Image Filtering

- Filtering in Spatial domain
- Filtering in Frequency domain
- Filters are used for smoothing and sharpening of images.

Fundamentals of Spatial Filtering

- Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors.
- If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*.
- *Otherwise, the* filter is a *nonlinear spatial filter*.

Fundamentals of Spatial Filtering

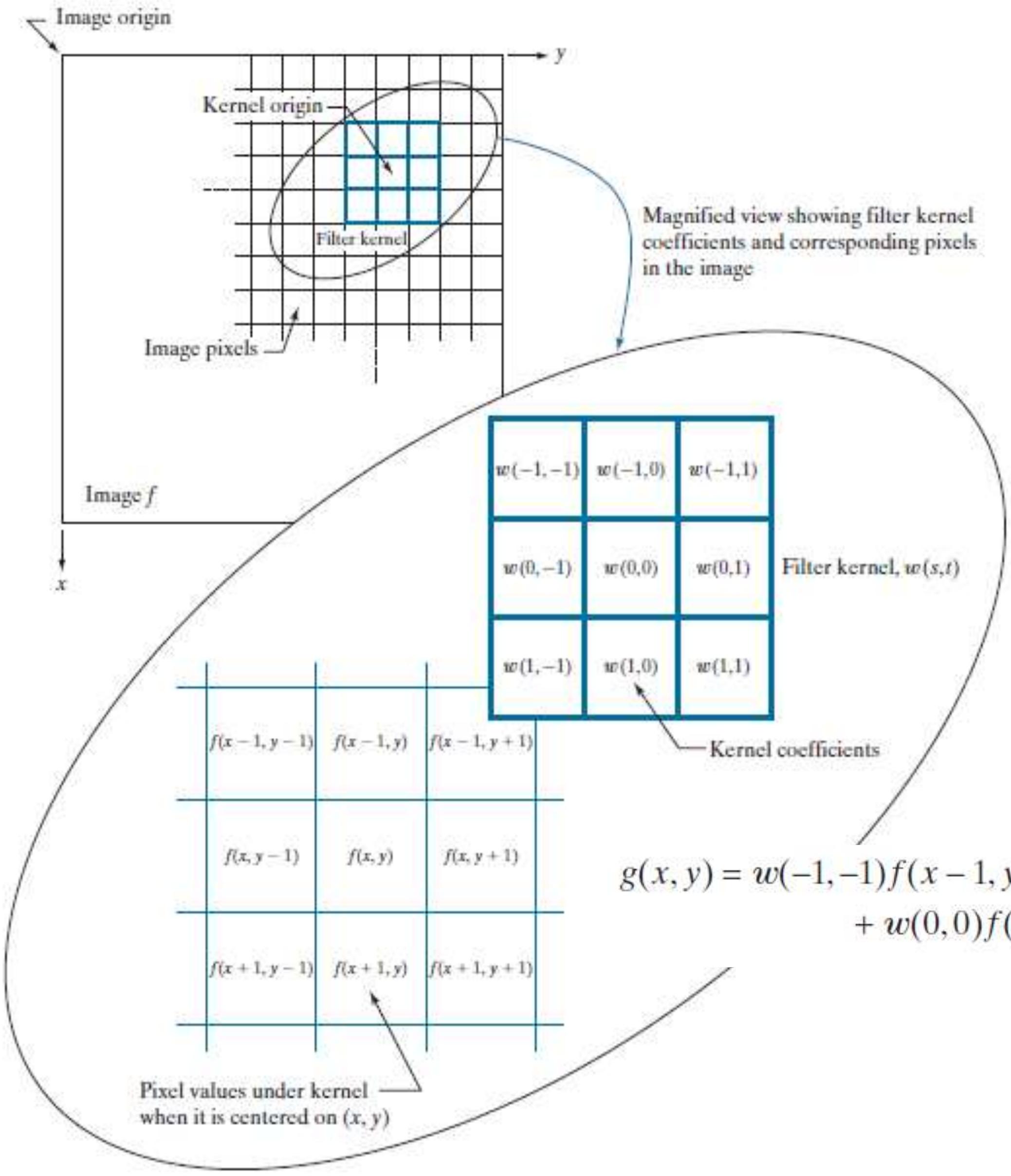


FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots$$
$$+ w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1)$$

Spatial Correlation

- Correlation consists of moving the center of a kernel over an image, and computing the sum of products at each location.
- Correlation of a filter w(x,y) of size m x n with an image f(x,y) is given by equation

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \qquad (3\text{-}34)$$

- This equation is evaluated for all values of displacement variables x and y.
- So that the center(origin) of w visit every pixel in f.
- f has to be padded appropriately.
- For a kernel of size $m * n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are nonnegative integers.
- This means that our focus is on kernels of odd size in both coordinate directions.
- a = (m-1)/2 , b = (n-1)/2

Spatial Convolution

- Convolution of a filter w(x,y) of size m x n with an image f(x,y) is given by equation

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \qquad (3\text{-}35)$$

Where the minus sign on the right flip f (i.e, rotate it by 180 degrees).

- This equation is evaluated for all values of displacement variables x and y.
- So that the centre(origin) of w visit every pixel in f.
- f has to be padded appropriately.
- For a kernel of size $m * n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are nonnegative integers.
- This means that our focus is on kernels of odd size in both coordinate directions.
- a = (m-1)/2 , b = (n-1)/2

Spatial 1-D correlation and convolution

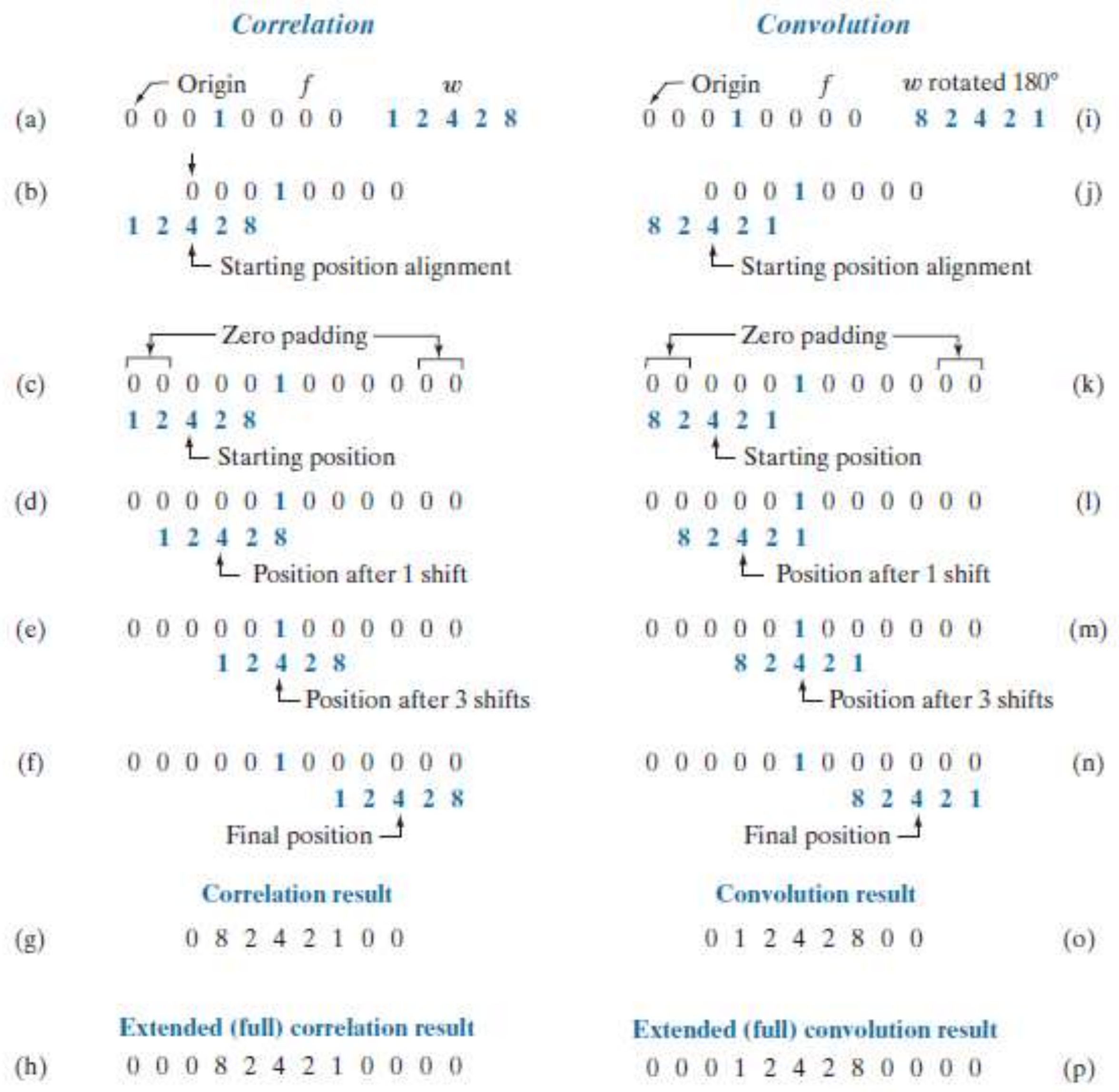
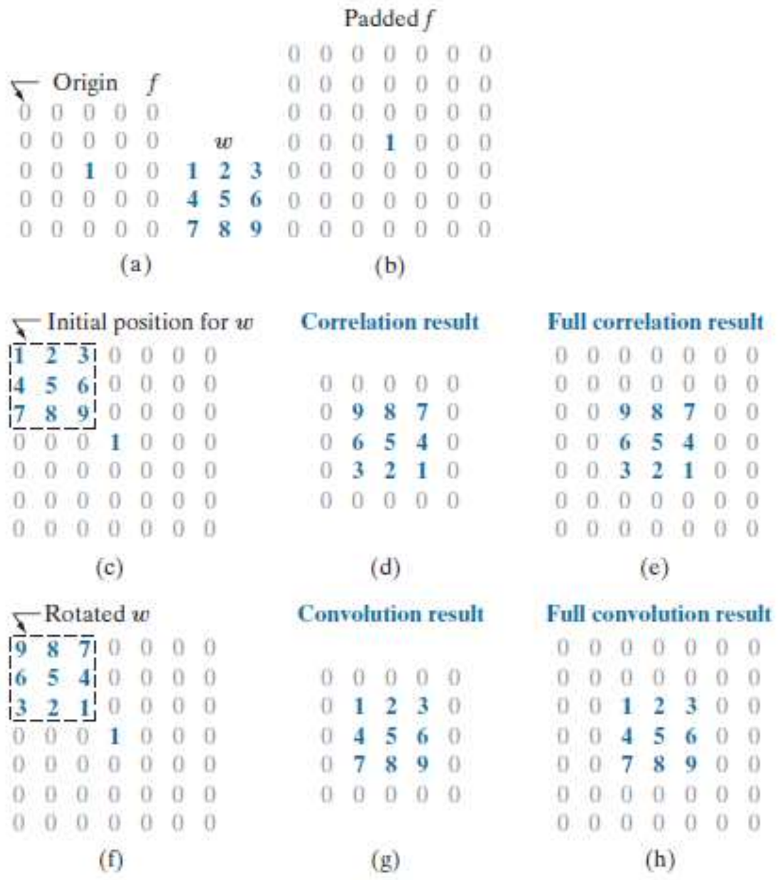

Correlation
Convolution
Origin
f
w
w rotated 180°
(a) 0 0 0 1 0 0 0 0 1 2 4 2 8
0 0 0 1 0 0 0 0 8 2 4 2 1 (i)
(b) 0 0 0 1 0 0 0 0
1 2 4 2 8
Starting position alignment
0 0 0 1 0 0 0 0 (j)
8 2 4 2 1
Starting position alignment
Zero padding
(c) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Starting position
0 0 0 0 0 1 0 0 0 0 0 0 (k)
8 2 4 2 1
Starting position
(d) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Position after 1 shift
0 0 0 0 0 1 0 0 0 0 0 0 (l)
8 2 4 2 1
Position after 1 shift
(e) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Position after 3 shifts
0 0 0 0 0 1 0 0 0 0 0 0 (m)
8 2 4 2 1
Position after 3 shifts
(f) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Final position
0 0 0 0 0 1 0 0 0 0 0 0 (n)
8 2 4 2 1
Final position
Correlation result
(g) 0 8 2 4 2 1 0 0
Convolution result
0 1 2 4 2 8 0 0 (o)
Extended (full) correlation result
(h) 0 0 0 8 2 4 2 1 0 0 0 0
Extended (full) convolution result
0 0 0 1 2 4 2 8 0 0 0 0 (p)

FIGURE 3.29 Illustration of 1-D correlation and convolution of a kernel, w, with a function f consisting of a discrete unit impulse. Note that correlation and convolution are functions of the variable x, which acts to *displace* one function with respect to the other. For the extended correlation and convolution results, the starting configuration places the right-most element of the kernel to be coincident with the origin of f. Additional padding must be used.

Spatial 2-D Correlation and Convolution

- For a kernel of size m × n, we pad the image with a minimum of (m − 1)/ 2 rows of 0’s at the top and bottom and (n − 1) /2 columns of 0’s on the left and right.
- In this case, m and n are equal to 3, so we pad f with one row of 0’s above and below and one column of 0’s to the left and right, as Fig. 3.30(b) shows.
- Figure 3.30(c) shows the initial position of the kernel for performing correlation, and Fig. 3.30(d) shows the final result after the center of w visits every pixel in *f, computing a sum of products* at each location.
- For convolution, we pre-rotate the kernel as before and repeat the sliding sum of products just explained. Figures 3.30(f) through (h) show the result.

FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0’s are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of *x* and *y*. As these variable change, they *displace* one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.



Extended(Full) Correlation / Convolution

- We can define correlation and convolution so that every element of w(instead of just its center) visits every pixel in f. This is called *extended, or full, correlation and convolution.*
- This requires that the starting configuration be such that the right, lower corner of the kernel coincides with the origin of the image.
- Similarly, the ending configuration will be with the top left corner of the kernel coinciding with the lower right corner of the image.
- If the kernel and image are of sizes m × n and M × N, respectively , we pad the image with a minimum of (m − 1) rows of 0's at the top and bottom and (n − 1) columns of 0's on the left and right.
- Under these conditions, the size of the resulting full correlation or convolution array will be of size $S_v \times S_h$, where

$$S_v = m + M - 1 \qquad (3\text{-}36)$$

and

$$S_h = n + N - 1 \qquad (3\text{-}37)$$

- Observations from the above example
  - Correlating a kernel function with a discrete unit impulse function yields a *rotated* (by 180 degrees) version of the kernel at the location of the impulse.
  - Convolving a kernel function with a discrete unit impulse function yields a *copy* of the kernel at the location of the impulse.

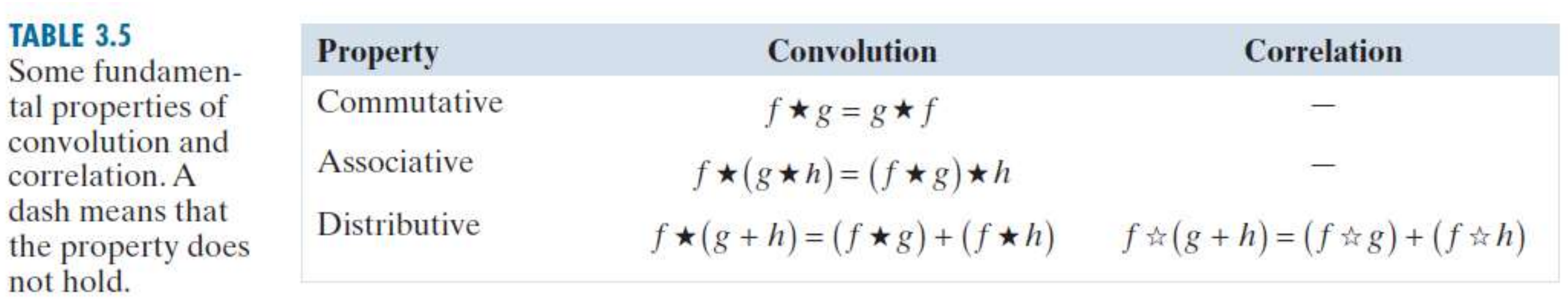
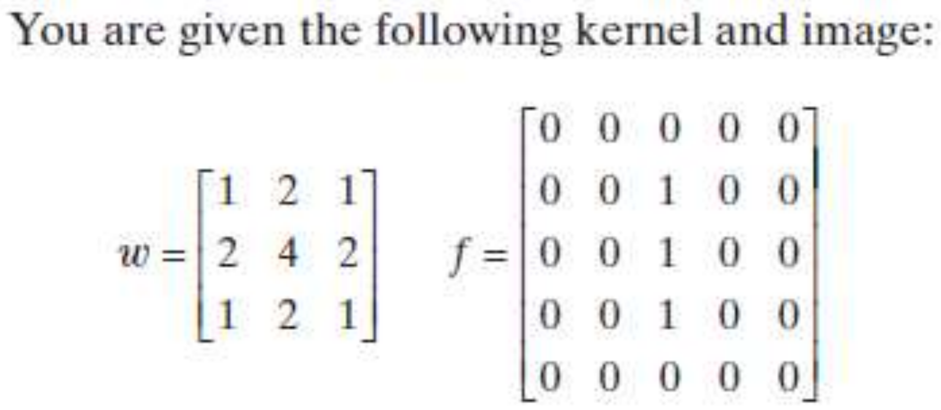


TABLE 3.5 Some fundamental properties of convolution and correlation. A dash means that the property does not hold.

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

Example



You are given the following kernel and image:

$$w = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Compute the convolution of w with *f using the minimum zero padding needed. Show the* details of your computations when the kernel is centered on point (2,3) of *f; and then* show the final full convolution result.

Smoothing Spatial filters

- *Smoothing* (also called *averaging*) spatial filters are used to reduce sharp transitions in intensity.
- Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction.
- Smoothing is used to reduce irrelevant detail in an image, where "irrelevant" refers to pixel regions that are small with respect to the size of the filter kernel.

- Another application is for smoothing the false contours that result from using an insufficient number of intensity levels in an image, as discussed in Section 2.4.
- Smoothing filters are used in combination with other techniques for image enhancement, such as the histogram processing techniques discussed in Section 3.3, and unsharp masking, as discussed later in this chapter.

- Types of smoothing filters
  - linear smoothing filters
    - Average filter
    - Weighted average filter
  - nonlinear smoothing filters
    - Median filter
    - Max filter
    - Min filter

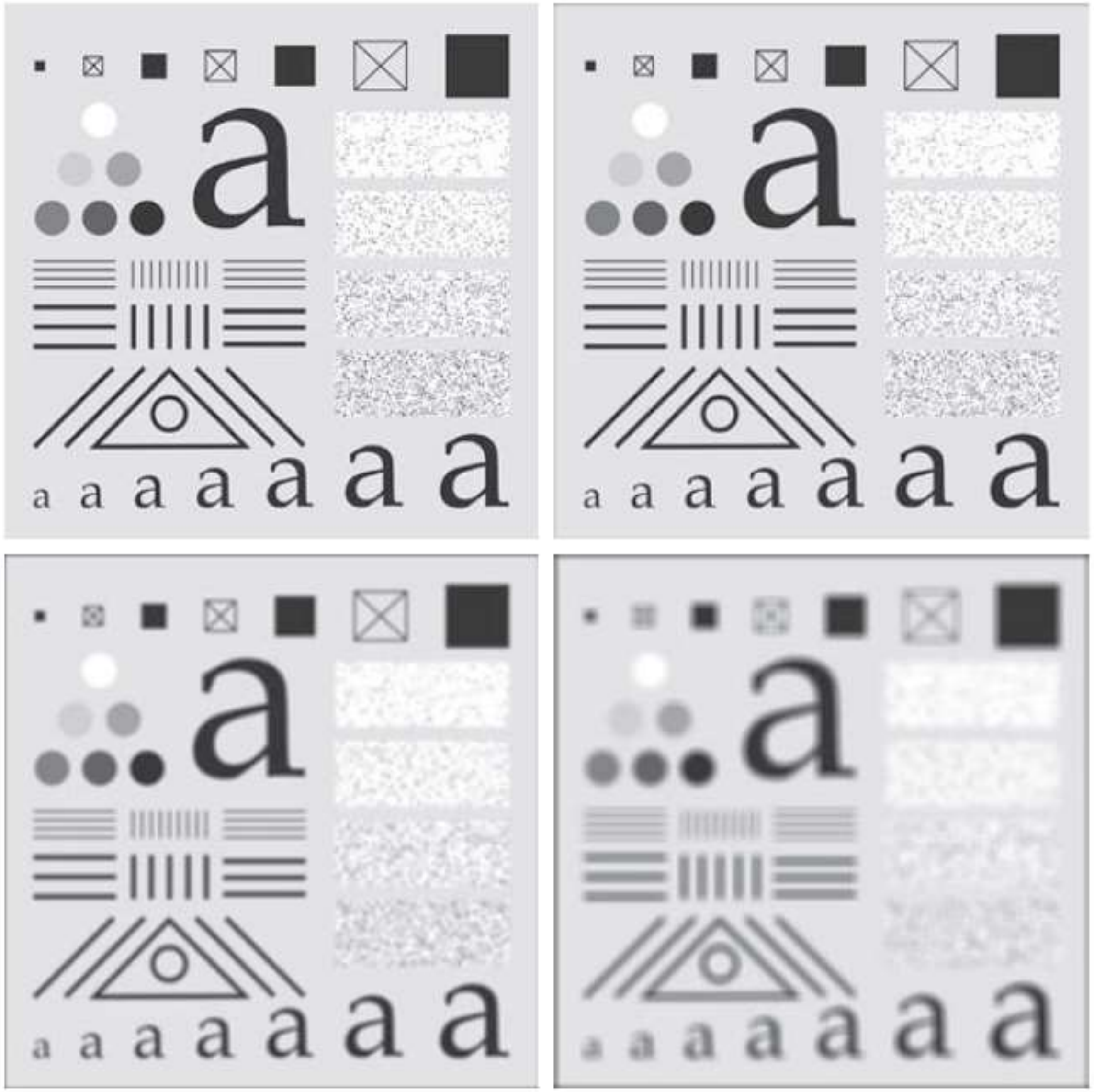
Linear Filters for smoothing: Average and Weighted average filters

- Also called as lowpass filters.
- In averaging filter all coefficients are equal.
- Also called box filter
- In weighted average filter pixels are multiplied by different coefficients .
- Thus giving more importance to some pixels at the expense of others.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

a b
c d

FIGURE 3.33 (a) Test pattern of size 1024×1024 pixels. (b)-(d) Results of lowpass filtering with box kernels of sizes 3×3, 11×11, and 21×21, respectively.



An Example

- For the given hypothetical 2-bit image of size 3*3 compute the smoothed image by convolving with average filter of size 3*3

$$f(x,y) = \begin{matrix} 1 & 3 & 3 \\ 0 & 0 & 0 \\ 2 & 2 & 1 \end{matrix}$$

Given image $f(x,y) = \begin{bmatrix} 1 & 3 & 3 \\ 0 & 0 & 0 \\ 2 & 2 & 1 \end{bmatrix}_{3\times 3}$

Average filter $w(x,y)$ of size 3x3 is $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{3\times 3}$

~~Convolving the~~

Convolution is equal to Rotating filter by 180° and performing correlation operation between filter & image.

The correlation is given as

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

Rotating $w(x,y)$ by 180° we get $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

padding the input image $f(x,y)$ we get

$$f(x,y) = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Now we compute $g(x,y)$ for $x = 0,1,2$. and $y = 0,1,2$.

$$g(0,0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix} \star \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$= 1\left(\tfrac{1}{9}\right) + 3\left(\tfrac{1}{9}\right) = 4/9$$

$$g(0,1) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 3 & 3 \\ 0 & 0 & 0 \end{bmatrix} \star \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$= 1\left(\frac{1}{9}\right) + 3\left(\frac{1}{9}\right) + 3\left(\frac{1}{9}\right) = 7/9$$

$$g(0,2) = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \star \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$= 3\left(\frac{1}{9}\right) + 3\left(\frac{1}{9}\right) = \frac{6}{9}.$$

Similarly we get

$$g(1,0) = \frac{8}{9} \qquad g(1,1) = \frac{12}{9} \qquad g(1,2) = \frac{9}{9}$$

$$g(2,0) = \frac{4}{9} \qquad g(2,1) = \frac{5}{9} \qquad g(2,2) = \frac{3}{9}$$

$$\therefore \quad g(x,y) = \begin{bmatrix} \frac{4}{9} & \frac{7}{9} & \frac{6}{9} \\ \frac{8}{9} & \frac{12}{9} & \frac{9}{9} \\ \frac{4}{9} & \frac{5}{9} & \frac{3}{9} \end{bmatrix} = \begin{bmatrix} 0.44 & 0.77 & 0.66 \\ 0.88 & 1.33 & 1 \\ 0.44 & 0.55 & 0.33 \end{bmatrix}$$

rounding off to nearest integer we get

$$g(x,y) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Now scale the range of g(x,y) values [0.33, 1.33] to [0,7] to get the final output image.
- We can scale a range of values [min, max] to [a, b] using the formula

$$f(x) = \frac{(b-a)(x-min)}{max-min} + a$$

- For the given hypothetical 2-bit image of size 3*3 compute the smoothed image by convolving with weighted average filter of size 3*3

$$f(x,y) = \begin{matrix} 1 & 3 & 3 \\ 0 & 0 & 0 \\ 2 & 2 & 1 \end{matrix}$$

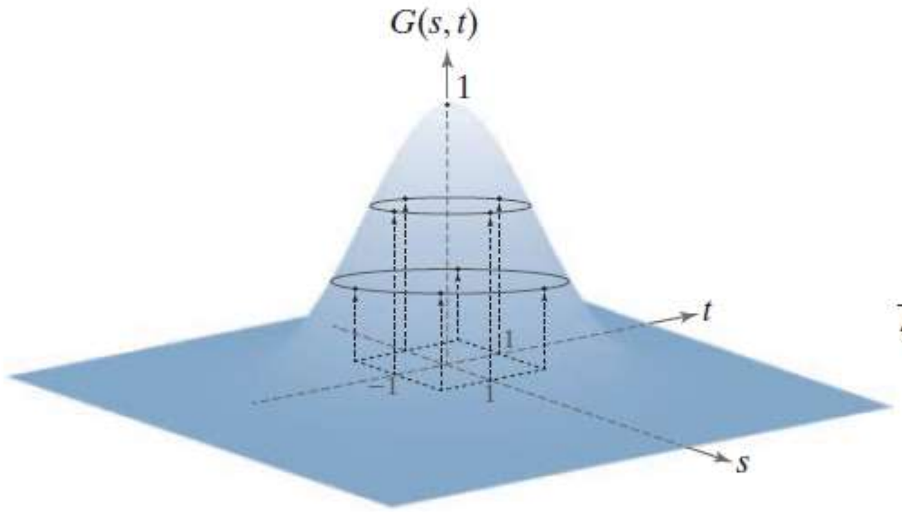
LOWPASS GAUSSIAN FILTER KERNELS

Gaussian kernels of the form

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

a b

FIGURE 3.35 (a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $K = 1$ and $\sigma = 1$. (b) Resulting 3×3 kernel [this is the same as Fig. 3.31(b)].



$\frac{1}{4.8976} \times$

0.3679	0.6065	0.3679
0.6065	1.0000	0.6065
0.3679	0.6065	0.3679

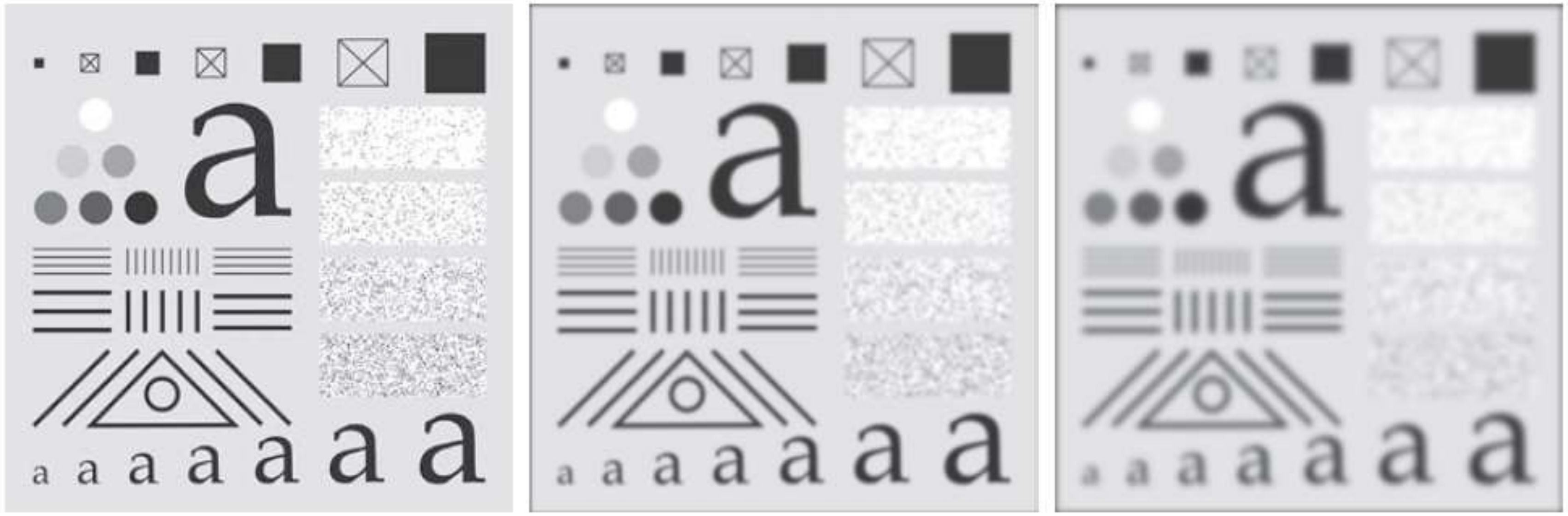
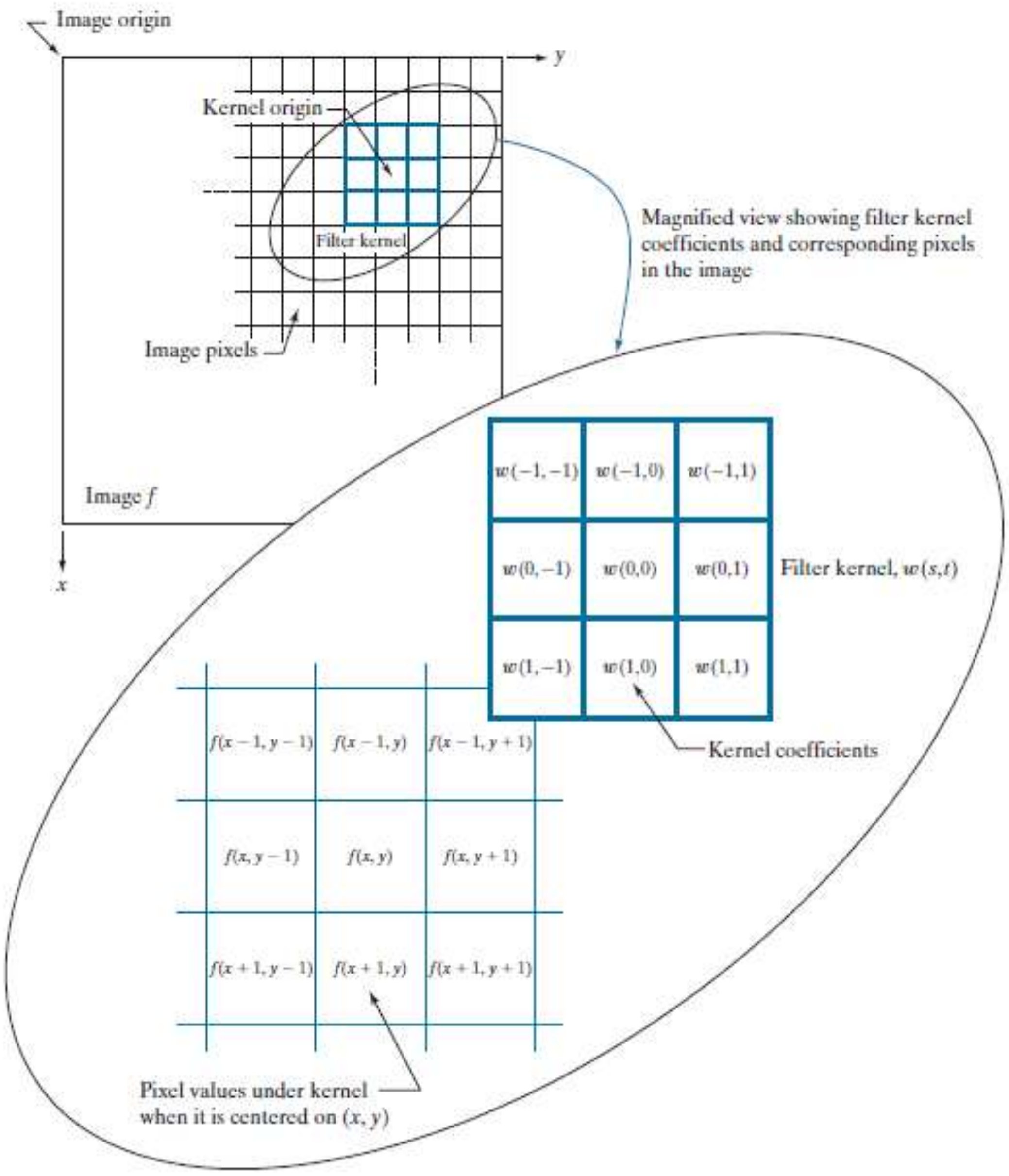


FIGURE 3.36 (a)A test pattern of size 1024×1024. (b) Result of lowpass filtering the pattern with a Gaussian kernel of size 21×21, with standard deviations $\sigma = 3.5$. (c) Result of using a kernel of size 43×43, with $\sigma = 7$. This result is comparable to Fig. 3.33(d). We used $K = 1$ in all cases.

Order statistic (Nonlinear) filters for smoothing

- Median filter
- Max filter
- Min filter

FIGURE 3.28
The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.



$f(x-1,y-1)$ z_1	$f(x-1,y)$ z_2	$f(x-1,y+1)$ z_3
$f(x,y+1)$ z_4	$f(x,y)$ z_5	$f(x,y+1)$ z_6
$f(x+1,y-1)$ z_7	$f(x+1,y)$ z_8	$f(x+1,y+1)$ z_9

$w(-1,-1)$ w_1	$w(-1,0)$ w_2	$w(-1,1)$ w_3
$w(0,-1)$ w_4	$w(0,0)$ w_5	$w(0,1)$ w_6
$w(1,-1)$ w_7	$w(1,0)$ w_8	$w(1,1)$ w_9

$$g(x,y) = f(x,y) \star w(x,y)$$

$$g(x,y) = w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + w_7z_7 + w_8z_8 + w_9z_9$$

Median filter

- Replaces the value of the center pixel, f(x,y), by the median of the intensity values in the neighborhood of that pixel (the value of the center pixel is included in computing the median).
- g(x,y) = median{ f(x-1,y-1), f(x-1,y), f(x-1,y+1), f(x, y-1), f(x,y), f(x,y+1), f(x+1,y-1), f(x+1,y), f(x+1,y+1) }
- For example, suppose that a 3 * 3 neighborhood has values

$$\begin{bmatrix} 10 & 20 & 20 \\ 20 & 15 & 20 \\ 20 & 25 & 100 \end{bmatrix}$$

- These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20.
- Median filter is effective in the presence of impulse noise(a.k.a salt and pepper noise)

- Max filter
  - the max of the intensity values in the neighborhood of the pixel, f(x,y) is assigned
  - $g(x,y) = \max\{ f(x-1,y-1), f(x-1,y), f(x-1,y+1), f(x, y-1), f(x,y), f(x,y+1), f(x+1,y-1), f(x+1,y), f(x+1,y+1) \}$
- Min filter
  - the min of the intensity values in the neighborhood of the pixel is assigned
  - $g(x,y) = \min\{ f(x-1,y-1), f(x-1,y), f(x-1,y+1), f(x, y-1), f(x,y), f(x,y+1), f(x+1,y-1), f(x+1,y), f(x+1,y+1) \}$

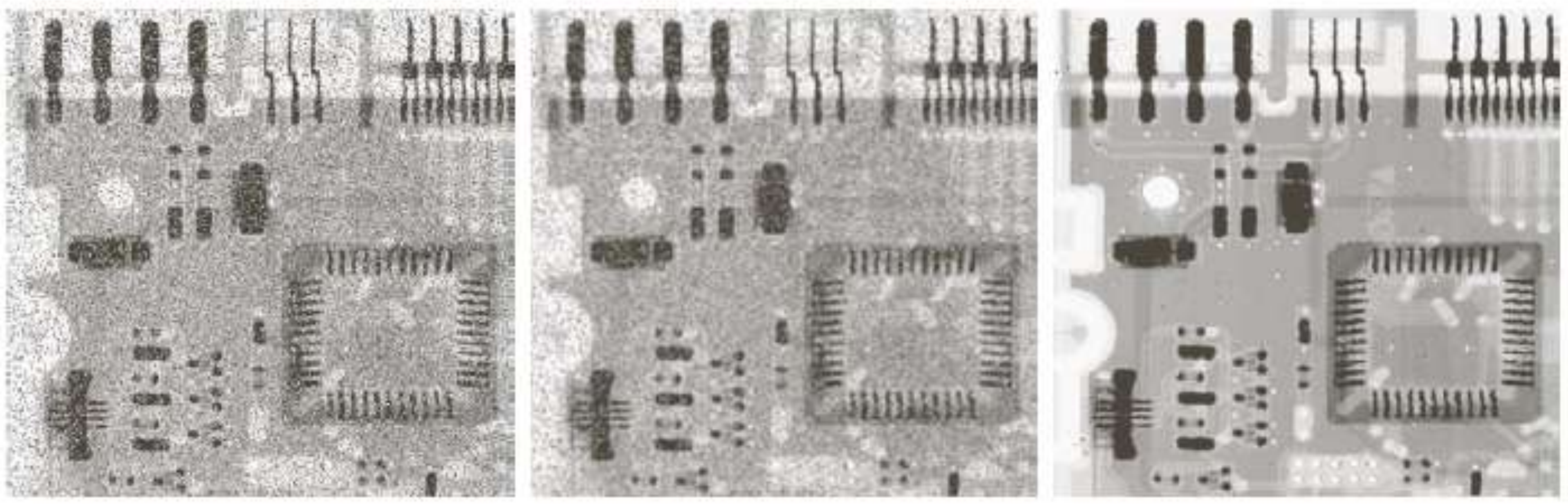

a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

- For the given hypothetical 2-bit image of size 5*5 compute the smoothed image, g(x,y),by applying median, max and min filters respectively.

$$\begin{matrix} 1 & 2 & 0 & 0 & 1 \\ 0 & 3 & 0 & 1 & 2 \\ 2 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 2 & 2 & 2 & 0 & 1 \end{matrix}$$

Given Image $f(x,y) = \begin{bmatrix} 1 & 2 & 0 & 0 & 1 \\ 0 & 3 & 0 & 1 & 2 \\ 2 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 2 & 2 & 2 & 0 & 1 \end{bmatrix}_{5\times5}$

~~Applying max filter we get~~ on

~~g(0,0)~~ $f_{padded}(x,y) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 1 & 2 & 0 \\ 0 & 2 & 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{7\times7}$

Applying the max filter ~~we get~~

$$g(x,y) = \max\{z_k \mid k = 1,2,3,\dots 9\}$$

$g(0,0) = \max\{0,0,0,0,1,2,0,0,3\} = 3$

$g(0,1) = \max\{0,0,0,1,2,0,0,3,0\} = 3$

Similarly $g(0,2) = 3$, $g(0,3) = 2$, $g(0,4) = 2$.

$g(1,0) = 3$, $g(1,1) = 3$, $g(1,2) = 3$, $g(1,3) = 3$, $g(1,4) = 3$

$g(2,0) = 3$ $g(2,1) = 3$ $g(2,2) = 3$ $g(2,3) = 3$ $g(2,4) = 3$

$g(3,0) = 2$ $g(3,1) = 2$ $g(3,2) = 3$ $g(3,3) = 3$ $g(3,4) = 3$

$g(4,0) = 2$ $g(4,1) = 2$ $g(4,2) = 2$ $g(4,3) = 2$ $g(4,4) = 1$

Sharpening spatial filters

- Objective of sharpening is to highlight transitions in intensity.
- Used in applications like
  - Electronic printing
  - Medical imaging
  - Industrial inspection
  - Autonomous guidance in military systems

Foundation of derivatives

- Derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, we require that any definition we use for a *first derivative:*
  1. **Must be zero in areas of constant intensity.**
  2. **Must be nonzero at the onset of an intensity step or ramp.**
  3. **Must be nonzero along intensity ramps.**
- Similarly, any definition of a *second derivative*
  1. **Must be zero in areas of constant intensity**.
  2. **Must be nonzero at the onset *and end of an intensity step or ramp*.**
  3. **Must be zero along intensity ramps.**
- We are dealing with digital quantities whose values are finite.
- Therefore, the maximum possible intensity change also is finite, and the shortest distance over which that change can occur is between adjacent pixels.

Foundation of derivatives

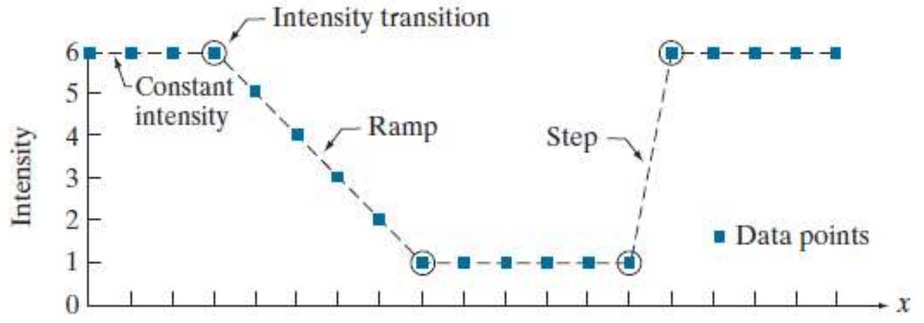
- The first-order derivative of a one-dimensional digital function f(x) is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{3-48}$$

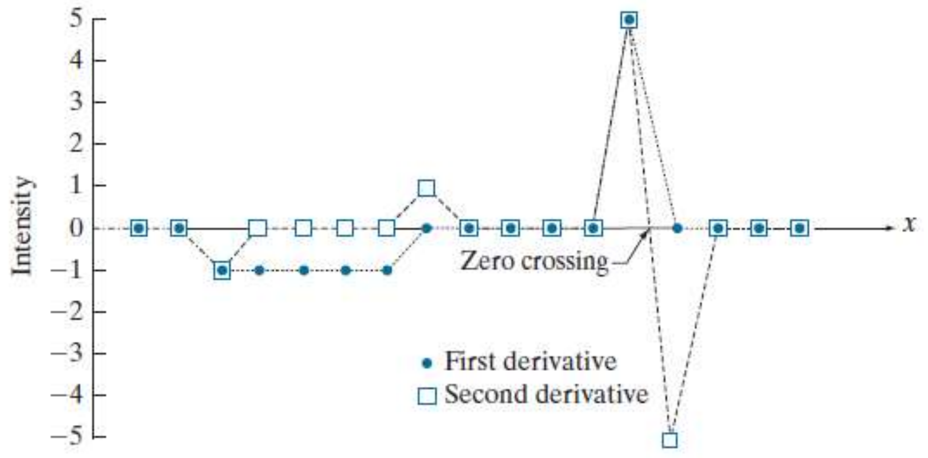
- The second-order derivative of a one-dimensional digital function f(x) is the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{3-49}$$

- These two definitions satisfy the conditions stated above, as we illustrate in Fig.
- The second derivative enhances fine detail much better than the first derivative



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	



a
b
c

FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

2-D second order derivative for Image Sharpening

- The second order derivative (Laplacian) for an image f(x,y) is given as

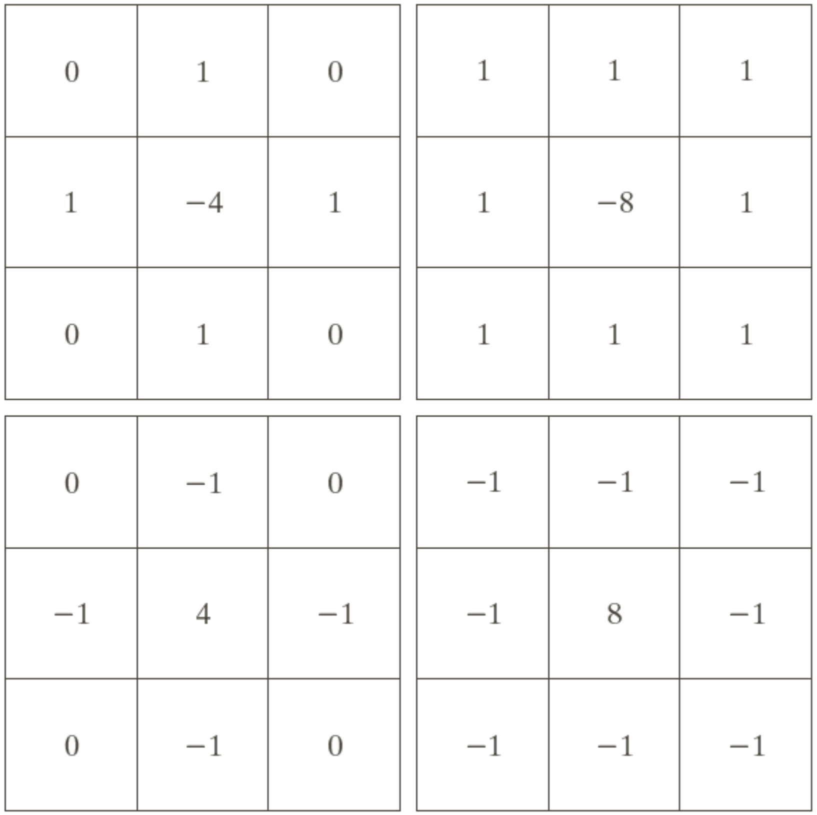
$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

- The above equation can be implemented using the filter mask as shown below



0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6).
(b) Mask used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) Two other implementations of the Laplacian found frequently in practice.

- The Laplacian image have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.
- The sharpened image is obtained by adding or subtracting the Laplacian image to the original image
- If the center coefficient is negative then we subtract
- Thus the basic way in which we use the Laplacian for image sharpening is

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

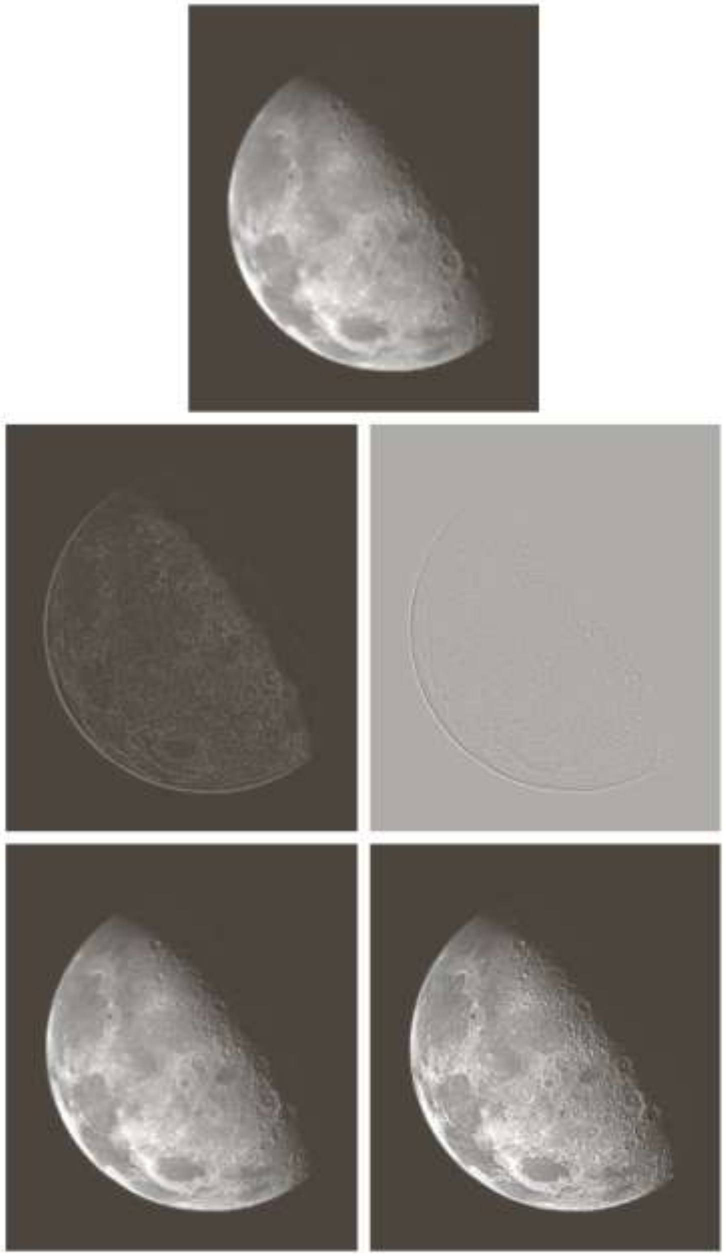


FIGURE 3.38 (a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. (c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b). (Original image courtesy of NASA.)

- Compute the Laplacian image for the given hypothetical 3-bit image of size 3*3 using the Laplacian mask mentioned in Fig 3.37(a)

 6 6 5

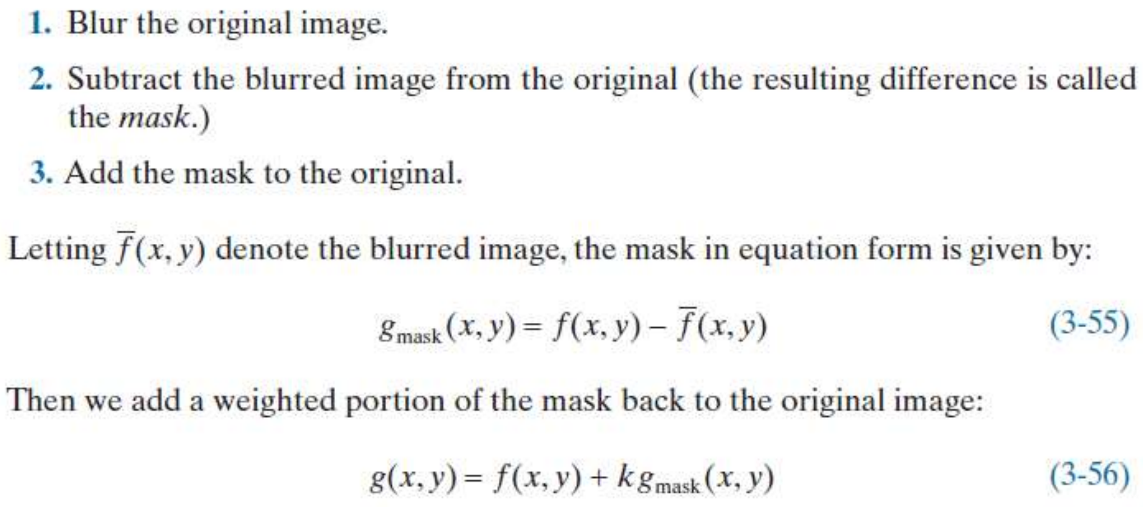
 2 5 2

 2 4 4

Sol: convolute the given image with Laplacian mask mentioned in Fig 3.37(a)

Unsharp Masking and High boost Filtering

- To sharp an image, printing and publishing industry, for many years, used a process called unsharp masking.
- This process consists of the following steps:



1. Blur the original image.
2. Subtract the blurred image from the original (the resulting difference is called the *mask*.)
3. Add the mask to the original.

Letting $\overline{f}(x, y)$ denote the blurred image, the mask in equation form is given by:

$$g_{\text{mask}}(x, y) = f(x, y) - \overline{f}(x, y) \tag{3-55}$$

Then we add a weighted portion of the mask back to the original image:

$$g(x, y) = f(x, y) + k\,g_{\text{mask}}(x, y) \tag{3-56}$$

- ‘k (k>=0)’ is included for generality.
- When k = 1, we have unsharp masking
- When k> 1, the process is called highboost filtering

Using First-order derivative for Image Sharpening– The Gradient

- First derivatives in image processing are implemented using the magnitude of the gradient. The *gradient* of an image *f* at coordinates (*x*, *y*) is defined as the two dimensional column vector

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- This vector has the important geometrical property that it points in the direction of the greatest rate of change of *f* at location (*x*, *y*).

- The magnitude of the vector is denoted as M(x,y), where

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

- M(x,y) is the gradient image having same size as the original image f(x,y)
- M(x,y) can be approximated to

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

- The definitions of gx and gy are given as (also called as 2*2 Gradient operators)

$$g_x = (z_8 - z_5) \text{ and } g_y = (z_6 - z_5).$$

- The other definitions of gx and gy proposed by Robert, considering cross differences are (also called as 2*2 Robert cross gradient operators)

$$g_x = (z_9 - z_5) \quad \text{and} \quad g_y = (z_8 - z_6)$$

$f(x-1, y-1)$ z_1	$f(x-1, y)$ z_2	$f(x-1, y+1)$ z_3
$f(x, y+1)$ z_4	$f(x, y)$ z_5	$f(x, y+1)$ z_6
$f(x+1, y-1)$ z_7	$f(x+1, y)$ z_8	$f(x+1, y+1)$ z_9

- The definitions of gx and gy given by Sobel are (also called as 3*3 Sobel gradient operators)

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

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

$$M(x,y) = |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

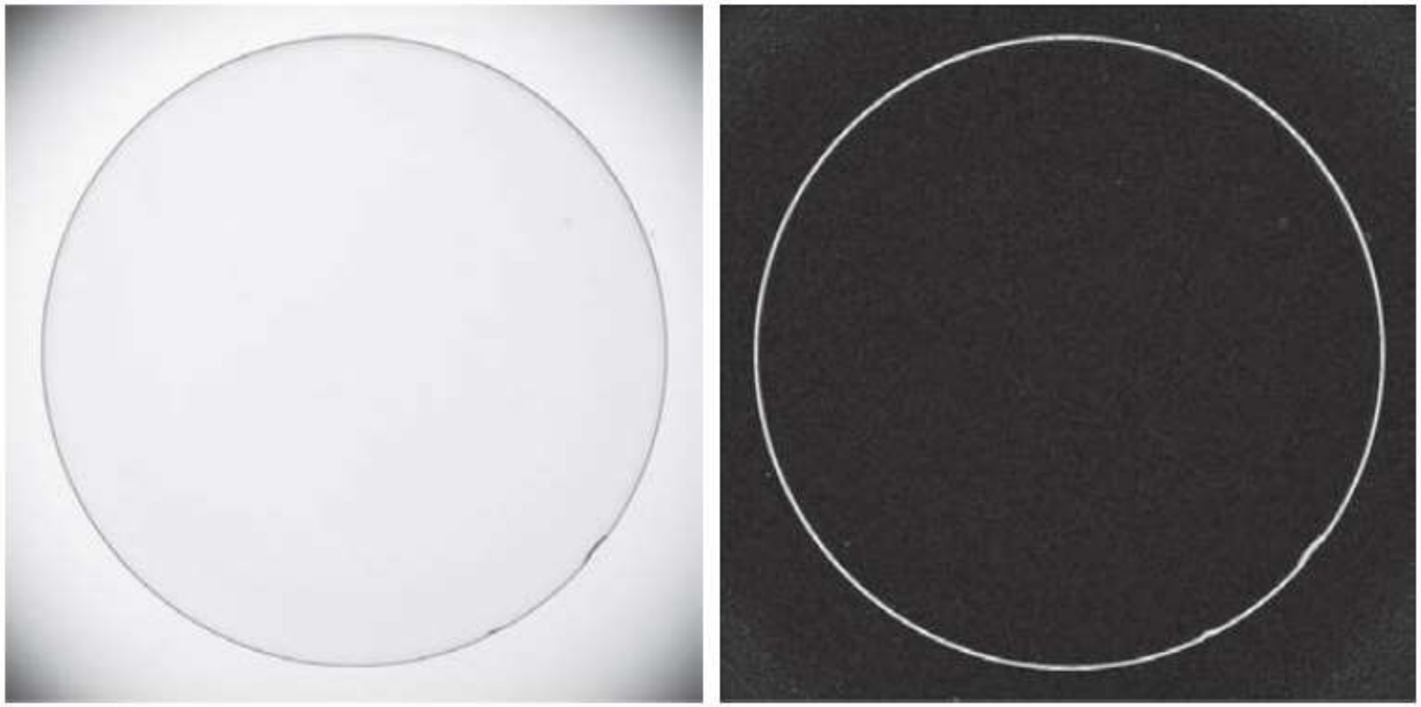
0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

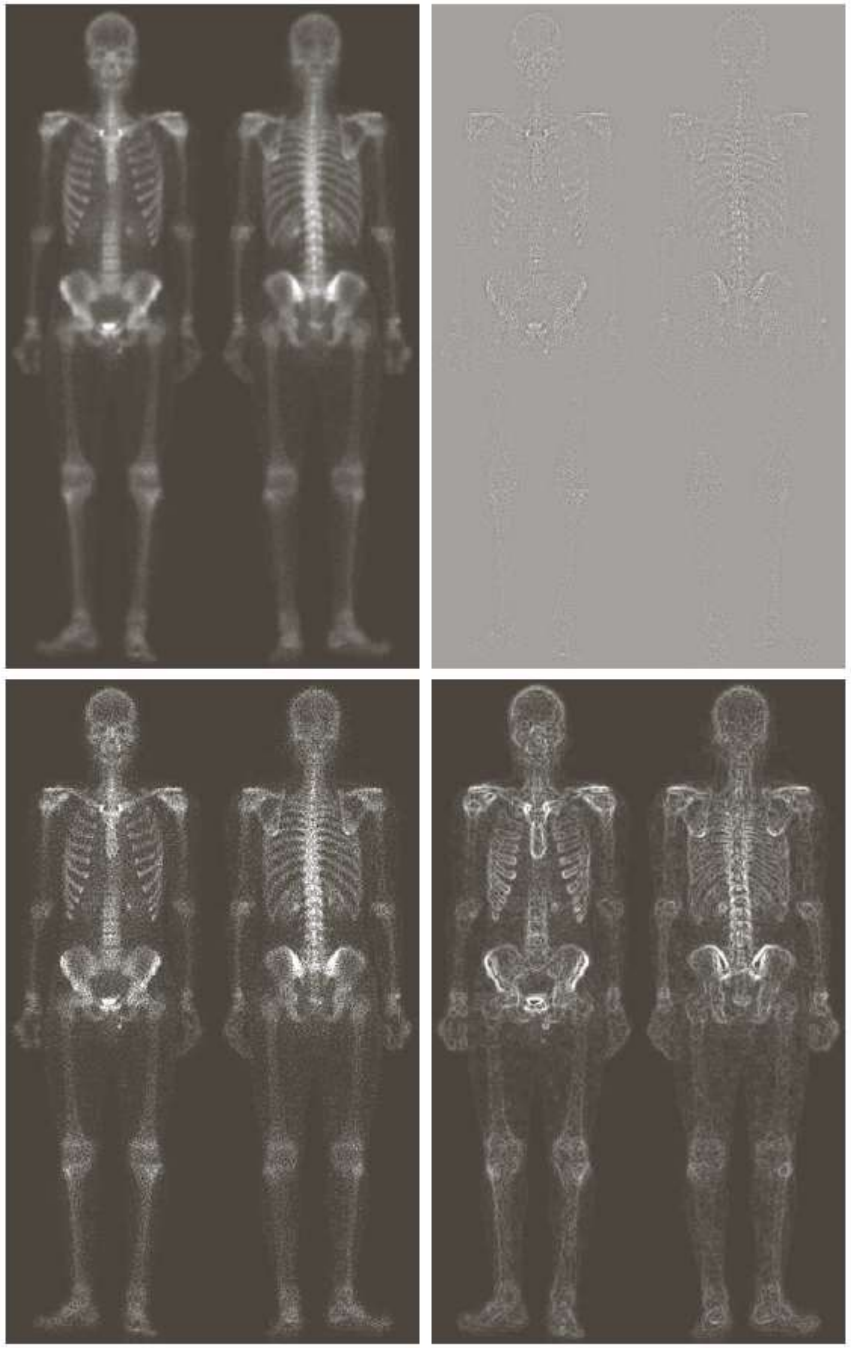
a	
b	c
d	e

FIGURE 3.41 A 3×3 region of an image (the zs are intensity values). (b)–(c) Roberts cross gradient operators. (d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.



a b

FIGURE 3.51
(a) Image of a contact lens (note defects on the boundary at 4 and 5 o'clock).
(b) Sobel gradient.
(Original image courtesy of Perceptics Corporation.)



a b
c d

FIGURE 3.43
(a) Image of whole body bone scan.
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b).
(d) Sobel gradient of (a).

- Compute the gradient image for the given hypothetical 3-bit image of size 3*3 using the Sobel operators mentioned in Fig 3.41(d) and (e)

6 6 5

2 5 2

2 4 4

Sol:

Step1: convolute the given image with sobel operator mentioned in Fig 3.41(d) to obtain gx image

Step2: convolute the given image with sobel operator mentioned in Fig 3.41(e) to obtain gy image

Step3: compute gradient image, M(x,y), by adding gx and gy images