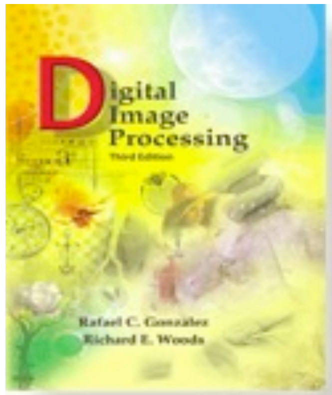


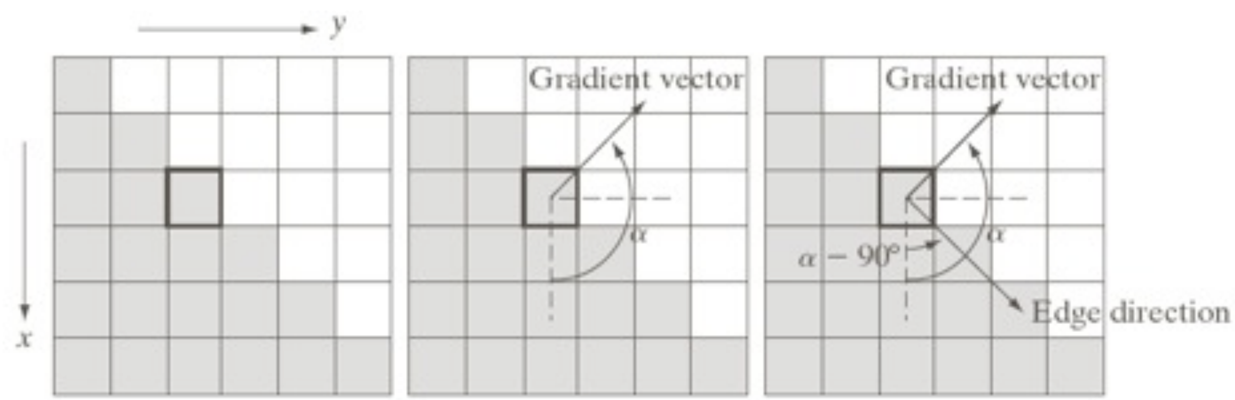
Last lecture:

•Finding edges using the gradient of an image and the laplacian.
•Edge models (ramp, roof, …) and what happens when they are contaminated with noise.

Edges and gradients:

The gradient points in the direction of maximal increase in the intensity.

=> The edge is orthogonal to the direction of the gradient.



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.

$$\nabla f = \text{grad}(f) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

−1
1

−1	1

a b

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

masks for vertical/horizontal edges

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Roberts

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−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

These masks do not have a “center pixel” and are less practical to apply

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−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

These masks do not have a "center pixel" and are less practical to apply

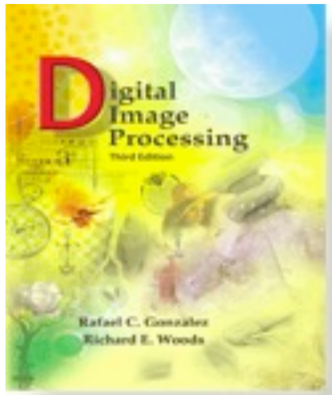
For diagonal edges

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



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−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

These masks do not have a "center pixel" and are less practical to apply

NB.
Prewitt masks are easier to implement

Sobel are somehow more robust wrt noise

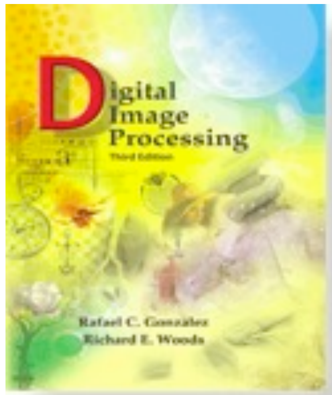
For diagonal edges

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

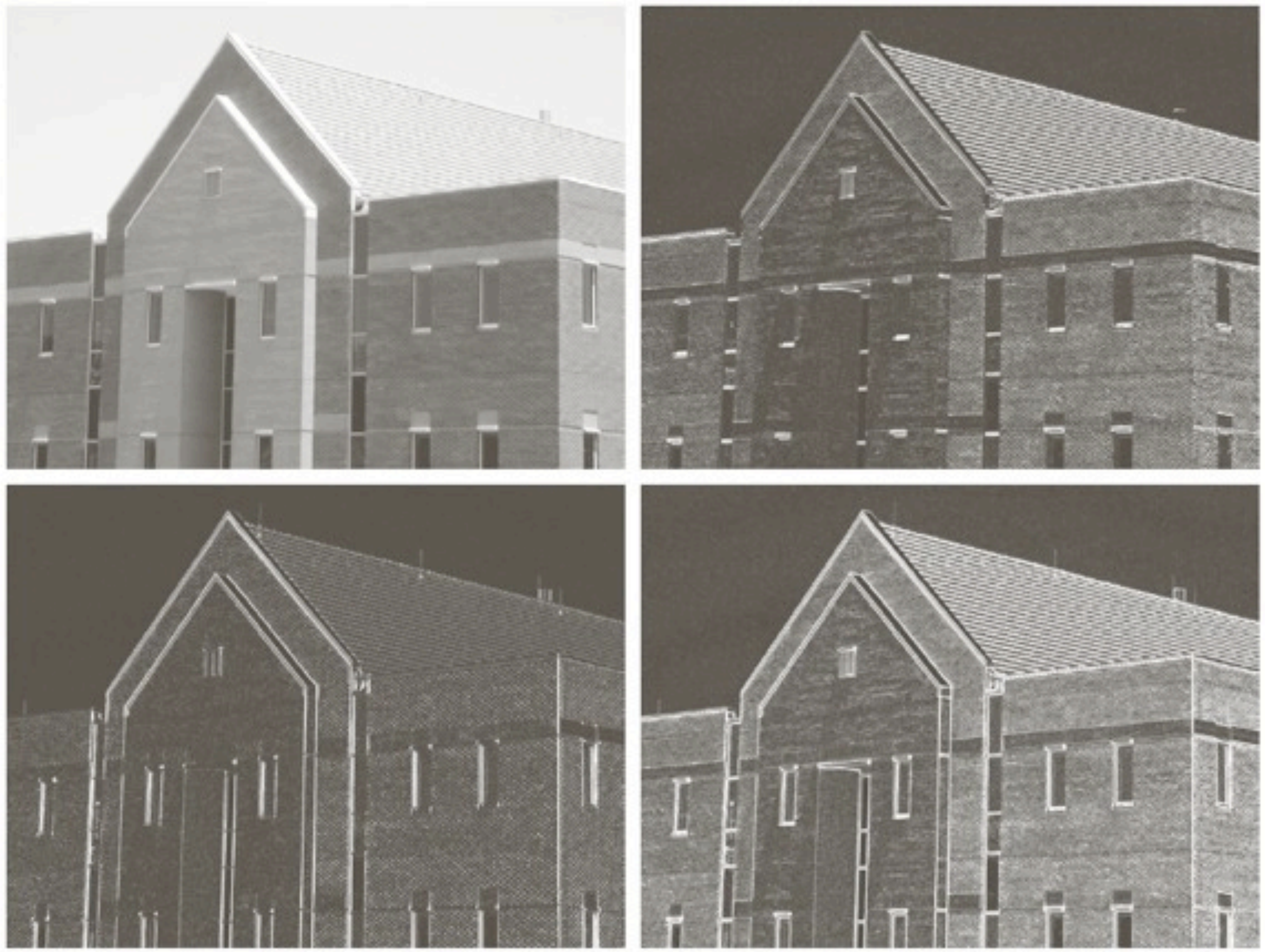


From the gradient components, we can estimate strength (magnitude) and direction.

Strength:

$$M(x,y) = \|\nabla f\| = \sqrt{f_x^2 + f_y^2} \qquad M(x,y) = |f_x| + |f_y|$$

Direction:
given that the gradient has direction $\alpha = \tan^{-1} \frac{f_y}{f_x}$
the edge will be
alpha +(-) 90°



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, $|g_x| + |g_y|$.

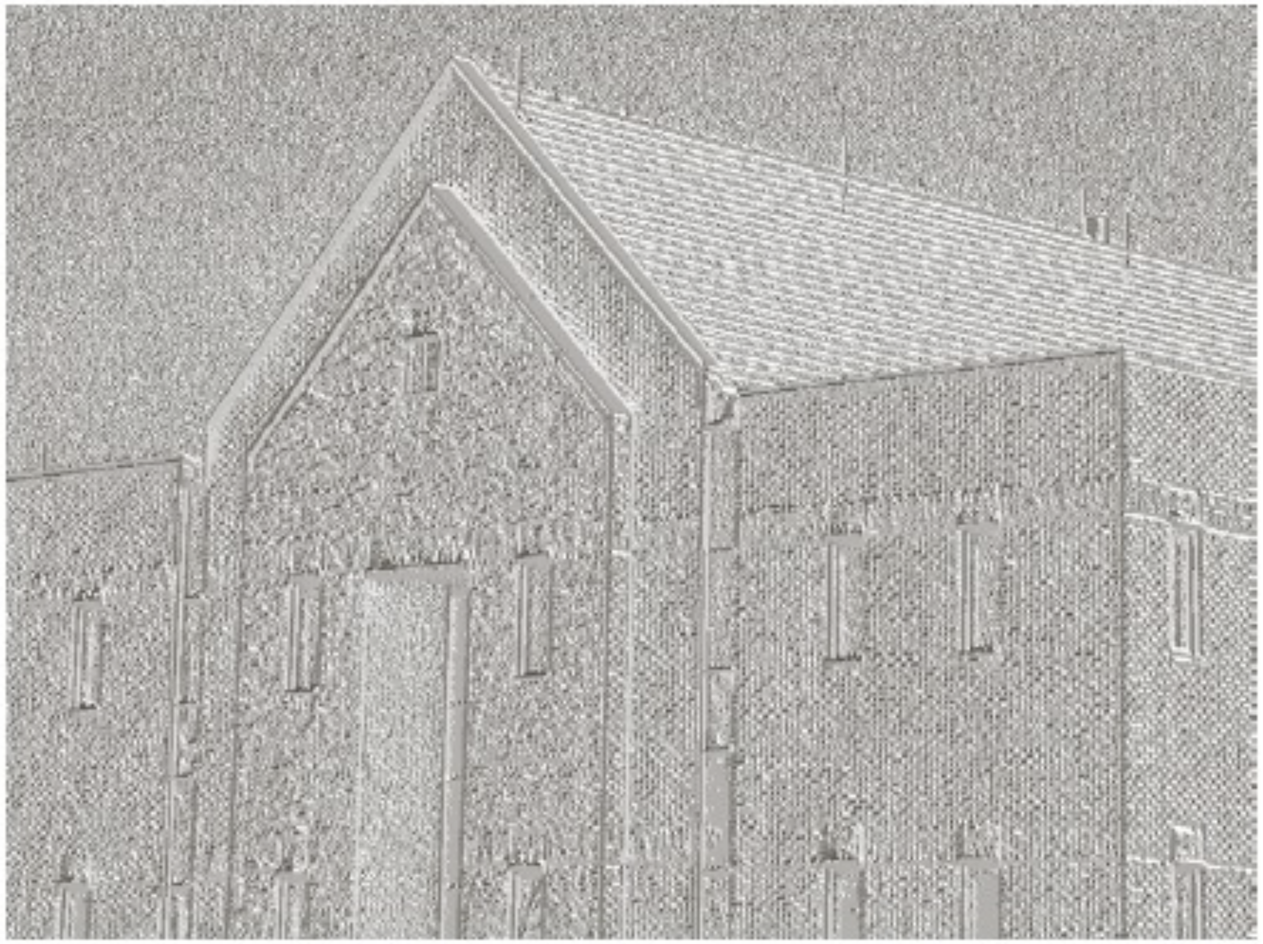
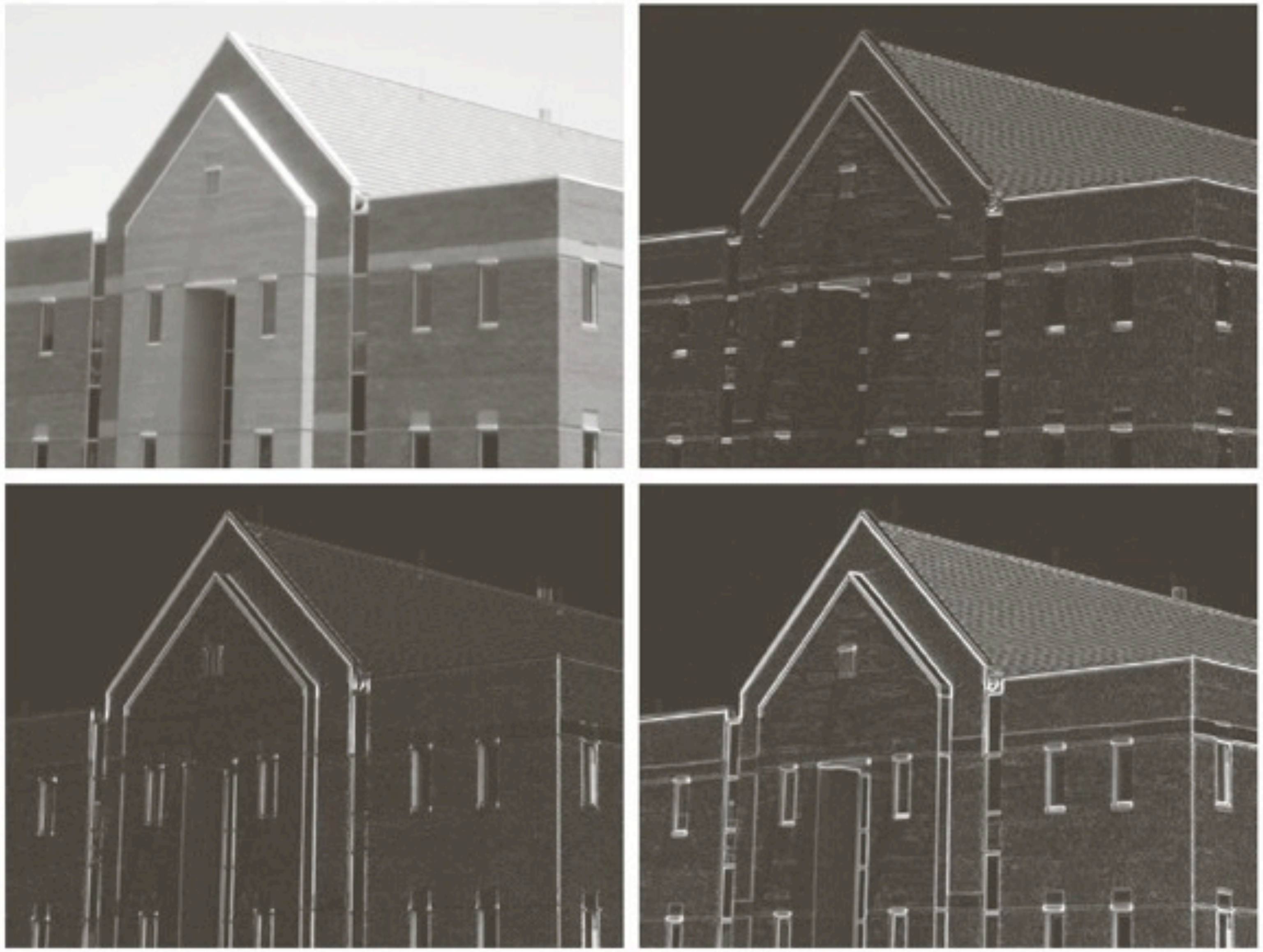
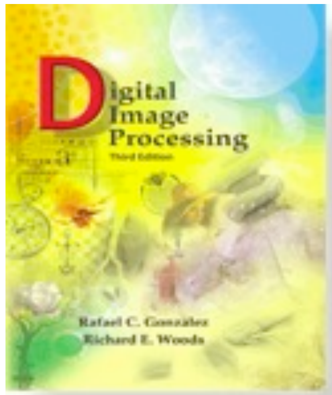


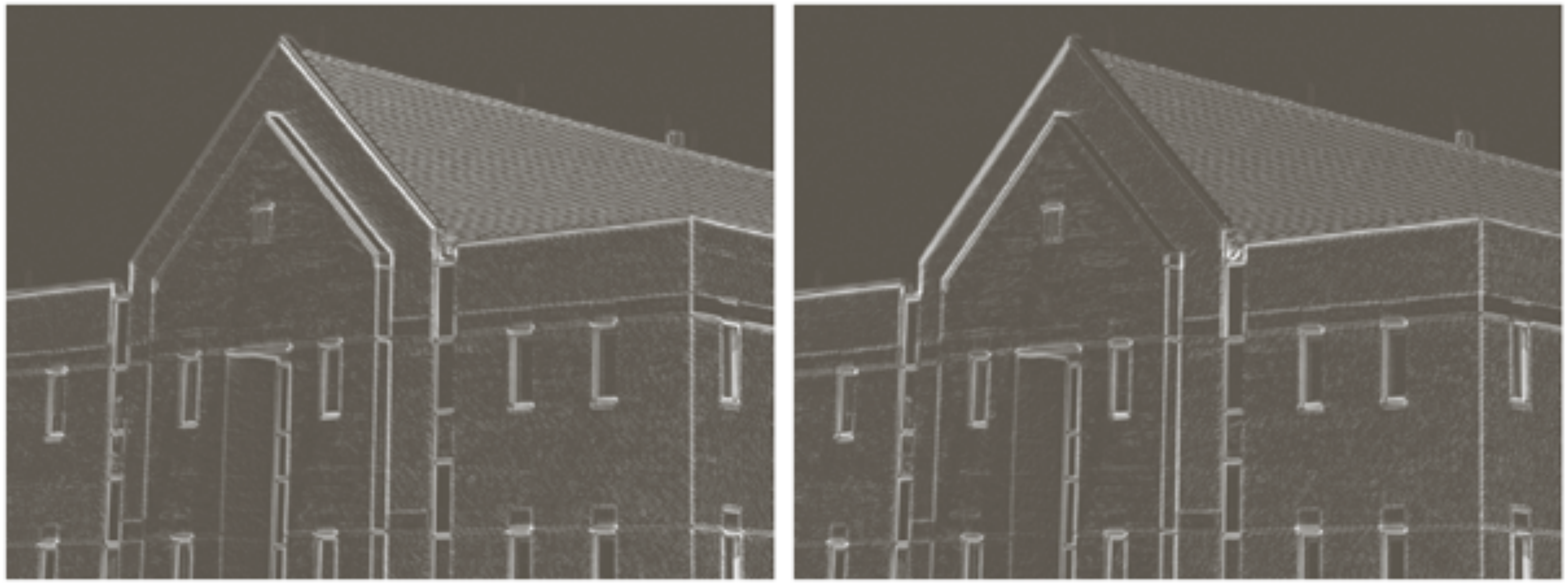
FIGURE 10.17 Gradient angle image computed using Eq. (10.2-11). Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.



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.





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).

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.

Marr-Hildreth (1980s) **edge detector (Laplacian of Gaussian)**

Before the 80s, edge detection was done using mainly operator with small masks (Sobel, Roberts, Prewitt,…)

Main observations that lay to the basis of MH approach:

1) Intensity changes are not independent of the image scale.
Hence intensity changes requires operators of different sizes.

2) A sudden intensity change gives raise to a peak in first derivative and a sign change in the second derivative

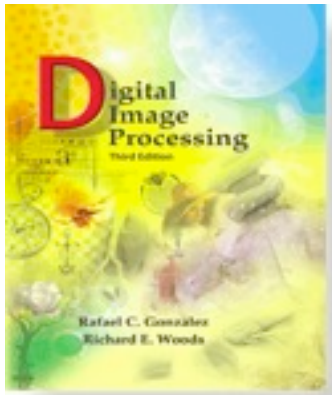
Main observations that lay to the basis of MH approach:

1) Intensity changes are not independent of the image scale. Hence intensity changes requires operators of different sizes.

2) A sudden intensity change gives raise to a peak in first derivative and a sign change in the second derivative.

Observation 2) suggests that the edge operator should be a *differential* operator, that can approximate ∂ and ∂^2

Observation 1) suggests that the operator should be able to act at different scales:
- large operators to detect blurry edges
- small operators to detect shaper/finer detail



Marr-Hildreth found that the most satisfactory operator fulfilling these requirements was the Laplacian of Gaussian

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

Derivation gives

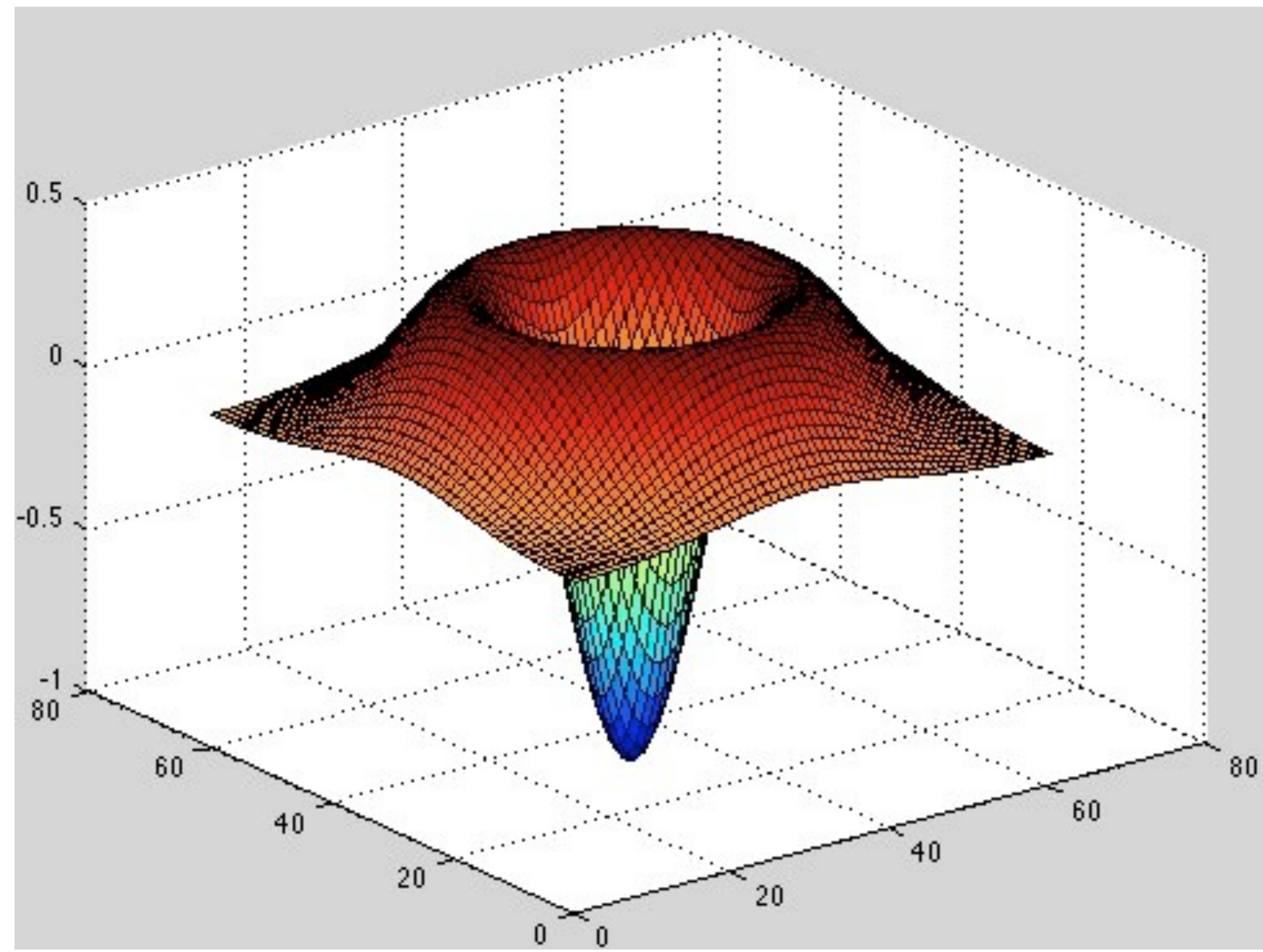
$$\nabla^2 G(x,y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

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

The change of sign happens when

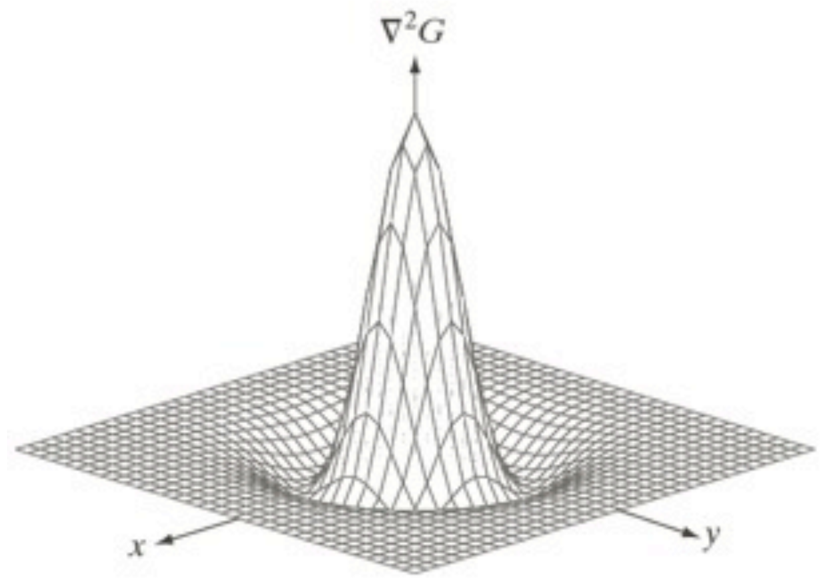
$$x^2 + y^2 = 2\sigma^2$$

defining a circle with radius $\sqrt{2}\sigma$



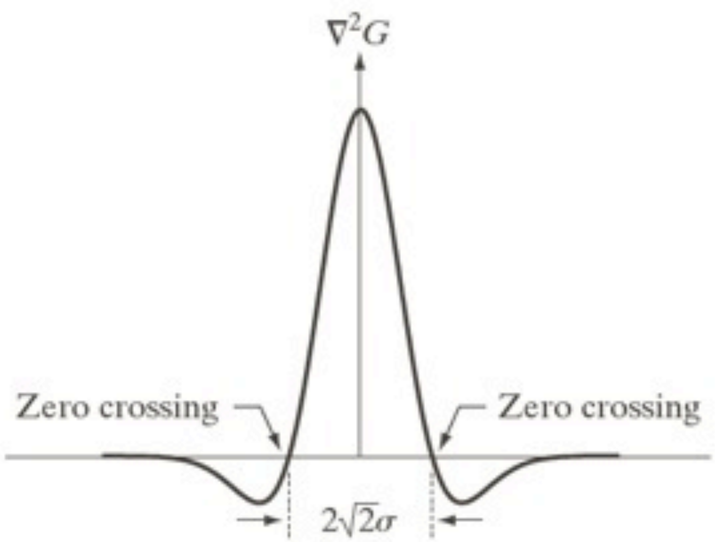
The operator can be then approximated by a mask of arbitray size, sampling the LoG function. The weights should be adjusted so that the sum of the weights is zero.

In the book, the negative of the LoG is displayed

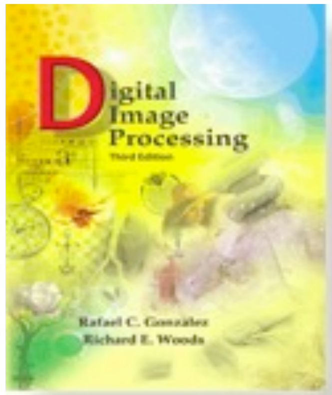


The mask is a suitable approximation of the LoG.

NB. The mask is used with a positive central weight



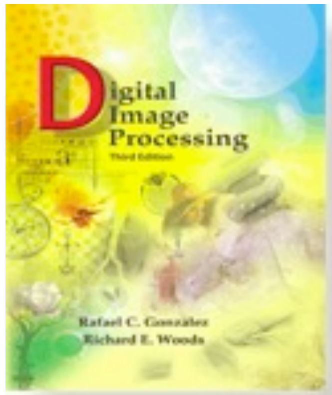
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0



Alternatively, one can

- sample the Gaussian function,
- convolve it with a Laplacian mask.

This convolution will guarantee that the sum of the weights is automatically zero.



How does the MH operator work on an image?

- The Gaussian part smoothes the image (combining the intensities at a scale/distance less than sigma, hence also the noise)
- The Laplacian detects the changes in intensity. The edges (can be double) are found by looking at the zero crossings.

- Why the Gaussian? The advantage of using the Gaussian is that the function is smooth both in spatial and frequency domain (the FT of a Gaussian is a Gaussian).

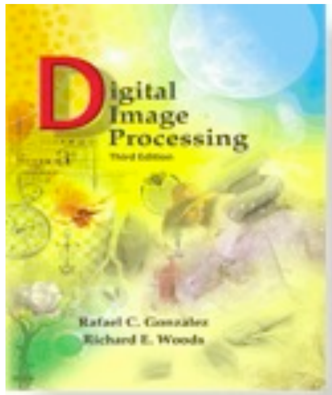
- Why the Laplacian? The Laplacian has the advantage of being isotropic, hence detects edges in all directions.

Marr-Hildreth algorithm:

- Convolve LoG with image $g(x,y) = \nabla^2 G(x,y) \star f(x,y)$
- Find zero crossing to determine the location of the egdes

Since differentiation is a diagonal operator in Fourier space, the Laplacian and convolution commute:

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

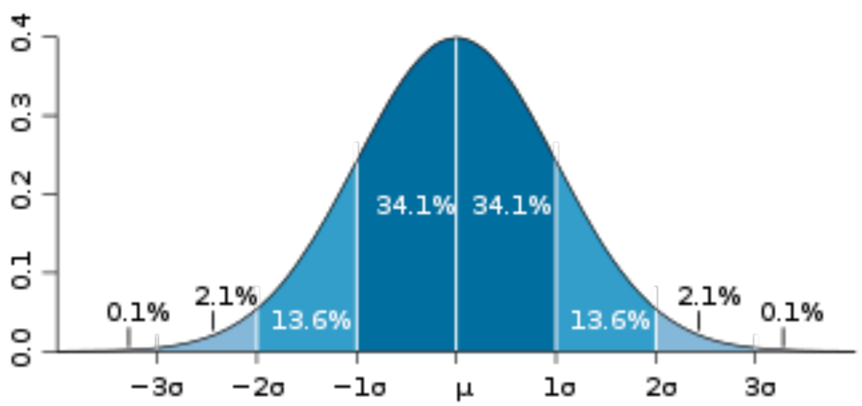


So the algorithm can also be summarized as follows:

1. Filter *f(x,y)* with a lowpass Gaussian filter mask
   (in Fourier space or by convolution by a Gaussian filter mask)
2. Compute the Laplacian (f.ex. by convolving with a Laplacian filter mask)
3. Find the zero-crossing to find the edges.

How to specify the width of the Gaussian filter?
99.7% of the volume lies between
-3sigma and 3sigma.

So the width *n* of the filter should be the
smallest integer ≥ 6sigma



Recall: The degree of truncation is inversely proportional to the size of the mask

How to find the zero crossing?

Look at a 3x3 neighborhood of a pixel p.
Signs of at least 2 opposing neighbors should be opposite

Cases:
- Left/Right
- Up/Down
- Diag +
- Diag -

1	2	-2
1	2	-2
3	2	1

Edge point

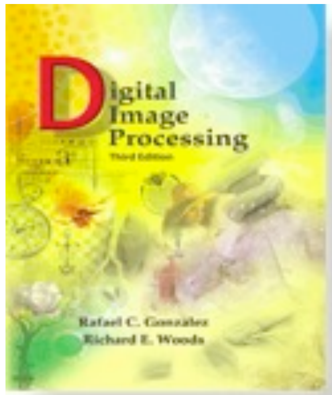
1	2	-2
1	2	2
3	2	1

Not edge point

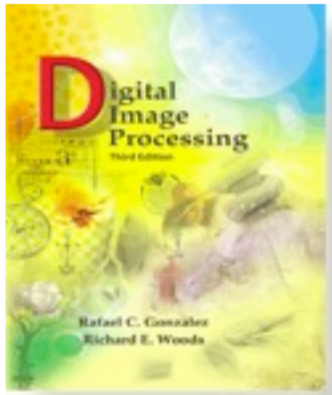
In addition, one might require that the changes should be higher than a certain threshold value.

In matlab (edge command)

```
Laplacian of Gaussian Method
    ----------------------------
    BW = EDGE(I,'log') specifies the Laplacian of Gaussian method.

    BW = EDGE(I,'log',THRESH) specifies the sensitivity threshold for the
    Laplacian of Gaussian method. EDGE ignores all edges that are not
    stronger than THRESH. If you do not specify THRESH, or if THRESH is
    empty ([]), EDGE chooses the value automatically.

    BW = EDGE(I,'log',THRESH,SIGMA) specifies the Laplacian of Gaussian
    method, using SIGMA as the standard deviation of the LoG filter. The
    default SIGMA is 2; the size of the filter is N-by-N, where
    N=CEIL(SIGMA*3)*2+1.

    [BW,thresh] = EDGE(I,'log',...) returns the threshold value.
```

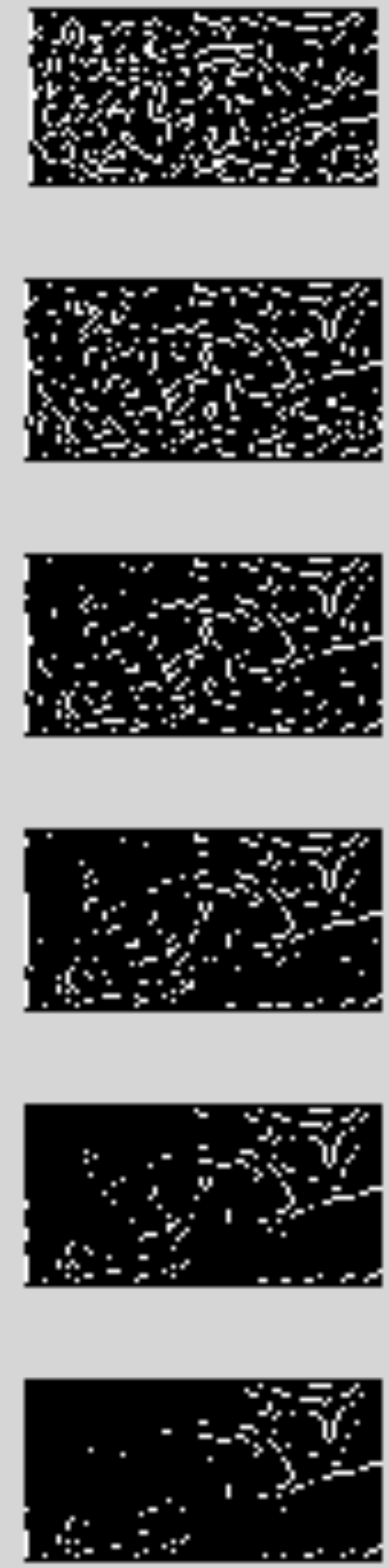


LoG edge detector
automatic threshold
(sigma=2)



LoG edge detector automatic threshold (sigma=2)

Results with different thresholds: T=0.5*k, k=1:6



Results with different thresholds: T=0.5*k, k=1:6

LoG edge detector automatic threshold (sigma=2)

T=0.5

T=0.4

sigma =3

It is possible to approximate the LoG by the difference of 2 Gaussians (DoG):

$$\frac{1}{2\pi\sigma_1^2}e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2}e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

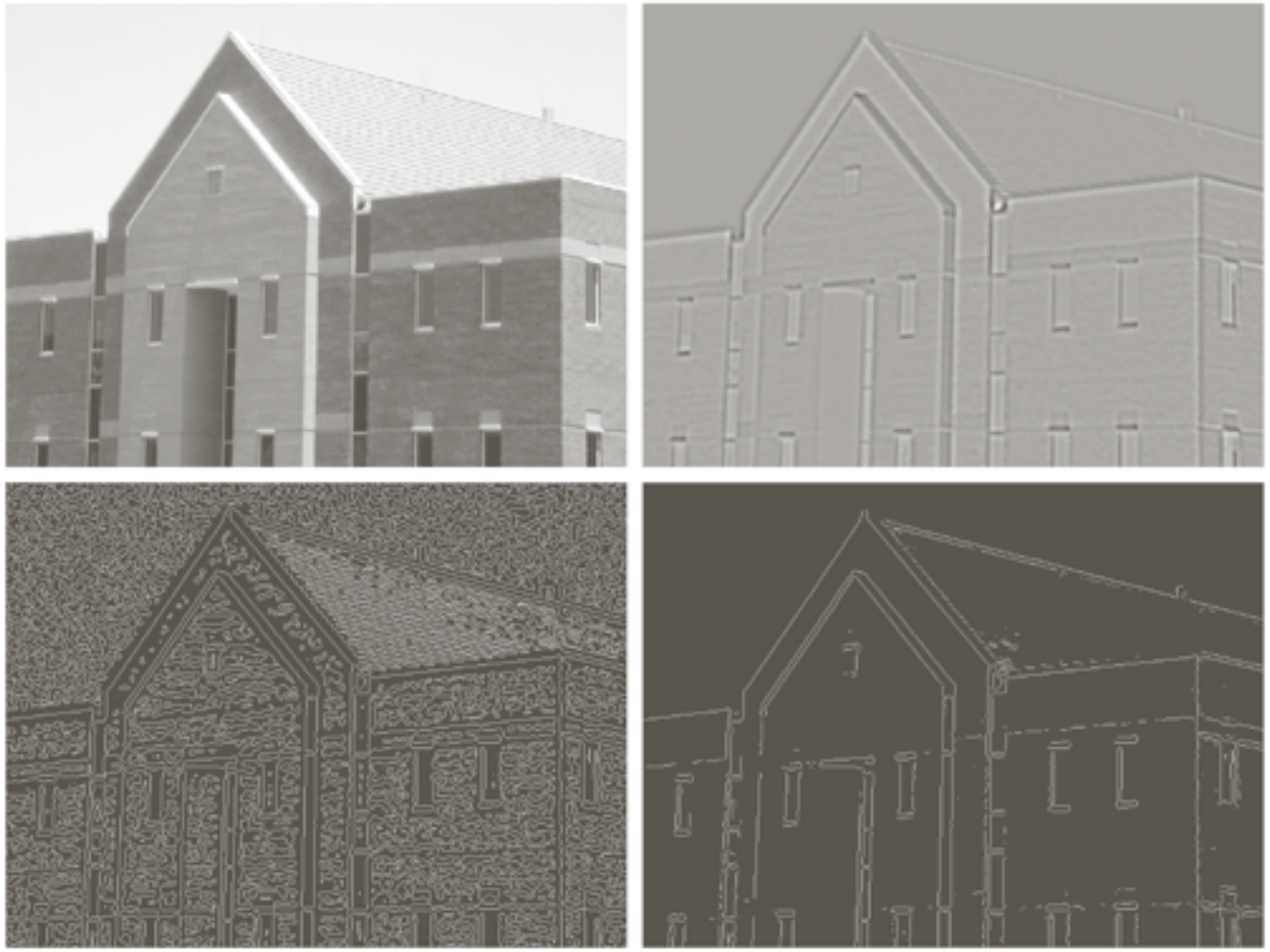
The sigma-values should be chosen so that the DoG has the same 0-crossing as LoG:

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2}\ln(\sigma_1^2/\sigma_2^2)$$

Moreover, as the amplitudes can be different, they should be scaled so that the two curves have the same amplitude at the origin.

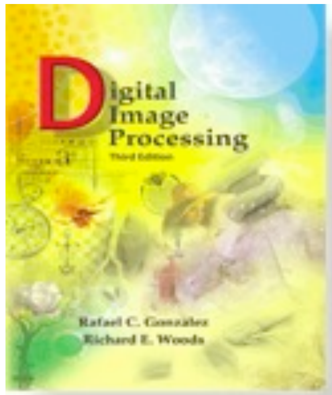
a b

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.



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.



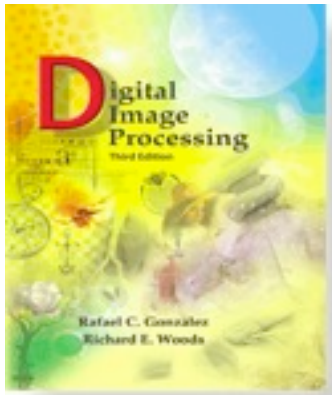
Canny edge detector

Objectives:

- *Low error rate*: no spurious edges

- *Edge points well localized*: The distance between a point marked as an edge and the true edge should be a minimum.

- *Single edge point response*: The edge detector returns only one point for each true edge point.

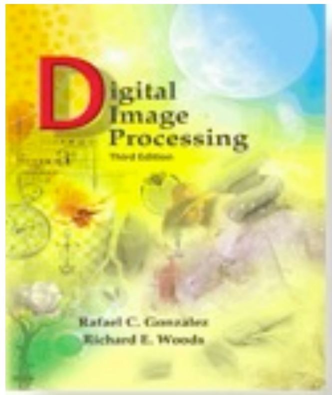


A good approximation to the *optimal* edge detector: first derivative of a Gaussian.

However, since the derivative is a *directional* operator, this would imply to apply a 1D edge detector in all directions:

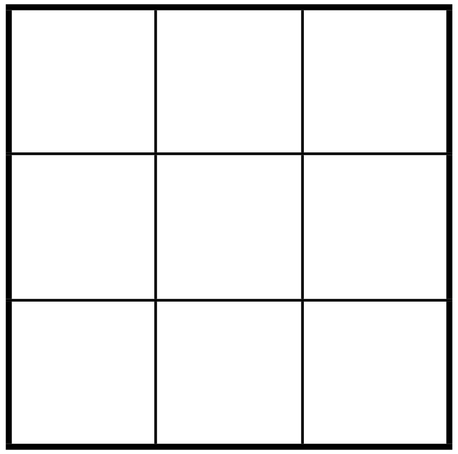
Solution: use a 2D Gaussian, compute the gradient of the result, then analyze magnitude and angle at every point.

$$g(x,y) = G(x,y) \star f(x,y), \qquad M(x,y) = \sqrt{g_x^2 + g_y^2}, \qquad \alpha(x,y) = \tan^{-1}\frac{g_y}{g_x}$$



Since the gradient returns thick edges, the edges are thinned by a method called *suppresion of non-maxima* (3x3 region):

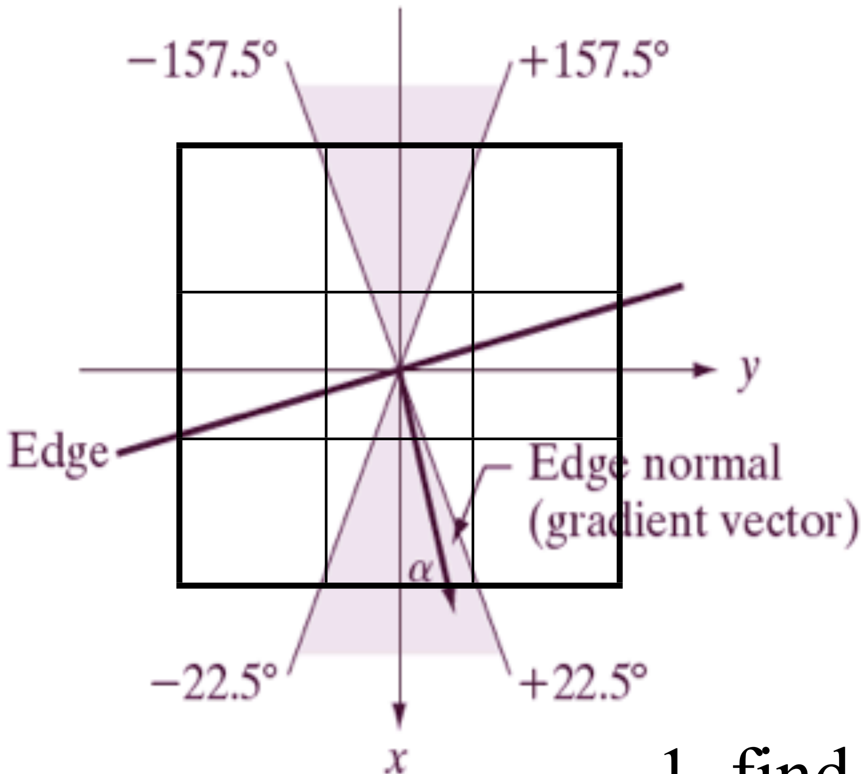
look at a discrete set of orientations, d1,d2,d3,d4, (v,h,+/-45º)



1. find the direction dk closest to alpha
2. Find M(x,y) less than at least one of its neigh. points on the direction dk, set gN(x,y)=0; otherwise gN(x,y) = M(x,y)
(gN non-maxima suppressed image)

Since the gradient returns thick edges, the edges are thinned by a method called *suppresion of non-maxima* (3x3 region):

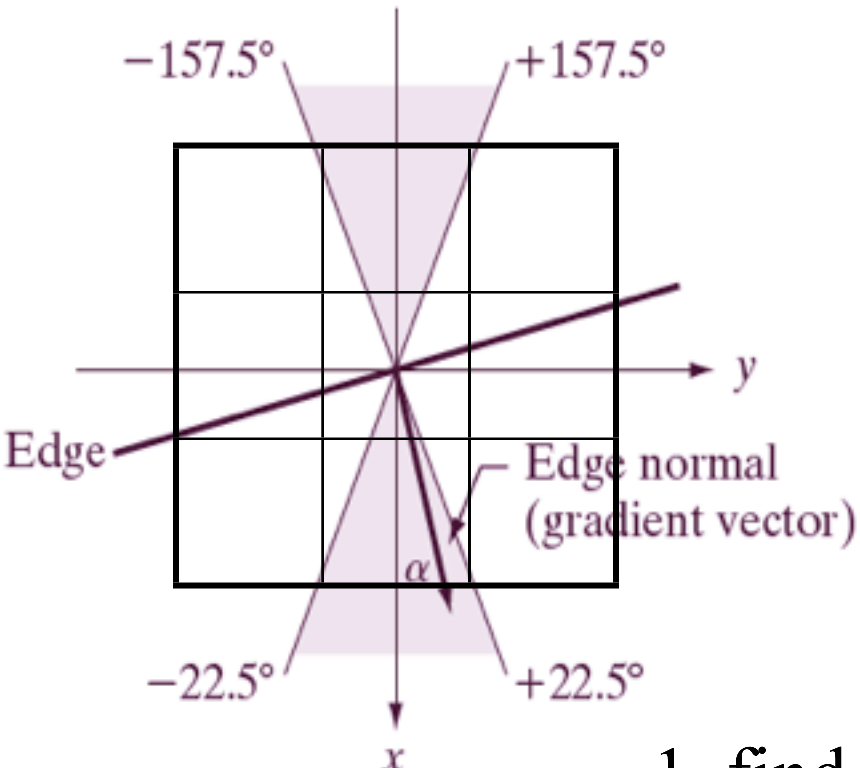
look at a discrete set of orientations, d1,d2,d3,d4, (v,h,+/-45º)

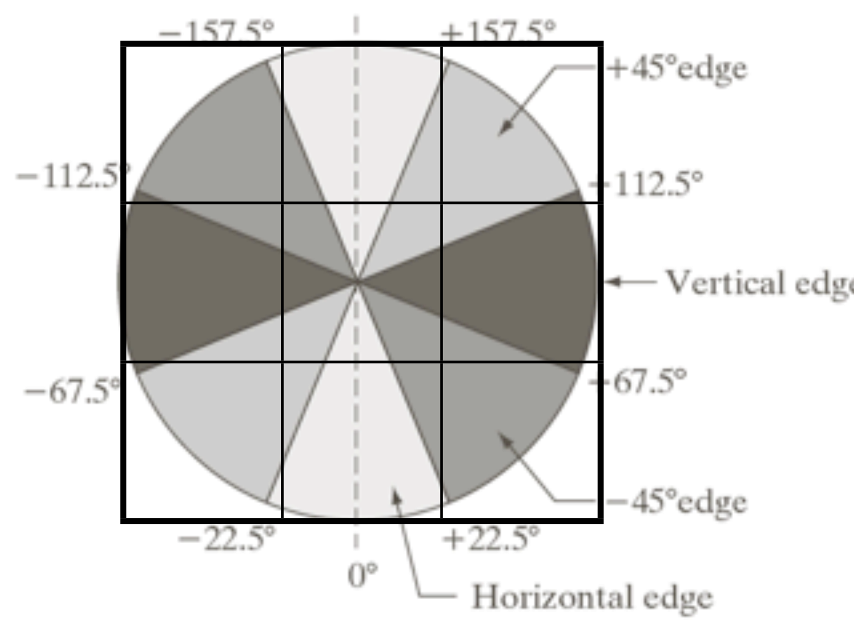


1. find the direction dk closest to alpha
2. Find M(x,y) less than at least one of its neigh. points on the direction dk, set gN(x,y)=0; otherwise gN(x,y) = M(x,y)
(gN non-maxima suppressed image)

Since the gradient returns thick edges, the edges are thinned by a method called *suppresion of non-maxima* (3x3 region):

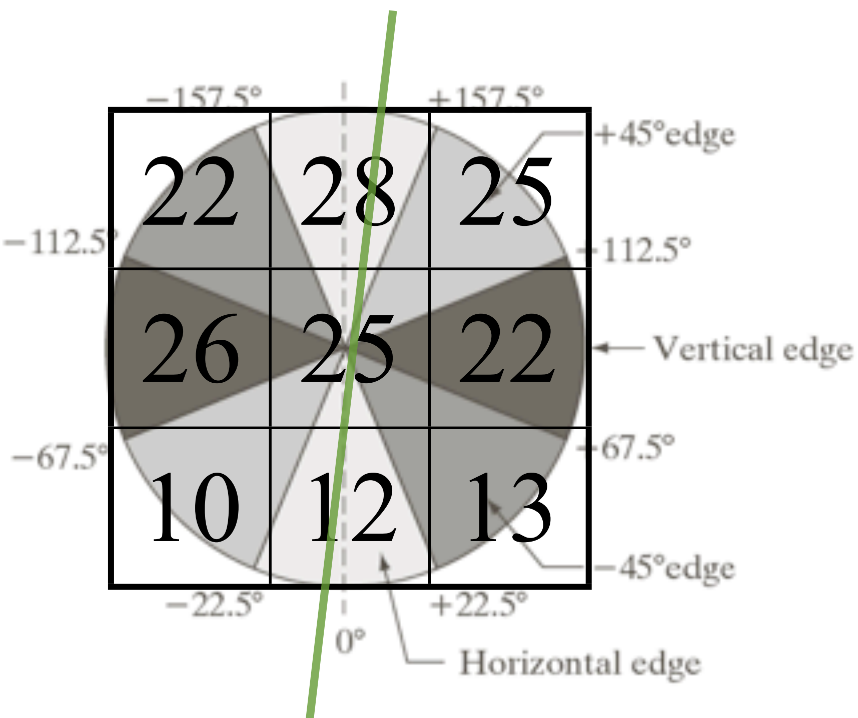
look at a discrete set of orientations, d1,d2,d3,d4, (v,h,+/-45º)





1. find the direction dk closest to alpha
2. Find M(x,y) less than at least one of its neigh. points on the direction dk, set gN(x,y)=0; otherwise gN(x,y) = M(x,y)

(gN non-maxima suppressed image)



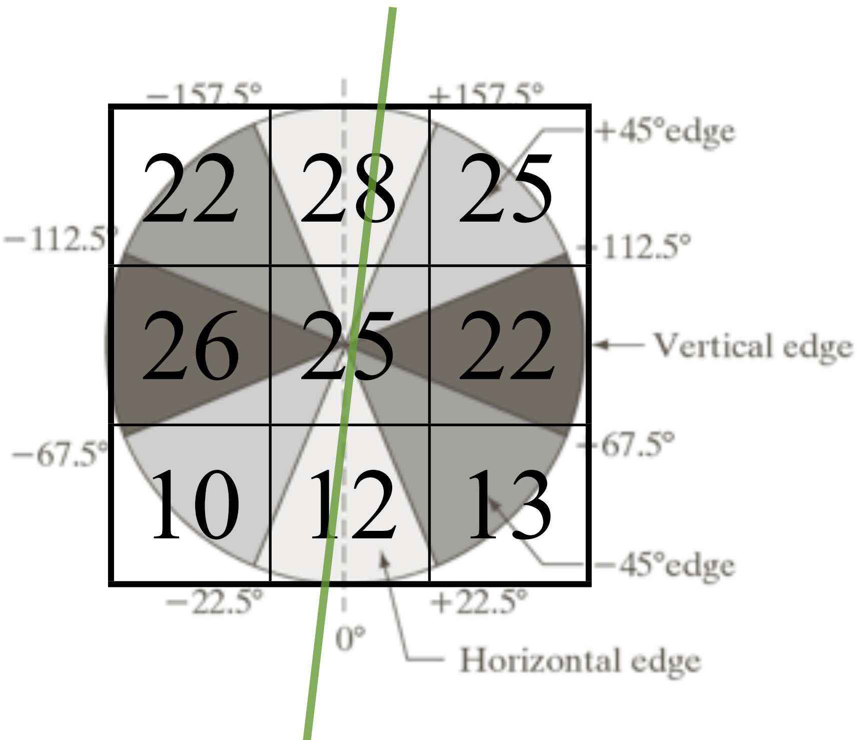
direction dk of the gradient

Horizontal edge.

We need to compare the magnitude of the gradient along the direction dk for the neighbouring pixels.

alpha=

0.19	1.55	0.48
-1.23	-0.03	1.23
4.57	0.09	2.38



direction dk of the gradient

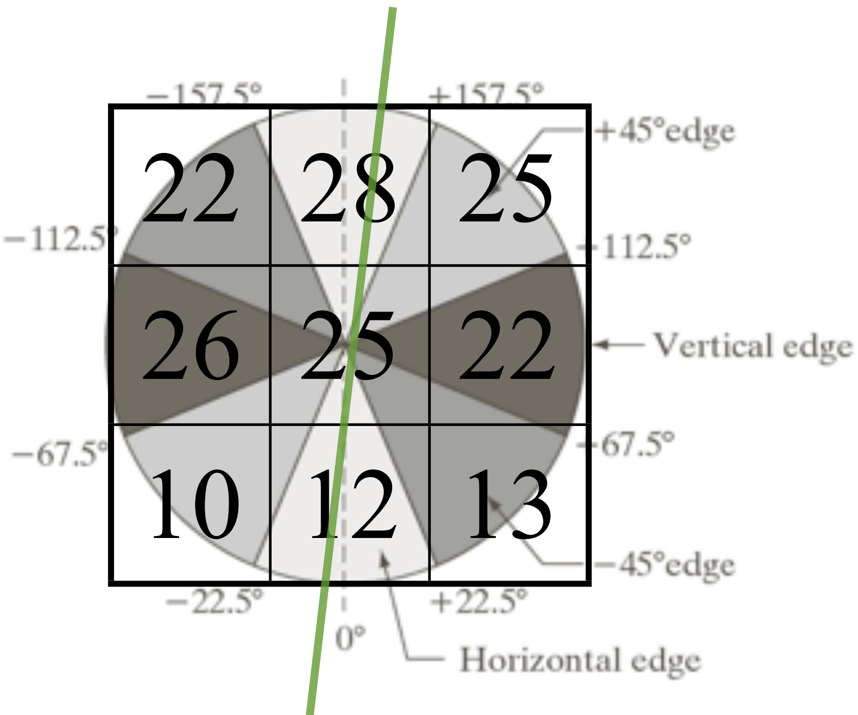
Horizontal edge.

We need to compare the magnitude of the gradient along the direction dk for the neighbouring pixels.

$$|f\_x|+|f\_y|= \begin{matrix} 114 & 10 & 118 \\ 130 & 58 & 130 \\ 106 & 56 & 92 \end{matrix}$$

alpha=

0.19	1.55	0.48
-1.23	-0.03	1.23
4.57	0.09	2.38



direction dk of the gradient

Horizontal edge.

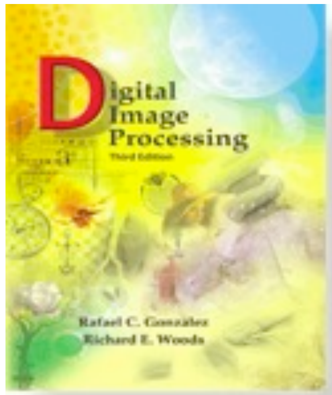
We need to compare the magnitude of the gradient along the direction dk for the neighbouring pixels.

alpha=

0.19	1.55	0.48
-1.23	-0.03	1.23
4.57	0.09	2.38

$$|f_x|+|f_y|= \begin{matrix} 114 & 10 & 118 \\ 130 & 58 & 130 \\ 106 & 56 & 92 \end{matrix}$$

58> 10, 56, hence the central point is a candidate for the edge (otherwise, we would have set it to 0)



Finally: threshold the maxima-suppressed image to reduce false edge points (false positive).

too high threshold removes too many edges (false negatives)
too low threshold includes false edges (false positives)

Use 2 thresholds (high and low): strong edges

gNH(x,y) = gN(x,y) >= TH
gNL(x,y) = gN(x,y) >= TL

(binary images)

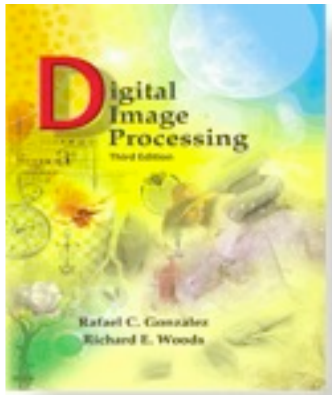
After thresholding, $gNH \subseteq gNL$

Remove gNH from gNL: $gNL = gNL - gNH$

After this step, gNL contains the *weak* edges.

The pixels in gNH contain the *strong* edges and are marked as edges immediately. They will typically have large gaps. Use now the information about weak edges to fill in some of the gaps and create longer edges.

This is done by visiting each valid pixel (those with 1) in gNH, and the corresponding pixels in gNL, adding pixels depending on the connectivity.



1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

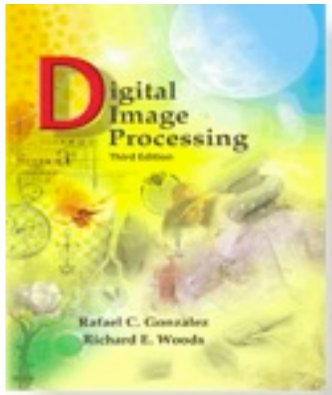
gNL

1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

gNL

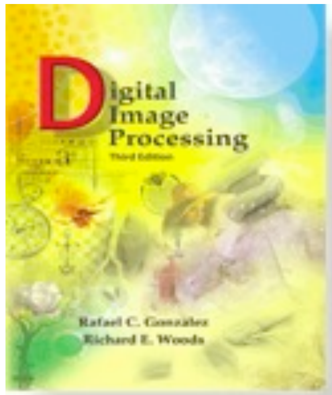


1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

gNL



1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

gNL

1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

gNL

1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1

gNH

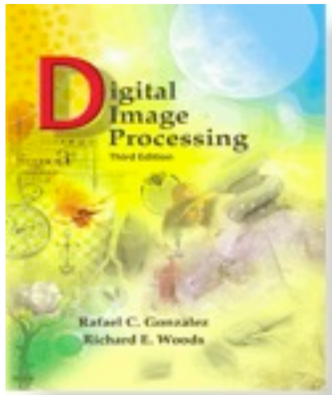
0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	0	0

gNL

Using 8-connectivity we add two extra edge points to the image

Summary: Canny edge detector:

1. Smooth the image by Gaussian filter
2. Compute gradient magnitude and angle
3. Apply nonmaxima suppression to the magnitude
4. Apply double thresholding and connectivity analysis to detect edges and link them.



In matlab (edge command)

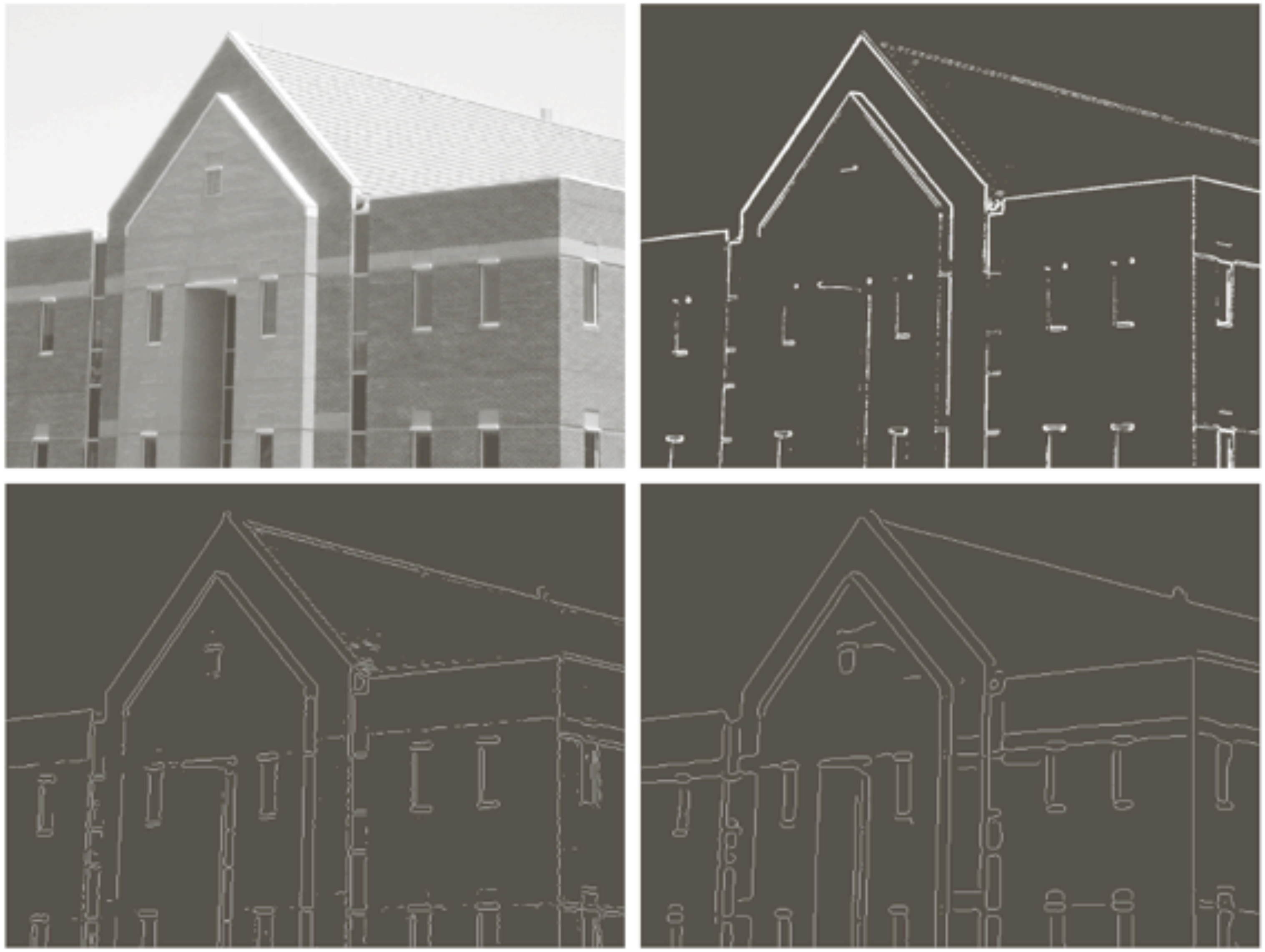
```
Canny Method
    ----------------------------
    BW = EDGE(I,'canny') specifies the Canny method.

    BW = EDGE(I,'canny',THRESH) specifies sensitivity thresholds for the
    Canny method. THRESH is a two-element vector in which the first element
    is the low threshold, and the second element is the high threshold. If
    you specify a scalar for THRESH, this value is used for the high
    threshold and 0.4*THRESH is used for the low threshold. If you do not
    specify THRESH, or if THRESH is empty ([]), EDGE chooses low and high
    values automatically.

    BW = EDGE(I,'canny',THRESH,SIGMA) specifies the Canny method, using
    SIGMA as the standard deviation of the Gaussian filter. The default
    SIGMA is 1; the size of the filter is chosen automatically, based
    on SIGMA.

    [BW,thresh] = EDGE(I,'canny',...) returns the threshold values as a
    two-element vector.
```

TH = 0.3

TH = 0.4

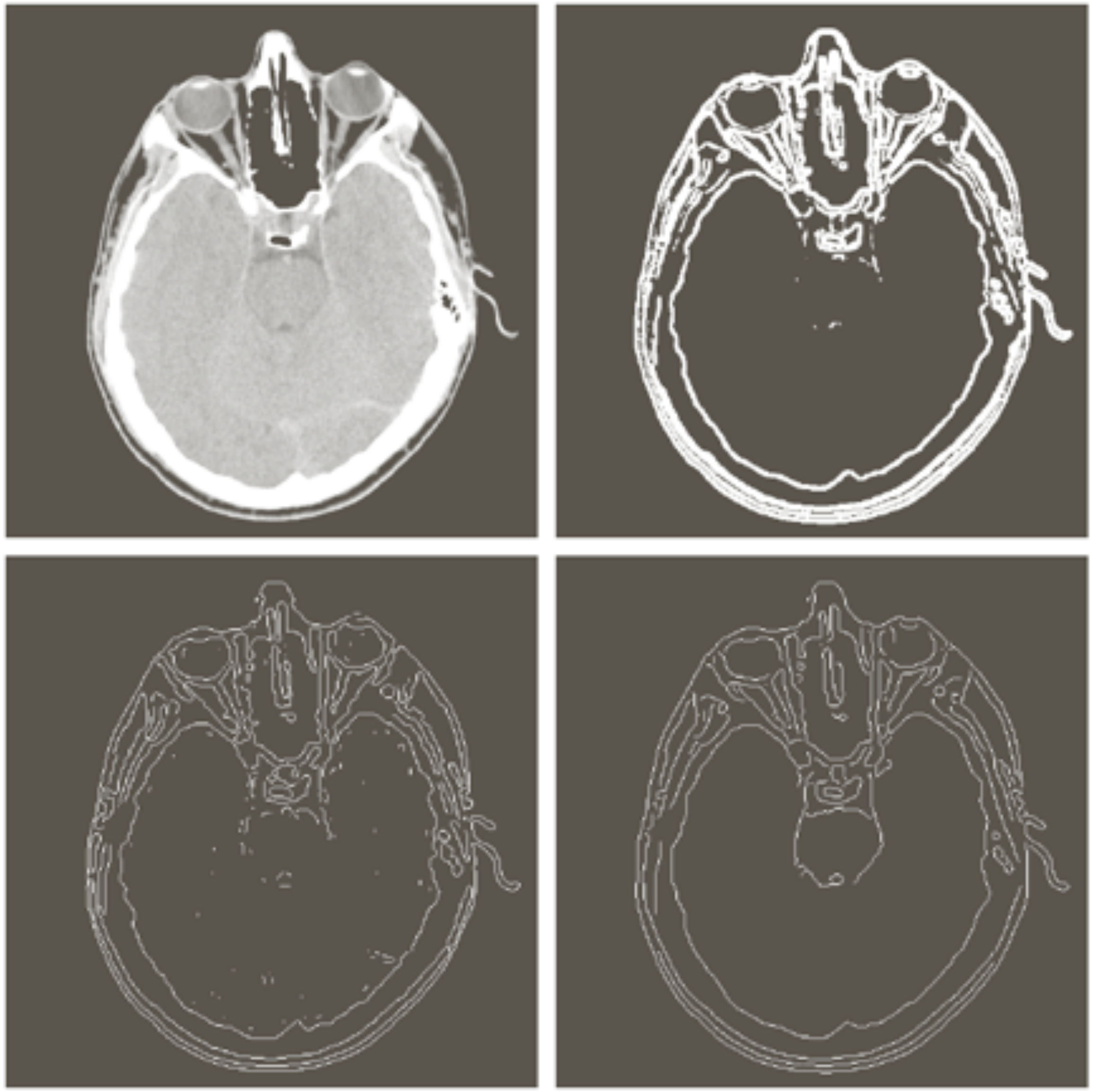
Canny threshold

thresh = 0.0750   0.1875



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.



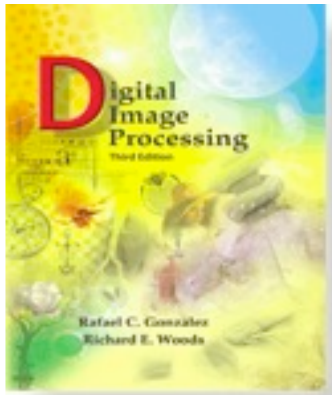
a b
c d

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.)

Linking edges

Local processing

Regional processing



Local processing:

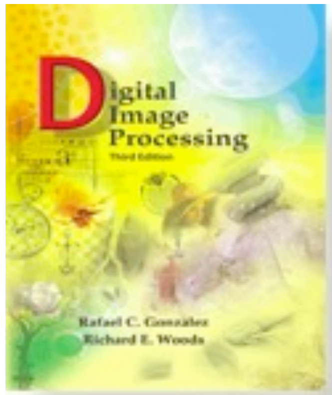
Analyze the pixels in a small neighborhood of every *(x, y)* for similarity.

1) strength of magnitude
2) direction of the gradient vector

A pixel *(s,t)* in *Sxy* has similar magnitude if: $|M(s,t) - M(x,y)| \leq E$

and similar angle if: $|\alpha(s,t) - \alpha(x,y)| \leq A$

(recall that the edge is orthogonal to the gradient)



Local processing:

Analyze the pixels in a small neighborhood of every *(x, y)* for similarity.

1) strength of magnitude
2) direction of the gradient vector

Two pixels are linked if they have similar magnitue and angle

A pixel *(s,t)* in *Sxy* has similar magnitude if: $|M(s,t) - M(x,y)| \leq E$

and similar angle if: $|\alpha(s,t) - \alpha(x,y)| \leq A$

(recall that the edge is orthogonal to the gradient)

The previous procedure can be costly because it has to analyze all the pixels.

Simplify by first thresholding on the magnitudes/angles of interest.

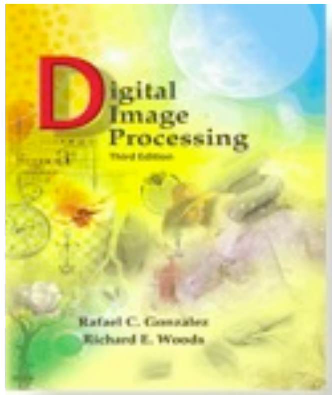
1. Compute M(x,y), alpha(x,y)
2. Threshold to get a binary mask:

$g(x,y)$ = 1 if M(x,y)> TM and alpha(x,y) = A+/- TA
$g(x,y)$ = 0 otherwise

3. Scan the rows of g and fill (horizontal) gaps by a prescribed length
4. For other directions, rotate by theta, and repeat the 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.)



Regional processing: polygonal approximations

Polygonal approximations are interesting because they are:
- simple (use lines) hence relatively fast
- can capture essential features like concavity, convexity, etc.

Will be defined on a binary image.

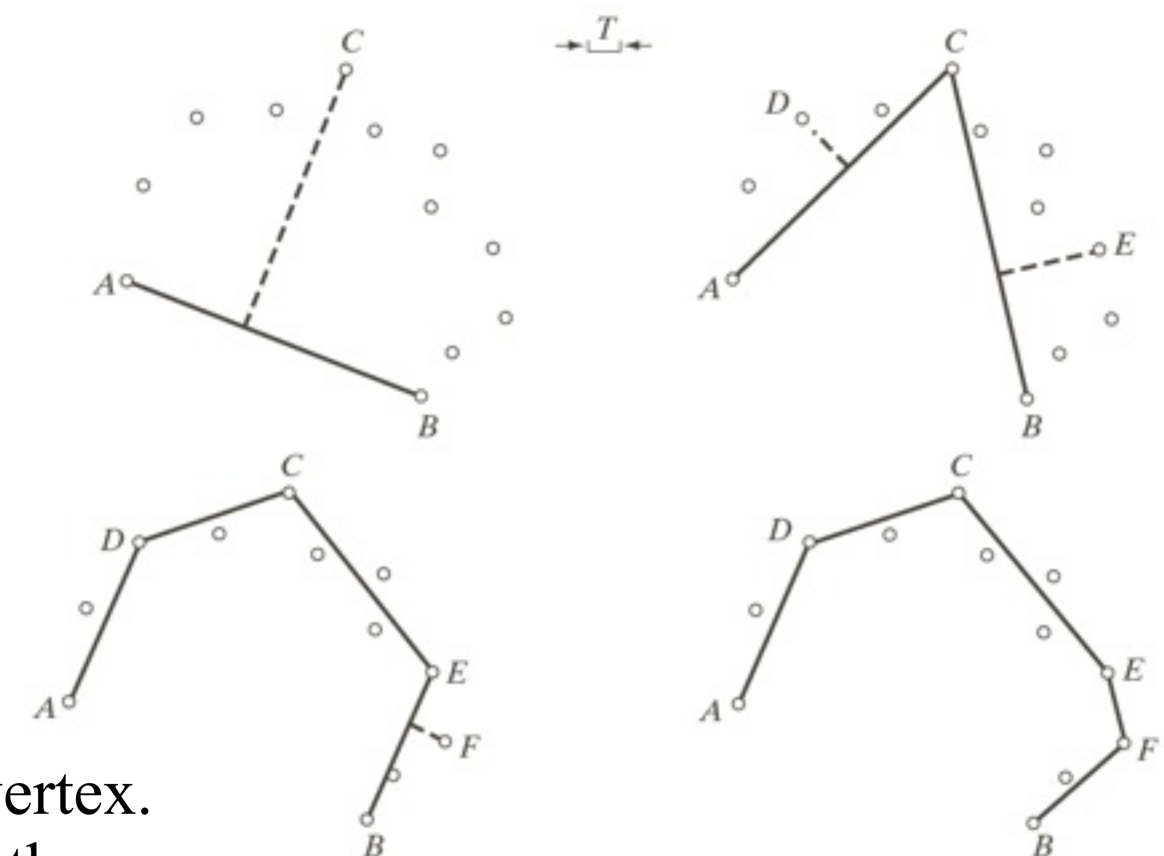
Given a ordered set of points (1 in a binary mask)

1. Define a start point (A) and end point (B). These are vertices of the polygon. Compute line between them.

2. Compute the distance (orthogonal) between other points and the AB line. Find furthest point (in case of a tie, resolve arbitrarily).

3. If the furthest point has a distance larger than a threshold T, define it as a vertex. Replace AB with AC and CB. Compute the distances of the subregions, furthest points, compare to threshold, evt. add vertex.

4. Continue until all points have distance ≤T from their edge

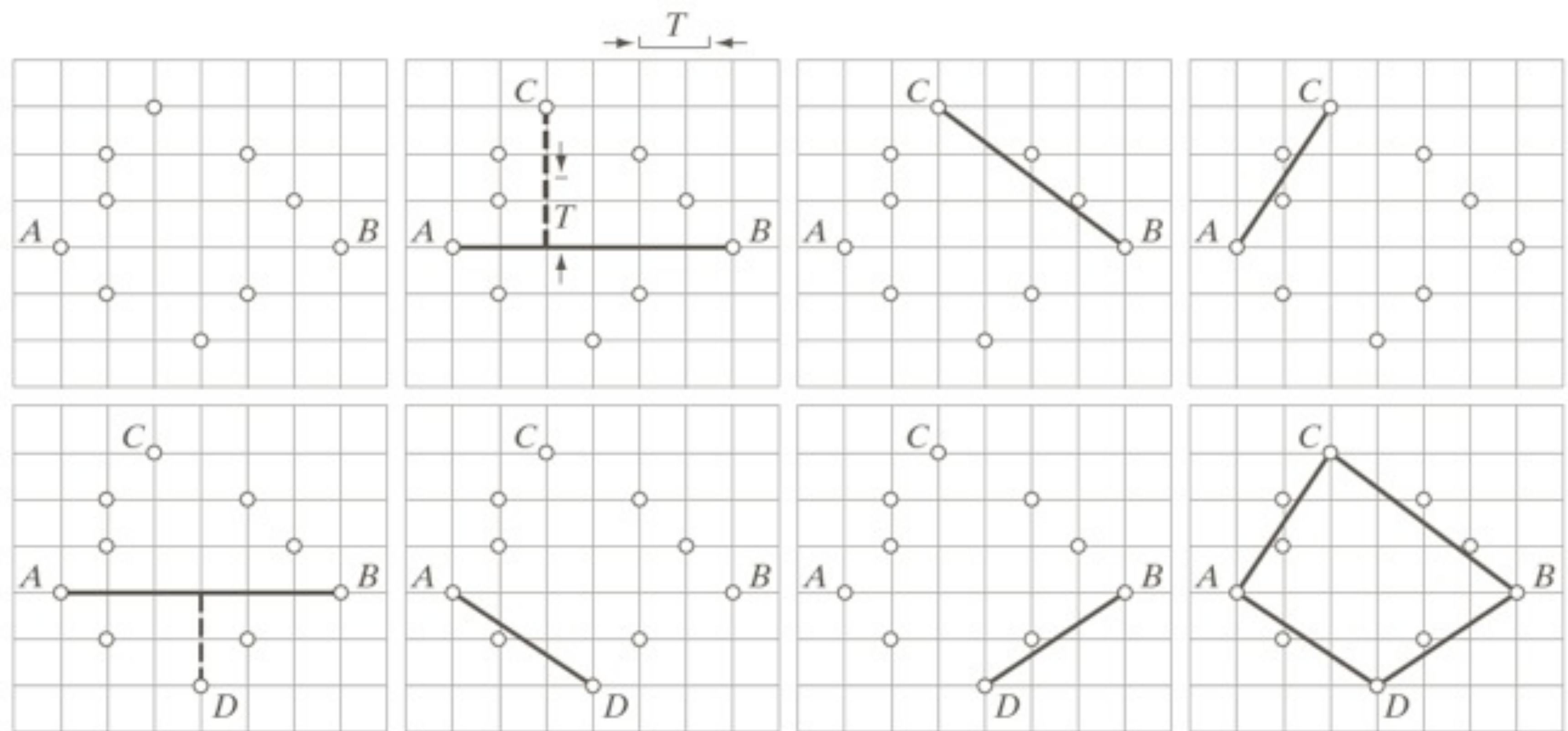


a b
c d

FIGURE 10.28 Illustration of t iterative polygonal fit algorithm.

In general, it is not clear whether a cloud of points forms a closed or open path.

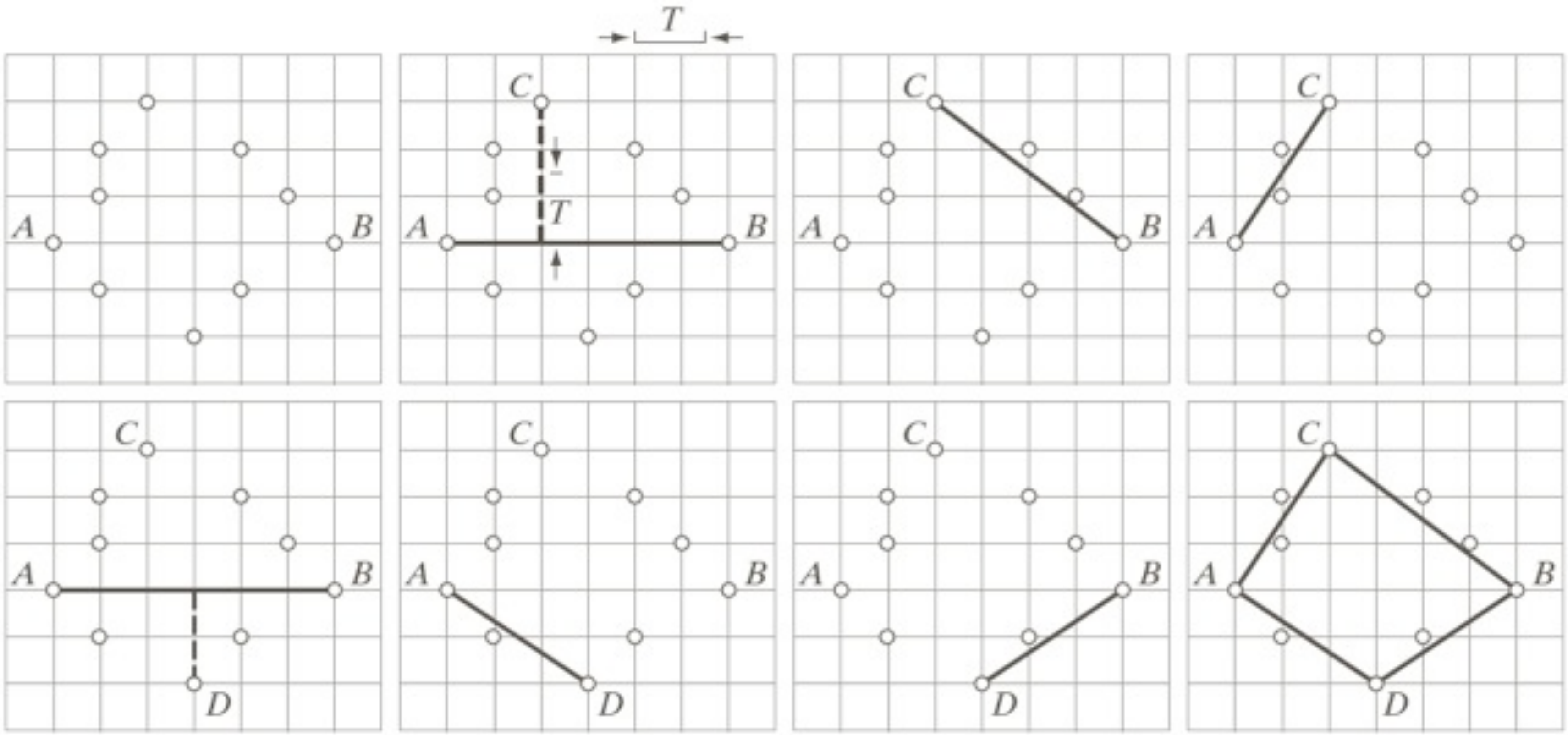
Look at the distance between points. If the distance is larger than the threshold, open the path.



a	b	c	d
e	f	g	h

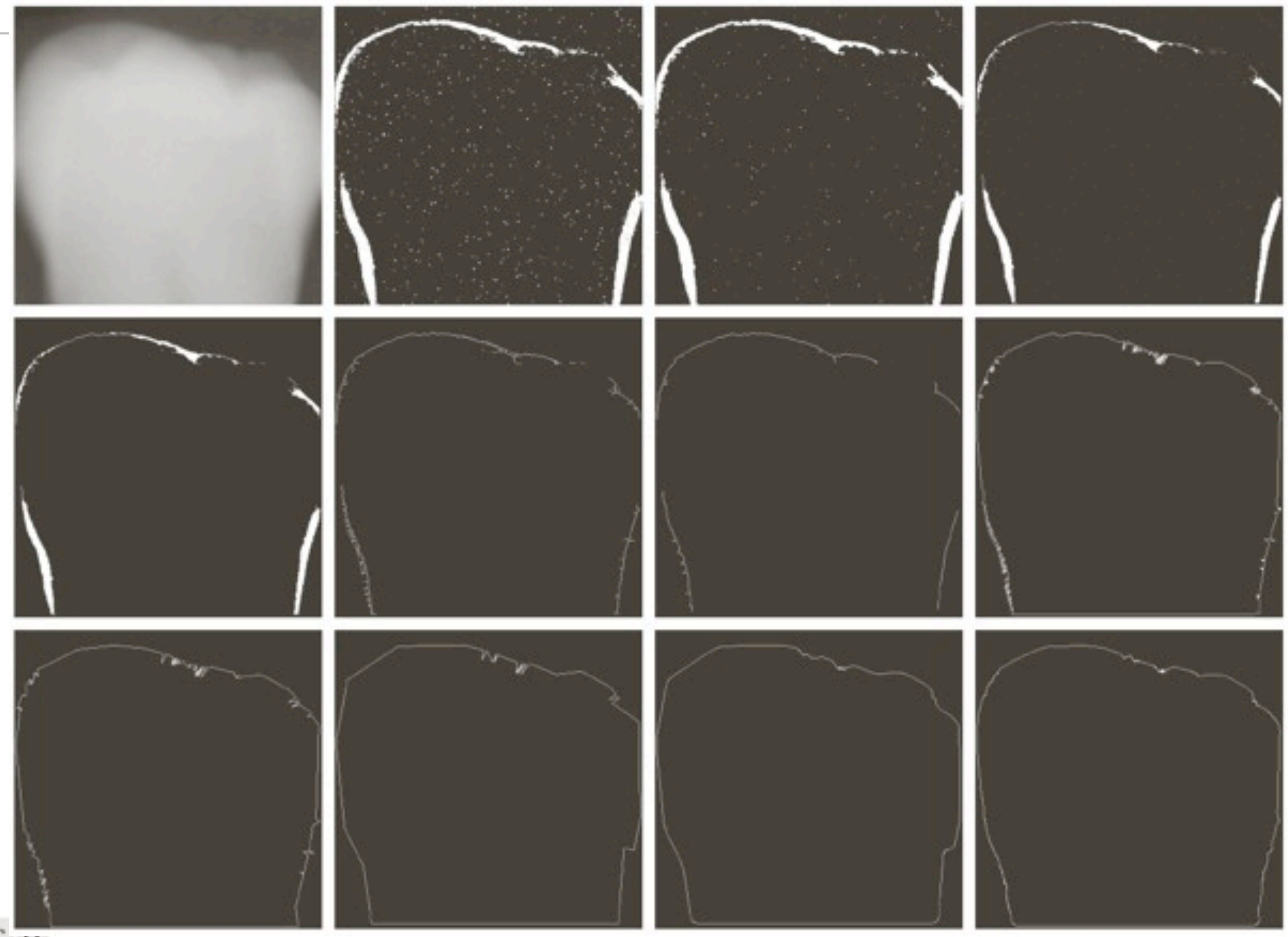
FIGURE 10.29 (a) A set of points in a clockwise path (the points labeled A and B were chosen as the starting vertices). (b) The distance from point C to the line passing through A and B is the largest of all the points between A and B and also passed the threshold test, so C is a new vertex. (d)–(g) Various stages of the algorithm. (h) The final vertices, shown connected with straight lines to form a polygon. Table 10.1 shows step-by-step details.

Chapter 10
Segmentation



CLOSED	OPEN	Curve segment processed	Vertex generated
B	*B*, *A*	—	*A*, *B*
B	*B*, *A*	(*BA*)	*C*
B	*B*, *A*, *C*	(*BC*)	—
B, *C*	*B*, *A*	(*CA*)	—
B, *C*, *A*	*B*	(*AB*)	*D*
B, *C*, *A*	*B*, *D*	(*AD*)	—
B, *C*, *A*, *D*	*B*	(*DB*)	—
B, *C*, *A*, *D*, *B*	Empty	—	—

Morphological operations + poligonal fit



a b c d
e f g h
i j k l

FIGURE 10.30 (a) A 550 × 566 X-ray image of a human tooth. (b) Gradient image. (c) Result of majority filtering. (d) Result of morphological shrinking. (e) Result of morphological cleaning. (f) Skeleton. (g) Spur reduction. (h)–(j) Polygonal fit using thresholds of approximately 0.5%, 1%, and 2% of image width ($T = 3$, 6, and 12). (k) Boundary in (j) smoothed with a 1-D averaging filter of size 1 × 31 (approximately 5% of image width). (l) Boundary in (h) smoothed with the same filter.