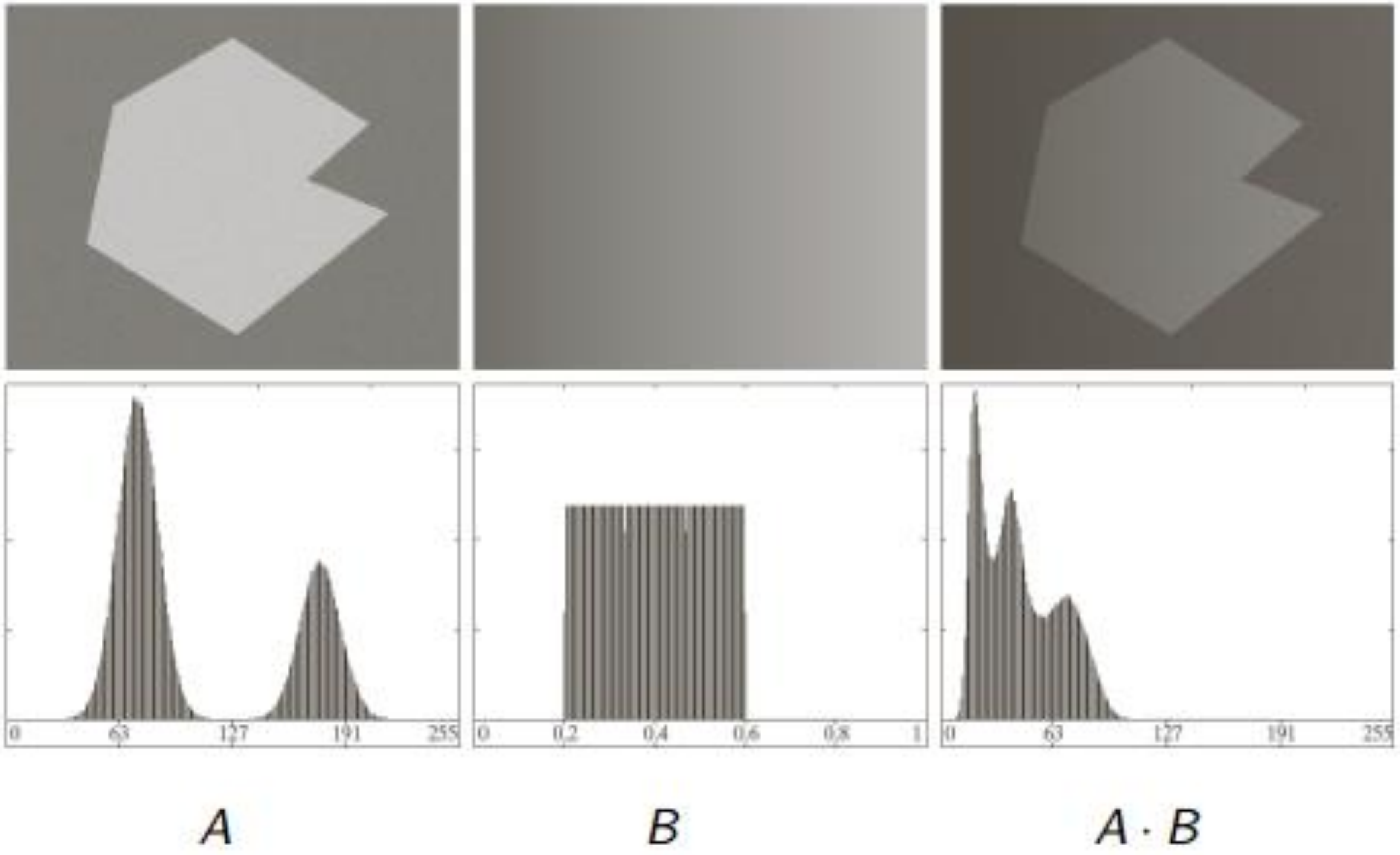
1. *the separation between peaks;*
2. *the noise content in the image;*
3. *the relative size of objects and background;*

Noise role in thresholding

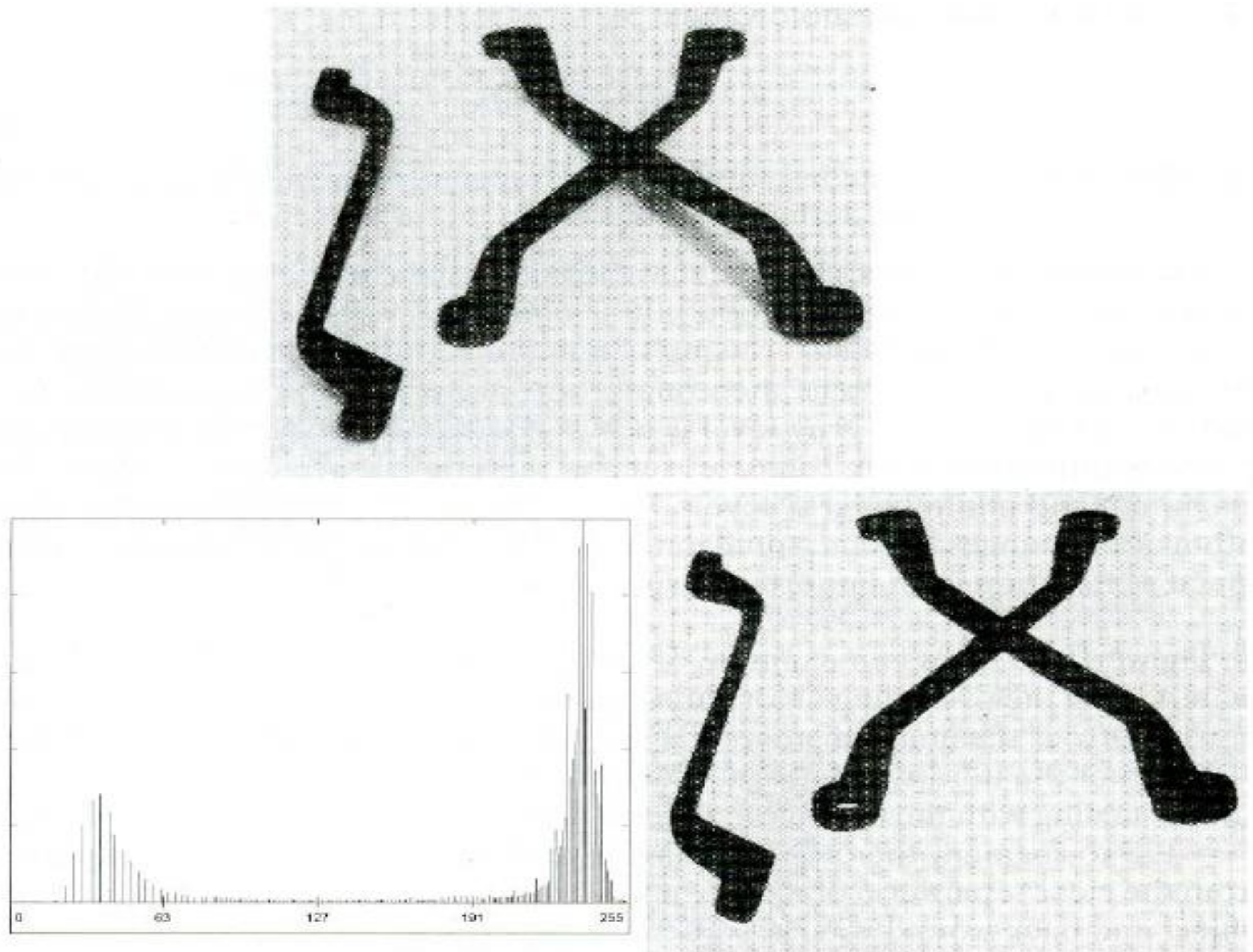


No noise 10% noise 50% noise

Illumination role in thresholding



Global thresholding



a
b c

FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

Estimated global thresholding

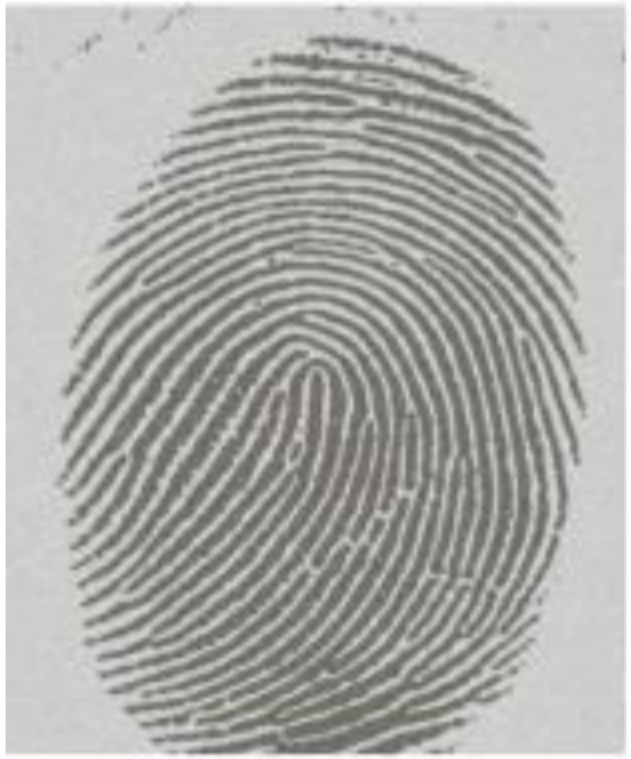
A simple algorithm:

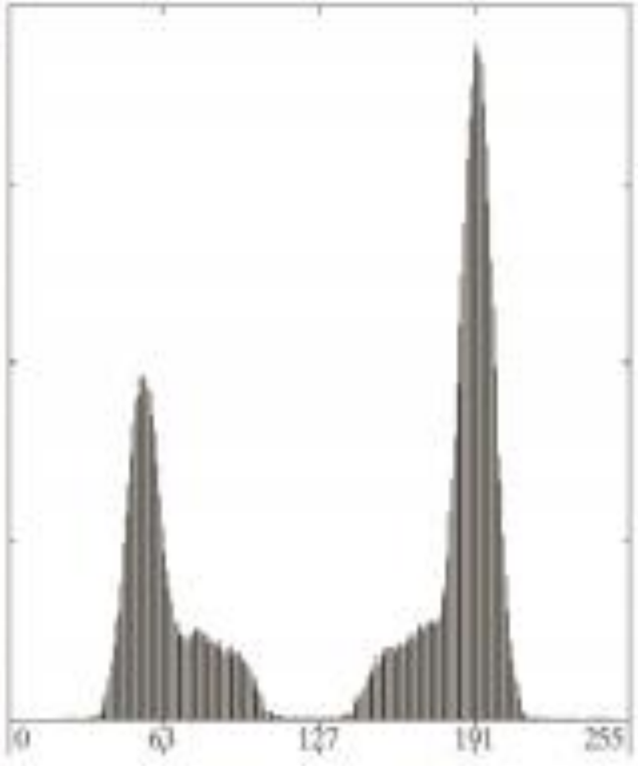
1. Initial estimate of T
2. Segmentation using T:
   - G_1, pixels brighter than T;
   - G_2, pixels darker than (or equal to) T.
3. Computation of the average intensities m_1 and m_2 of G_1 and G_2.
4. New threshold value:

$$T_{\text{new}} = \frac{m_1 + m_2}{2}$$

5. If $|T - T_{\text{new}}| > \Delta T$, back to step 2, otherwise stop.

Global thresholding: an example





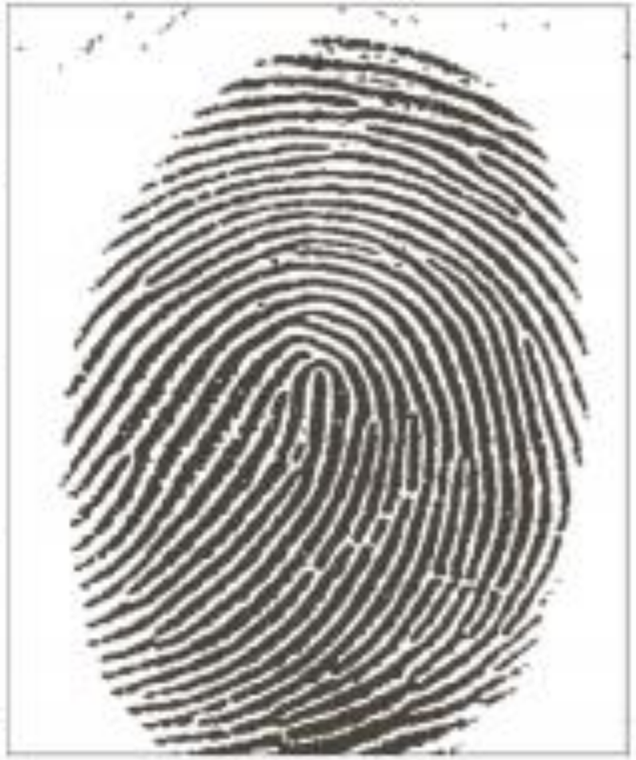
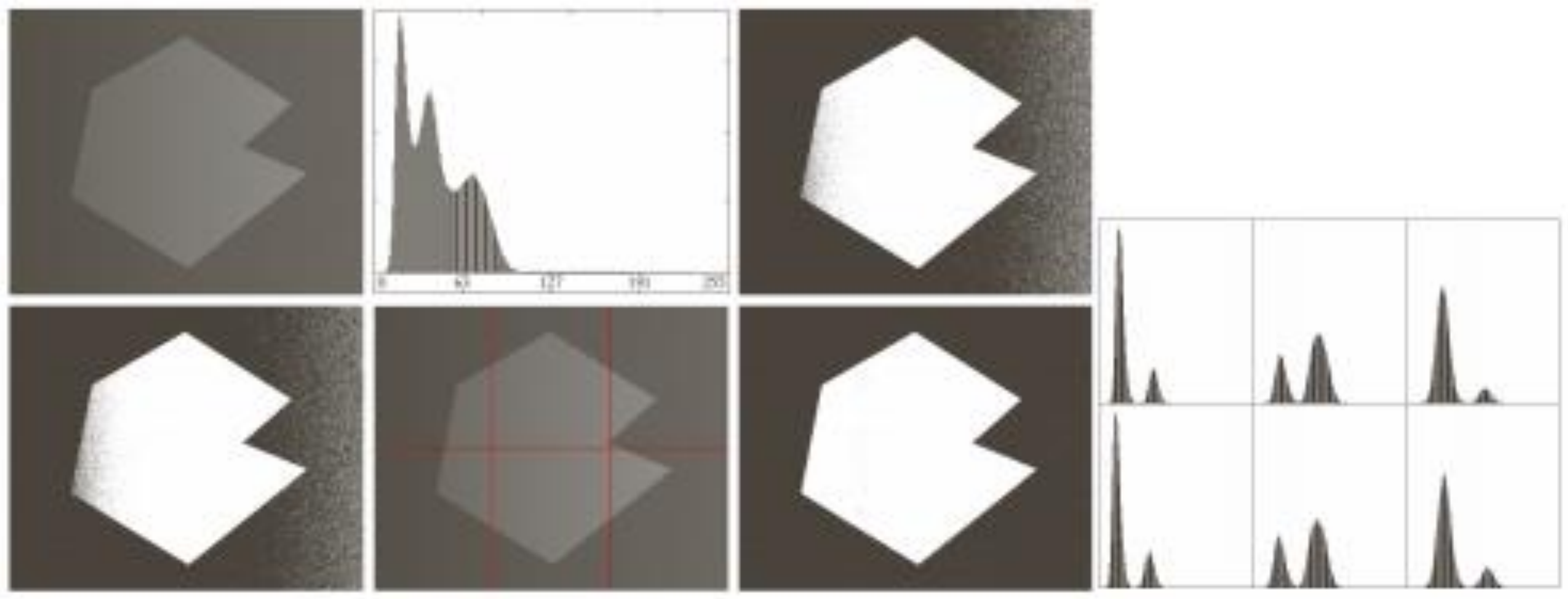


Image partitioning based Thresholding (Adaptive or dynamic thresh holding)

- ✓ ***In order to face nonuniform illumination, the image is partitioned, and the thresholding is operated on each partition.***
  - ✓ ***In each partition, the illumination is supposed uniform.***
  - ✓ ***In each partition, objects and backgrounds have to be equally represented.***



Edge Linking and Boundary Detection
Local Processing

- Two properties of edge points are useful for edge linking:
  - the strength (or magnitude) of the detected edge points
  - their directions (determined from gradient directions)
- This is usually done in local neighborhoods.
- Adjacent edge points with similar magnitude and direction are linked.

Edge Linking and Boundary Detection
Local Processing: Example

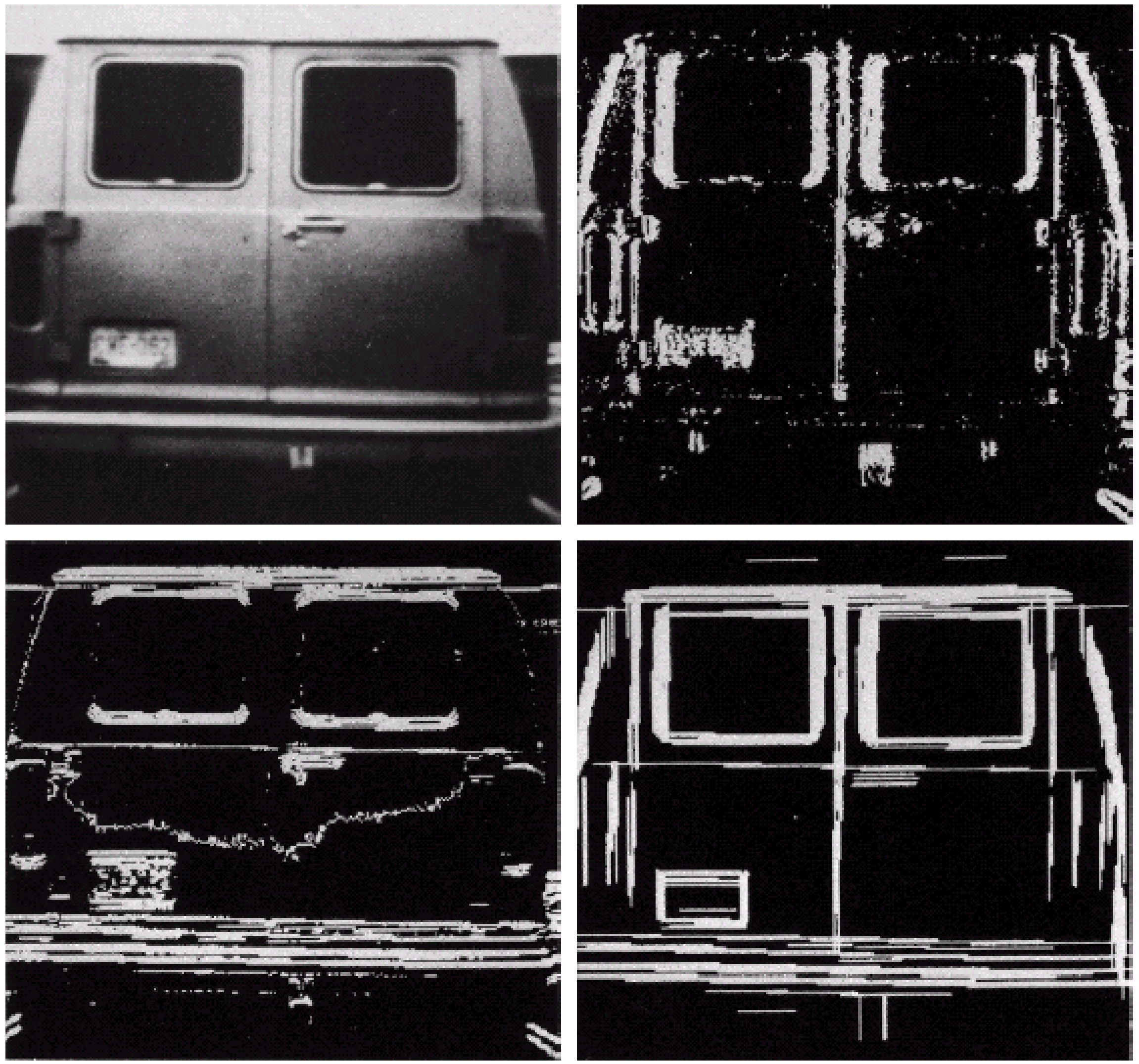


FIGURE 10.16 (a) Input image. (b) G_y component of the gradient. (c) G_x component of the gradient. (d) Result of edge linking. (Courtesy of Perceptics Corporation.)

In this example, we can find the license plate candidate after edge linking process.

Region-Based Segmentation

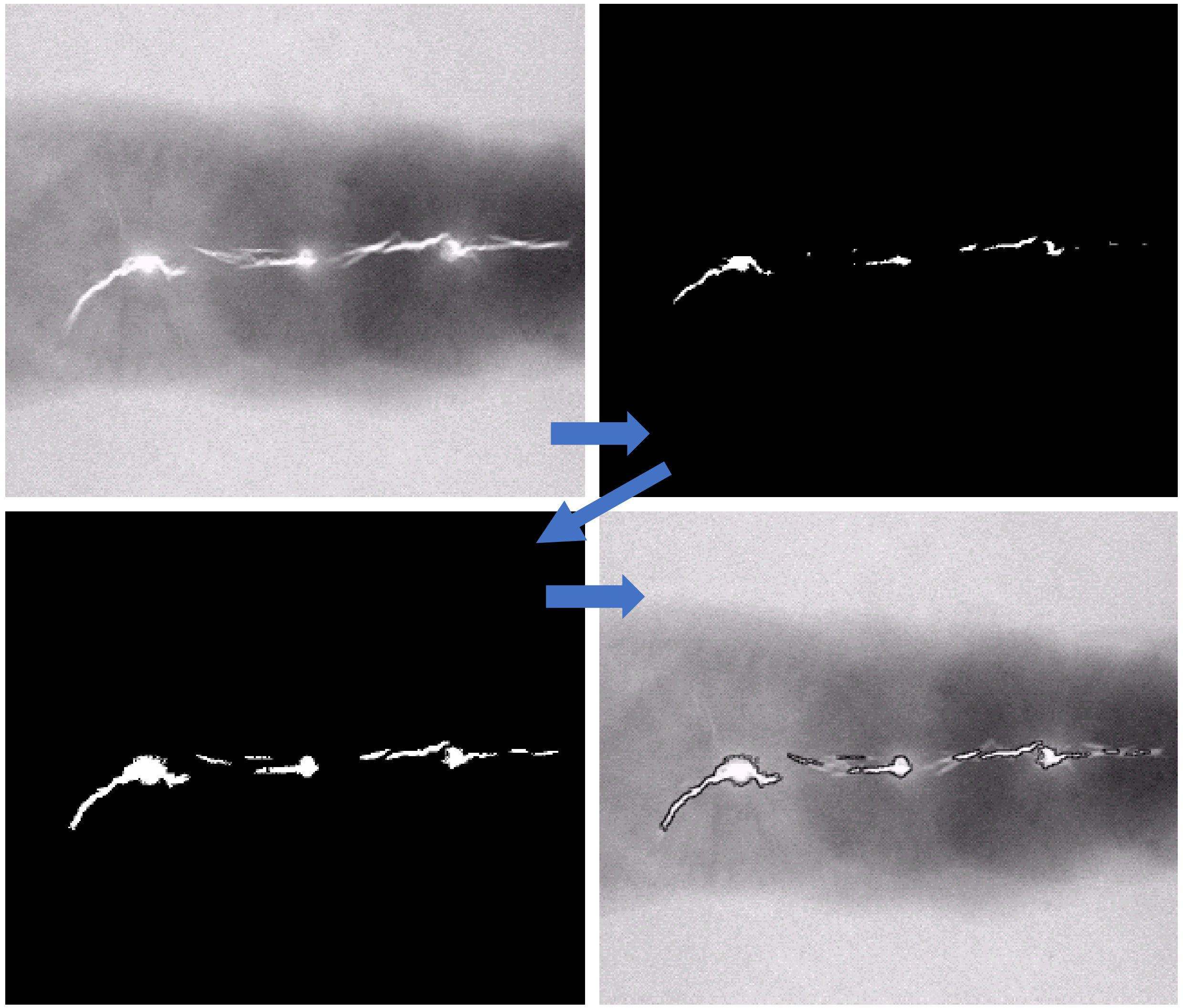
- Edges and thresholds sometimes do not give good results for segmentation.
- Region-based segmentation is based on the connectivity of similar pixels in a region.
  - Each region must be uniform.
  - Connectivity of the pixels within the region is very important.
- There are two main approaches to region-based segmentation: region growing and region splitting.

Region-Based Segmentation

Region Growing

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-Based Segmentation
Region Growing

- Fig. 10.41 shows the histogram of Fig. 10.40 (a). It is difficult to segment the defects by thresholding methods. (Applying region growing methods are better in this case.)

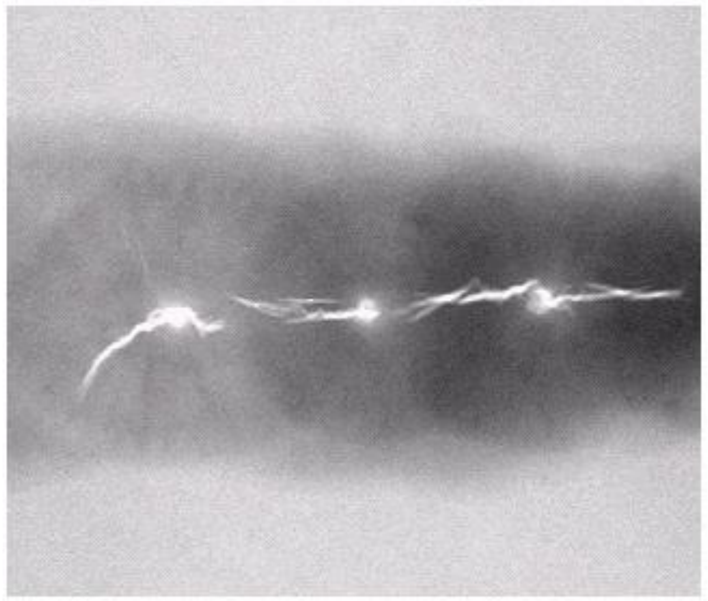


Figure 10.40(a)

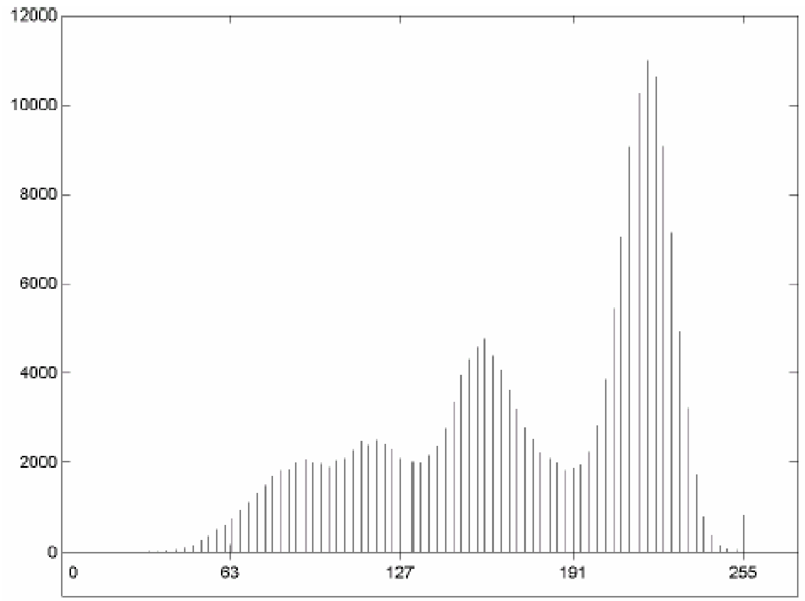


Figure 10.41

Region-Based Segmentation
Region Splitting and Merging

- Region splitting is the opposite of region growing.
  - First there is a large region (possible the entire image).
  - Then a predicate (measurement) is used to determine if the region is uniform.
  - If not, then the method requires that the region be split into two regions.
  - Then each of these two regions are independently tested by the predicate (measurement).
  - This procedure continues until all resulting regions are uniform.

Region splitting on the following image Assume the Threshold value be $<= 4$.

R1a R2a

5	6	6	6	7	7	6	6
6	7	6	$\textcircled{7}$	5	5	4	7
6	6	$\textcircled{4}$	4	3	2	5	6
5	4	5	4	2	3	4	6
0	3	2	3	3	2	4	7
0	0	0	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

R3a

overall Max, min

$7 - 0 = 7$

$R_1a = 7 - 4 = 3$

$R_2a = 7 - 2 = 5$

$R_2a_1 = 7 - 5 = 2$

$R_2a_2 = 7 - 4 = 3$

$R_2a_3 = 3 - 2 = 1$

$R_2a_4 = 5 - 4 = 2$

$R_3a = 3 - 0 = 3$

$R_4a = 7 - 0 = 7$

R1a R2a R3a R4a

$$\begin{tikzcd}
 & & R_2a \arrow[dll, no head] \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
R_2a_1 & R_2a_2 & R_2a_3 & R_2a_4
\end{tikzcd}$$

$$\begin{tikzcd}
 & & R_4a \arrow[dll, no head] \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
R_4a_1 & R_4a_2 & R_4a_3 & R_4a_4
\end{tikzcd}$$

$3 - 2 = 1$

$7 - 4 = 3$

$3 - 0 = 3$

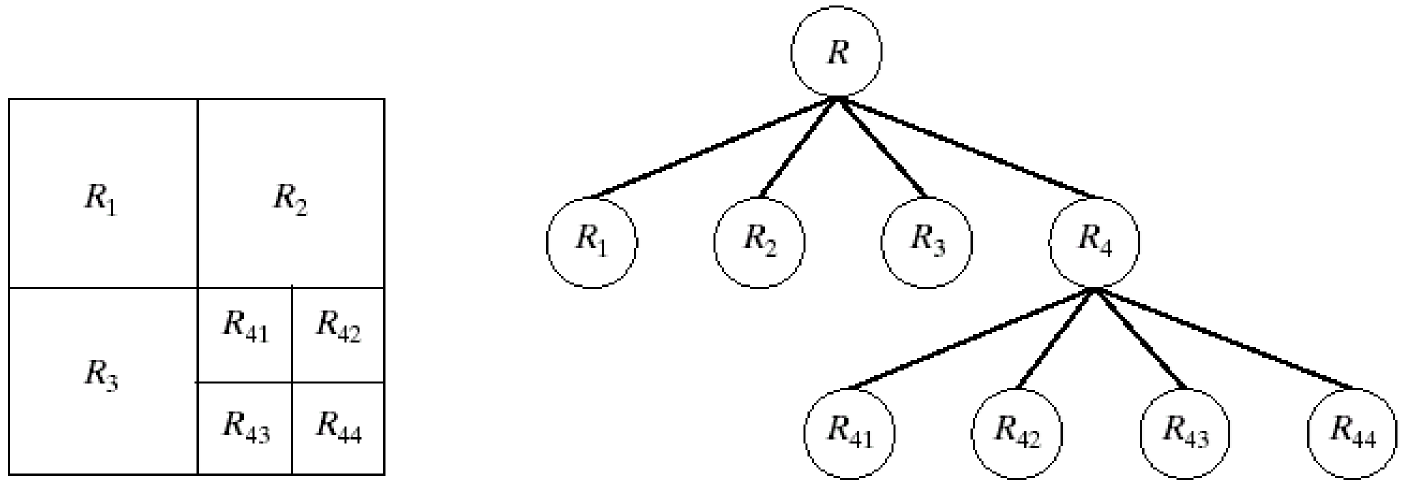
$5 - 4 = 1$

Region-Based Segmentation
Region Splitting

- The main problem with region splitting is determining where to split a region.
- One method to divide a region is to use a quadtree structure.
- Quadtree: a tree in which nodes have exactly four descendants.

a b

FIGURE 10.42
(a) Partitioned image.
(b) Corresponding quadtree.



region-based segmentation
Seeded Region Growing (SRG)

- Region growing: Groups pixels or sub-region into larger regions.
  - step1:
    - Start with a set of “seed” points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed.
  - step2:
    - Region splitting and merging

region-based segmentation
Seeded Region Growing (SRG)

- Advantage:
  - With good connectivity
- Disadvantage:
  - Initial seed-points:
    - different sets of initial seed-point cause different segmented result
  - Time-consuming problem

region-based segmentation
Unseeded Region Growing (USRG)

- Advantage:
  - easy to use
  - can readily incorporate high level knowledge of the image composition through region threshold

- Disadvantage:
  - slow speed

Region Growing Algorithm

Example: For a given image, show the results of region growing algorithm

5×5

1	0	7	8	7
0	1	8	9	8
0	0	7	9	8
0	1	8	8	9
1	2	8	8	9

↑

* Seed point → s_1 9 s_2 1

* Threshold, T $\boxed{T \leq 4}$

* $s_1 = 9$

$|f(x,y) - f(x',y')| \leq 4$

$|f(x,y) - 9| \leq 4$

$\hookrightarrow f(x,y) = \{5, 6, 7, 8, 9\} \Rightarrow$ Ⓐ ✓

* $s_2 = 1$

$|f(x,y) - f(x',y')| \leq 4$

$|f(x,y) - 1| \leq 4$ ✓

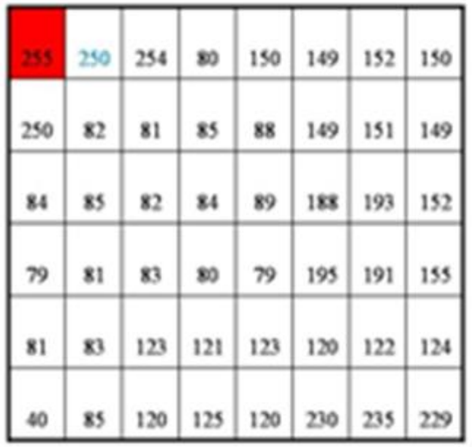
$f(x,y) = \{0, 1, 2, 3, 4, 5\} \Rightarrow$ Ⓑ ✓

Resultant Image

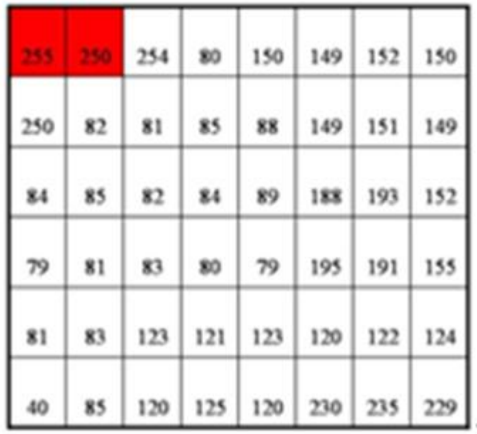
B	B	A	A	A
B	B	A	A	A
B	B	A	A	A
B	B	A	A	A
B	B	A	A	A

region-based segmentation
fast scanning

- Fast scanning Algorithm:
  - The fast scanning algorithm somewhat resembles unseeded region growing
  - the number of clusters of both two algorithm would not be decided before image passing through them.

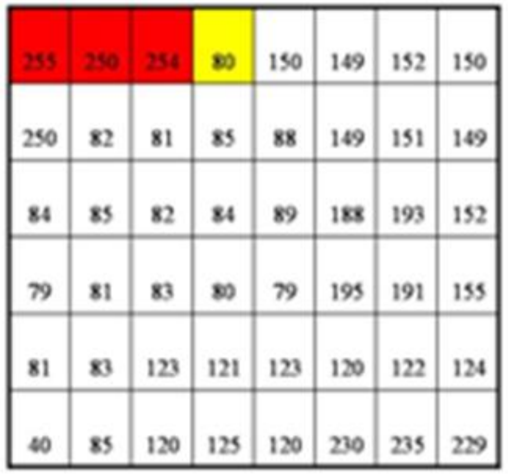


255	250	254	80	150	149	152	150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229

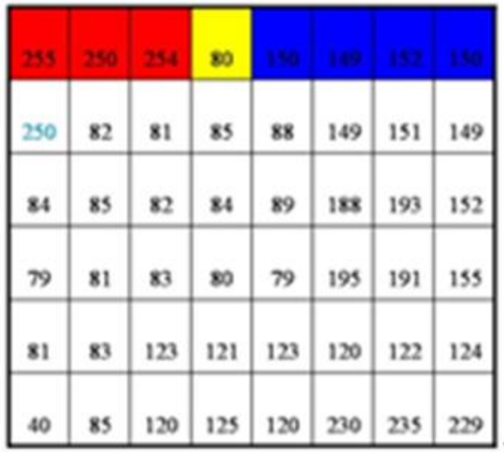


255	250	254	80	150	149	152	150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229

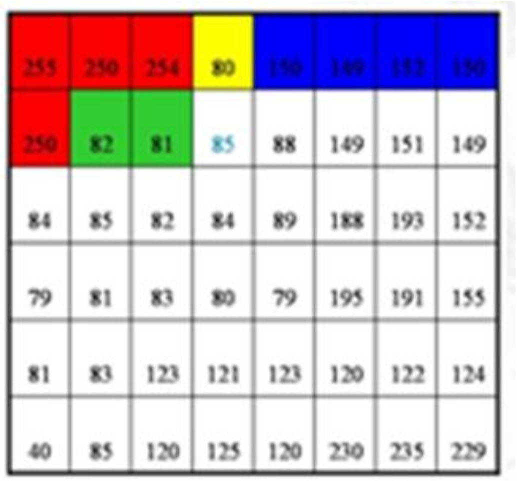
region-based segmentation
fast scanning



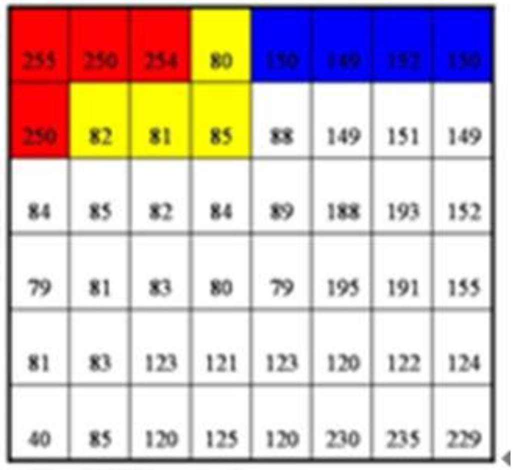
🟥 255	🟥 250	🟥 254	🟨 80	150	149	152	150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229



🟥 255	🟥 250	🟥 254	🟨 80	🟦 150	🟦 149	🟦 152	🟦 150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229

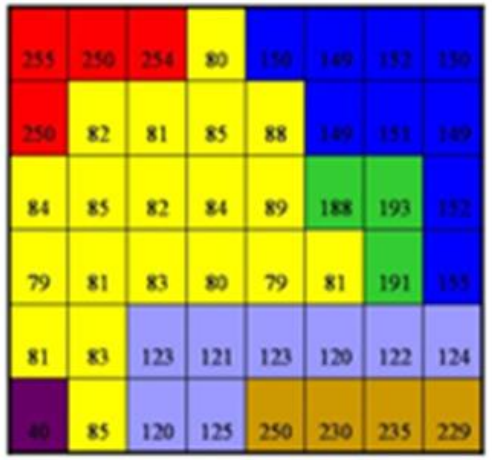


🟥 255	🟥 250	🟥 254	🟨 80	🟦 150	🟦 149	🟦 152	🟦 150
🟥 250	🟩 82	🟩 81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229

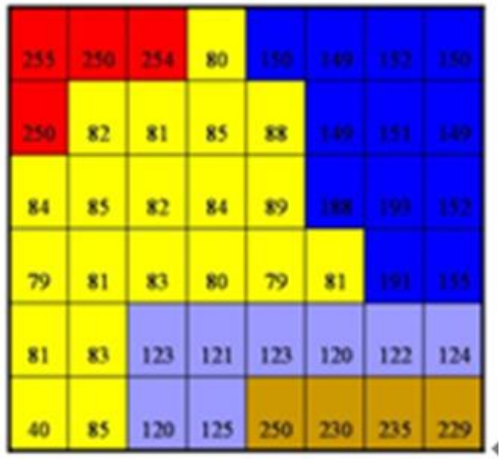


🟥 255	🟥 250	🟥 254	🟨 80	🟦 150	🟦 149	🟦 152	🟦 150
🟥 250	🟨 82	🟨 81	🟨 85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	195	191	155
81	83	123	121	123	120	122	124
40	85	120	125	120	230	235	229

region-based segmentation
fast scanning



255	250	254	80	150	149	152	150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	81	191	155
81	83	123	121	123	120	122	124
40	85	120	125	250	230	235	229



255	250	254	80	150	149	152	150
250	82	81	85	88	149	151	149
84	85	82	84	89	188	193	152
79	81	83	80	79	81	191	155
81	83	123	121	123	120	122	124
40	85	120	125	250	230	235	229

- Last step:
  - merge small region to big region

The Canny Edge Detector

- Although the algorithm is more complex, the performance of the Canny edge detector (Canny [1986]) discussed in this section is superior in general to the edge detectors discussed thus far. Canny's approach is based on three basic objectives:
- *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.

Details

- There are 5 steps involved in Canny edge detection, as shown in fig. We will be using the following image for illustration.
- **Image Smoothening**
- we convert the image to grayscale as edge detection does not dependent on colors. Then we remove the noise in the image with a Gaussian filter as edge detection is prone to noise.
- **Finding Intensity Gradients of the Image**
- apply the Sobel kernel in horizontal and vertical directions to get the first derivative in the horizontal direction (Gx) and vertical direction (Gy) on the smoothened image. We then calculate the edge gradient(G) and Angle(ϑ) as given below,
- $Edge\_Gradient(G) = \sqrt{(G_x^2+G_y^2)}$
- $Angle(\vartheta)=tan^{-1}(G_y/G_x)$
- We know that the gradient direction is perpendicular to the edge. We round the angle to one of four angles representing vertical, horizontal, and two diagonal directions.

Details

- **Non-Max Suppression**
- Now we remove all the unwanted pixels which may not constitute the edge. For this, every pixel is checked in the direction of the gradient if it is a local maximum in its neighbourhood. If it is a local maximum, it is considered for the next stage, otherwise, it is darkened with 0. This will give a thin line in the output image.
- **Double Threshold**
- Pixels due to noise and color variation would persist in the image. So, to remove this, we get two thresholds from the user, lowerVal and upperVal. We filter out edge pixels with a weak gradient(lowerVal) value and preserve edge pixels with a high gradient value(upperVal). Edges with an intensity gradient more than upperVal are sure to edge, and those below lowerVal are sure to be non-edges, so discarded. The pixels that have pixel value lesser than the upperVal and greater than the lowerVal are considered part of the edge if it is connected to a “sure-edge”. Otherwise, they are also discarded.

Details

- **Edge Tracking by Hysteresis**
- A pixel is made as a strong pixel if either of the 8 pixels around it is strong(pixel value=255) else it is made as 0.