10.2.5 Basic edge detection

The image gradient and its properties

Definition

$$\nabla \mathbf{f} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Magnitude

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = [g_x^2 + g_y^2]^{1/2} = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

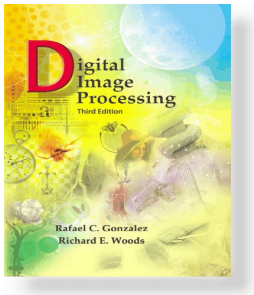
Direction

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

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

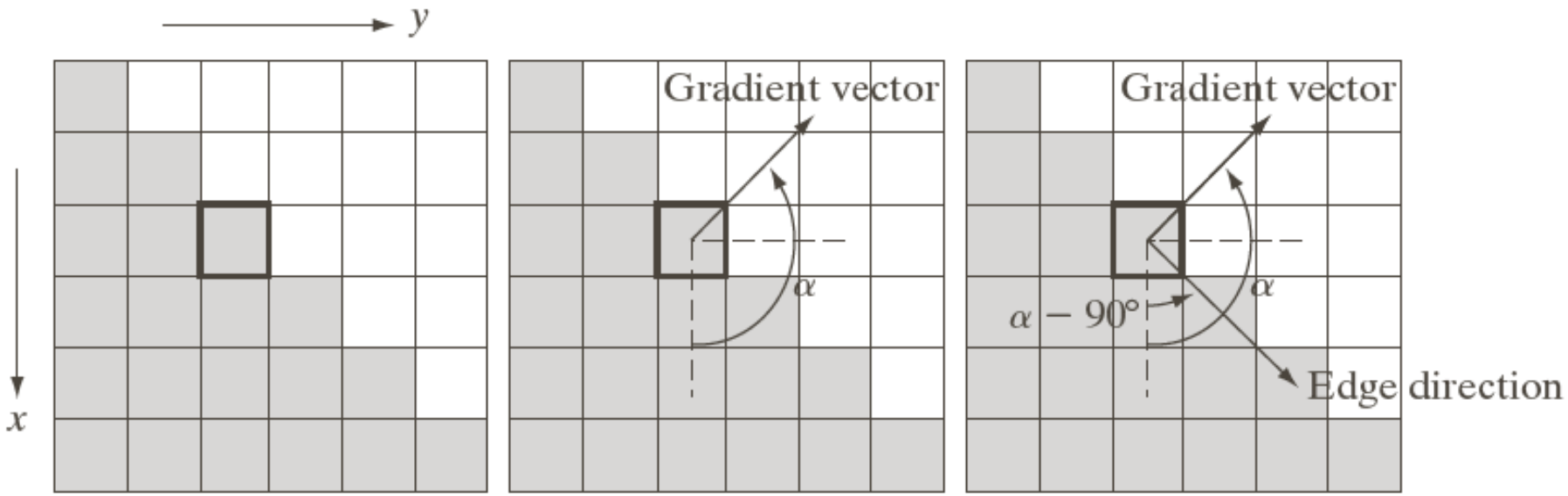
- The magnitude of gradient provides information about the strength of the edge.

Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa



Chapter 10

Segmentation



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

Gradient operators

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

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

−1
1

−1	1

a b

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

Roberts:

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

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

Prewitt:

$$g_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

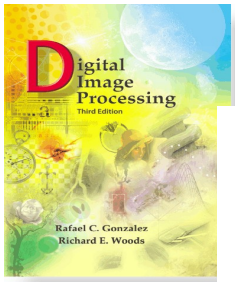
$$g_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Sobel:

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa



Digital Image Processing, 3rd ed.

Gonzalez & Woods

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

a
b c
d e
f g

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

−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

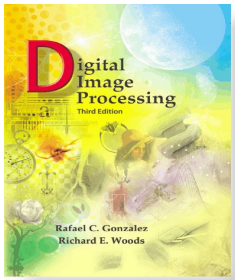
−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	**1**	−1	0	1

Sobel

Convenient approximation:

$$M(x, y) \approx |g_x| + |g_y|$$

Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa



Chapter 10

Segmentation

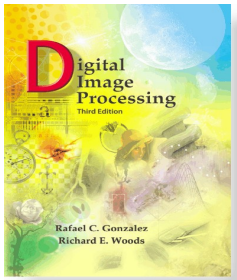
The Prewitt masks are simpler to implement than the Sobel masks, but, the slight computational difference between them typically is not an issue. The fact that the Sobel masks have better noise-suppression (smoothing) characteristics makes them preferable because, as mentioned in the previous section, noise suppression is an important issue when dealing with derivatives. Note that the coefficients of all the masks in Fig. 10.14 sum to zero, thus giving a response of zero in areas of constant intensity, as expected of a derivative operator.

The masks just discussed are used to obtain the gradient components g_x and g_y at every pixel location in an image. These two partial derivatives are then used to estimate edge strength and direction. Computing the magnitude of the gradient requires that g_x and g_y be combined in the manner shown in Eq. (10.2-10). However, this implementation is not always desirable because of the computational burden required by squares and square roots. An approach used frequently is to approximate the magnitude of the gradient by absolute values:

$$M(x, y) \approx |g_x| + |g_y| \qquad (10.2\text{-}20)$$

This equation is more attractive computationally, and it still preserves relative changes in intensity levels. The price paid for this advantage is that the resulting filters will not be isotropic (invariant to rotation) in general. However, this is not an issue when masks such as the Prewitt and Sobel masks are used to compute g_x and g_y, because these masks give isotropic results only for vertical and horizontal edges. Results would be isotropic only for edges in those two directions, regardless of which of the two equations is used. In addition, Eqs. (10.2-10) and (10.2-20) give identical results for vertical and horizontal edges when the Sobel or Prewitt masks are used (Problem 10.8).

It is possible to modify the 3×3 masks in Fig. 10.14 so that they have their strongest responses along the diagonal directions. Figure 10.15 shows the two additional Prewitt and Sobel masks needed for detecting edges in the diagonal directions.



Chapter 10

Segmentation

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

Prewitt

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

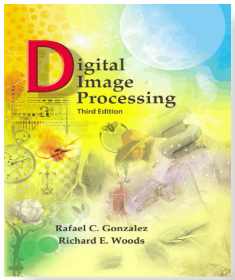
Sobel

a b
c d

FIGURE 10.15 Prewitt and Sobel masks for detecting diagonal edges.

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, $|g_x| + |g_y|$.



Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10
Segmentation

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

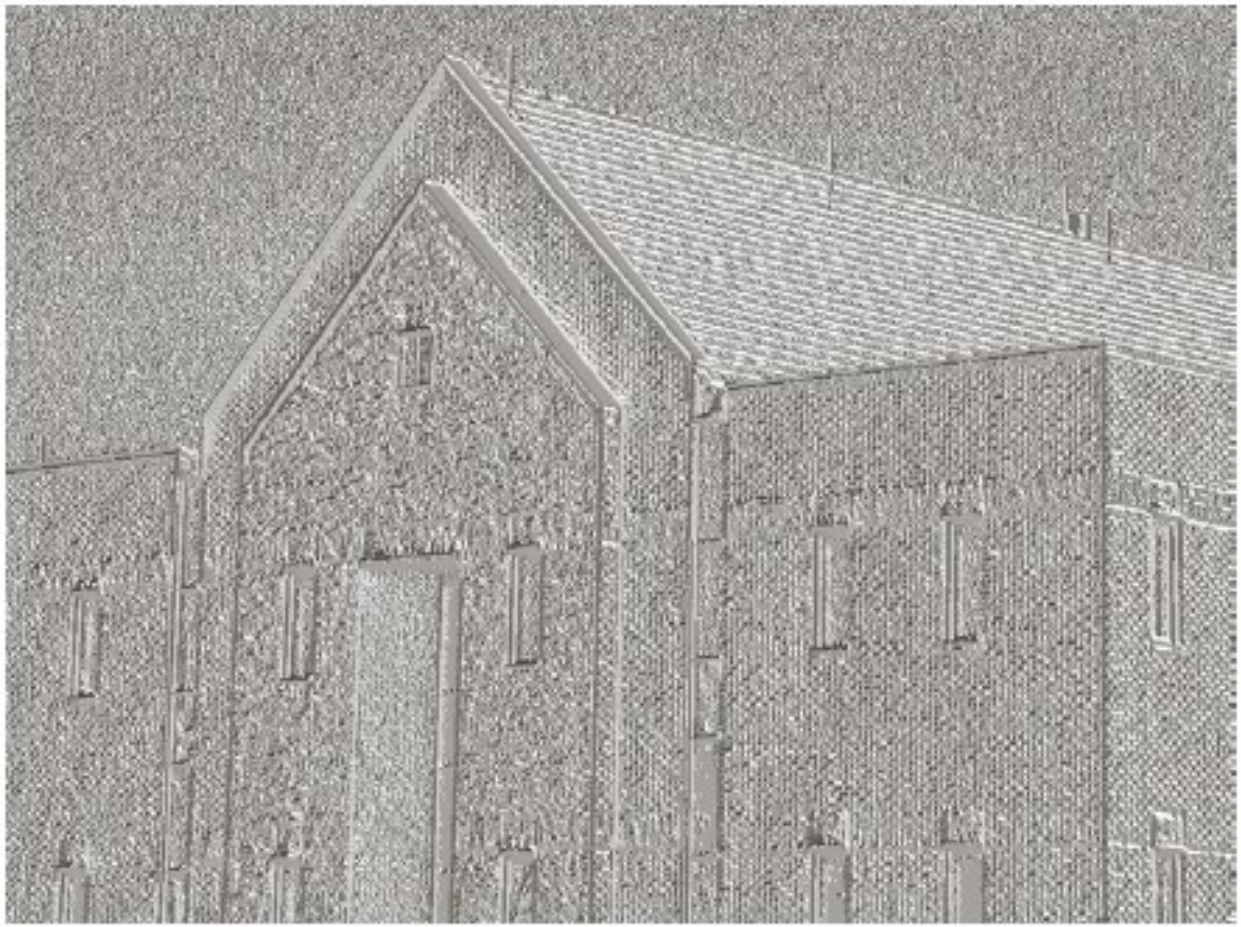
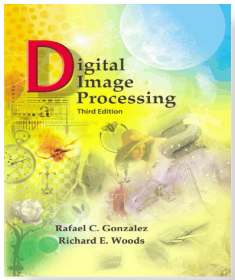


FIGURE 10.17 Gradient angle image computed using Eq. (10.2-11). Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.

Angle info plays key supporting role in Canny edge detection

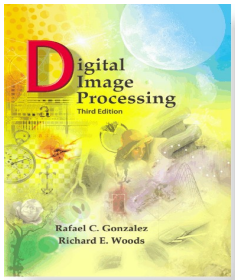


Chapter 10

Segmentation

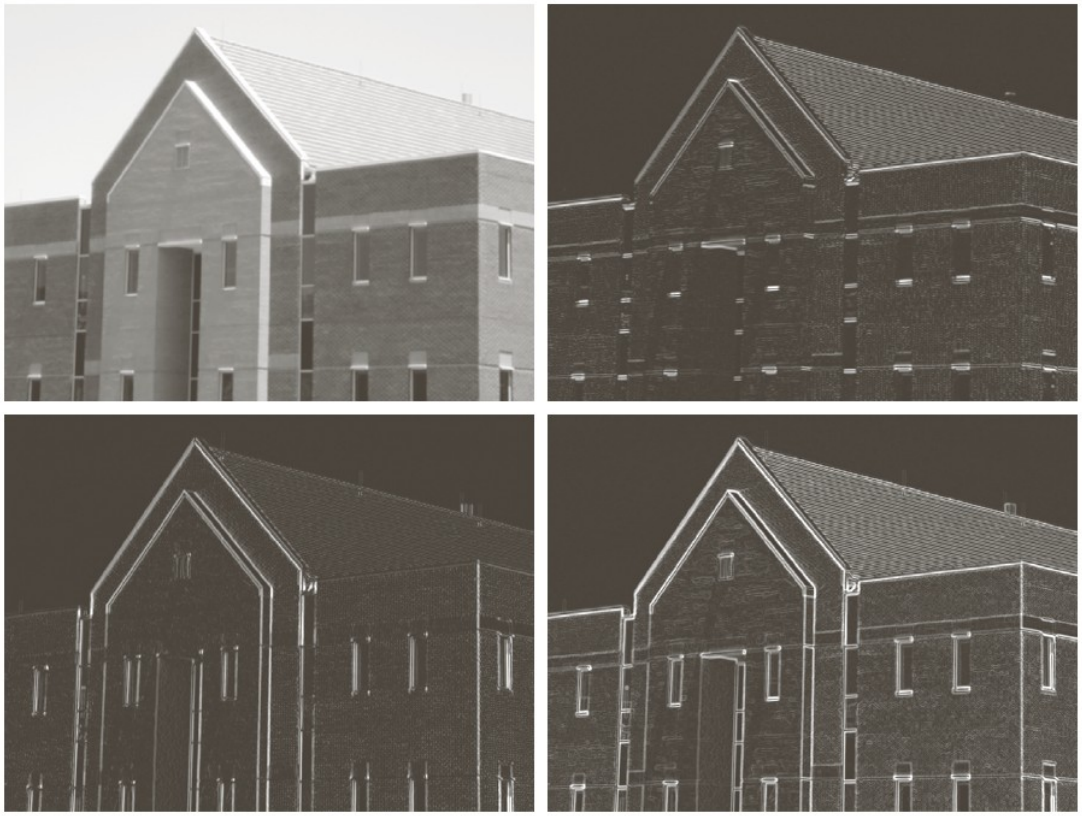
Figure 10.17 shows the gradient angle image computed using Eq. (10.2-11). **In general,** angle images are not as useful as gradient magnitude images for **edge detection,** but they do complement the information extracted from an **image using** the magnitude of the gradient. For instance, the constant intensity **areas in Fig.** 10.16(a), such as the front edge of the sloping roof and top hori**zontal bands** of the front wall, are constant in Fig. 10.17, indicating that the **gradient** vector direction at all the pixel locations in those regions is the same.

Angle info plays key supporting role in Canny edge detection



Result with prior smoothing

Fine detail info (e.g. bricks contribution in original image) often is undesirable in edge detection because it tends to act as noise, which is enhanced by derivative computations and thus complicates detection of principal edges in an image. One way to reduce the fine detail is to smooth the image



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.

Now almost no contribution due to the bricks)

a b

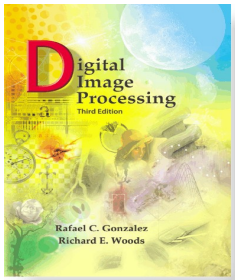
FIGURE 10.19 Diagonal edge detection. (a) Result of using the mask in Fig. 10.15(c). (b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

Diagonal edge detection

Combining the gradient with thresholding

The results in Fig. 10.18 show that edge detection can be made more selective by smoothing the image prior to computing the gradient. Another approach aimed at achieving the same basic objective is to threshold the gradient image. For example, Fig. 10.20(a) shows the gradient image from Fig. 10.16(d) thresholded, in the sense that pixels with values greater than or equal to 33% of the maximum value of the gradient image are shown in white, while pixels below the threshold value are shown in black. Comparing this image with Fig. 10.18(d), we see that there are fewer edges in the thresholded image, and that the edges in this image are much sharper (see, for example, the edges in the roof tile). On the other hand, numerous edges, such as the 45° line defining the far edge of the roof, are broken in the thresholded image.

When interest lies both in highlighting the principal edges and on maintaining as much connectivity as possible, it is common practice to use both smoothing and thresholding. Figure 10.20(b) shows the result of thresholding Fig. 10.18(d), which is the gradient of the smoothed image. This result shows a reduced number of broken edges; for instance, compare the 45° edges in Figs. 10.20(a) and (b). Of course, edges whose intensity values were severely attenuated due to blurring (e.g., the edges in the tile roof) are likely to be totally eliminated by thresholding. We return to the problem of broken edges in Section 10.2.7.



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.

- **Criteria for optimal edge detection**

(1) Good detection: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges).

(2) Good localization: the edges detected must be as close as possible to the true edges.

Single response constraint: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge (created by noise).

(Trucco, Chapt 4 AND Jain et al., Chapt 5)

10.2.6 More advanced techniques for edge detection

The Marr-Hildreth edge detector [1980]

- Image should be smoothed first (to reduce noise)
- Larger operators should be used for larger images
- Zero-crossings of second derivative should be exploited
- The Laplacian of a Gaussian (LoG) operator is therefore empolyed

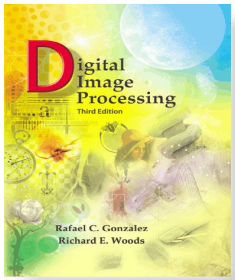
2-D Gaussian operator:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \ldots \quad (1)$$

Laplacian of Gaussian (LoG)

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

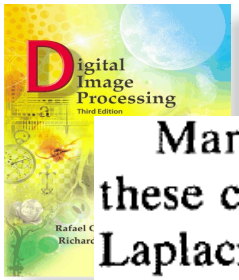
Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa



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 [1980]. Edge-detection methods in use at the time were based on using small operators (such as the Sobel masks), as discussed in the previous section. Marr and Hildreth argued (1) that intensity changes are not independent of image scale and so their detection requires the use of operators of different sizes; and (2) that 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 (as we saw in Fig. 10.10).

These ideas suggest that an operator used for edge detection should have two salient features. First and foremost, it should be a differential operator capable of computing a digital approximation of the first or second derivative at every point in the image. Second, it should be capable of being "tuned" to act at any desired scale, so that large operators can be used to detect blurry edges and small operators to detect sharply focused fine detail.



Marr and Hildreth argued that the most satisfactory operator fulfilling these conditions is the filter $\nabla^2 G$ where, as defined in Section 3.6.2, ∇^2 is the Laplacian operator, $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$, and G is the 2-D Gaussian function

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2-21}$$

with standard deviation σ (sometimes σ is called the *space constant*). To find an expression for $\nabla^2 G$ we perform the following differentiations:

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

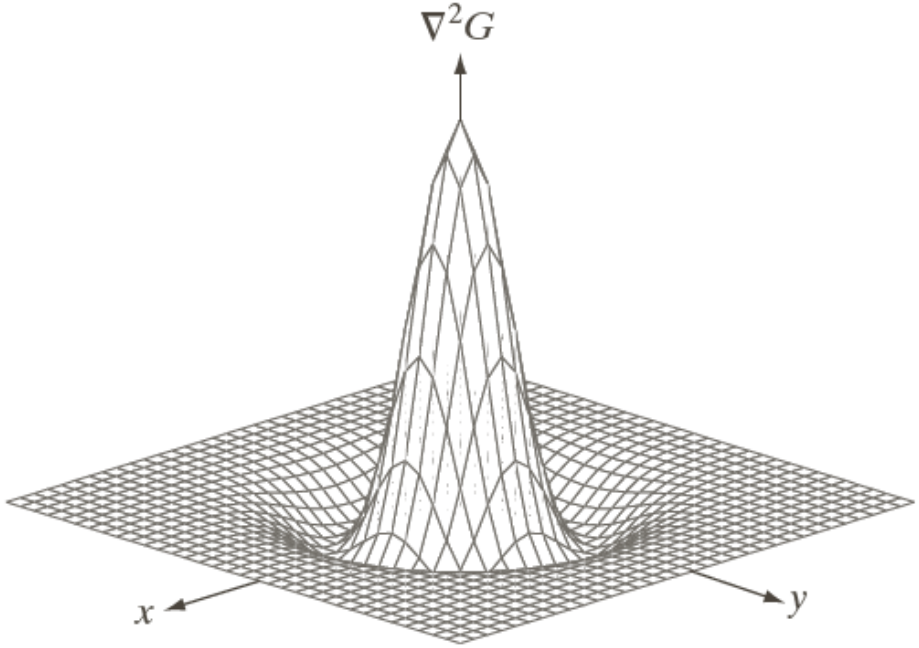
$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \tag{10.2-22}$$

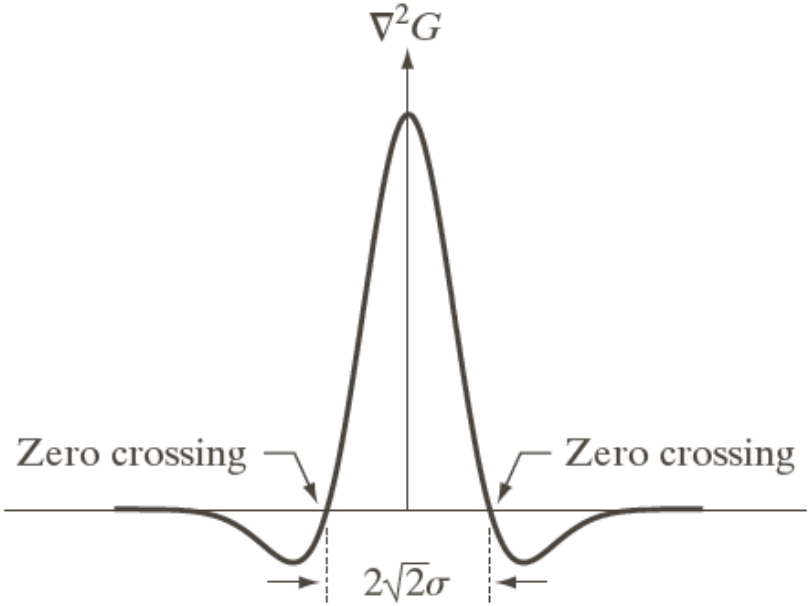
$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Collecting terms gives the final expression:

$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2-23}$$

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





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

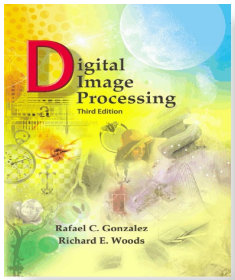
5 × 5 Laplacian of Gaussian mask

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

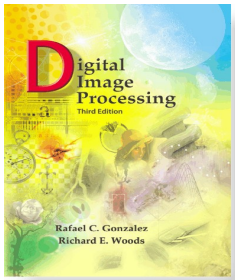
17 × 17 Laplacian of Gaussian mask

0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0

(Trucco, Chapt 4 AND Jain et al., Chapt 5)



Figures 10.21(a) through (c) show a 3-D plot, image, and cross section of the *negative* of the LoG function (note that the zero crossings of the LoG occur at $x^2 + y^2 = 2\sigma^2$, which defines a circle of radius $\sqrt{2}\sigma$ centered on the origin). Because of the shape illustrated in Fig. 10.21(a), the LoG function sometimes is called the *Mexican hat* operator. Figure 10.21(d) shows a 5×5 mask that approximates the shape in Fig. 10.21(a) (in practice we would use the *negative* of this mask). This approximation is not unique. Its purpose is to capture the essential *shape* of the LoG function; in terms of Fig. 10.21(a), this means a positive, central term surrounded by an adjacent, negative region whose values increase as a function of distance from the origin, and a zero outer region. The coefficients must sum to zero so that the response of the mask is zero in areas of constant intensity.



Chapter 10

Segmentation

There are two fundamental ideas behind the selection of the operator $\nabla^2 G$. First, the Gaussian part of the operator blurs the image, thus reducing the intensity of structures (including noise) at scales much smaller than σ. Unlike averaging of the form discussed in Section 3.5 and used in Fig. 10.18, the Gaussian function is smooth in both the spatial and frequency domains (see Section 4.8.3), and is thus less likely to introduce artifacts (e.g., ringing) not present in the original image. The other idea concerns ∇^2, the second derivative part of the filter. Although first derivatives can be used for detecting abrupt changes in intensity, they are directional operators. The Laplacian, on the other hand, has the important advantage of being isotropic (invariant to rotation), which not only corresponds to characteristics of the human visual system (Marr [1982]) but also responds equally to changes in intensity in any mask direction, thus avoiding having to use multiple masks to calculate the strongest response at any point in the image.

• The Laplacian

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

- Approximating $\nabla^2 f$:

$$\nabla^2 f = -4f(i,j) + f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j)$$

• Properties of the Laplacian

- It is an isotropic operator.

- It is cheaper to implement (one mask only).

- It does not provide information about edge direction.

- It is more sensitive to noise (differentiates twice).

Example:

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\nabla^2 f = -4z_5 + (z_2 + z_4 + z_6 + z_8)$$

- The Laplacian can be implemented using the mask shown below:

0	1	0
1	-4	1
0	1	0

Example:

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

(Trucco, Chapt 4 AND Jain et al., Chapt 5)

The LoG filter is first convolved with the input image $f(x, y)$,

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

after which the zero-crossings of $g(x, y)$ is found

Summary of algorithm:

(1) Filter input image with $n \times n$ Gaussian lowpass filter (sample eqn (1))

(2) Compute the Laplacian of the result in step (1)

(3) Find the zero-crossings in the result in step (2)

- **Rule of thumb: $n \equiv$ smallest odd integer greater than or equal to 6σ**
- **A zero-crossing at p implies that the sign of at least two of its oposing neibouring pixels must differ and that the absolute value of this difference must exceed a certain threshold**

Gaussian kernel coefficients are sampled from the 2D Gaussian function.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where σ is the standard deviation of the distribution.

The distribution is assumed to have a mean of zero.

We need to discretize the continuous Gaussian functions to store it as discrete pixels.

An integer valued 5 by 5 convolution kernel approximating a Gaussian with a σ of 1 is shown to the right,

$$\frac{1}{273}$$

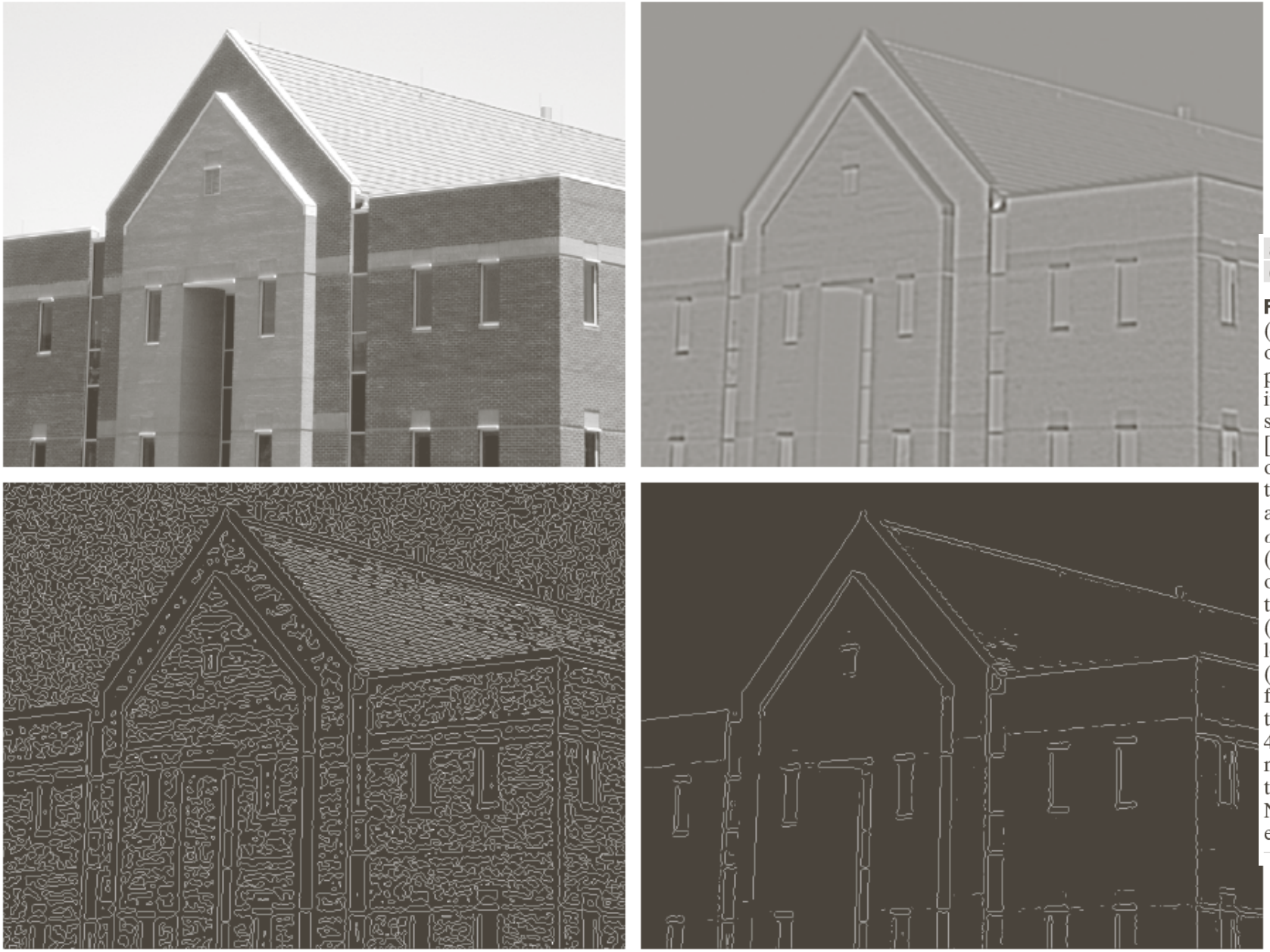
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

3x3 Laplacian mask (example)

1	1	1
1	−8	1
1	1	1

To specify the size of the Gaussian filter, recall that about 99.7% of the volume under a 2-D Gaussian surface lies between $\pm 3\sigma$ about the mean. Thus, as a rule of thumb, the size of an $n \times n$ LoG discrete filter should be such that n is the smallest odd integer greater than or equal to 6σ. Choosing a filter mask smaller than this will tend to "truncate" the LoG function, with the degree of truncation being inversely proportional to the size of the mask; using a larger mask would make little difference in the result.

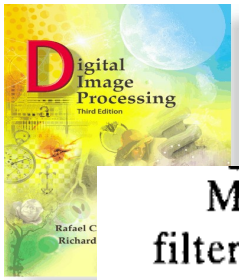
One approach for finding the zero crossings at any pixel, p, of the filtered image, $g(x, y)$, is based on using a 3×3 neighborhood centered at p. A zero crossing at p implies that the *signs* of at least two of its opposing neighboring pixels must differ. There are four cases to test: left/right, up/down, and the two diagonals. If the values of $g(x, y)$ are being compared against a threshold (a common approach), then not only must the signs of opposing neighbors be different, but the absolute value of their numerical difference must also exceed the threshold before we can call p a zero-crossing pixel. We illustrate this approach in Example 10.7 below.



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

- $\sigma = 4$ ($\approx 0.5\%$ of smallest image dimension)



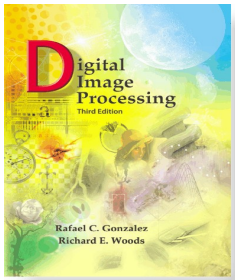
Marr and Hildreth [1980] noted that it is possible to approximate the LoG filter in Eq. (10.2-23) by a difference of Gaussians (DoG):

$$\mathrm{DoG}(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \qquad (10.2\text{-}26)$$

with $\sigma_1 > \sigma_2$. Experimental results suggest that certain "channels" in the human vision system are selective with respect to orientation and frequency, and can be modeled using Eq. (10.2-26) with a ratio of standard deviations of 1.75:1. Marr and Hildreth suggested that using the ratio 1.6:1 preserves the basic characteristics of these observations and also provides a closer "engineering" approximation to the LoG function. To make meaningful comparisons between the LoG and DoG, the value of σ for the LoG must be selected as in the following equation so that the LoG and DoG have the same zero crossings (Problem 10.17):

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \qquad (10.2\text{-}27)$$

Although the zero crossings of the LoG and DoG will be the same when this value of σ is used, their amplitude scales will be different. We can make them compatible by scaling both functions so that they have the same value at the origin.



Chapter 10

Segmentation

a b

FIGURE 10.23
(a) Negatives of the LoG (solid) and DoG (dotted) profiles using a standard deviation ratio of 1.75:1. (b) Profiles obtained using a ratio of 1.6:1.

The profiles in Figs. 10.23(a) and (b) were generated with standard deviation ratios of 1:1.75 and 1:1.6, respectively (by convention, the curves shown are inverted, as in Fig. 10.21). The LoG profiles are shown as solid lines while the DoG profiles are dotted. The curves shown are intensity profiles through the center of LoG and DoG arrays generated by sampling Eq. (10.2-23) (with the constant in $1/2\pi\sigma^2$ in front) and Eq. (10.2-26), respectively. The amplitude of all curves at the origin were normalized to 1. As Fig. 10.23(b) shows, the ratio 1:1.6 yielded a closer approximation between the LoG and DoG functions.

Both the LoG and the DoG filtering operations can be implemented with 1-D convolutions instead of using 2-D convolutions directly (Problem 10.19). For an image of size $M \times N$ and a filter of size $n \times n$, doing so reduces the number of multiplications and additions for each convolution from being proportional to n^2MN for 2-D convolutions to being proportional to nMN for 1-D convolutions. This implementation difference is significant. For example, if $n = 25$, a 1-D implementation will require on the order of 12 times fewer multiplication and addition operations than using 2-D convolution.

How To Compute Zero Crossings

Important step in the Laplacian edge detector. Do the following for each pixel $I(u, v)$:

1. Look at your four neighbors, left, right, up and down
2. If they all have the same sign as you, then you are not a zero crossing
3. Else, if you have the smallest absolute value compared to your neighbors with opposite sign, then you are a zero crossing

- How do we estimate the edge strength?
- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.
- To mark an edge:
  - compute slope of zero-crossing
  - apply a threshold to slope

- The Laplacian-of-Gaussian (LOG) – cont.

I

$I * (\Delta^2 G)$

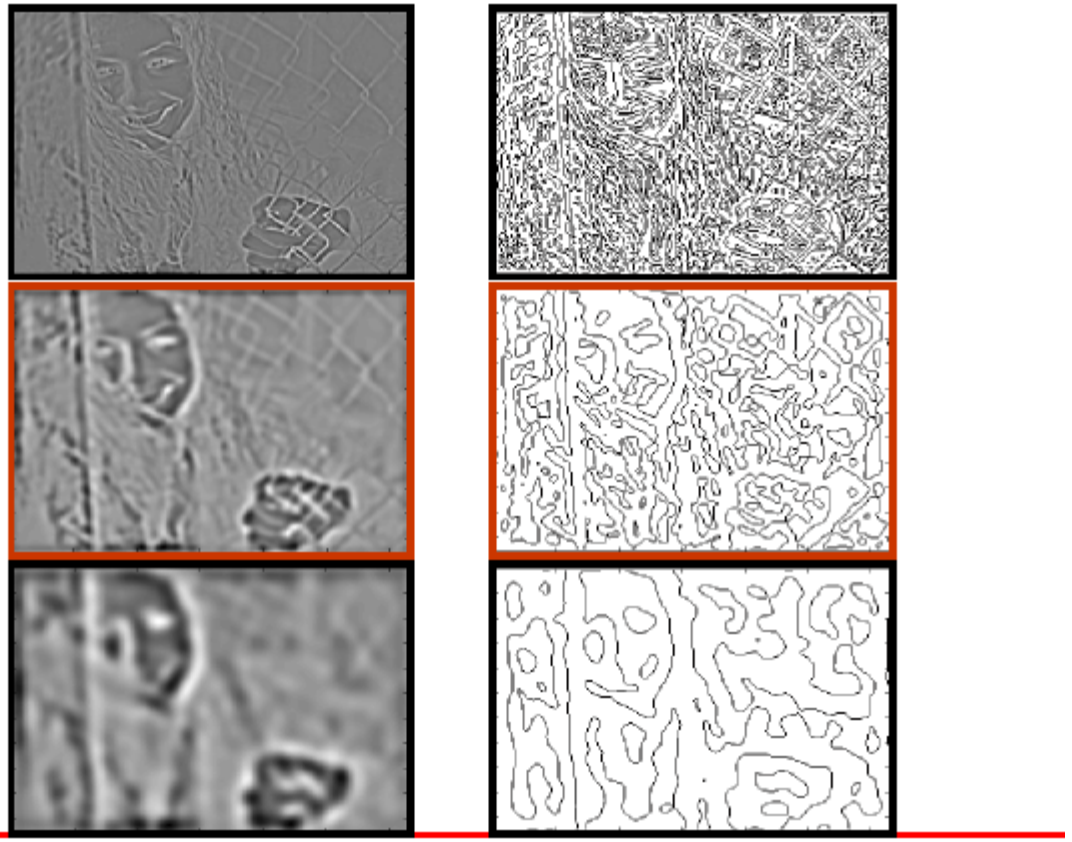
Zero crossings of $I * (\Delta^2 G)$

- The Laplacian-of-Gaussian (LOG) – cont.

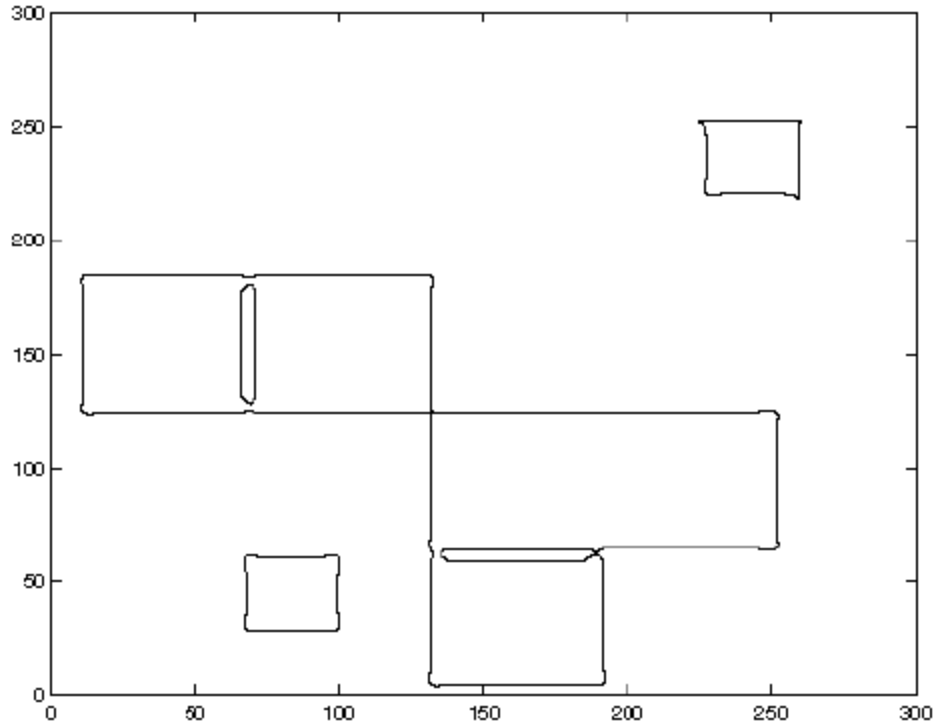
$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



- Disadvantage of LOG edge detection: #102
  - Does not handle corners well



• Gradient vs LOG: a comparison

- Gradient works well when the image contains sharp intensity transitions and low noise

- Zero-crossings of LOG offer better localization, especially when the edges are not very sharp

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

A sample image containing a vertical step edge.

0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

A sample image containing a vertical ramp edge.

0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0

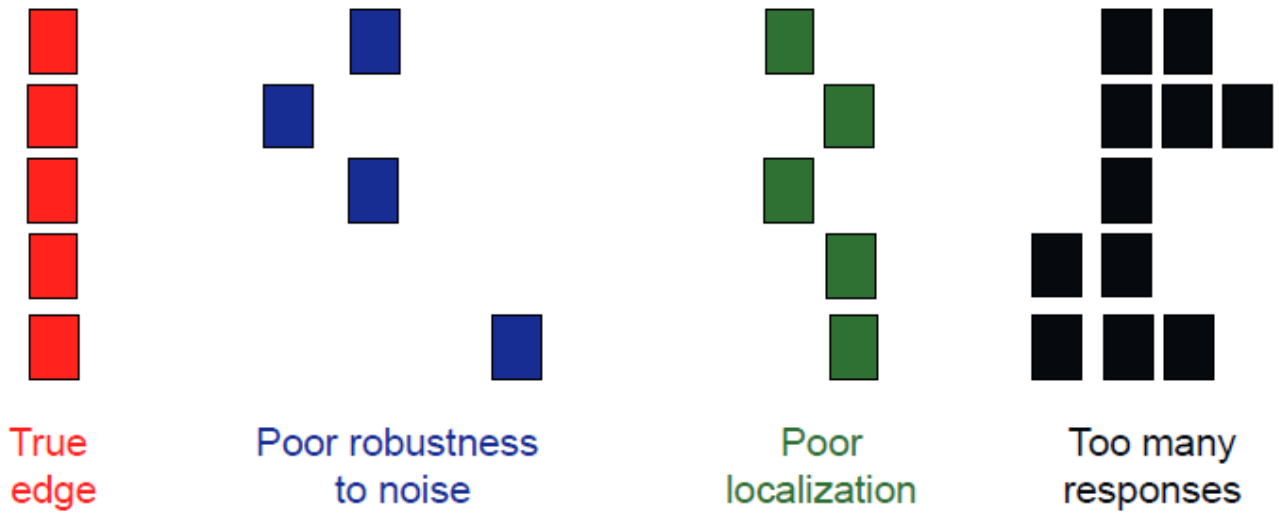
(Trucco, Chapt 4 AND Jain et al., Chapt 5)

Criteria for Optimal Edge Detection

- **(1) Good detection**
  - Minimize the probability of false positives (i.e., spurious edges).
  - Minimize the probability of false negatives (i.e., missing real edges).
- **(2) Good localization**
  - Detected edges must be as close as possible to the true edges.
- **(3) Single response**
  - Minimize the number of local maxima around the true edge. (one point for the true edge)

- Examples:

Factors in Edge Detection



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:

1. *Low error rate.* All edges should be found, and there should be no spurious responses. That is, the edges detected must be as close as possible to the true edges.
2. *Edge points should be well localized.* The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
3. *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.

The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solutions to these formulations. In general, it is difficult (or impossible) to find a closed-form solution

The Canny edge detector [1986]

- Driven by 3 objectives:

(1) Low error rate: All edges should be found

(2) Edge points should be well localized: Edges found must be as close as possible to true edges

(3) Single edge point response: Only one point for each true edge point should be returned

Canny concluded that a good approximation to the optimal step edge detector is the first derivative of a Gaussian

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

For 2-D, the 1-D approach still applies in the direction of the edge normal

Since normal direction is unknown beforehand, 1-D detector has to be applied for all directions

This is approximated by (1) smoothing image with 2-D Gaussian; (2) compute the gradient of the result; (3) use gradient magnitude and direction to estimate edge strength and direction at every point

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

$$G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \tag{10.2-29}$$

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

$$f_s(x, y) = G(x, y) \star f(x, y) \tag{10.2-30}$$

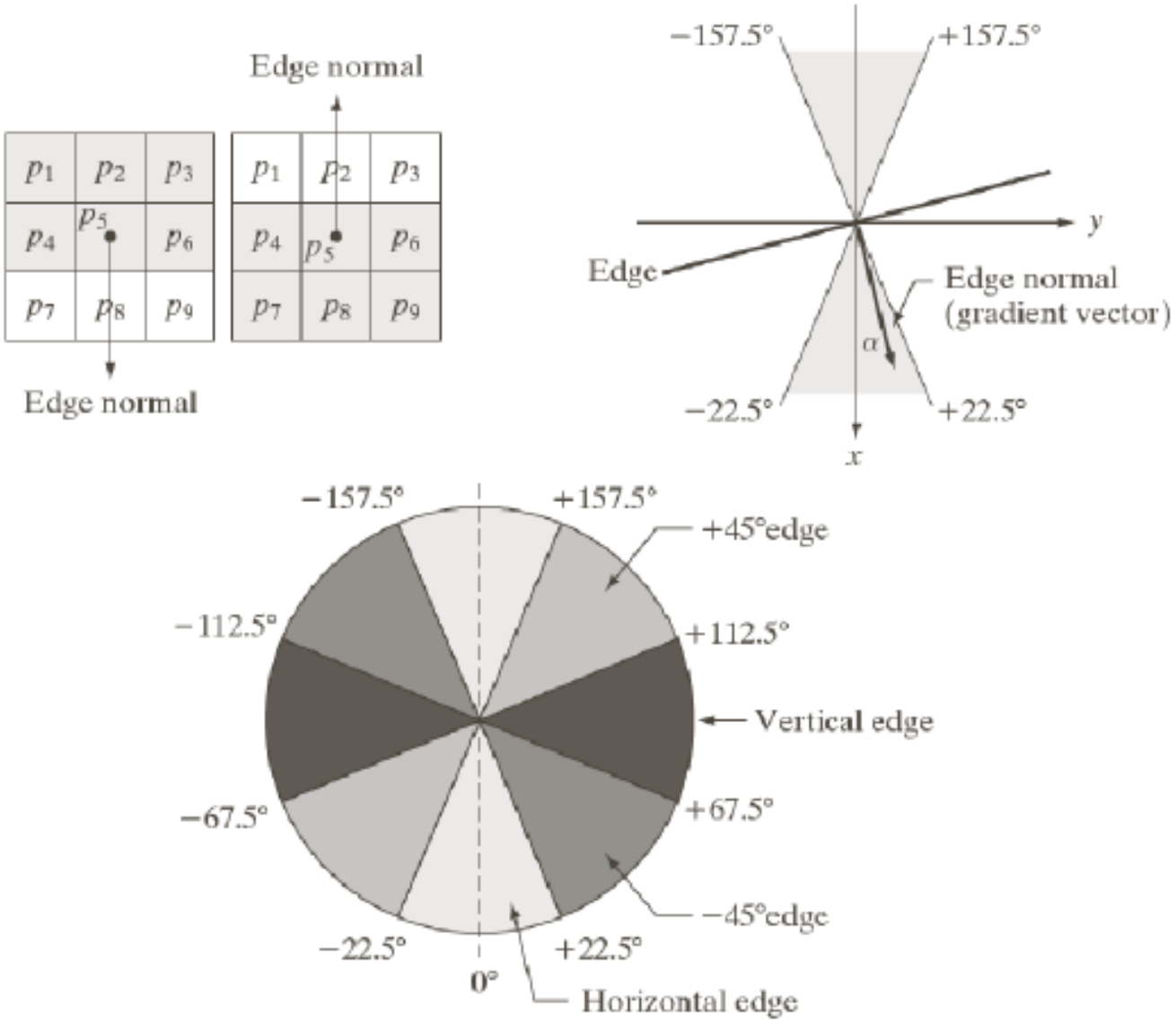
This operation is followed by computing the gradient magnitude and direction (angle), as discussed in Section 10.2.5:

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{10.2-31}$$

and

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{10.2-32}$$

with $g_x = \partial f_s/\partial x$ and $g_y = \partial f_s/\partial y$. Any of the filter mask pairs in Fig. 10.14 can be used to obtain g_x and g_y. Equation (10.2-30) is implemented using an $n \times n$ Gaussian mask whose size is discussed below. Keep in mind that $M(x, y)$ and $\alpha(x, y)$ are arrays of the same size as the image from which they are computed.



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (in gray) in a 3 × 3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3 × 3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa

1. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

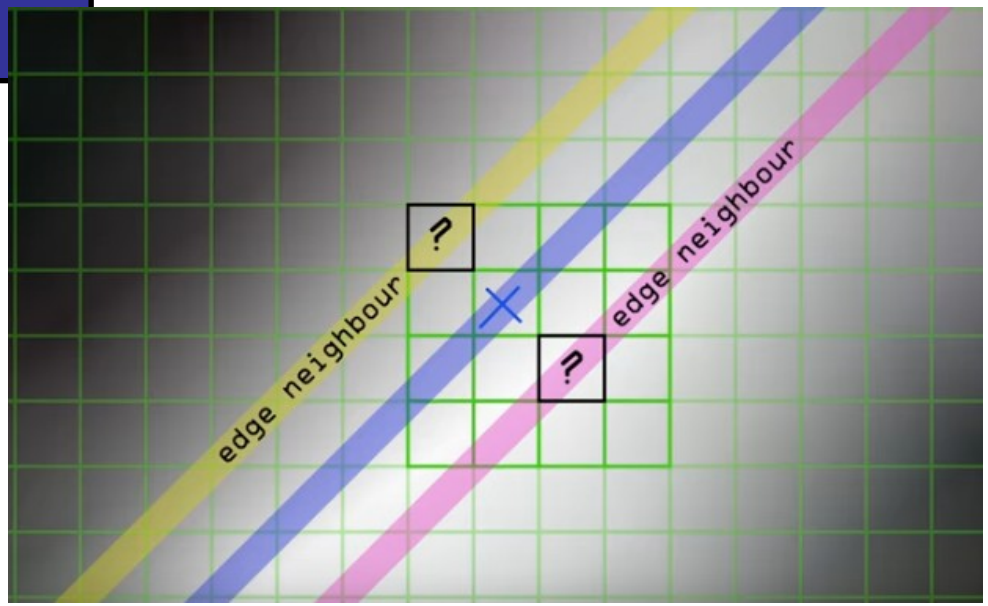
In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. For example,

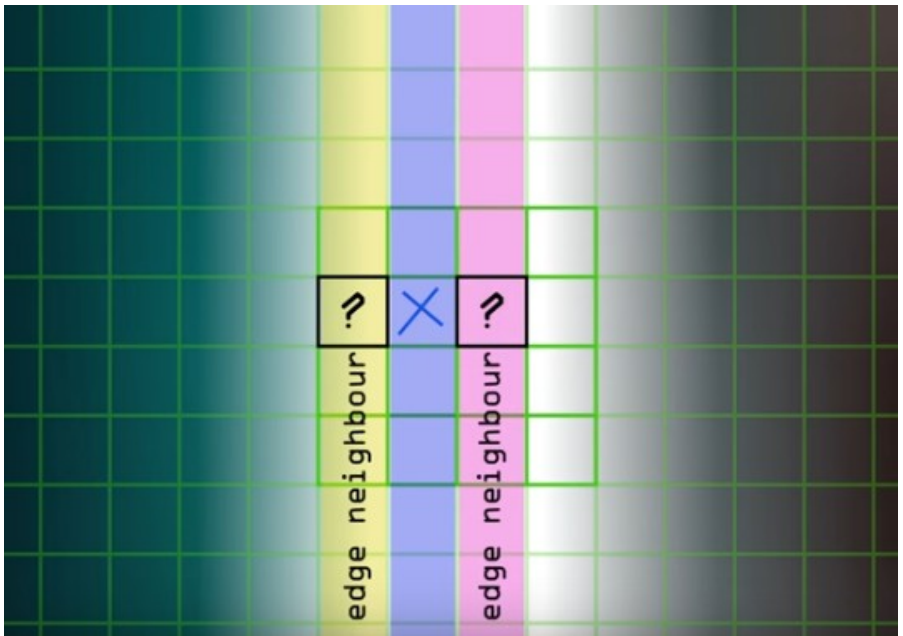
- if the rounded gradient angle is 0° (i.e. the edge is in the north–south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **east and west** directions,
- if the rounded gradient angle is 90° (i.e. the edge is in the east–west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north and south** directions,
- if the rounded gradient angle is 135° (i.e. the edge is in the northeast–southwest direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north west and south east** directions,
- if the rounded gradient angle is 45° (i.e. the edge is in the north west–south east direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north east and south west** directions.

In more accurate implementations, linear interpolation is used between the two neighbouring pixels that straddle the gradient direction. For example, if the gradient angle is between 45° and 90°, interpolation between gradients at the **north** and **north east** pixels will give one interpolated value, and interpolation between the **south** and **south west** pixels will give the other (using the conventions of last paragraph). The gradient magnitude at the central pixel must be greater than both of these for it to be marked as an edge.

Non-Maximum Suppression

https://en.wikipedia.org/wiki/Canny_edge_detector



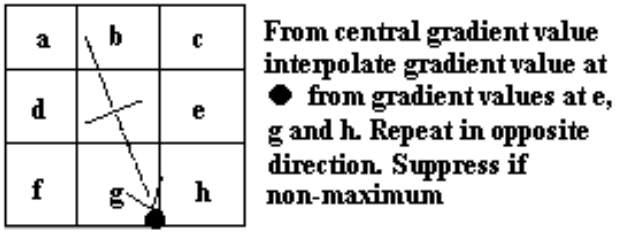


https://www.youtube.com/watch?v=sRFM5IEqR2w

Stage Three - Non-maximum Suppression

Having found the rate of intensity change at each point in the image, edges must now be placed at the points of maxima, or rather non-maxima must be suppressed. A local maximum occurs at a peak in the gradient function, or alternatively where the derivative of the gradient function is set to zero. However, in this case we wish to surpress non-maxima perpendicular to the edge direction, rather than parallel to (along) the edge direction, since we expect continuity of edge strength along an extended contour. (This assumption creates a problem at corners !)

Rather than perform an explicit differentiation perpendicular to each edge, another approximation is often used. Each pixel in turn forms the centre of a nine pixel neighbourhood. By interpolation of the surrounding discrete grid values, the gradient magnitudes are calculated at the neighbourhood boundary in both directions perpendicular to the centre pixel, as shown in Figure 6, below. If the pixel under consideration is not greater than these two values (i.e. non-maximum), it is suppressed.



Non-maximum Suppression

Nonmaxima supression scheme for 3×3 region centered at every point (x, y) in $\alpha(x, y)$:

(1) Find the direction d_k that is closest to $\alpha(x, y)$

(2) If the value of $M(x, y)$ is less than at least one of its two neighbours along d_k, then $g_N(x, y) = 0$ (supression); otherwise $g_N(x, y) = M(x, y)$

Final operation is to threshold $g_N(x, y)$ to reduce false edge points

• Hysteresis thresholding (Section 10.3.6): Two thresholds, T_L and T_H are selected using Otsu's method

$$g_{\mathrm{NH}}(x, y) = (g_N(x, y) \geq T_H) \qquad g_{\mathrm{NL}}(x, y) = (g_N(x, y) \geq T_L)$$

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

$g_{\mathrm{NH}}(x, y)$ **≡ "strong" edge pixels** $\qquad$ $g_{\mathrm{NL}}(x, y)$ **≡ "weak" edge pixels**

Strong pixels in $g_{\mathrm{NH}}(x, y)$ are assumed to be valid and marked accordingly

Also, the edges in $g_{\mathrm{NH}}(x, y)$ typically have gaps!

Credits: H. Coetzer, Department of Applied Mathematics, Stellenbosch University, S. Africa

Longer edges are formed using the following procedure:

(a) Locate the next unvisited pixel, p, in $g_{\mathrm{NH}}(x, y)$

(b) Mark as valid edge pixels all the weak pixels in $g_{\mathrm{NL}}(x, y)$ that are connected to p (8-connectivity)

(c) If all nonzero pixels in $g_{\mathrm{NH}}(x, y)$ have been visited, go to step (d). Else, return to step (a)

(d) Set to zero all pixels in $g_{\mathrm{NL}}(x, y)$ that were not marked as valid edge pixels

Final output image is formed by appending to $g_{\mathrm{NH}}(x, y)$ all the nonzeo pixels from $g_{\mathrm{NL}}(x, y)$

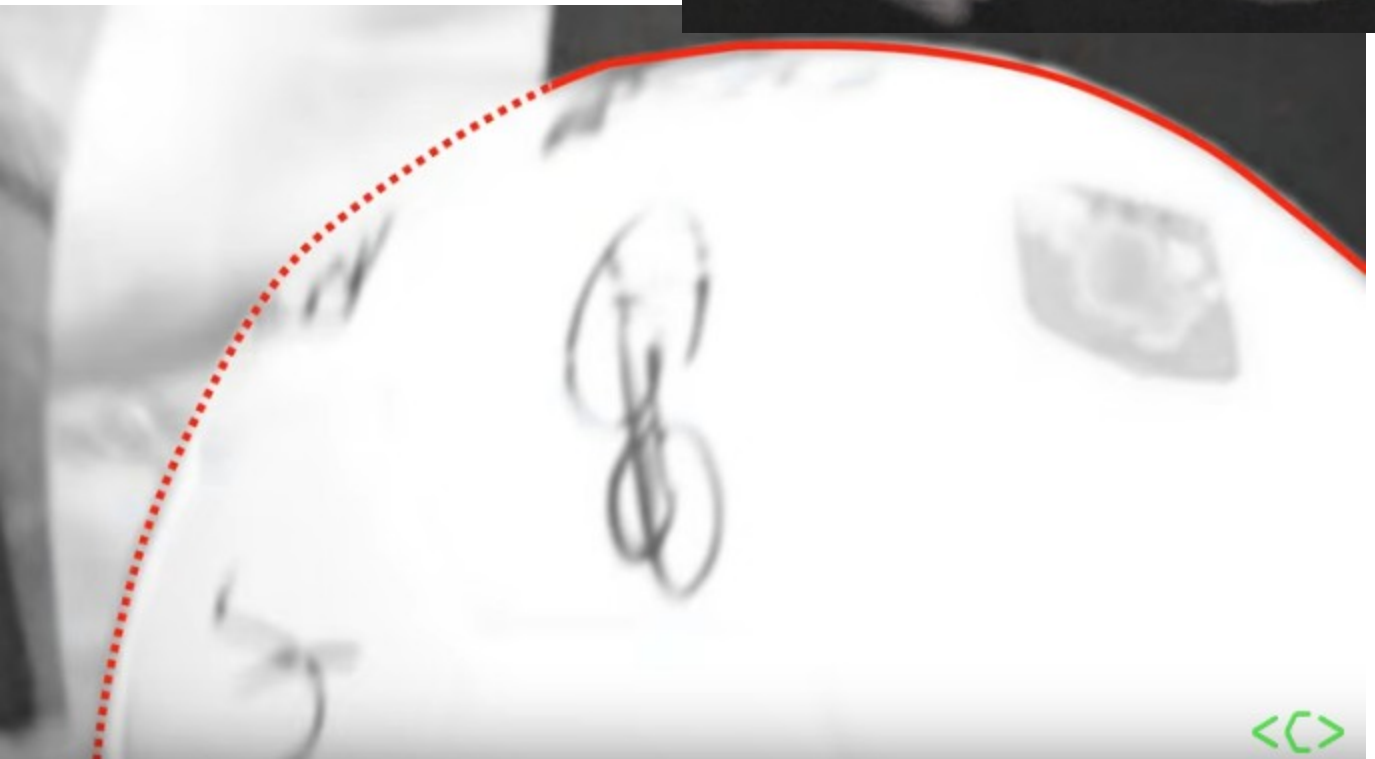
Summary: (Step (4) is typically followed by one pass of edge thinning)

(1) Smooth the input image with a Gaussian filter

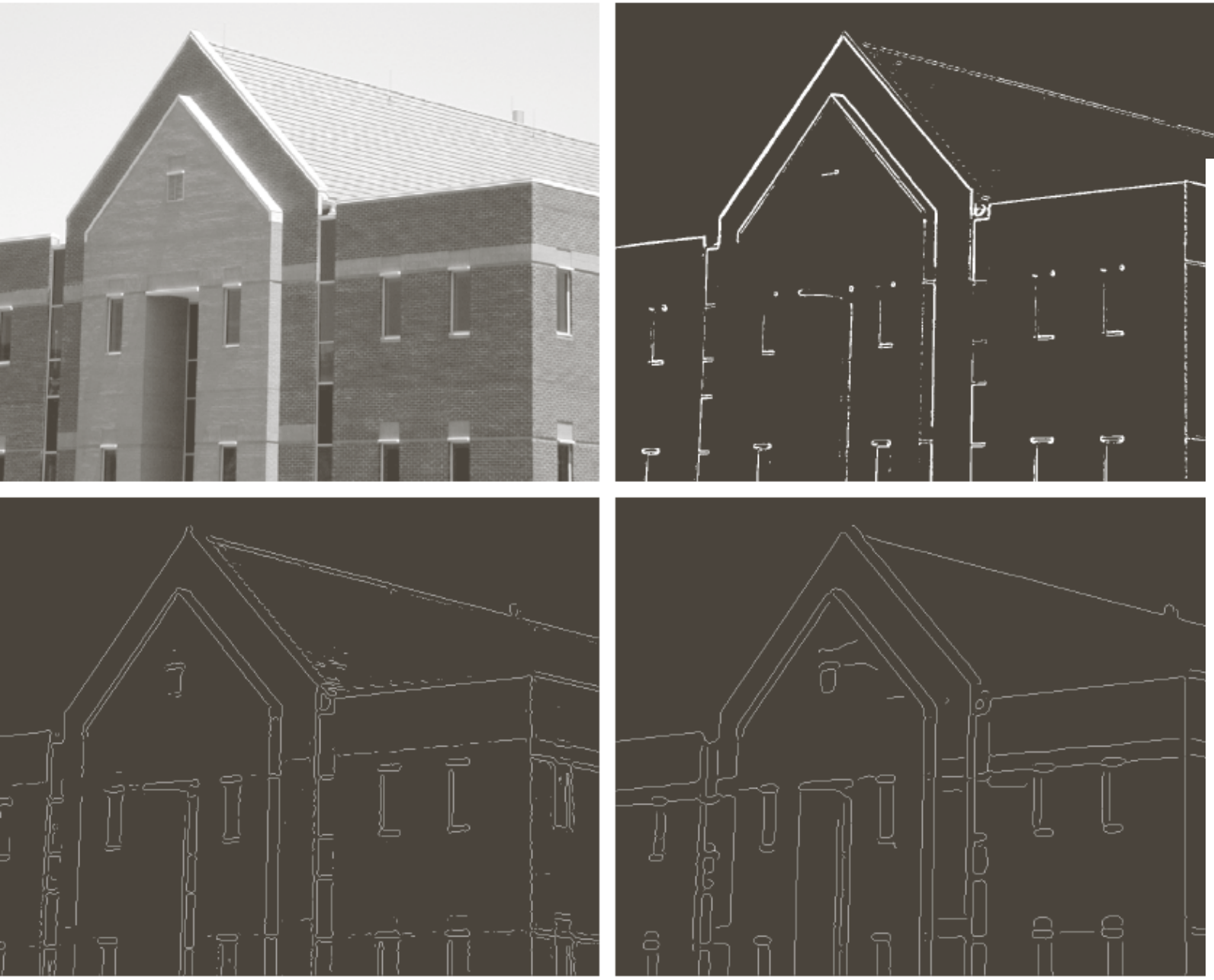
(2) Compute the gradient magnitude and angle images

(3) Apply nonmaxima supression to the gradient magnitude image

(4) Use double thresholding and connectivity analysis to detect and link edges



https://www.youtube.com/watch?v=sRFM5IEqR2w



a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 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. Note the significant improvement of the Canny image compared to the other two.

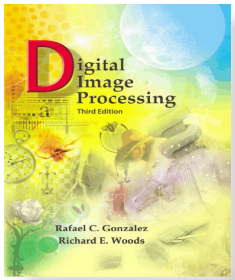
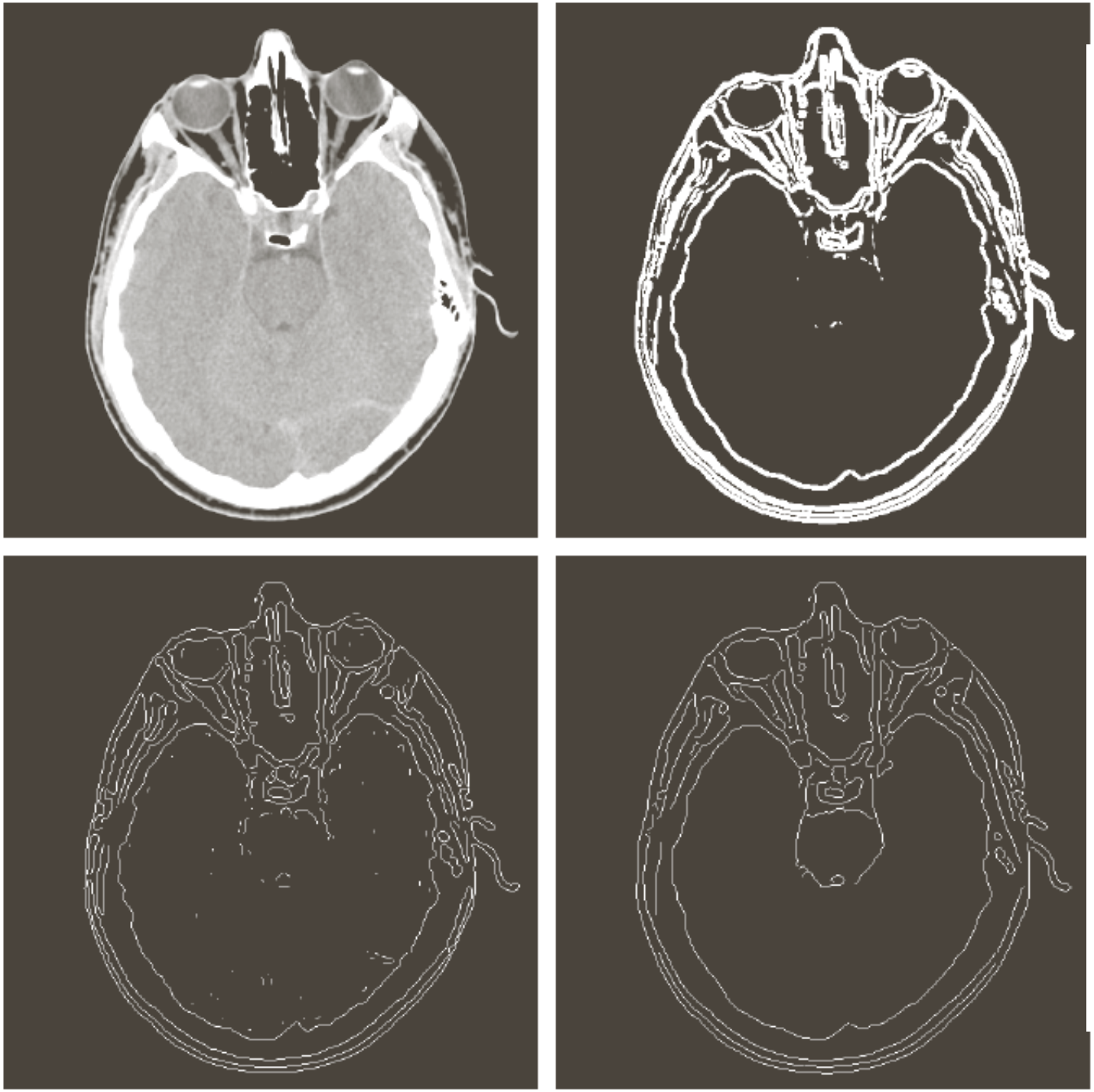


Figure 10.25(d) shows the result obtained with the Canny algorithm using the parameters $T_L = 0.04$, $T_H = 0.10$ (2.5 times the value of the low threshold), $\sigma = 4$ and a mask of size 25×25, which corresponds to the smallest odd integer greater than 6σ. These parameters were chosen interactively to achieve the objectives stated in the previous paragraph for the gradient and Marr-Hildreth images. Comparing the Canny image with the other two images, we see significant improvements in detail of the principal edges and, at the same time, more rejection of irrelevant features in the Canny result. Note, for example, that both edges of the concrete band lining the bricks in the upper section of the image were detected by the Canny algorithm, whereas the thresholded gradient lost both of these edges and the Marr-Hildreth image contains only the upper one. In terms of filtering out irrelevant detail, the Canny image does not contain a single edge due to the roof tiles; this is not true in the other two images. The quality of the lines with regard to continuity, thinness, and straightness is also superior in the Canny image. Results such as these have made the Canny algorithm a tool of choice for edge detection.



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.)

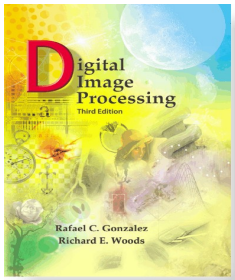
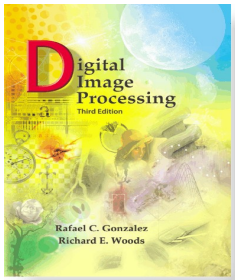


Figure 10.26(b) shows a thresholded gradient image that was first smoothed with a 5×5 averaging filter. The threshold required to achieve the result shown was 15% of the maximum value of the gradient image. Figure 10.26(c) shows the result obtained with the Marr-Hildreth edge-detection algorithm with a threshold of 0.002, $\sigma = 3$, and a mask of size 19×19 pixels. Figure 10.26(d) was obtained using the Canny algorithm with $T_L = 0.05$, $T_H = 0.15$ (3 times the value of the low threshold), $\sigma = 2$, and a mask of size 13×13, which, as in the Marr-Hildreth case, corresponds to the smallest odd integer greater than 6σ.

The results in Fig. 10.26 correspond closely to the results and conclusions in the previous example in terms of edge quality and the ability to eliminate irrelevant detail. Note also that the Canny algorithm was the only procedure capable of yielding a totally unbroken edge for the posterior boundary of the brain. It was also the only procedure capable of finding the best contours while eliminating all the edges associated with the gray matter in the original image. ■



10.2.7 Edge Linking and Boundary Detection

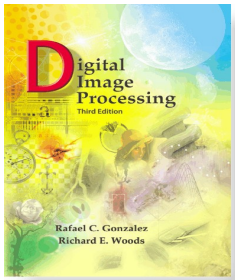
Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edges completely because of noise, breaks in the edges due to nonuniform illumination, and other effects that introduce spurious discontinuities in intensity values. Therefore, edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries.

Local processing

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are (1) the strength (magnitude) and (2) the direction of the gradient vector. The first property is based on Eq. (10.2-10). Let S_{xy} denote the set of coordinates of a neighborhood centered at point (x, y) in an image. An edge pixel with coordinates (s, t) in S_{xy} is similar in *magnitude* to the pixel at (x, y) if

$$|M(s, t) - M(x, y)| \leq E \qquad (10.2\text{-}36)$$

where E is a positive threshold.



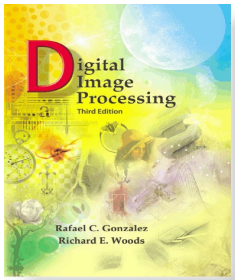
Local processing

The direction angle of the gradient vector is given by Eq. (10.2-11). An edge pixel with coordinates (s, t) in S_{xy} has an *angle* similar to the pixel at (x, y) if

$$|\alpha(s, t) - \alpha(x, y)| \leq A \qquad (10.2\text{-}37)$$

where A is a positive angle threshold. As noted in Section 10.2.5, the direction of the edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

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



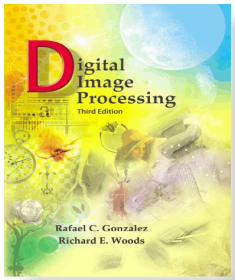
Local processing

The preceding formulation is computationally expensive because all neighbors of every point have to be examined. A simplification particularly well suited for real time applications consists of the following steps:

1. Compute the gradient magnitude and angle arrays, $M(x, y)$ and $\alpha(x, y)$, of the input image, $f(x, y)$.
2. Form a binary image, g, whose value at any pair of coordinates (x, y) is given by:

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

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



Local processing

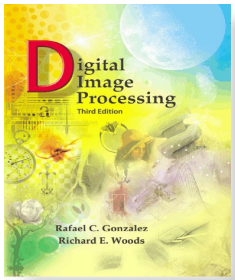
3. Scan the rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length, K. Note that, by definition, a gap is bounded at both ends by one or more 1s. The rows are processed individually, with no memory between them.
4. To detect gaps in any other direction,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

When interest lies in horizontal and vertical edge linking, Step 4 becomes a simple procedure in which g is rotated ninety degrees, the rows are scanned, and the result is rotated back. This is the application found most frequently in practice and, as the following example shows, this approach can yield good results. In general, image rotation is an expensive computational process so, when linking in numerous angle directions is required, it is more practical to combine Steps 3 and 4 into a single, radial scanning procedure.

a b c
d e f

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

■ Figure 10.27(a) shows an image of the rear of a vehicle. The objective of this example is to illustrate the use of the preceding algorithm for finding rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges. Figure 10.27(b) shows the gradient magnitude image, $M(x, y)$, and Figs. 10.27(c) and (d) show the result of Steps (3) and (4) of the algorithm obtained by letting T_M equal to 30% of the maximum gradient value, $A = 90°$, $T_A = 45°$, and filling in all gaps of 25 or fewer pixels (approximately 5% of the image width). Use of a large range of allowable angle directions was required to detect the rounded corners of the license plate enclosure, as well as the rear windows of the vehicle. Figure 10.27(e) is the result of forming the logical OR of the two preceding images, and Fig. 10.27(f) was obtained by thinning 10.27(e) with the thinning procedure discussed in Section 9.5.5. As Fig. 10.16(f) shows, the rectangle corresponding to the license plate was clearly detected in the image. It would be a simple matter to isolate the license plate from all the rectangles in the image using the fact that the width-to-height ratio of license plates in the U.S. has a distinctive 2:1 proportion. ■



Global processing using the Hough transform

Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to find first all lines determined by every pair of points and then find all subsets of points that are close to particular lines. This approach involves finding $n(n-1)/2 \sim n^2$ lines and then performing $(n)(n(n-1))/2 \sim n^3$ comparisons of every point to all lines. This is a computationally prohibitive task in all but the most trivial applications.

Hough [1962] proposed an alternative approach, commonly referred to as the *Hough transform*. Consider a point (x_i, y_i) in the xy-plane and the general equation of a straight line in slope-intercept form, $y_i = ax_i + b$. Infinitely many lines pass through (x_i, y_i), but they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. However, writing this equation as $b = -x_i a + y_i$ and considering the ab-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and, unless they are parallel, this line intersects the line associated with (x_i, y_i) at some point (a', b'), where a' is the slope and b' the intercept of the line containing *both* (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, *all* the points on this line have lines in parameter space that intersect at (a', b'). Figure 10.31 illustrates these concepts.

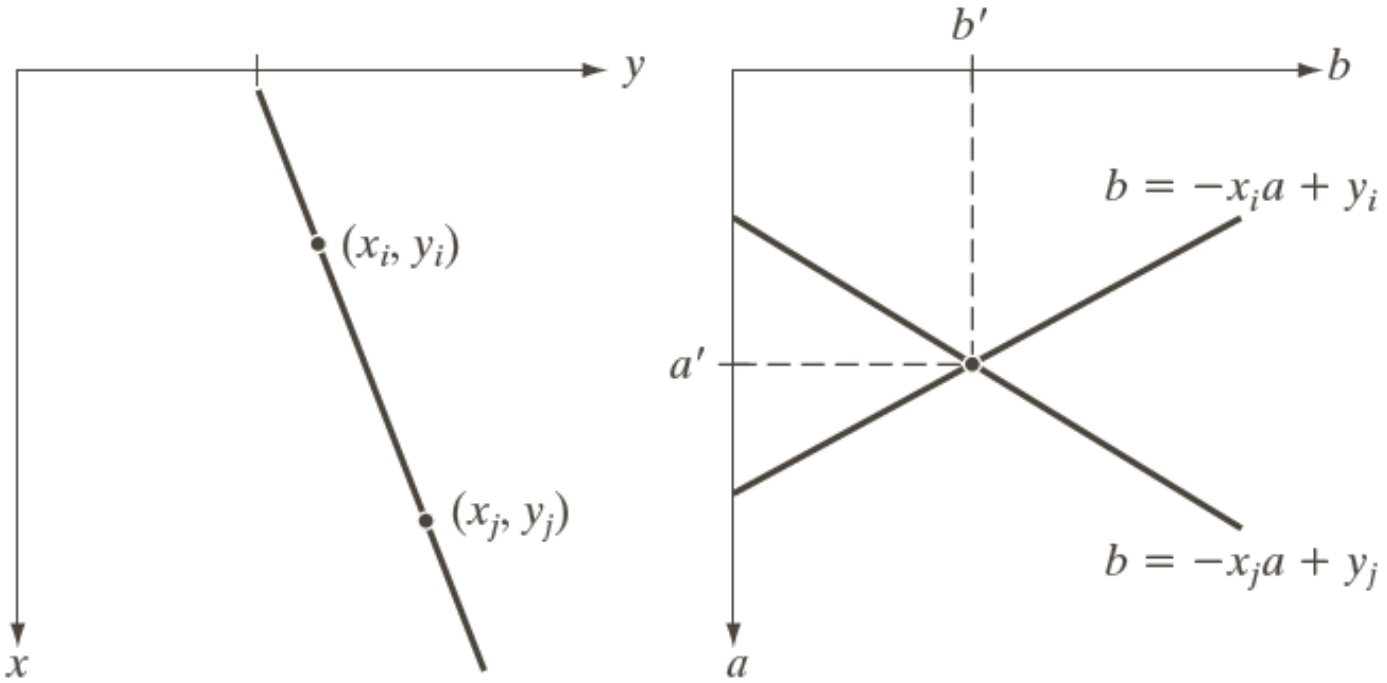
Hough Transform

- Straight line case
  - Consider a single isolated edge point (x_i, y_i)
    - There are an infinite number of lines that could pass through the points
  - Each of these lines can be characterized by some particular equation

$$y_i = mx_i + c$$

Chapter 10

Segmentation

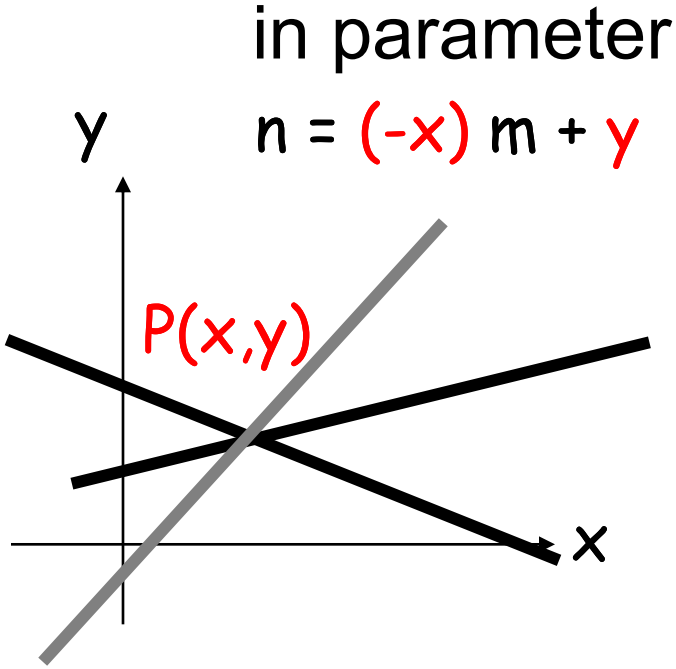


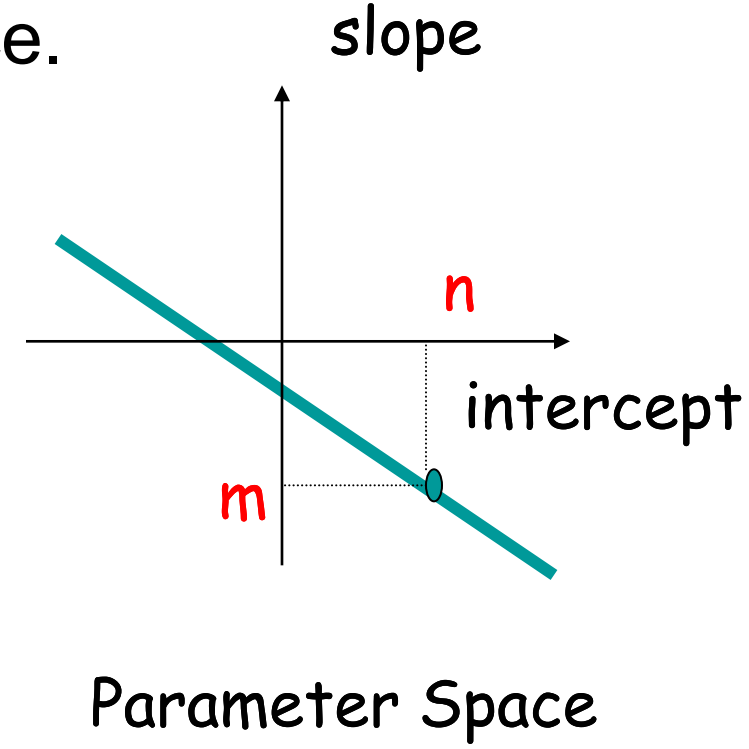
a b

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

Hough Transform Technique

- Given an edge point, there is an infinite number of lines passing through it (Vary m and n).
  - These lines can be represented as a line in parameter space.

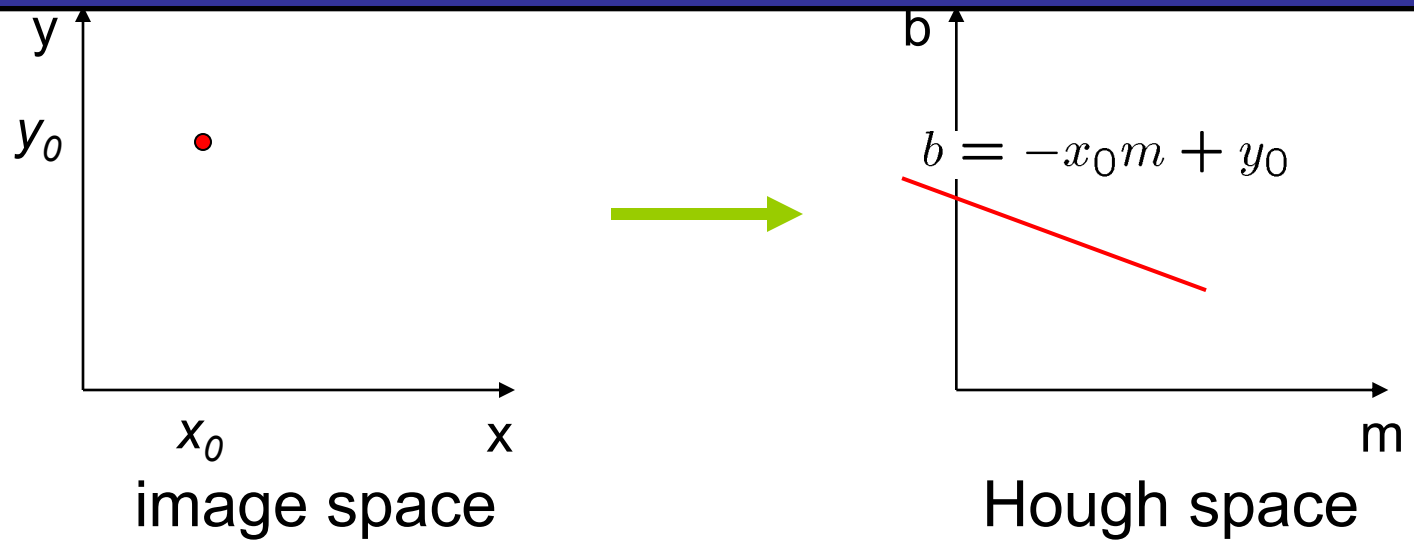




Hough Transform Technique

- Given a set of collinear edge points, each of them have associated a line in parameter space.
  - These lines intersect at the point (m,n) corresponding to the parameters of the line in the image space.

Finding lines in an image



- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space
  - To go from image space to Hough space:
    - given a set of points (x,y), find all (m,b) such that y = mx + b
  - What does a point (x_0, y_0) in the image space map to?
    - A: the solutions of $b = -x_0 m + y_0$
    - this is a line in Hough space

Image Parameter Spaces

- Image Space
  - Lines
  - Points
  - Collinear points
- Parameter Space
  - Points
  - Lines
  - Intersecting lines

Hough Transform Philosophy

- H.T. is a method for detecting straight lines, shapes and curves in images.

- Main idea:
  - Map a difficult pattern problem into a simple **peak detection** problem

Hough Transform Technique

- At each point of the (discrete) parameter space, count how many lines pass through it.
  - Use an array of counters
  - Can be thought as a " parameter **image**"
- The higher the count, the more edges are collinear in the image space.
  - Find a peak in the counter array
  - This is a "bright" point in the parameter image
  - It can be found by thresholding

HT properties

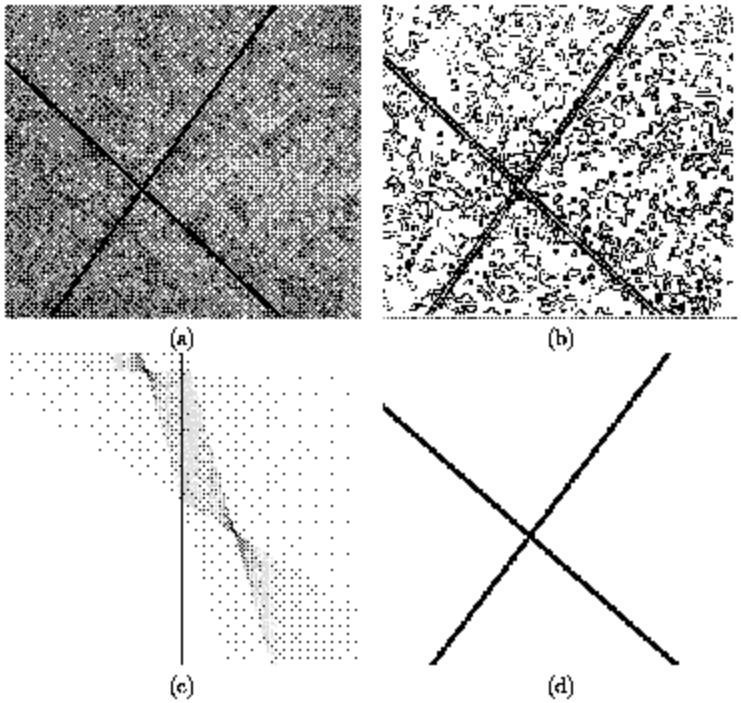
- Original HT designed to detect straight lines and curves
- Advantage - robustness of segmentation results
  - segmentation not too sensitive to imperfect data or noise
  - better than edge linking
  - works through occlusion
- Any ***part*** of a straight line can be mapped into parameter space

Accumulators

- Each edge pixel (x,y) votes in (k,q) space for each possible line through it
  - i.e. all combinations of k & q
- This is called the accumulator
- If position (k,q) in accumulator has ***n*** votes
  - ***n*** feature points lie on that line in image space
- Large ***n*** in parameter space, more probable that line exists in image space
- Therefore, find max ***n*** in accumulator to find lines

HT Algorithm (briefly)

- Find all desired feature points in image space
  - i.e. edge detect (low pass filter)
- Take each feature point
  - increment appropriate values in parameter space
  - i.e. all values of (k,q) for give (x,y)
- Find maxima in accumulator array
- Map parameter space back into image space to view results



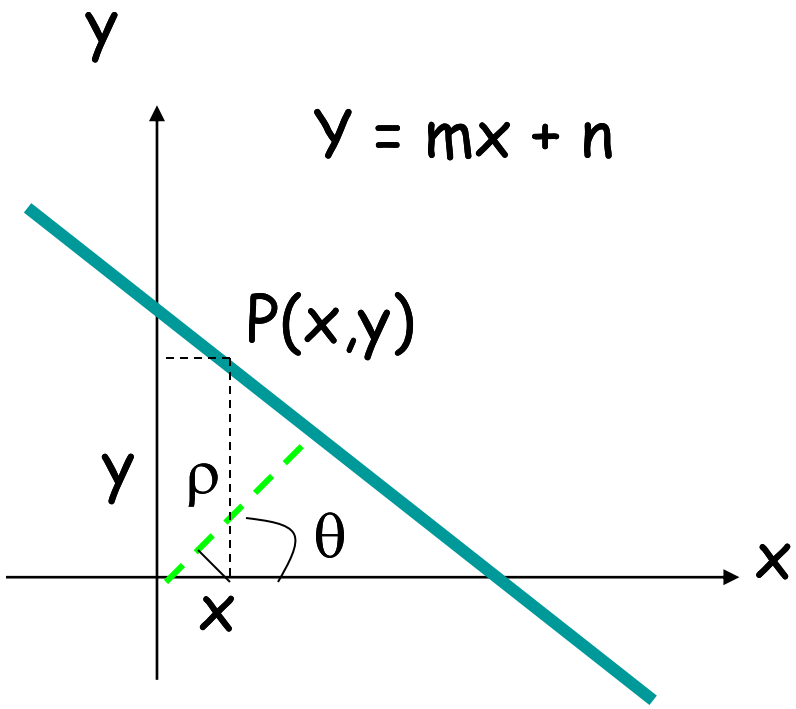
Practical Issues with This Hough Parameterization

- The slope of the line is $-\infty < m < \infty$
  - The parameter space is INFINITE
- The representation $y = mx + n$ does not express lines of the form $x = k$

Solution

Use the "Normal" equation of a line:

Using the "Normal" equation of a line:



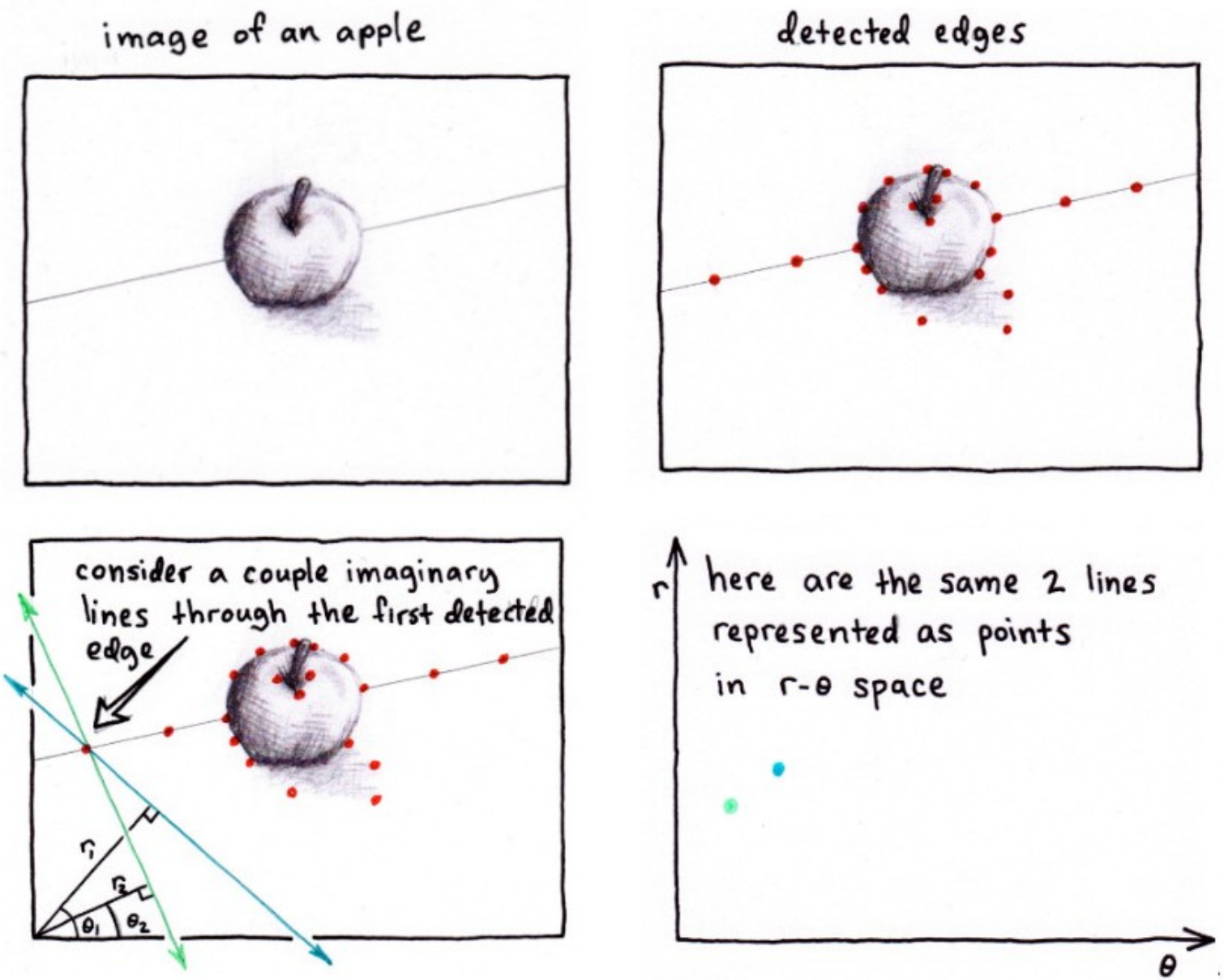
$\rho = x\cos\theta + y\sin\theta$

θ Is the line orientation

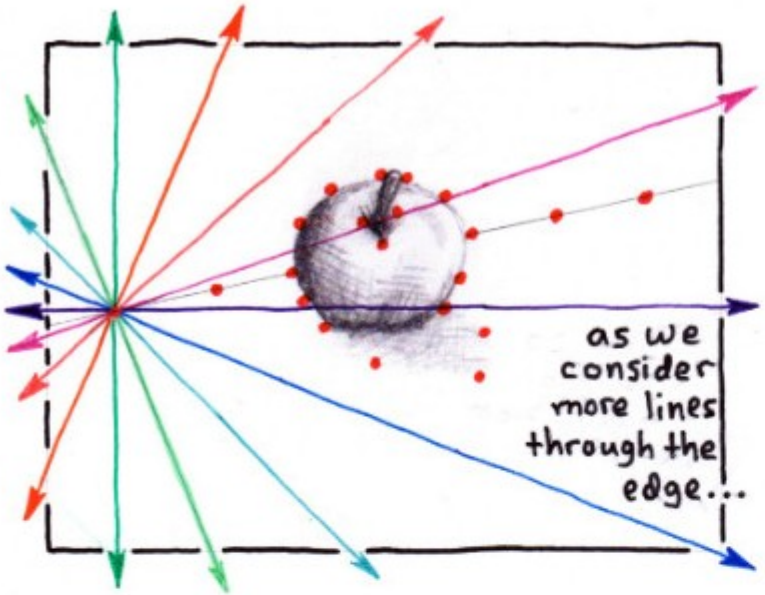
ρ Is the distance between the origin and the line

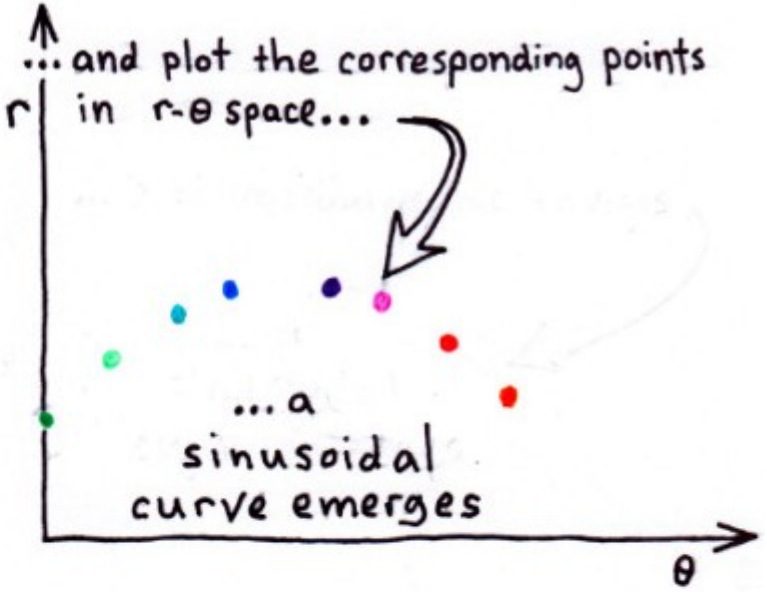
- A Point in Image Space is now represented as a SINUSOID
  - $\rho = x\cos\theta + y\sin\theta$

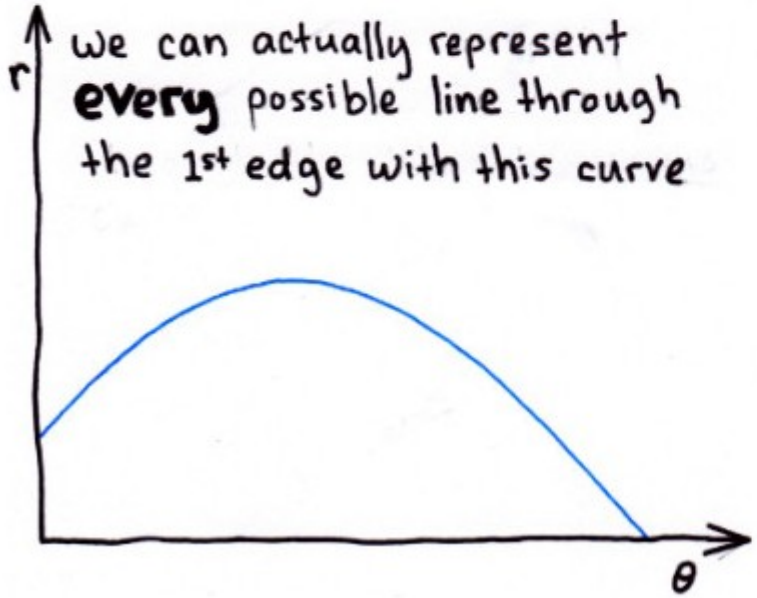
A very basic, visual explanation of how a Hough Transform works for detecting lines in an image

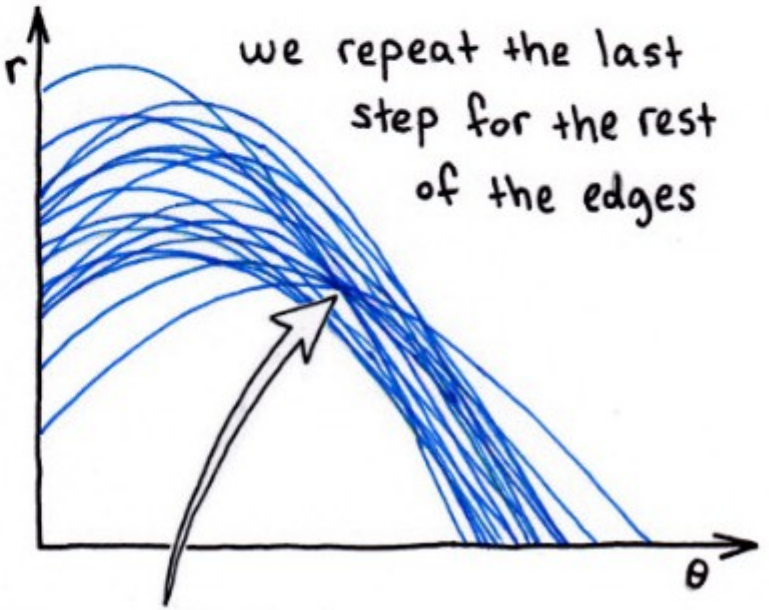


Credits: http://stackoverflow.com/questions/4709725/explain-hough-transformation (mlai's answer)

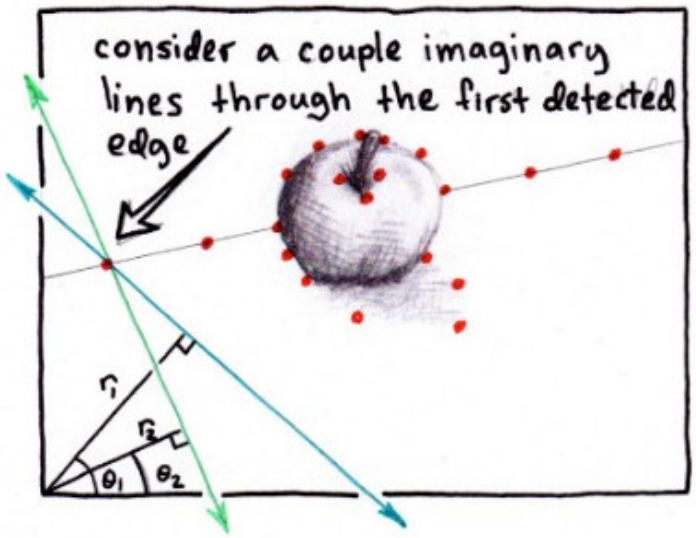


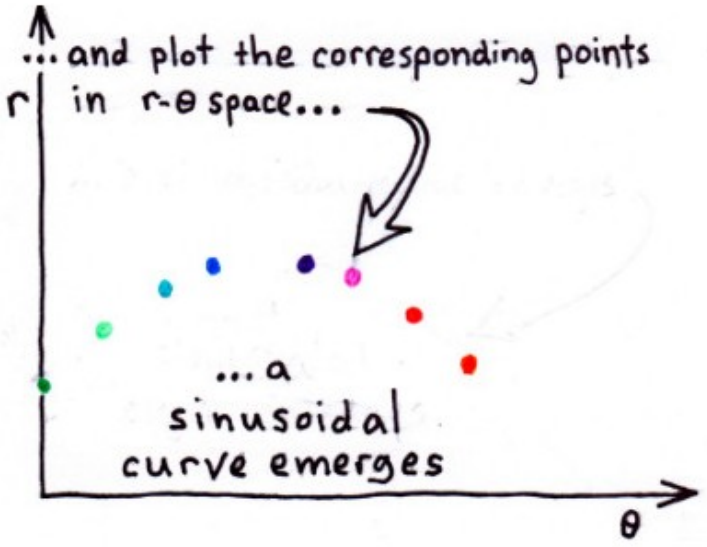


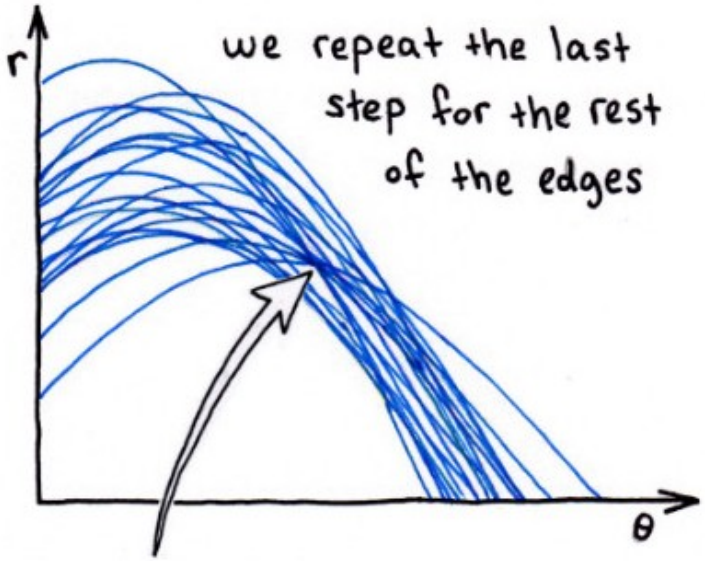




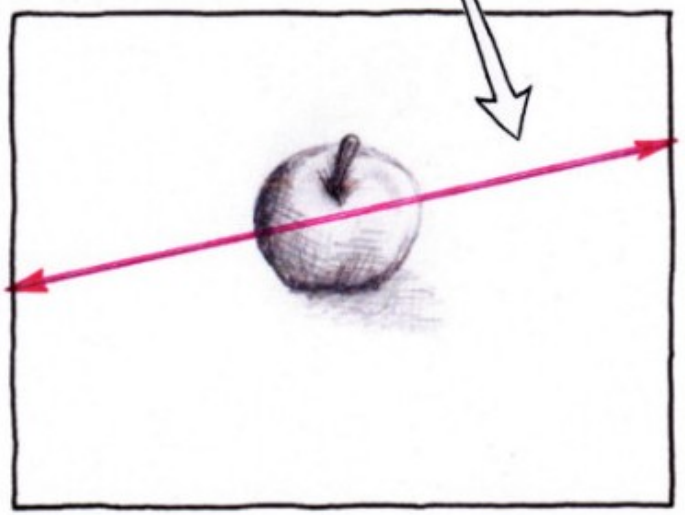
Credits: http://stackoverflow.com/questions/4709725/explain-hough-transformation (mlai's answer)





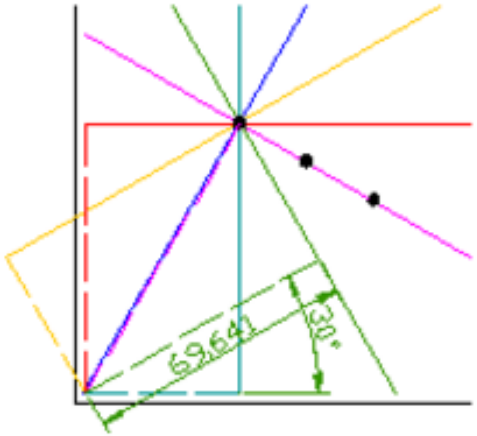


The point with the most curve-crossings identifies an imaginary line in the original image that passes through the most edges

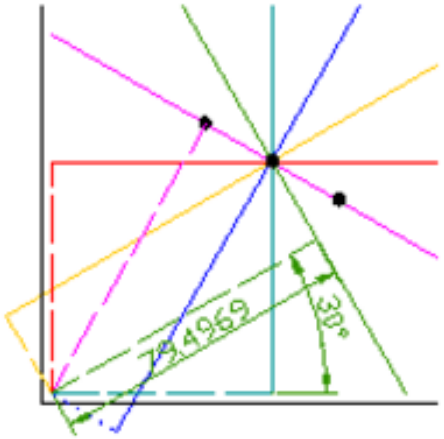


Credits: http://stackoverflow.com/questions/4709725/explain-hough-transformation (mlai's answer)

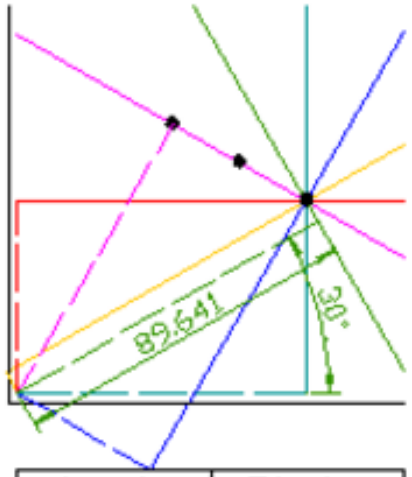
Consider three data points, shown here as black dots.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

- For each data point, a number of lines are plotted going through it, all at different angles. These are shown here as solid lines.
- For each solid line a line is plotted which is perpendicular to it and which intersects the origin. These are shown as dashed lines.
- The length (i.e. perpendicular distance to the origin) and angle of each dashed line is measured. In the diagram above, the results are shown in tables.

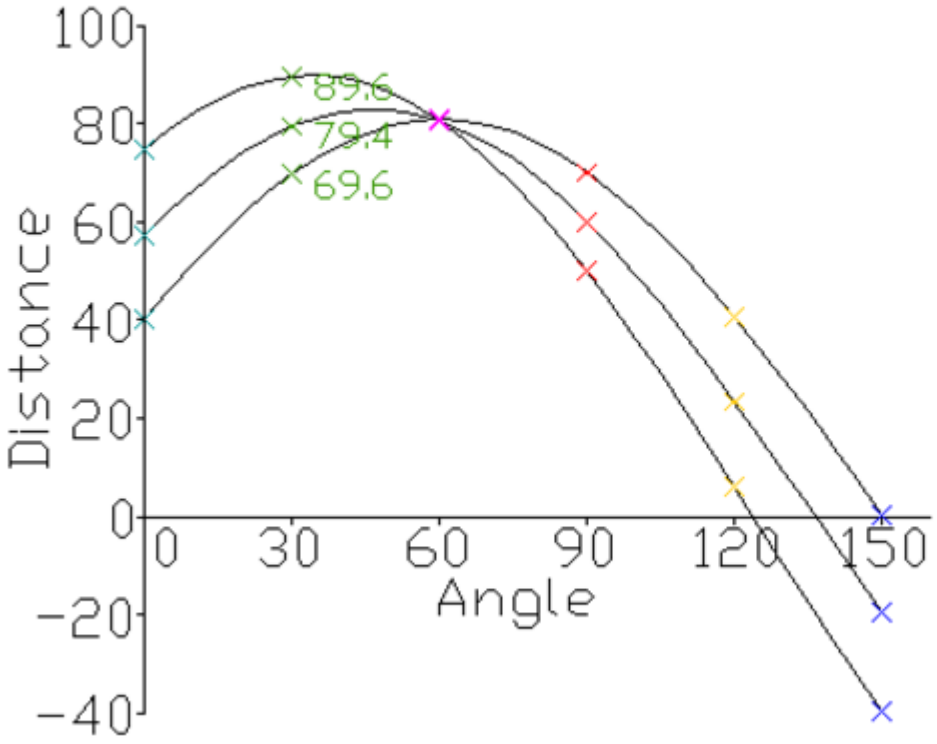
Credits: https://en.wikipedia.org/wiki/Hough_transform

Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

- A graph of the line lengths for each angle, known as a Hough space graph, is then created.



The point where the curves intersect gives a distance and angle.

This distance and angle indicate the line which intersects the points being tested.

In the graph shown the lines intersect at the pink point; this corresponds to the solid pink line in the diagrams above, which passes through all three points.

Credits: https://en.wikipedia.org/wiki/Hough_transform

Algorithm 3.1 Simple Hough algorithm for detecting straight lines. It returns a list containing the parameters $\langle \theta, r \rangle$ of the K strongest lines in the binary edge image I.

1: HOUGHLINES(I, N_θ, N_r, K)

Computes the Hough transform to detect straight lines in the binary image I (of size $M \times N$), using N_θ, N_r discrete steps for the angle and radius, respectively. Returns the list of parameter pairs $\langle \theta_i, r_i \rangle$ for the K strongest lines found.

2: $(u_c, v_c) \leftarrow (\frac{M}{2}, \frac{N}{2})$ ▷ image center

3: $r_{\max} \leftarrow \sqrt{u_c^2 + v_c^2}$ ▷ max. radius is half the image diagonal

4: $\Delta_\theta \leftarrow \frac{\pi}{N_\theta}$ ▷ angular increment

5: $\Delta_r \leftarrow \frac{2 \cdot r_{\max}}{N_r}$ ▷ radial increment

6: Create the accumulator array $Acc(i_\theta, i_r)$ of size $N_\theta \times N_r$

7: **for all** accumulator cells (i_θ, i_r) **do**

8: $Acc(i_\theta, i_r) \leftarrow 0$ ▷ initialize the accumulator array

9: **for all** image coordinates (u, v) **do** ▷ scan the image

10: **if** $I(u, v)$ is an edge point **then**

11: $(x, y) \leftarrow (u - u_c, v - v_c)$ ▷ coordinate relative to center

12: **for** $i_\theta \leftarrow 0 \ldots N_\theta - 1$ **do** ▷ angular index i_θ

13: $\theta \leftarrow \Delta_\theta \cdot i_\theta$ ▷ real angle, $0 \le \theta < \pi$

14: $r \leftarrow x \cdot \cos(\theta) + y \cdot \sin(\theta)$ ▷ real radius (pos./neg.)

15: $i_r \leftarrow \frac{N_r}{2} + \text{round}(\frac{r}{\Delta_r})$ ▷ radial index i_r

16: $Acc(i_\theta, i_r) \leftarrow Acc(i_\theta, i_r) + 1$ ▷ increment $Acc(i_\theta, i_r)$

Find the parameters pairs $\langle \theta_j, r_j \rangle$ for the K strongest lines:

17: $MaxLines \leftarrow$ FINDMAXLINES(Acc, K)

18: **return** $MaxLines$.

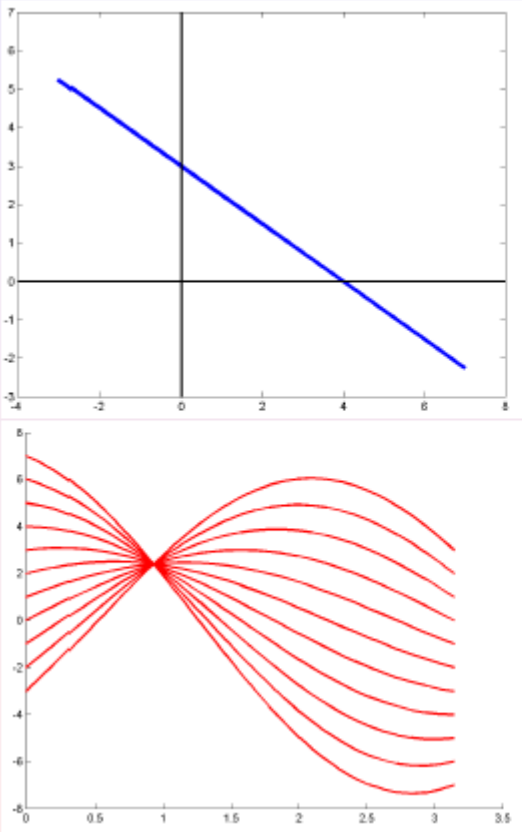
1. Hough Transform for Analytical Shapes
   - Voting in Parameter Space
   - Using Directional Information
   - Error Compensation: Smoothing
2. Generalizing to Non-Analytical Shapes

References

D. H. Ballard
Generalizing the Hough Transform to Detect Arbitrary Shapes
Pattern Recognition, **13**(2):111-122, 1981.

R. O. Duda and P. E. Hart
Use of the Hough Transformation to Detect Lines and Curves in Pictures
Communications of Association for Computing Machinery, **15**(1):11-15, 1972.

Straight line

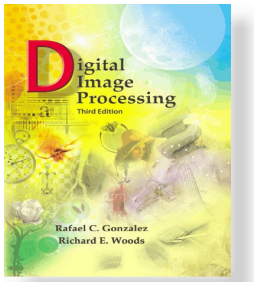


- Normal parameterization: $x\cos\theta + y\sin\theta = \rho$
- Points in picture $\leftrightarrow$ sinusoids in parameter space
- Points in parameter space $\leftrightarrow$ lines in picture
- Sinusoids corresponding to co-linear points intersect at an unique point

Example

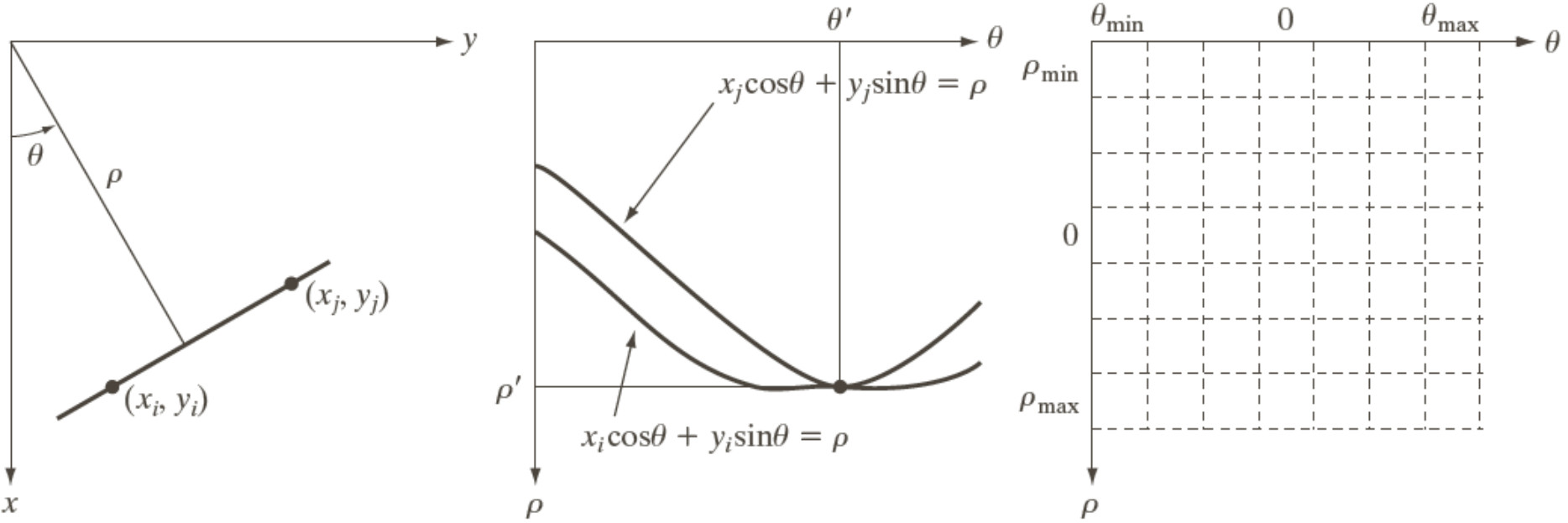
Line: $0.6x + 0.4y = 2.4$

Sinusoids intersect at: $\rho = 2.4$, $\theta = 0.9273$



Chapter 10

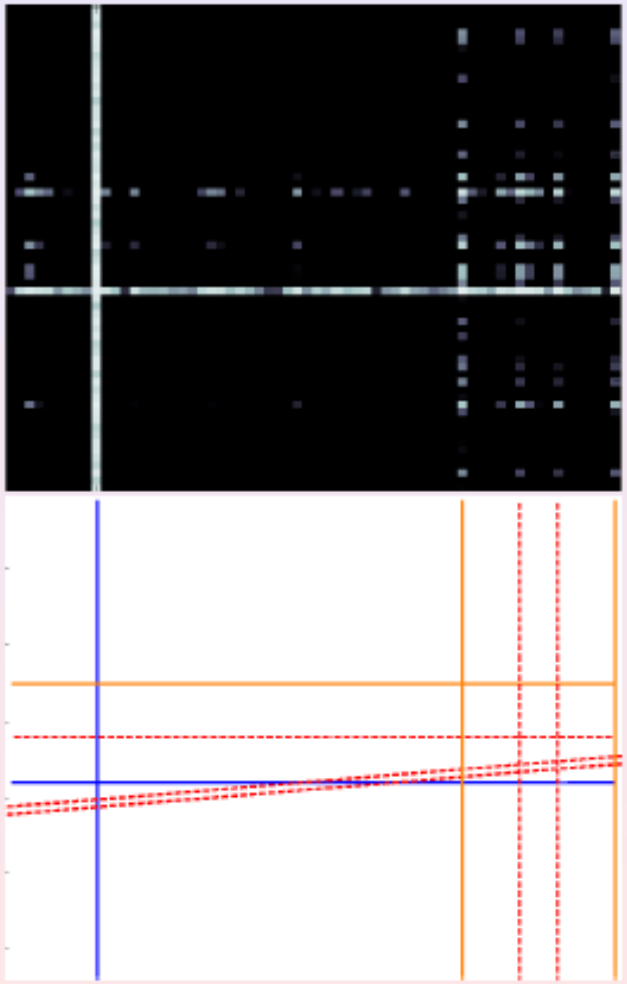
Segmentation



a b c

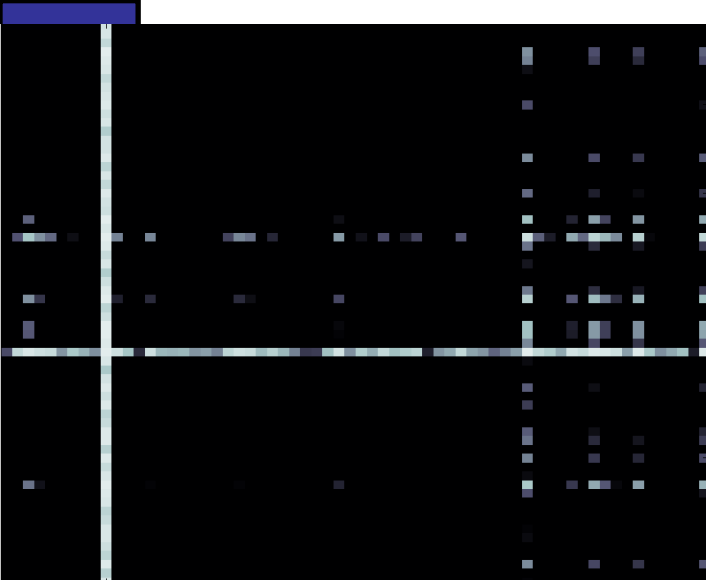
FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.

Quantize parameter space and vote into bins



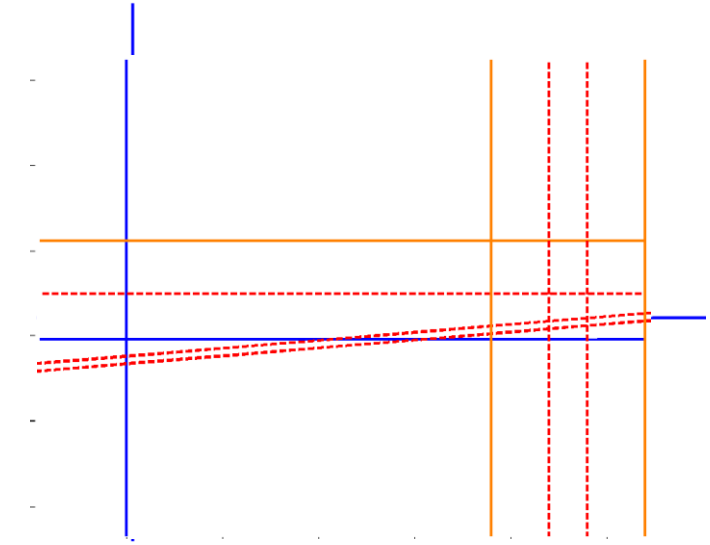
- Let $\rho \in [-R, R]$ and $\theta \in [0, \pi)$
- For each edge point (x_i, y_i), calculate:
  $\hat{\rho} = x_i \cos\hat{\theta} + y_i \sin\hat{\theta} \qquad \forall\hat{\theta} \in [0, \pi)$
- Accumulator: $\mathcal{A}(\hat{\rho}, \hat{\theta}) = \mathcal{A}(\hat{\rho}, \hat{\theta}) + 1$
- Threshold the accumulator values to get parameters for detected lines
  - Threshold at $\mathcal{A}(\hat{\rho}, \hat{\theta}) = 15$
- Same general idea applies to other analytical shapes

- $\mathcal{O}(nd_1)$ computation, instead of $\mathcal{O}(n^2)$
- Can we do better?



Let $\rho = [-R, R]$ and $\theta = [0, \pi)$
For each edge point (x_i, y_i),
calculate:
$\theta' = x_i \cos\theta' + y_i \sin\theta$

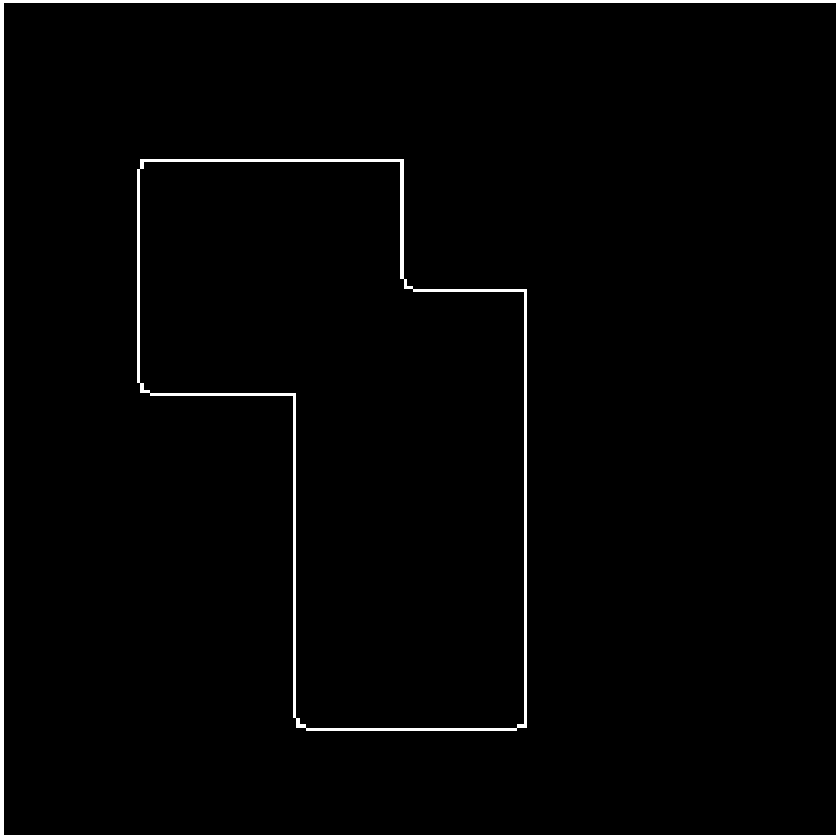
Accumulator:
$A(\rho, \theta) = A(\rho, \theta) + 1$

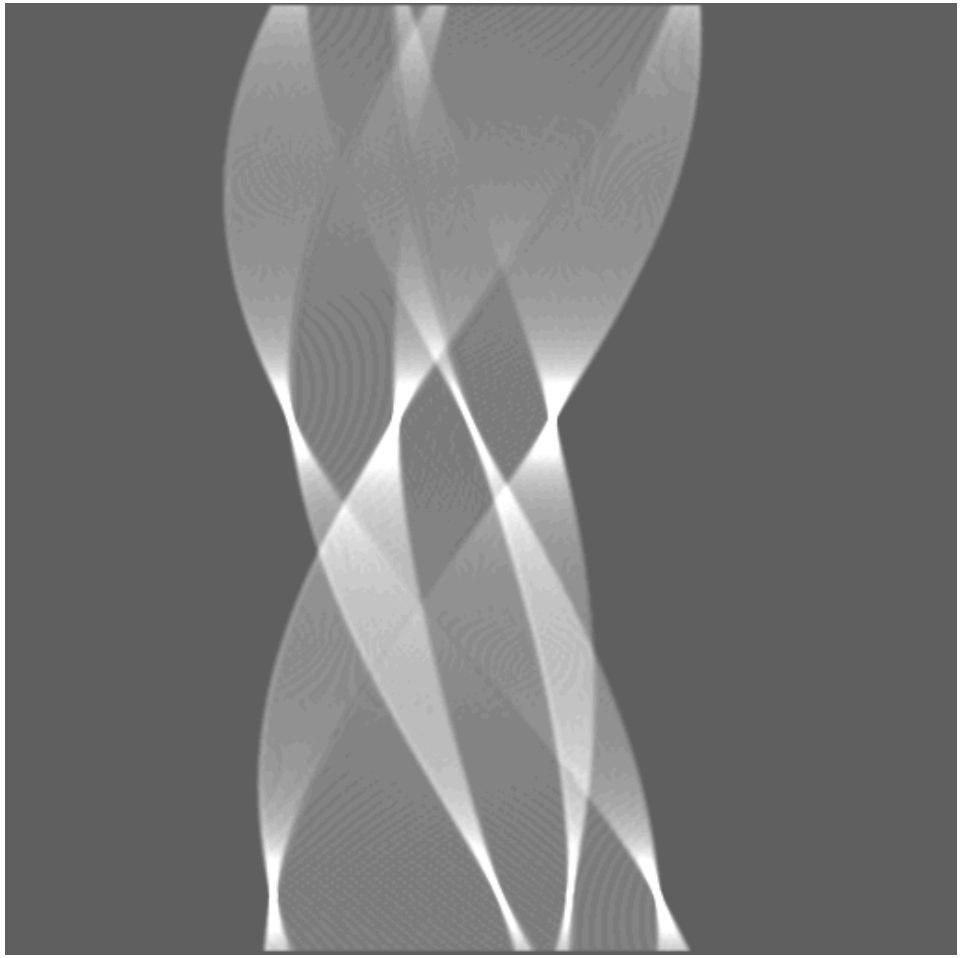


Threshold the accumulator values to get parameters for detected lines

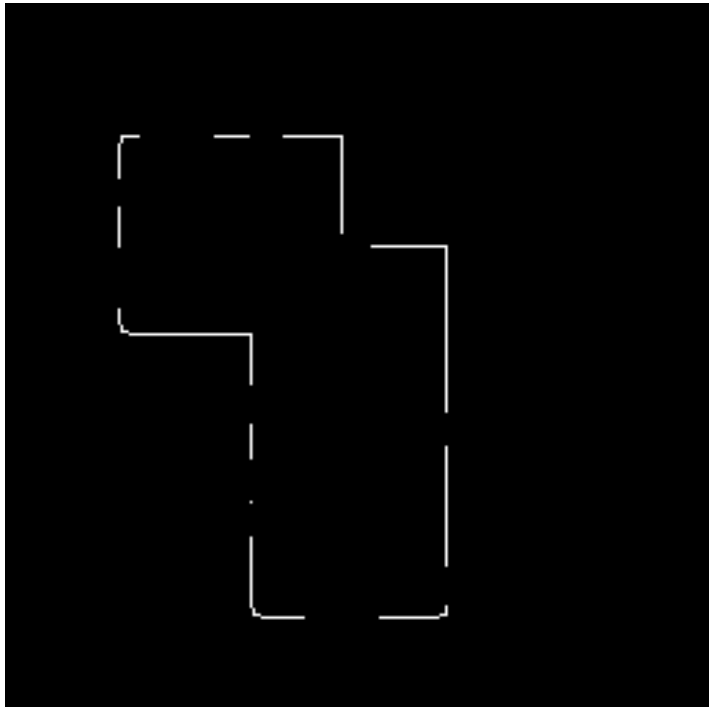
Threshold at $A(\rho, \theta) = 30$
Threshold at $A(\rho, \theta) = 20$
Threshold at $A(\rho, \theta) = 15$

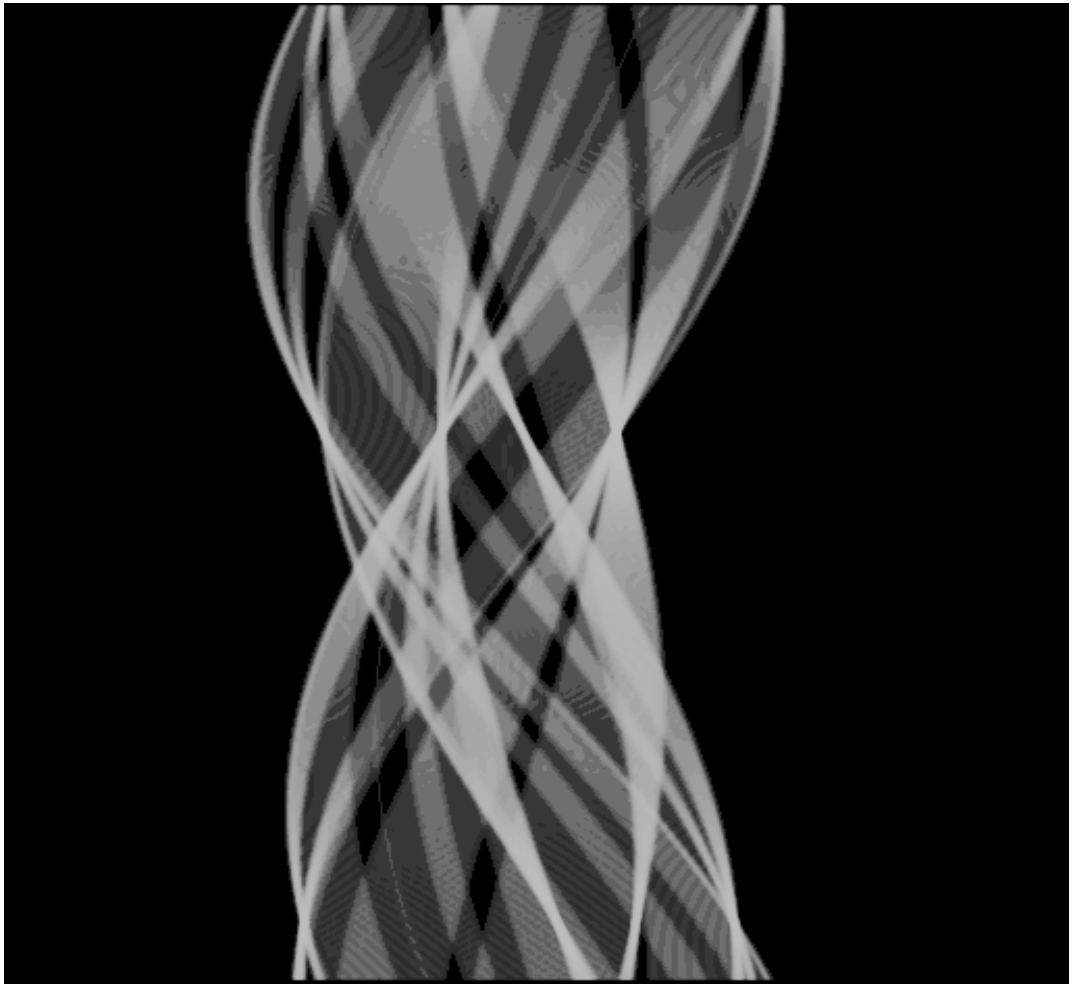
Example



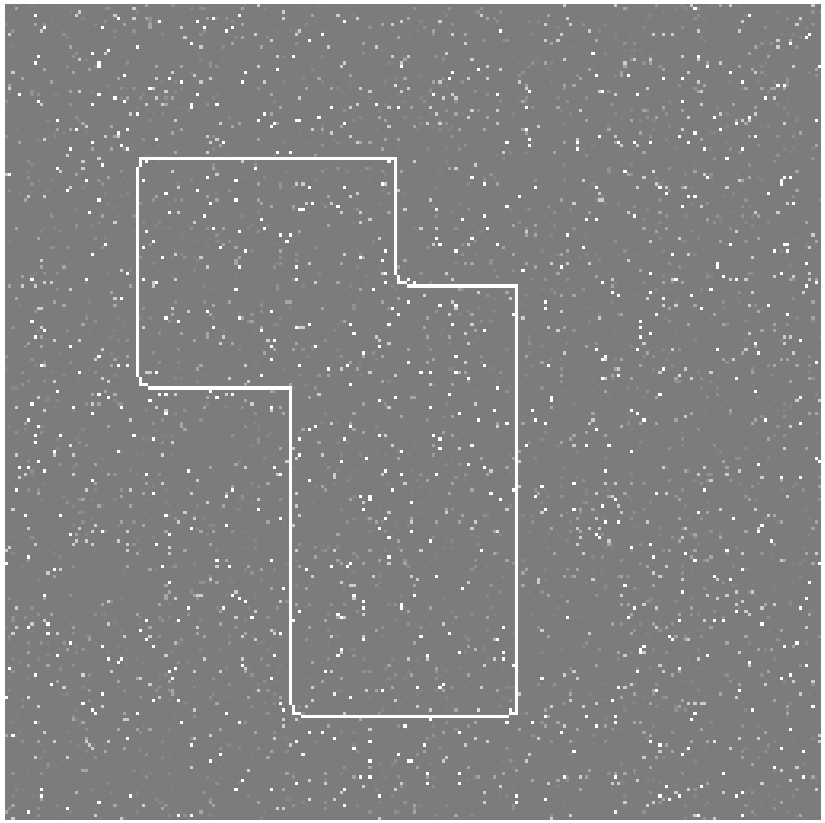


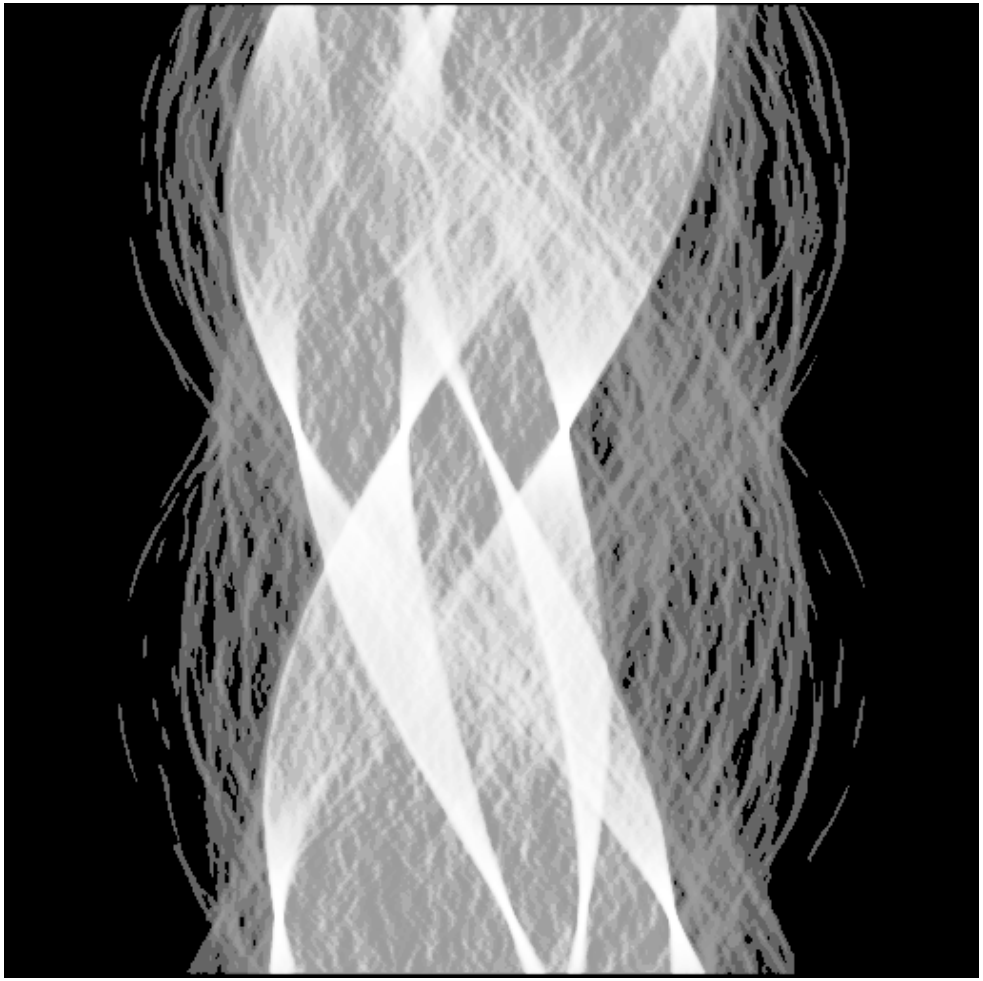
Hough Transform Example with breaks in lines

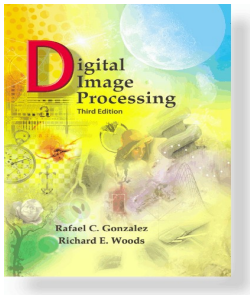




Hough Transform Example with 1% salt and pepper noise

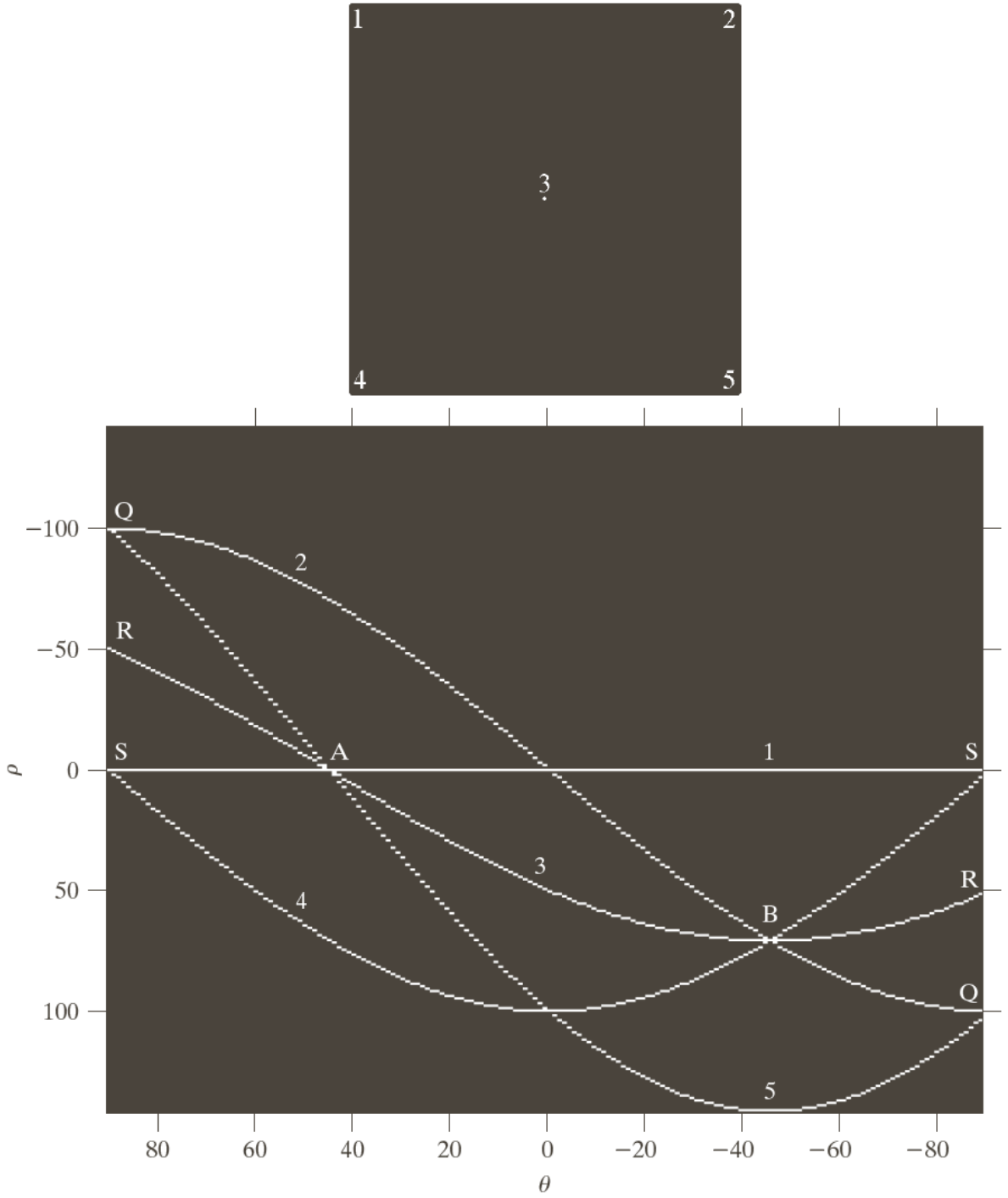


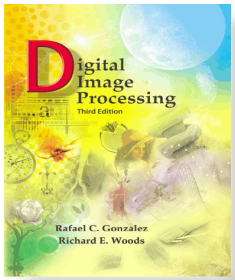




a
b

FIGURE 10.33
(a) Image of size 101 × 101 pixels, containing five points.
(b) Corresponding parameter space.
(The points in (a) were enlarged to make them easier to see.)





Chapter 10

Segmentation

■ **Figure** 10.33 illustrates the Hough transform based on Eq. (10.2-38). **Figure 10.33**(a) **shows** an image of size 101 × 101 pixels with five labeled **points, and** Fig. 10.33(b) shows each of these points mapped onto the **$\rho\theta$-plane using** subdivisions of one unit for the ρ and θ axes. The range of θ **values is** $\pm 90°$, and the range of the ρ axis is $\pm\sqrt{2}D$, where D is the dis**tance between** corners in the image. As Fig. 10.33(c) shows, each curve has **a different sinusoidal** shape. The horizontal line resulting from the map**ping of point** 1 is a special case of a sinusoid with zero amplitude.

Hough transform for circles

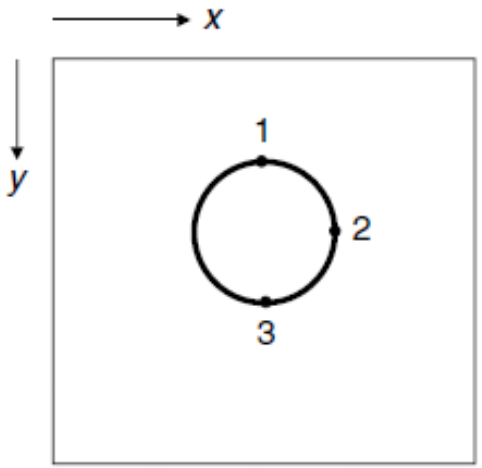
Hough transform for circles

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

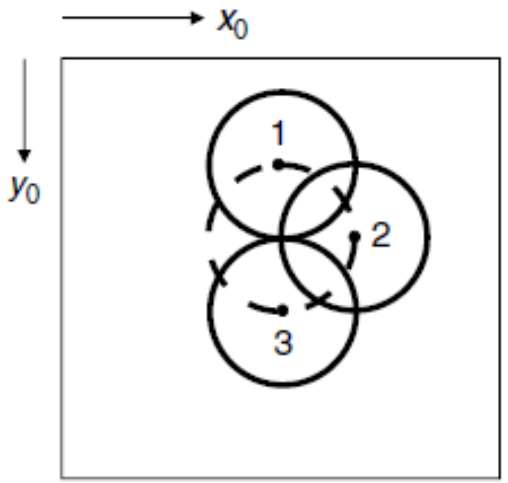
$$x = x_0 + r\cos(\theta) \quad y = y_0 + r\sin(\theta)$$

HT mapping is defined by

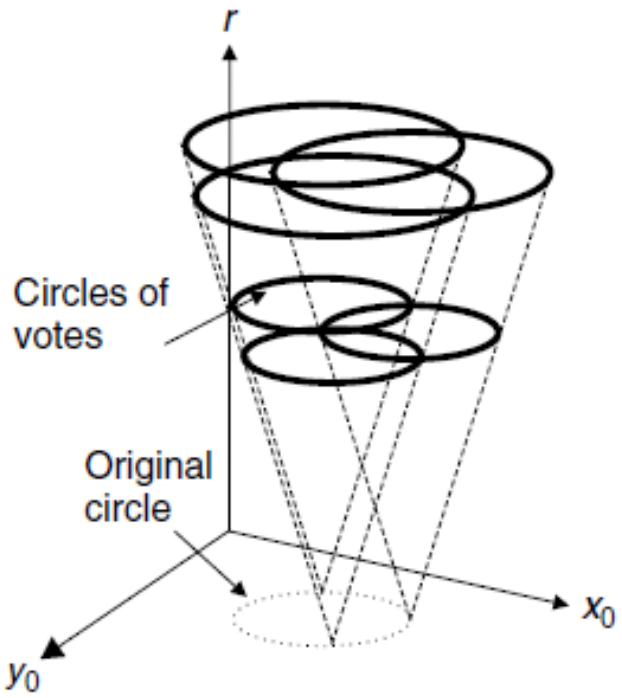
$$x_0 = x - r\cos(\theta) \quad y_0 = y - r\sin(\theta)$$



(a) Image containing a circle



(b) Accumulator space



(c) 3D accumulator space