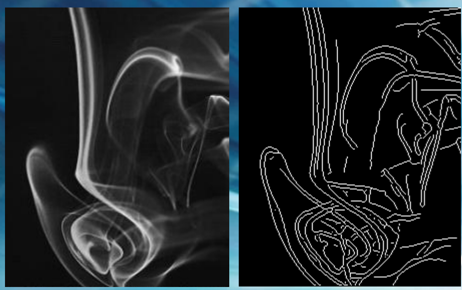
Introduction to Digital Image Processing with MATLAB® Asia Edition
McAndrew‧Wang‧Tseng



Chapter 9: Image Segmentation

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.1 Introduction

- **Segmentation** refers to the operation of partitioning an image into component parts or into separate objects
- In this chapter, we will investigate two very important topics: **thresholding** and **edge detection**

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.2 Thresholding

- **9.2.1 Single thresholding**

$$\text{A pixel becomes } \begin{cases} \text{white if its gray level is} > T, \\ \text{black if its gray level is} \leq T. \end{cases}$$

```
>> r=imread('rice.tif');
>> imshow(r),figure,imshow(r>110)
```

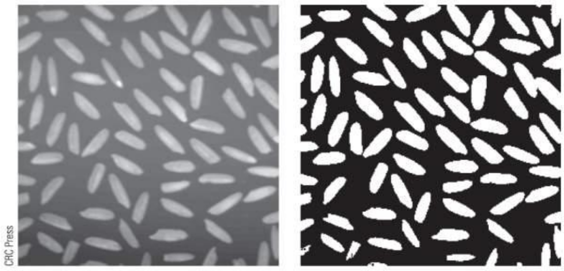


FIGURE 9.1 *Thresholded image of rice grains.*

3 Ch9-p.221

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.2

```
>> b=imread('bacteria.tif');
>> imshow(b),figure,imshow(b>100)
```

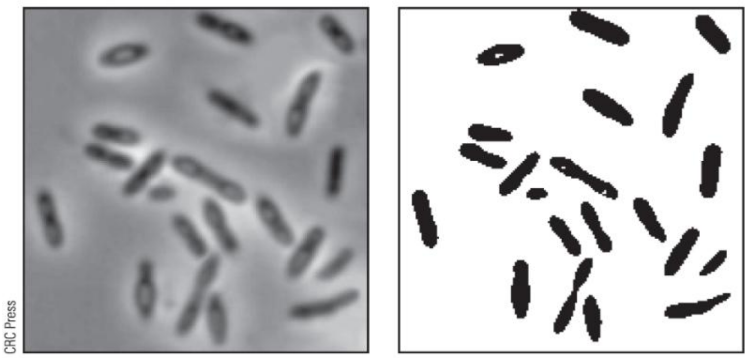


FIGURE 9.2 *Thresholded image of bacteria.*

4 Ch9-p.223

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.2 Thresholding

- MATLAB has the `im2bw` function, which thresholds an image of **any data type**, using the general syntax

```
im2bw(image,level)
```

e.g.

```
>> im2bw(r,0.43);
>> im2bw(b,0.39);
```

5 **Ch9-p.222** © 2010 Cengage Learning Engineering. All Rights Reserved. CENGAGE Learning

FIGURE 9.3

- Showing hidden aspects of an image

```
>> p=imread('paper1.tif');
>> imshow(p),figure,imshow(p>241)
```

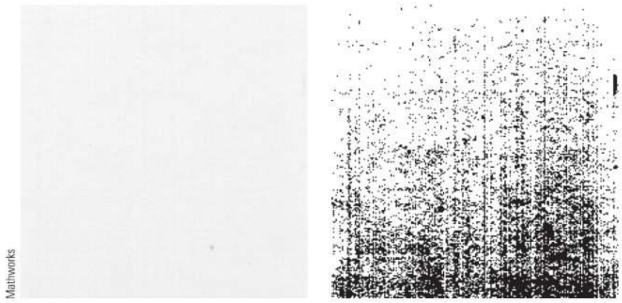


FIGURE 9.3 *The paper image and result after thresholding.*

6 **Ch9-p.223** © 2010 Cengage Learning Engineering. All Rights Reserved. CENGAGE Learning

9.2.2 Double Thresholding

a pixel becomes $\begin{cases} \text{white if its gray level is between } T_1 \text{ and } T_2, \\ \text{black if its gray level is otherwise.} \end{cases}$

```
>> [x,map]=imread('spine.tif');
>> s=ind2gray(x,map);
>> imshow(s),figure,imshow(s>115 & s<125)
```

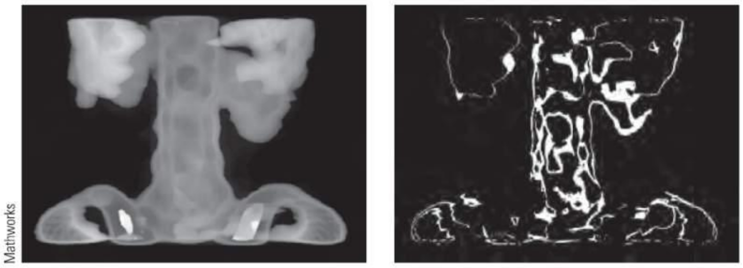


FIGURE 9.4 *The image* `spine.tif` *as the result after double thresholding.*

7 **Ch9-p.224**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

9.3 Applications of Thresholdng

- Remove a varying background from text or a drawing

```
>> r=rand(256)*128+127;
>> t=imread('text.tif');
>> tr=uint8(r.*double(not(t)));
>> imshow(tr)
```

```
>> imshow(tr>100)
```

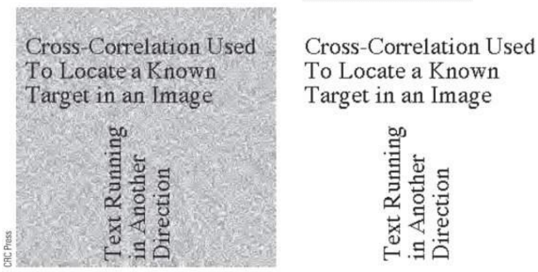


FIGURE 9.5 *Text on a varying background and thresholding.*

8 **Ch9-p.225**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

9.4 Choosing an Appropriate Thresholding Value

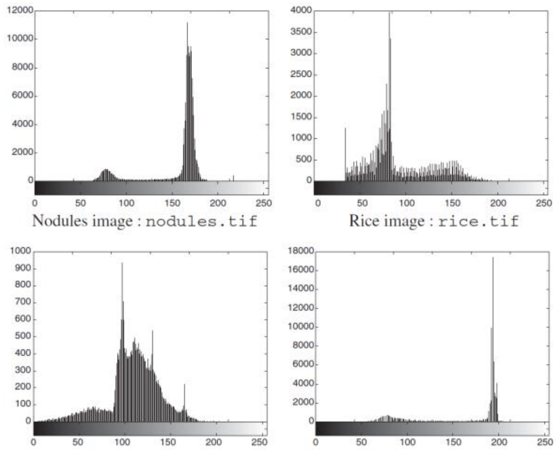
```
>> n=imread('nodules1.tif');
>> imshow(n);
>> n1=im2bw(n,0.35);
>> n2=im2bw(n,0.75);
>> figure,imshow(n1),figure,imshow(n2)
```

n: Original image n1: Threshold too low n2: Threshold too high

FIGURE 9.6 *Attempts at thresholding.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.7



Nodules image : nodules.tif Rice image : rice.tif

Bacteria image : bacteria.tif Coins image : eight.tif

FIGURE 9.7 *Histograms.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.8

- Describe the image histogram as a probability distribution

$$p_i = n_i/N$$

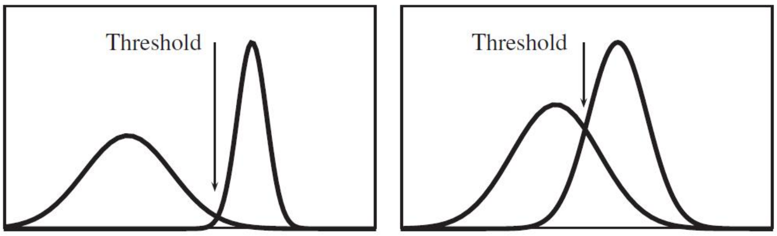


FIGURE 9.8 *Splitting up a histogram for thresholding.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.4 Choosing an Appropriate Thresholding Value

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

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

$$\omega(k) + \mu(k) = \sum_{i=0}^{L-1} p_i = 1$$

- We would like to find k to maximize the difference between $\omega(k)$ and $\mu(k)$

Define the image average as $\mu_T = \sum_{i=0}^{L-1} i p_i,$

then find a k, which maximizes $\dfrac{(\mu_T \omega(k) - \mu(k))^2}{\omega(k)\mu(k)}$

(Otsu's method, MATLAB function `graythresh`)

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.4 Choosing an Appropriate Thresholding Value

```
>> tn=graythresh(n)

tn =

    0.5804

>> r=imread('rice.tif');
>> tr=graythresh(r)

tr =

    0.4902

>> b=imread('bacteria.tif');
>> tb=graythresh(b)

tb =

    0.3765

>> e=imread('eight.tif');
>> te=graythresh(e)

te =

    0.6490
```

```
>> imshow(im2bw(n,tn))
>> figure,imshow(im2bw(r,tr))
>> figure,imshow(im2bw(b,tb))
>> figure,imshow(im2bw(e,te))
```


© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.9

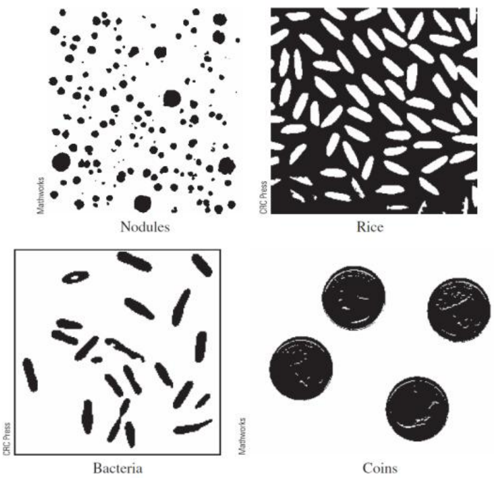
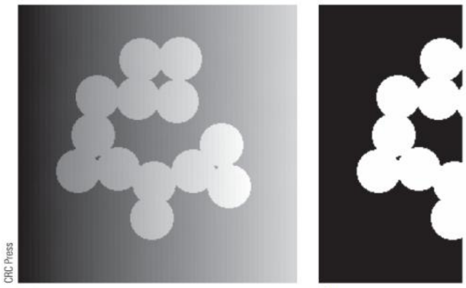


FIGURE 9.9 *Thresholding with values from* graythresh.

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.5 Adaptive Thresholding

```
>> c=imread('circles.tif');
>> x=ones(256,1)*[1:256];
>> c2=double(c).*(x/2+50)+(1-double(c)).*x/2;
>> c3=uint8(255*mat2gray(c2));
```



(a) (b)

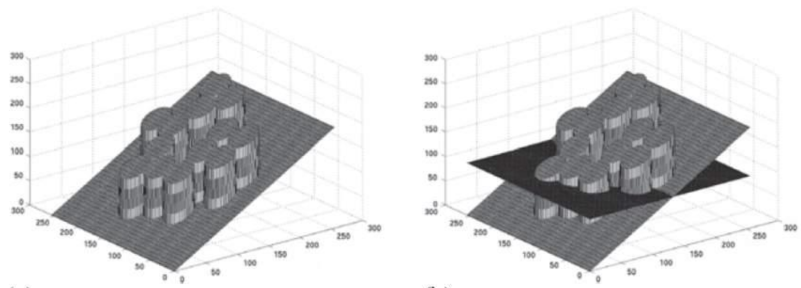
```
>> t=graythresh(c3)

t =

    0.4196

>> ct=im2bw(c3,t);
```

FIGURE 9.10 *An attempt at thresholding. (a) Circles image:* c3. *(b) Thresholding attempt:* ct.

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.11



(a) (b)

FIGURE 9.11 *An attempt at thresholding—functional version. (a) The image as a function. (b) Thresholding attempt.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

9.5 Adaptive Thresholding

```
>> p1=c3(:,1:64);
>> p2=c3(:,65:128);
>> p3=c3(:,129:192);
>> p4=c3(:,193:256);
```

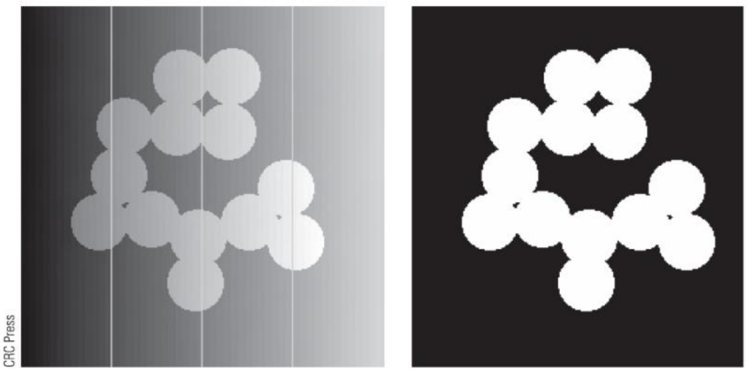
```
>> g1=im2bw(p1,graythresh(p1));
>> g2=im2bw(p2,graythresh(p2));
>> g3=im2bw(p3,graythresh(p3));
>> g4=im2bw(p4,graythresh(p4));
```

```
>> imshow([g1 g2 g3 g4])
```

17 Ch9-p.231

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.12



(a) (b)

FIGURE 9.12 *Adaptive thresholding. (a) Cutting up the image. (b) Thresholding each part separately.*

18 Ch9-p.232

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.6 Edge Detection

- The general MATLAB command for finding edges is

```
edge(image,'method',parameters. . .)
```

51	52	53	59
54	52	53	62
50	52	53	68
55	52	53	55

(a)

50	53	155	160
51	53	160	170
52	53	167	190
51	53	162	155

(b)

FIGURE 9.13 *Blocks of pixels.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.7 Derivatives and Edges

- **9.7.1 Fundamental Definitions**

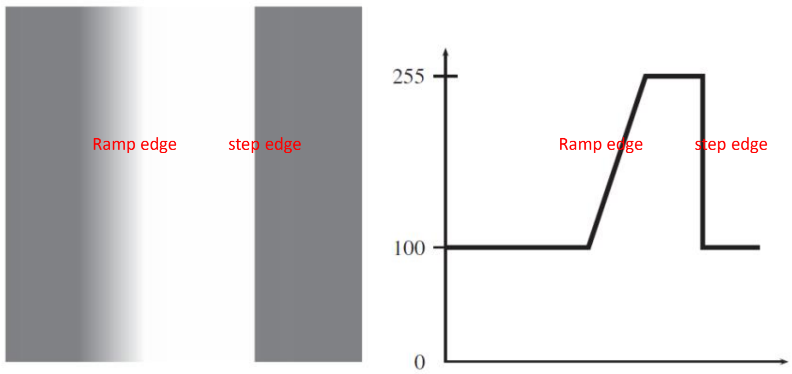


FIGURE 9.14 *Edges and their profiles.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.15

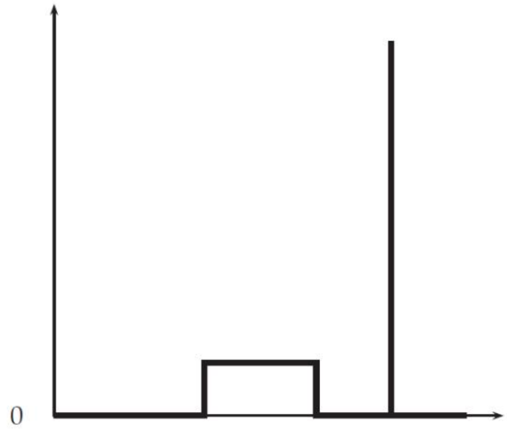


FIGURE 9.15 *The derivative of the edge profile.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.7.2 Some Edge Detection Filters

- **Prewitt filters**

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

```
>> ic=imread('ic.tif');
```

```
>> px=[-1 0 1;-1 0 1;-1 0 1];
>> icx=filter2(px,ic);
>> figure,imshow(icx/255)
```


© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.16 & 17

```
>> py=px';
>> icy=filter2(py,ic);
>> figure,imshow(icy/255)
```

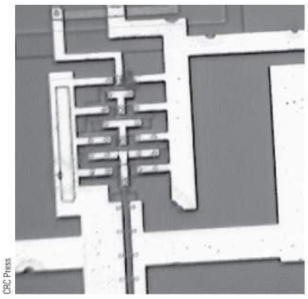
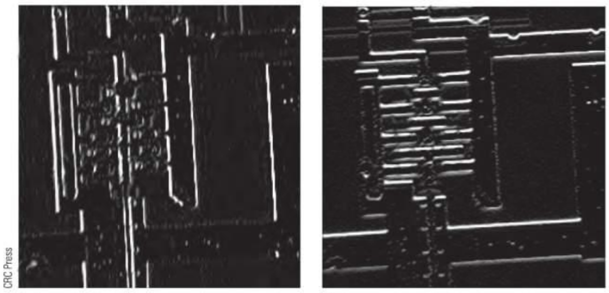


FIGURE 9.16 *An integrated circuit.*



(a) (b)

FIGURE 9.17 *The circuit after filtering with the Prewitt filters.*

23 **Ch9-p.237**

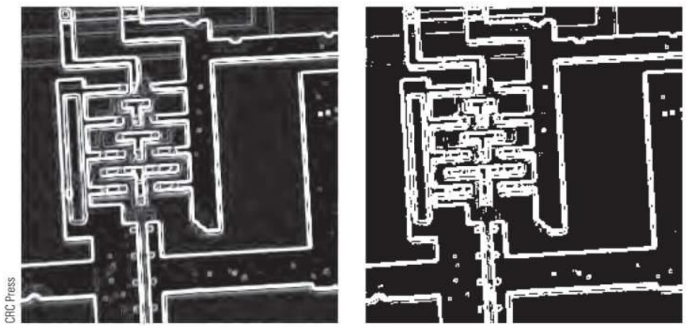
© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.18

```
>> edge_p=sqrt(icx.^2+icy.^2);
>> figure,imshow(edge_p/255)
```

```
>> edge_t=im2bw(edge_p/255,0.3);
```



(a) (b)

FIGURE 9.18 *All the edges of the circuit.*

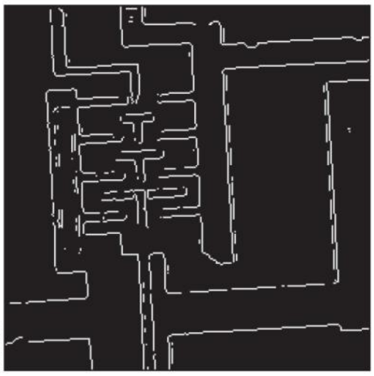
24 **Ch9-p.238**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.19

```
>> edge_p=edge(ic,'prewitt');
```



CRC Press

FIGURE 9.19 *The* `prewitt` *option of* `edge`.

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

9.7.2 Some Edge Detection Filters

- **Roberts cross-gradient filters**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- **Sobel filters**

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.20

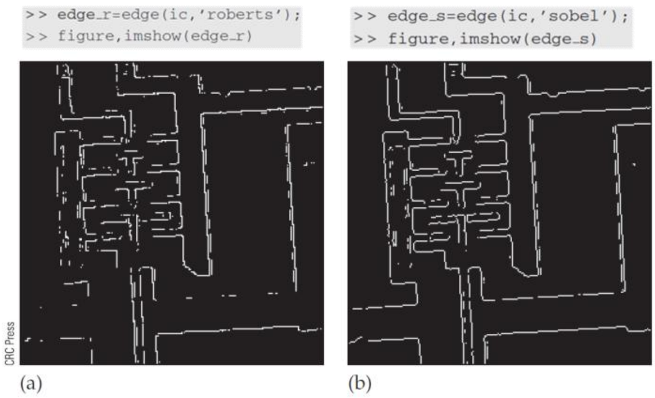


FIGURE 9.20 *Results of the Roberts and Sobel filters. (a) Roberts edge detection. (b) Sobel edge detection.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.8 Second Derivatives

- **9.8.1 The Laplacian**

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

✓ Isotropic filter

✓ Very sensitive to noise

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.21

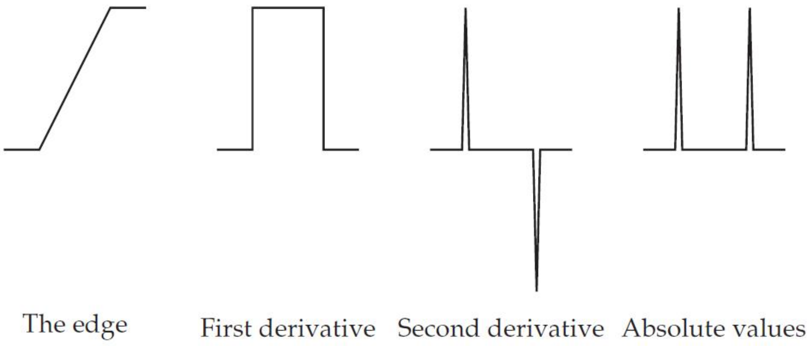


FIGURE 9.21 *Second derivatives of an edge function.*

29 **Ch9-p.241**

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.22

```
>> l=fspecial('laplacian',0);
>> ic_l=filter2(l,ic);
>> figure,imshow(mat2gray(ic_l))
```

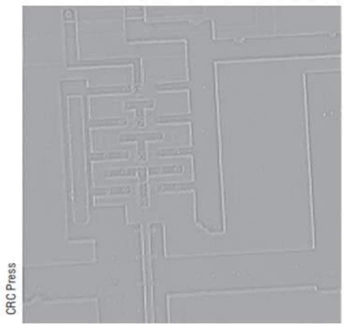


FIGURE 9.22 *Result after filtering with a discrete Laplacian.*

30 **Ch9-p.242**

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.8.2 Zero Crossing

- The position where the result of the filter **changes sign**
- e.g., consider the simple image given in Figure 9.23(a) and the result after filtering with a Laplacian mask in Figure 9.23(b)

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.23

50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	50	50	200	200	200	200	50	50
50	50	50	50	200	200	200	200	50	50
50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50

(a)

−100	−50	−50	−50	−50	−50	−50	−50	−50	−100
−50	0	150	150	150	150	150	150	0	−50
−50	150	−300	−150	−150	−150	−150	−300	150	−50
−50	150	−150	0	0	0	0	−150	150	−50
−50	150	−150	0	0	0	0	−150	150	−50
−50	150	−300	−150	0	0	0	−150	150	−50
−50	0	150	300	−150	0	0	−150	150	−50
−50	0	0	150	−300	−150	−150	−300	150	−50
−50	0	0	0	150	150	150	150	0	−50
−100	−50	−50	−50	−50	−50	−50	−50	−50	−100

(b)

FIGURE 9.23 *Locating zero crossings in an image. (a) A simple image. (b) After laplace filtering.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.8.2 Zero Crossing

- We define the **zero crossings** in such a filtered image to be pixels that satisfy either of the following:
  - ✓ They have a negative gray value and are orthogonally adjacent to a pixel whose gray value is positive
  - ✓ They have a value of zero and are between negative- and positive-valued pixels

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.24

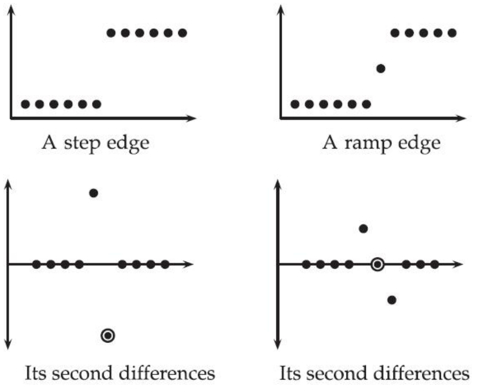


FIGURE 9.24 *Edges and second differences.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.8.2 Zero Crossing

- **Marr-Hildreth** method
  - ✓ Smooth the image with a Gaussian filter
  - ✓ Convolve the result with a Laplacian filter
  - ✓ Find the zero crossings

```
>> fspecial('log',13,2)
```

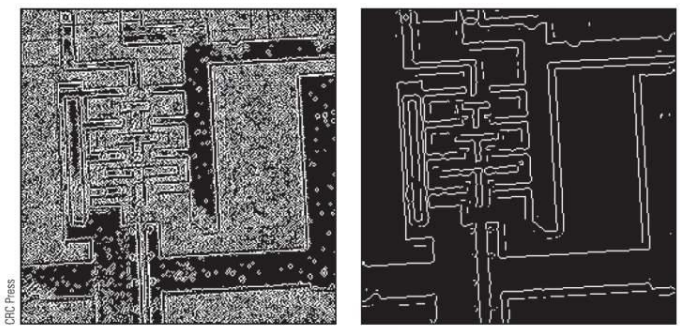
```
>> edge(ic,'zerocross');
```


© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.25

```
>> l=fspecial('laplace',0);
>> icz=edge(ic,'zerocross',1);
>> imshow(icz)
```

```
>> log=fspecial('log',13,2);
>> edge(ic,'zerocross',log);
```



(a) (b)

FIGURE 9.25 *Edge detection using zero crossings. (a) Zero crossings. (b) Using an LoG filter first.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.9 The Canny Edge Detector

- ✓ Take our image x
- ✓ Create a one-dimensional Gaussian filter g
- ✓ Create a one-dimensional filter dg corresponding to the expression given in

$$\left(-\frac{x}{\sigma^2}\right)e^{-\frac{x^2}{2\sigma^2}}$$

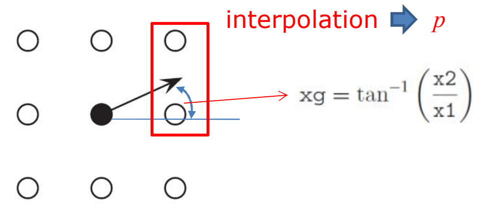
- ✓ 4. Convolve g with dg to obtain gdg
- ✓ 5. Apply gdg to x producing x1
- ✓ 6. Apply gdg' to x producing x2

We can now form an edge image with $\mathrm{xe} = \sqrt{\mathrm{x1}^2 + \mathrm{x2}^2}$

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.26

- ✓ **Non-maximum suppression**



The edge direction at a pixel

FIGURE 9.26 *Nonmaximum suppression in the Canny edge detector.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.27

Region	Degree	Pixel Location
1	0°	$y \le 0$ and $x > -y$
(1)	0°	$y \ge 0$ and $x < -y$
2	45°	$x > 0$ and $x \le -y$
(2)	45°	$x < 0$ and $x \ge -y$
3	90°	$x \le 0$ and $x > y$
(3)	90°	$x \ge 0$ and $x < y$
4	135°	$y < 0$ and $x \le y$
(4)	135°	$y > 0$ and $x \ge y$

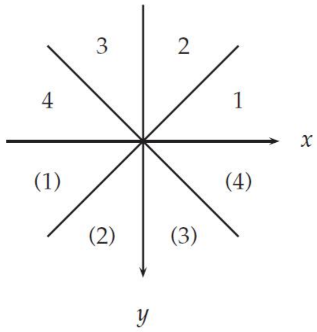


FIGURE 9.27 *Using pixel locations to quantize the gradient.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.28

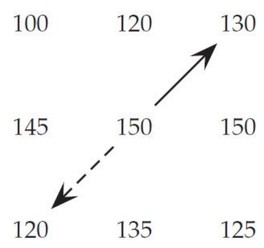


FIGURE 9.28 *Quantizing in non-maximum suppression.*

- ✓ **Hysteresis thresholding**

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.29

```
>> [icc,t]=edge(ic,'canny');
>> t

t =

    0.0500    0.1250

>> imshow(icc)
```

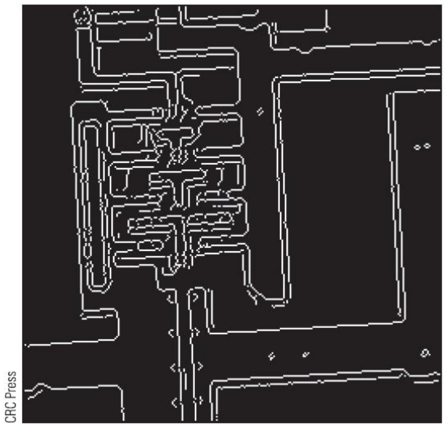
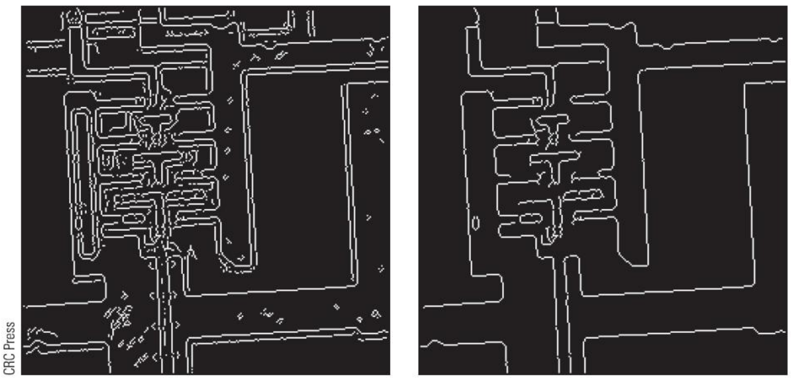


FIGURE 9.29 *Canny edge detection.*

41 **Ch9-p.248**

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.30



`edge(ic,'canny',[0,0.05])` `edge(ic,'canny',[0.01,0.5])`

FIGURE 9.30 *Canny edge detection with different thresholds.*

42 **Ch9-p.249**

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.10 The Hough Transform

$$y = ax + b$$

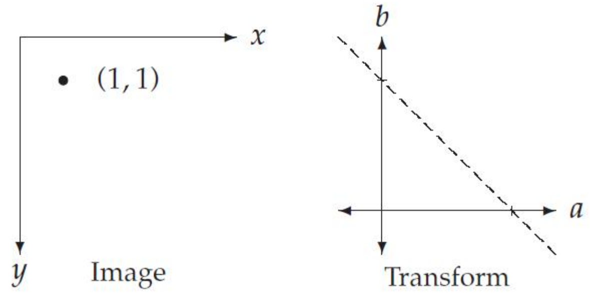


FIGURE 9.31 *A point in an image and its corresponding line in the transform.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.32

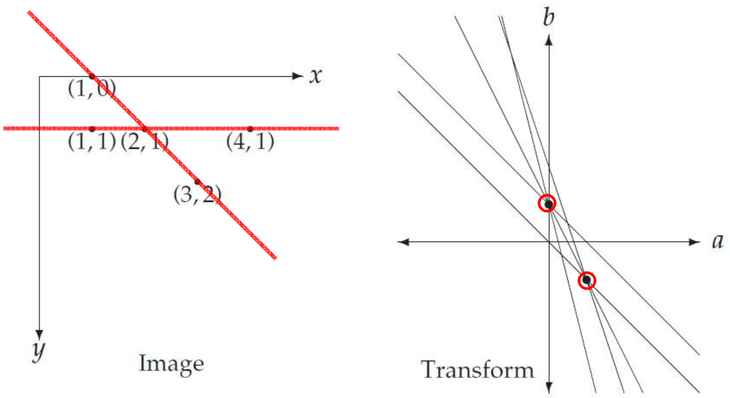


FIGURE 9.32 *An image and its corresponding lines in the transform.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.33

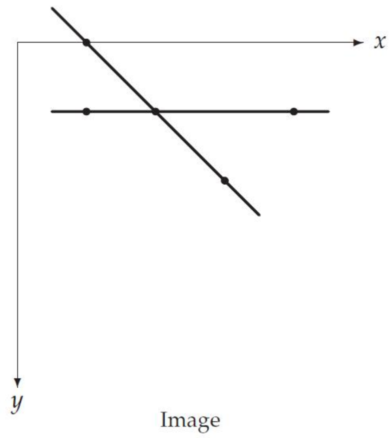


FIGURE 9.33 *Lines found by the Hough transform.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.10 The Hough Transform

- We cannot express a vertical line in the form $y = mx+c$, because m represents the gradient and a vertical line has infinite gradient
- Any line can be described in terms of the two parameters r and θ
  - ✓ r is the perpendicular distance from the line to the origin
  - ✓ θ is the angle of the line's perpendicular to the x axis

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.34

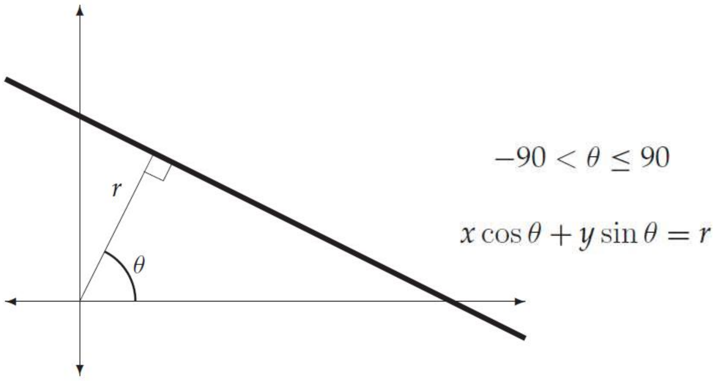


FIGURE 9.34 *A line and its parameters.*

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

FIGURE 9.35

- If we discretize θ to use only four values: -45°, 0°, 45°, 90°

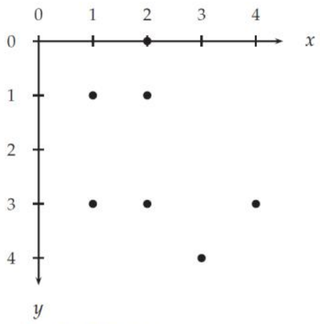


FIGURE 9.35 *A small image.*

(x, y)	$-45°$	$0°$	$45°$	$90°$
(2, 0)	1.4	2	1.4	0
(1, 1)	0	1	1.4	1
(2, 1)	0.7	2	2.1	1
(1, 3)	−1.4	1	2.8	3
(2, 3)	−0.7	2	3.5	3
(4, 3)	0.7	4	4.9	3
(3, 4)	−0.7	3	4.9	4

© 2010 Cengage Learning Engineering. All Rights Reserved.
CENGAGE Learning

9.10 The Hough Transform

- The accumulator array contains the number of times each value of (r, θ) appears in the above table

	-1.4	-0.7	0	0.7	1	1.4	2	2.1	2.8	3	3.5	4	4.9
-45°	1	2	1	2		1							
0°					2		3			1		1	
45°						2		1	1		1		2
90°			1		2					3		2	

© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.36

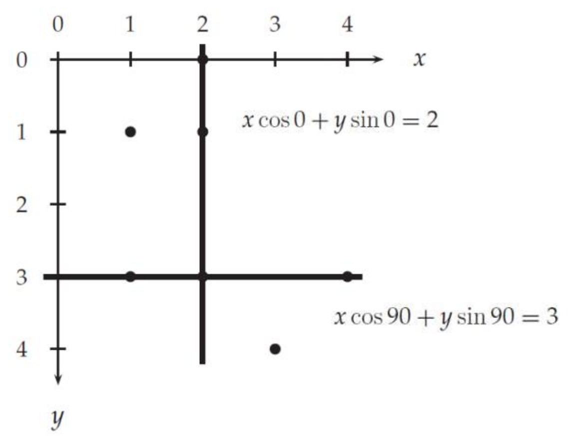


FIGURE 9.36 *Lines found by the Hough transform.*

© 2010 Cengage Learning Engineering. All Rights Reserved.

9.11 Implementing the Hough Transform in MATLAB

- ✓ Decide on a discrete set of values of θ and r to use
- ✓ Calculate for each foreground pixel (x, y) in the image the values of $r = x\cos\theta + y\sin\theta$ for all of our chosen values of θ
- ✓ Create an accumulator array whose sizes are the numbers of angles θ and values r in our chosen discretizations from Step 1, and
- ✓ Step through all of our r values, updating the accumulator array as we go

51 **Ch9-p.255**
© 2010 Cengage Learning Engineering. All Rights Reserved.

FIGURE 9.37

- **9.11.1 Discretizing r and θ**

```
>> angles = [-90:180]*pi/180;
```

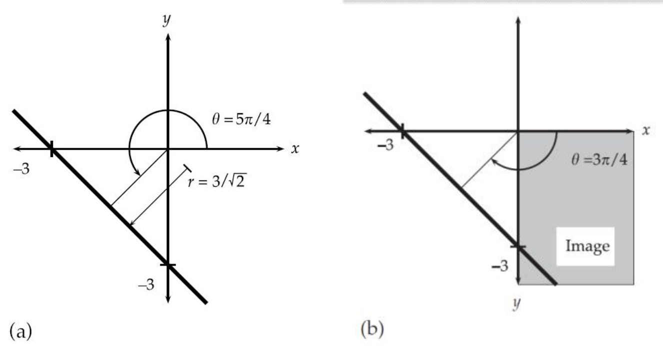


FIGURE 9.37 *A line parameterized with r and θ. (a) Using ordinary Cartesian axes. (b) Using matrix axes.*

52 **Ch9-p.256**
© 2010 Cengage Learning Engineering. All Rights Reserved.

9.11 Implementing the Hough Transform in MATLAB

9.11.2 Calculating the r Values

✓ If im is a **binary image**

We can create a binary edge image by use of the `edge` function

```
>> [x,y]=find(im);
```

```
>> r=floor(x*cos(angles)+y*sin(angles));
```

9.11.3 Forming the Accumulator Array

```
>> rmax=max(r(find(r>0)));
>> acc=zeros(rmax+1,270);
```

53 **Ch9-p.257**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

9.11 Implementing the Hough Transform in MATLAB

- **9.11.4 Updating the Accumulator Array**

```
function res=hough2(image)

%
% HOUGH2(IMAGE) creates the Hough transform corresponding to the image IMAGE
%

if ~isbw(image)
  edges=edge(image,'canny');
else
  edges=image;
end;
[x,y]=find(edges);
angles=[-90:180]*pi/180;
r=floor(x*cos(angles)+y*sin(angles));
rmax=max(r(find(r>0)));
acc=zeros(rmax+1,270);
for i=1:length(x),
  for j=1:270,
    if r(i,j)>=0
      acc(r(i,j)+1,j)=acc(r(i,j)+1,j)+1;
    end;
  end;
end;
res=acc;
```

FIGURE 9.38 *A simple MATLAB function for implementing the Hough transform.*

54 **Ch9-p.257**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.39

```
>> c=imread('cameraman.tif');
>> hc=hough2(c);
```

```
>> imshow(mat2gray(hc)*1.5)
```

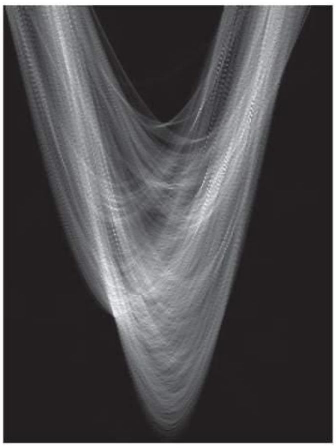


FIGURE 9.39 *The result of a Hough Transform.*

55 **Ch9-p.258**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.40

```
>> max(hc(:))

ans =

    91
```

```
>> [r,theta]=find(hc==91)

r =

   138


theta =

   181
```

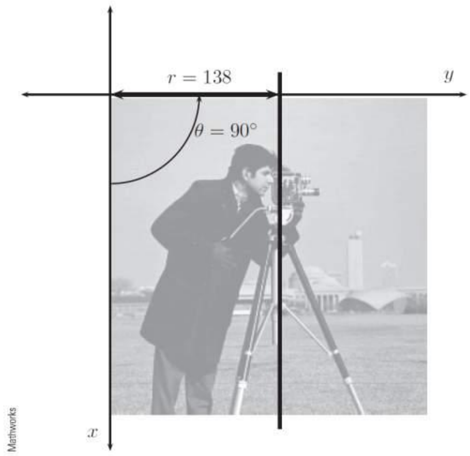


FIGURE 9.40 *A line from the Hough Transform.*

56 **Ch9-p.259**

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.41

```
function houghline(image,r,theta)
%
% Draws a line at perpendicular distance R from the upper left corner of the
% current figure, with perpendicular angle THETA to the left vertical axis.
% THETA is assumed to be in degrees.
%
[x,y]=size(image);
angle=pi*(181-theta)/180;
X=[1:x];
if sin(angle)==0
  line([r r],[0,y],'Color','black')
else
  line([0,y],[r/sin(angle),(r-y*cos(angle))/sin(angle)],'Color','black')
end;
```

FIGURE 9.41 *A simple MATLAB function for drawing lines on an image.*

57 Ch9-p.261

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning

FIGURE 9.42

(a) (b)

FIGURE 9.42 *The* `houghline` *function in action.*

58 Ch9-p.261

© 2010 Cengage Learning Engineering. All Rights Reserved.

CENGAGE Learning