Spatial Domain Enhancement

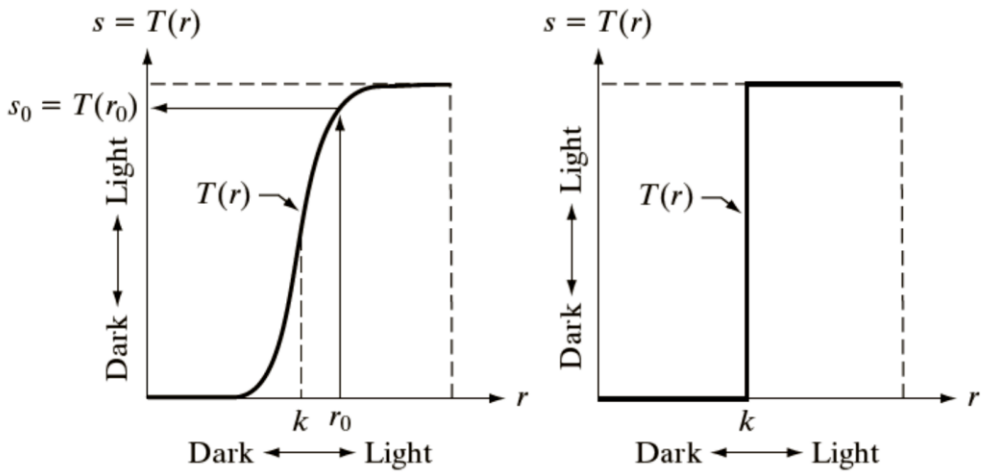
Dr. Badri Narayan Subudhi

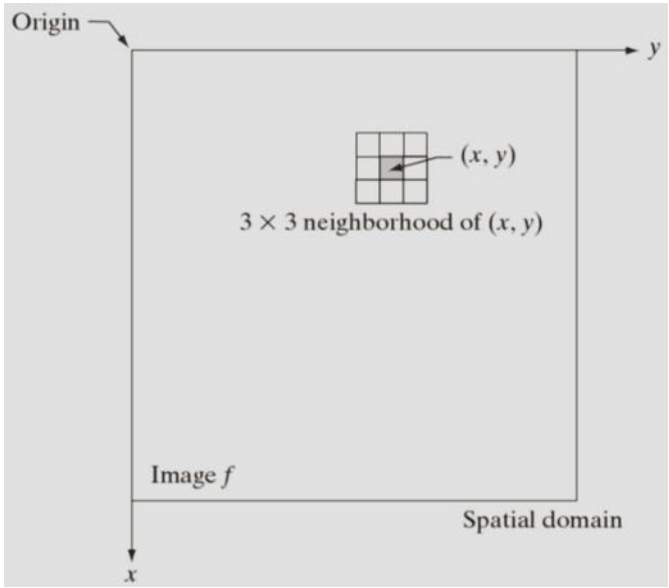
Dept. of EE, IIT Jammu

Enhancement

Spatial Domain Processing

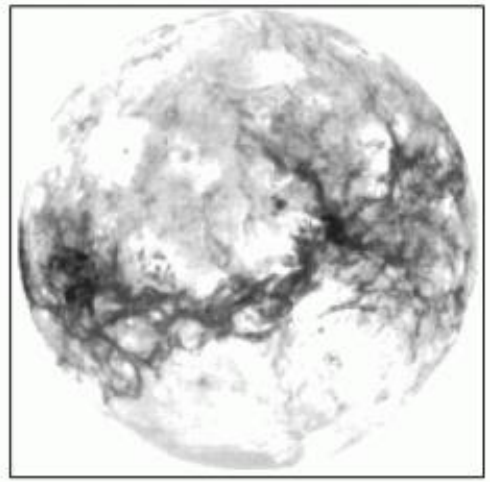
- **Point based processing**
- **Window based processing**



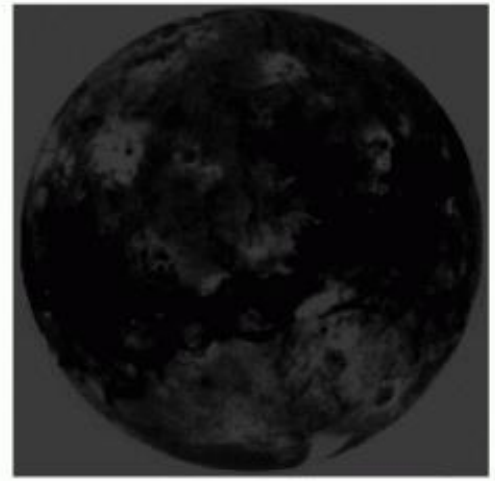


Contrast and Brightness

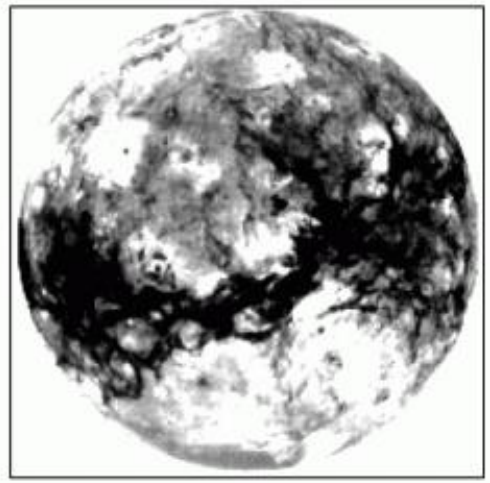
- An image must have the proper **brightness** and **contrast** for easy viewing.
- Brightness refers to the overall lightness or darkness of the image.
- Contrast is the *difference* in brightness between objects and background.
- For example, a white rabbit running across a snowy field has *poor* contrast, while a black dog against the same white background has *good* contrast.



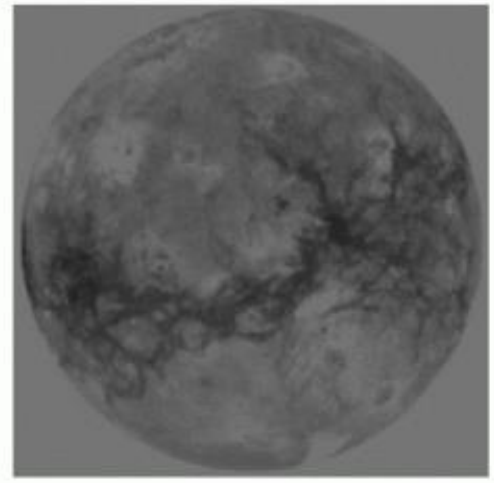
a. Brightness too high



b. Brightness too low



c. Contrast too high



d. Contrast too low

Brightness and Contrast

Brightness:

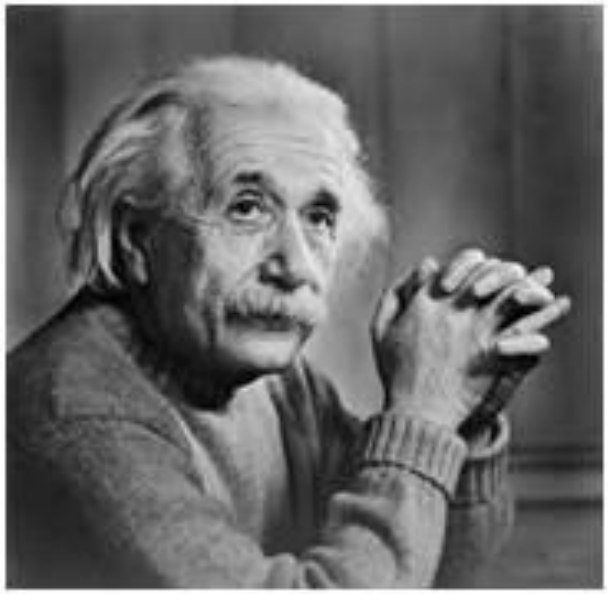
$$s=k+r$$

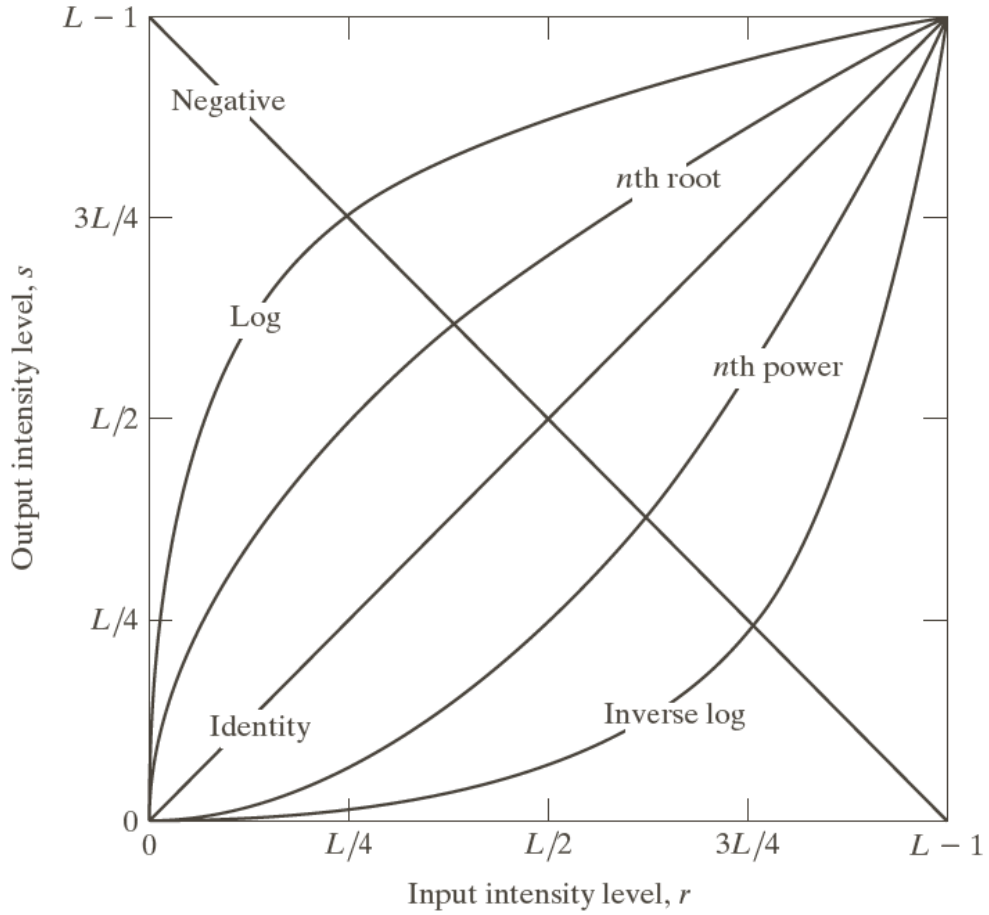
Contrast:

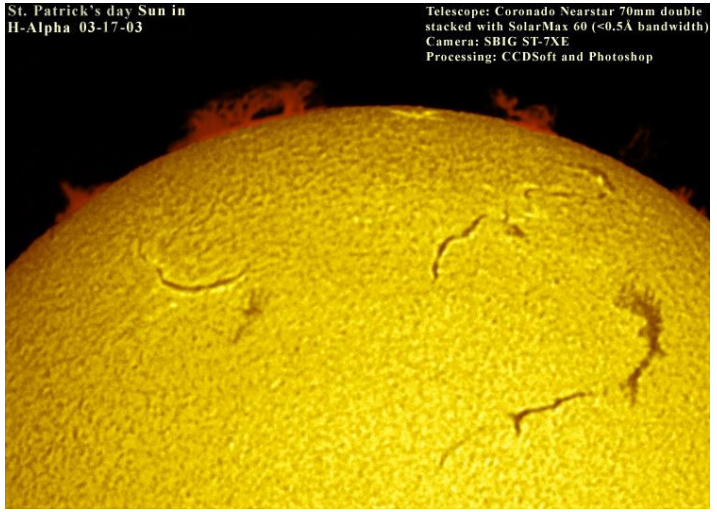
$$s=k \times r$$

Image Negative

S= L-1-r

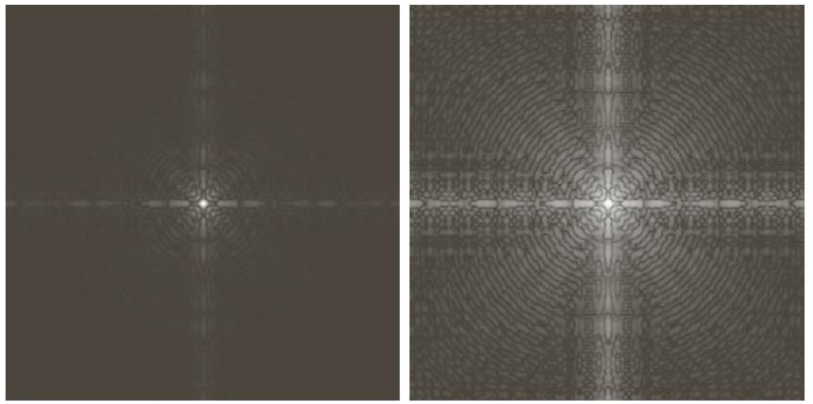





St. Patrick's day Sun in
H-Alpha 03-17-03
Telescope: Coronado Nearstar 70mm double
stacked with SolarMax 60 (<0.5Å bandwidth)
Camera: SBIG ST-7XE
Processing: CCDSoft and Photoshop


St. Patrick's day Sun in
H-Alpha 03-17-03
Telescope: Coronado Nearstar 70mm double
stacked with SolarMax 60 (<0.5Å bandwidth)
Camera: SBIG ST-7XE
Processing: CCDSoft and Photoshop

Logarithmic Transformation

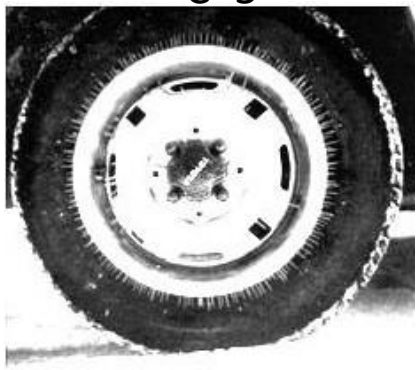


$$S=c \times \log(1+r)$$

C=1

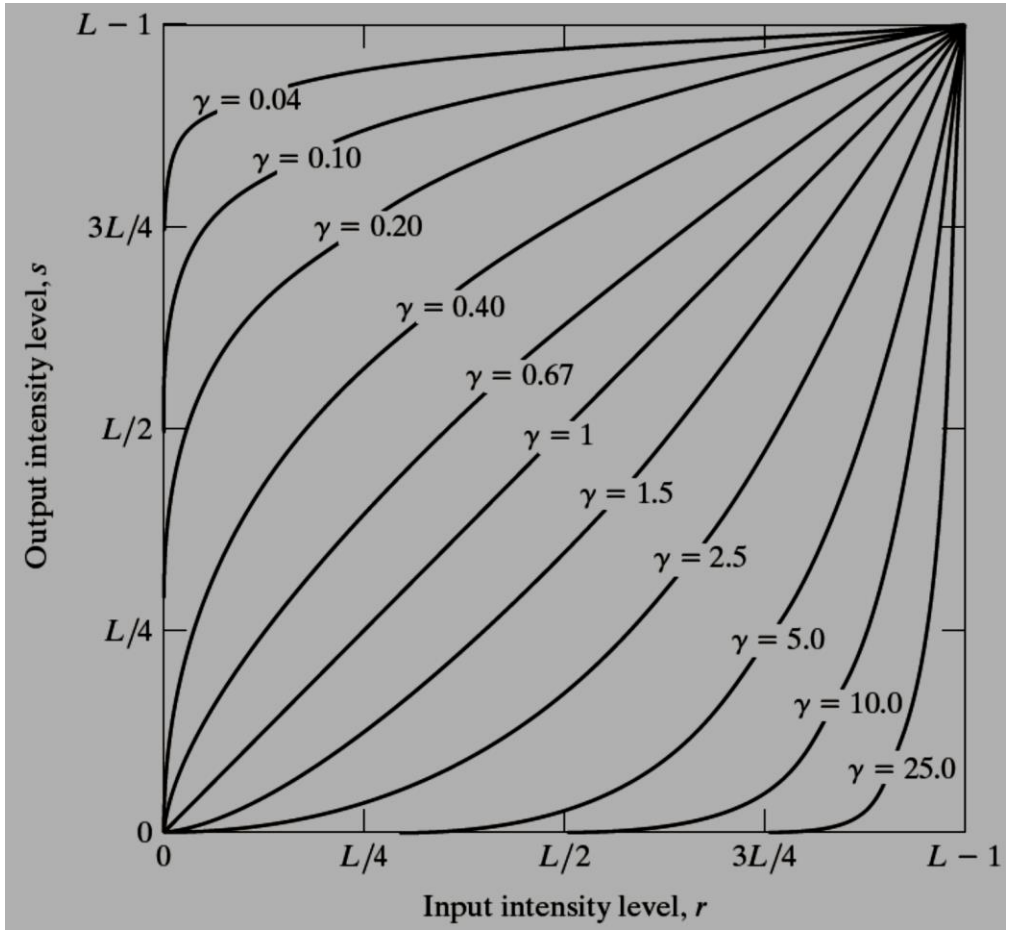
C=2

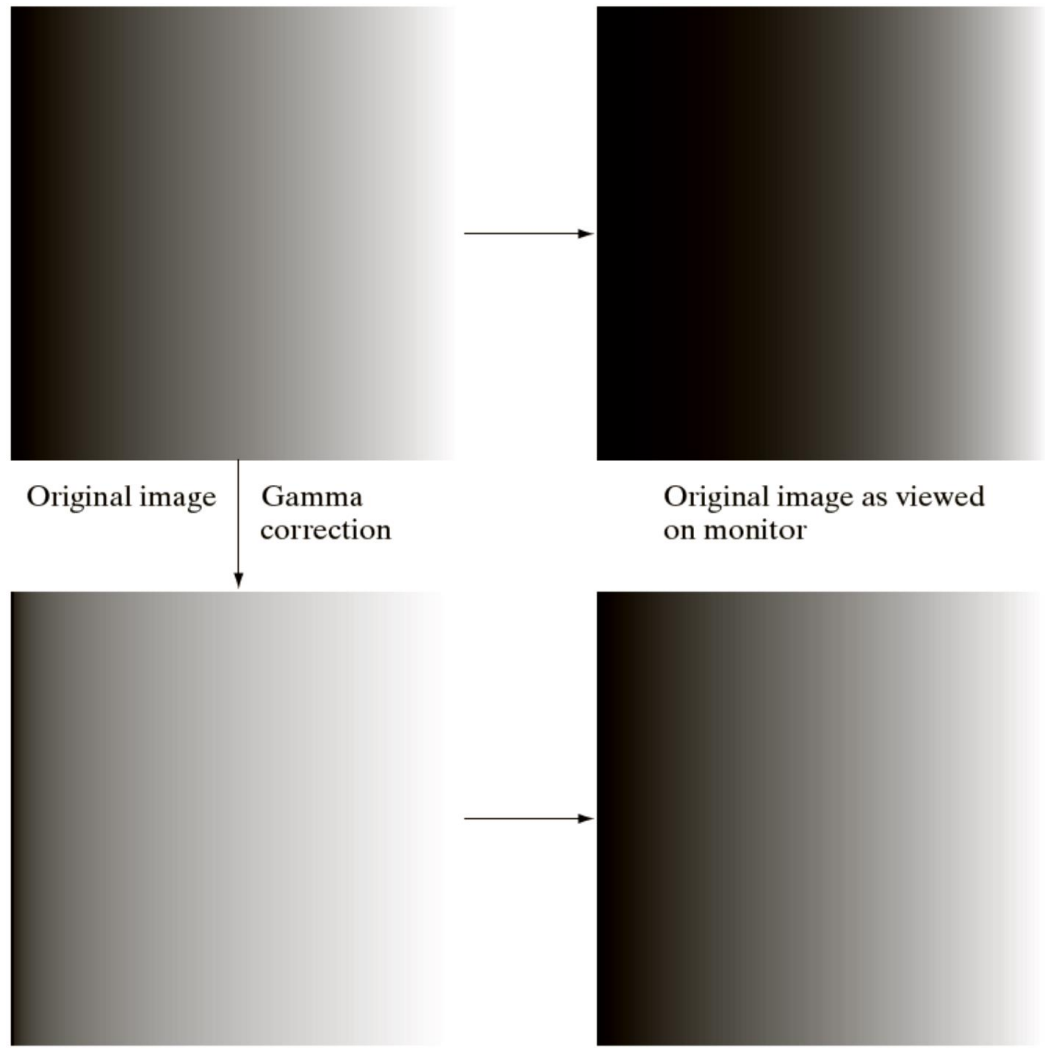
C=5

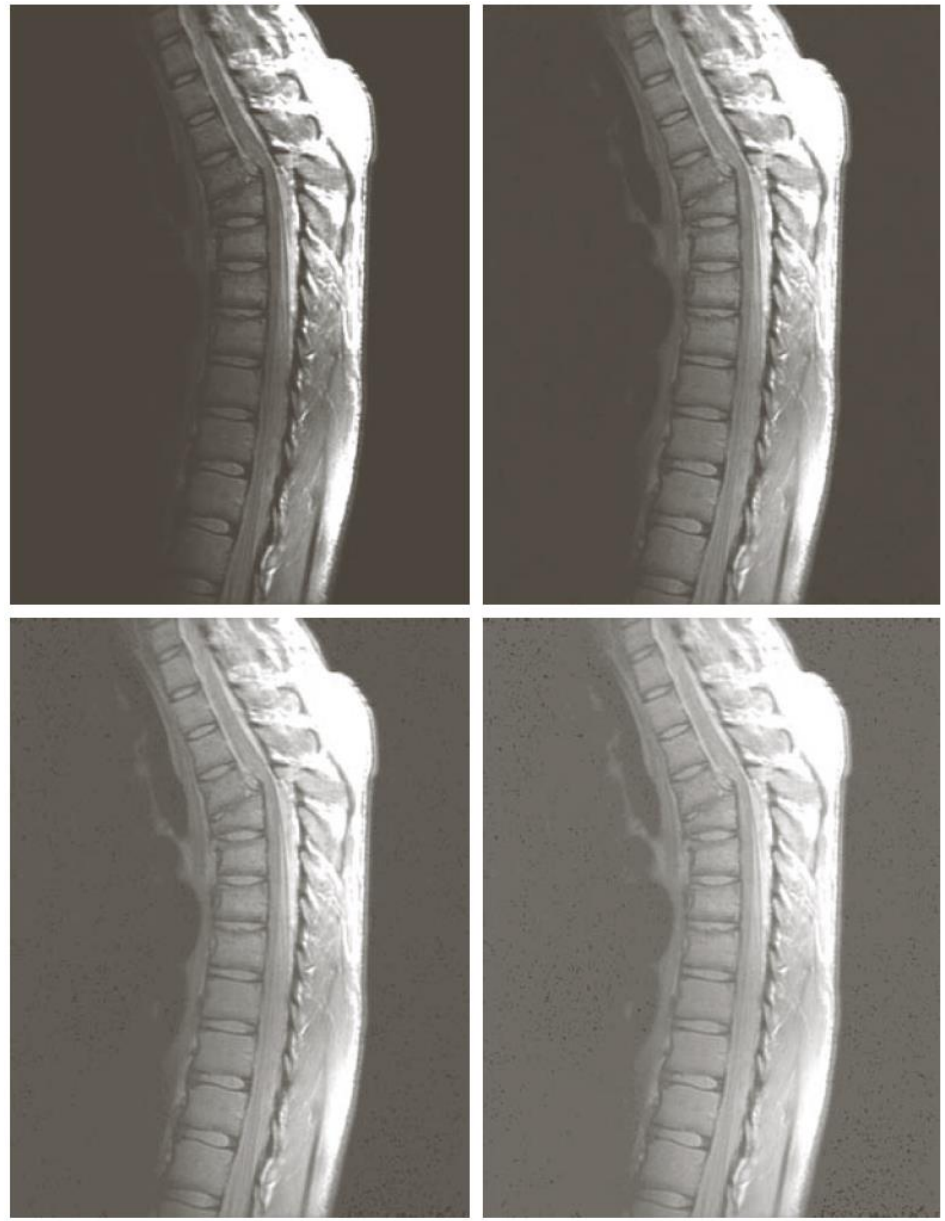


Gamma-Transformation

$$S=C(r+\varepsilon)^{\gamma}$$




Original image
Gamma correction
Original image as viewed on monitor
Gamma-corrected image
Gamma-corrected image as viewed on the same monitor



γ=0.6

γ=0.4

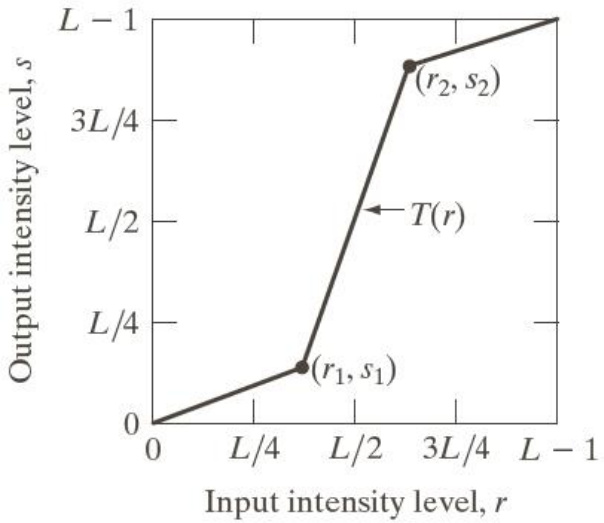
γ=0.3

γ=3

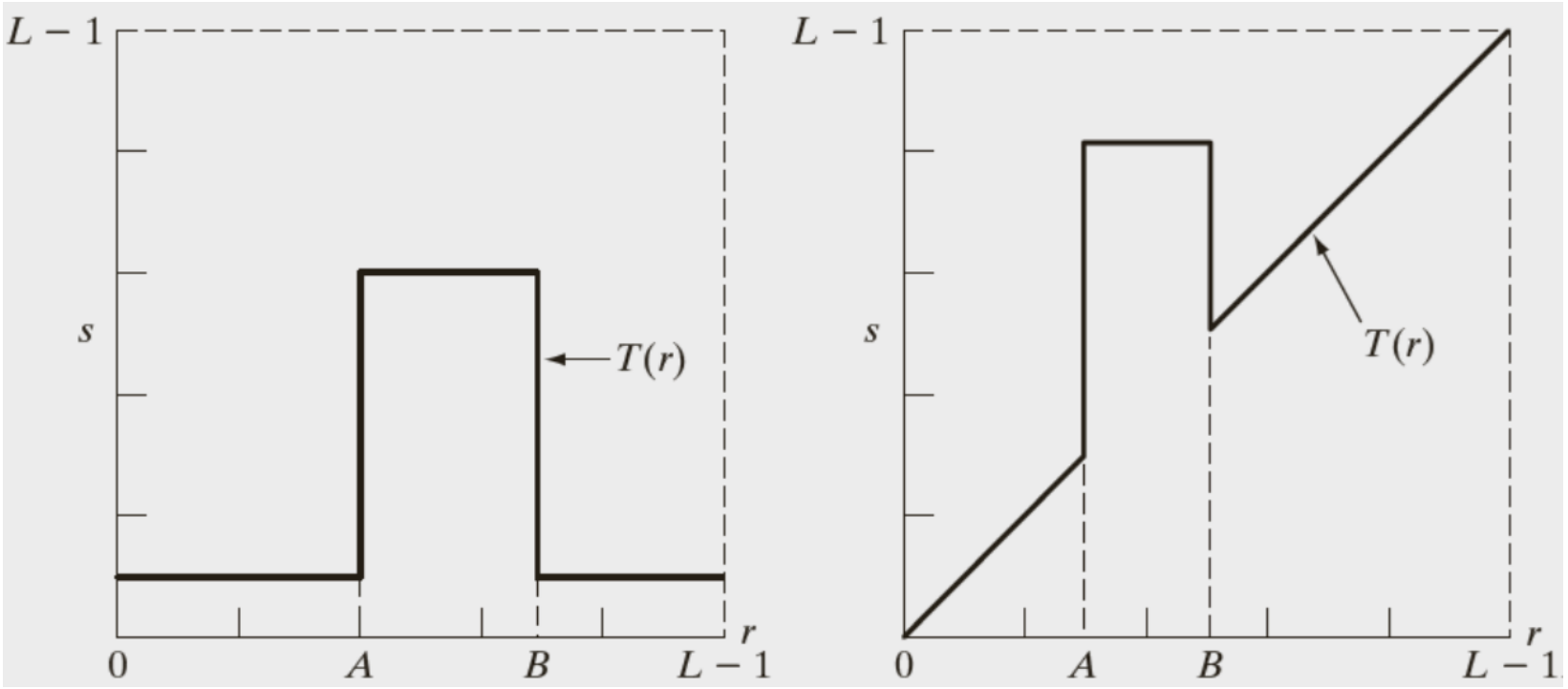
γ=4

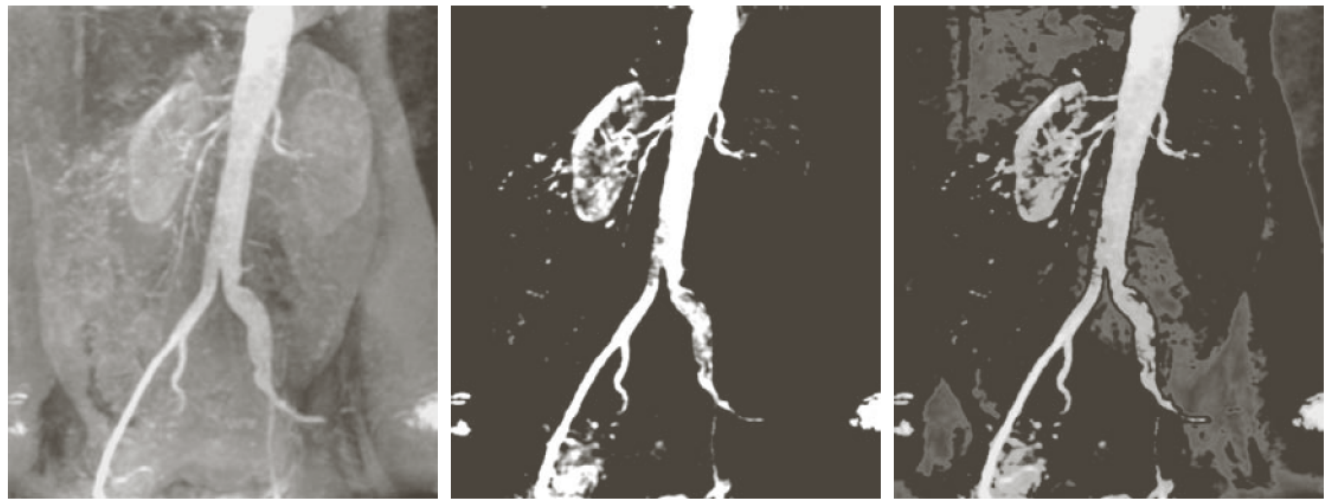
γ=5

Contrast Stretching



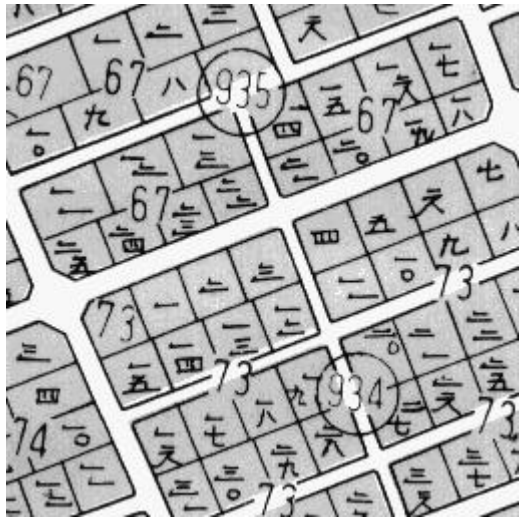
Gray-level Slicing

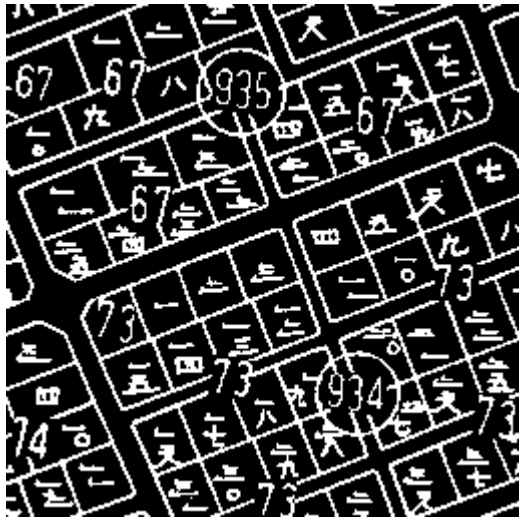




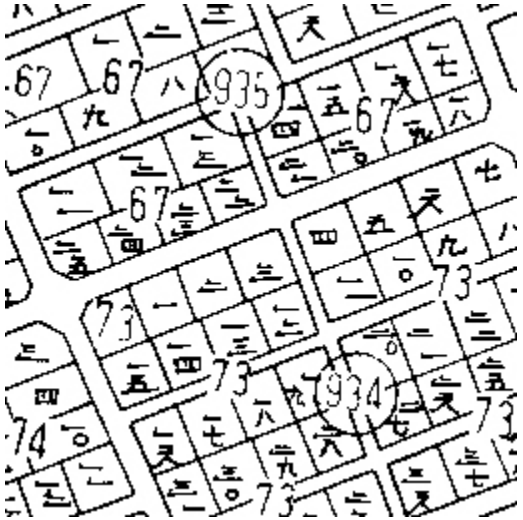
a b c

FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)



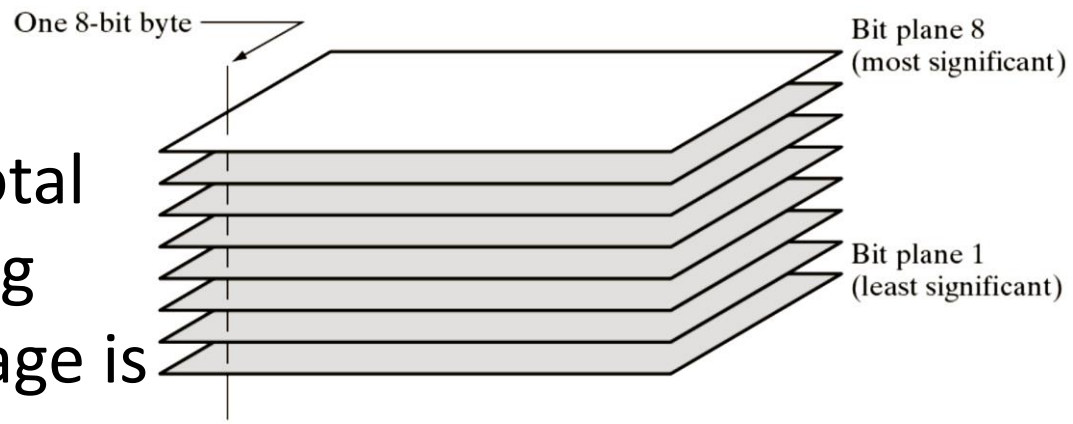


0-160



128-255

Bit-plane Slicing



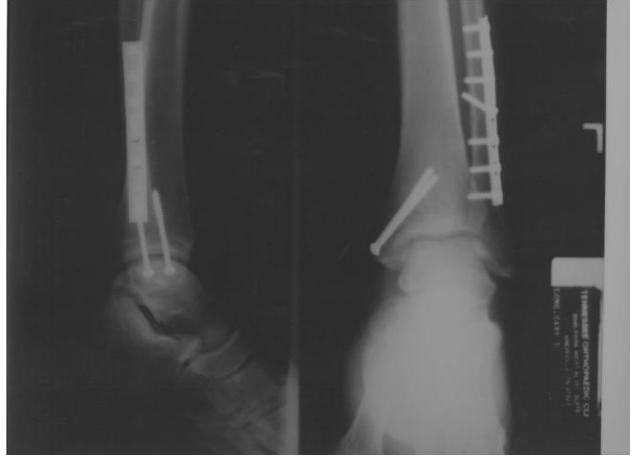
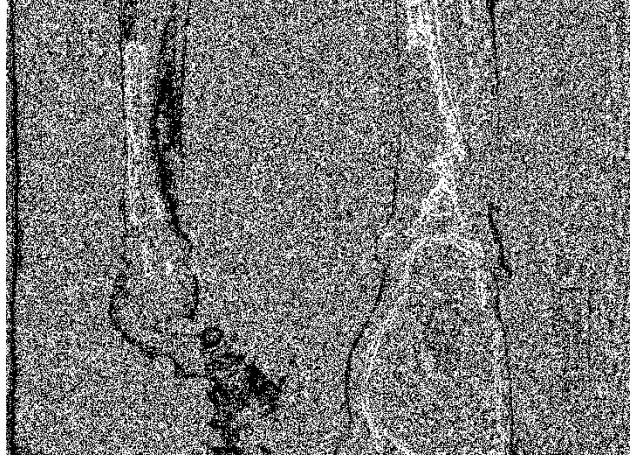
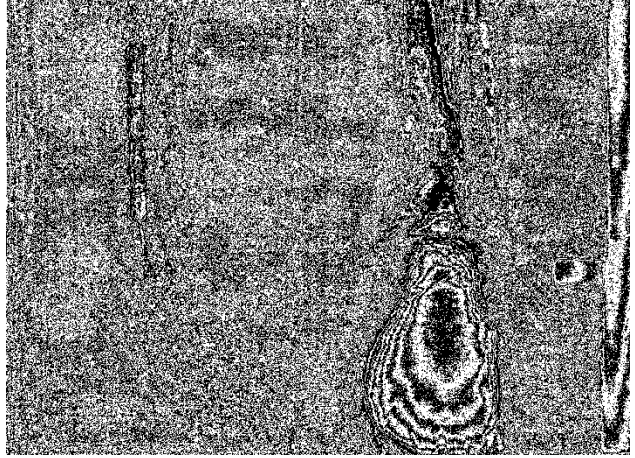
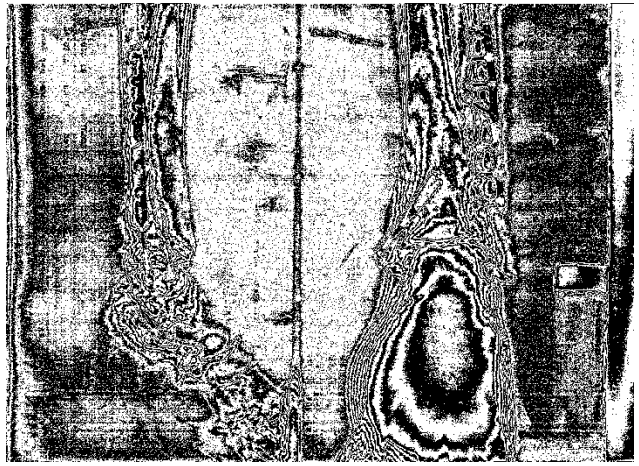
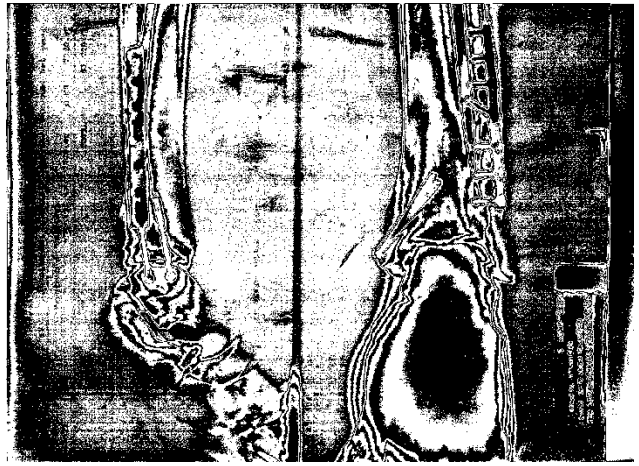
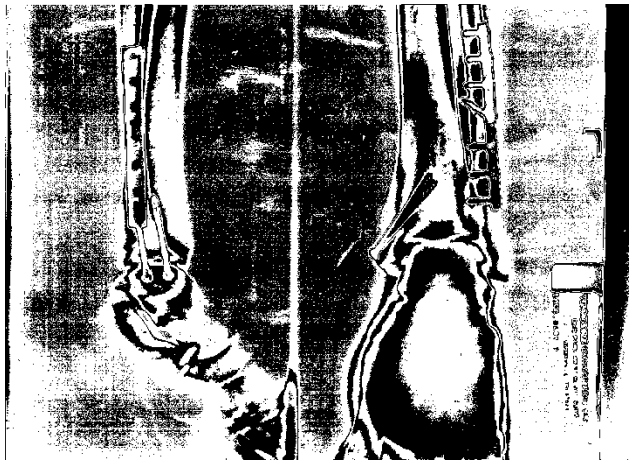
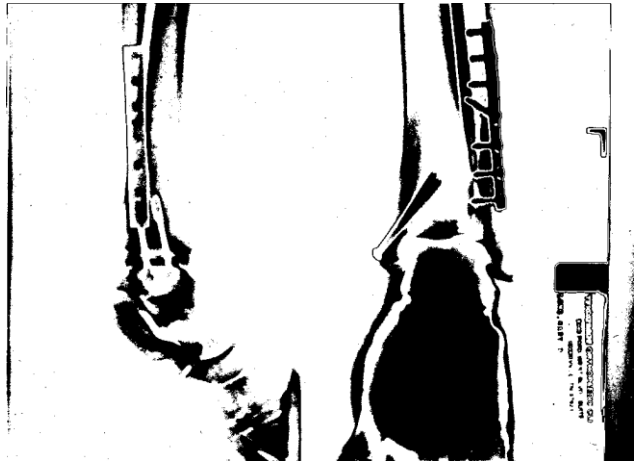
- To highlight the contribution made to the total image appearance by specific bits. i.e. Assuming that each pixel is represented by 8 bits, the image is composed of 8 1-bit planes.

– Plane 1 contains the least significant bit and plane 8 contains the most significant bit.

– Useful for analyzing the relative importance played by each bit of the image

– Only the higher order bits (top four) contain visually significant data. The other bit planes contribute the more subtle details

– Plane 8 corresponds exactly with an image thresholded at gray level 128


AE 57148129 A
E5
THE UNITED STATES OF AMERICA
100
ONE HUNDRED DOLLARS

a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

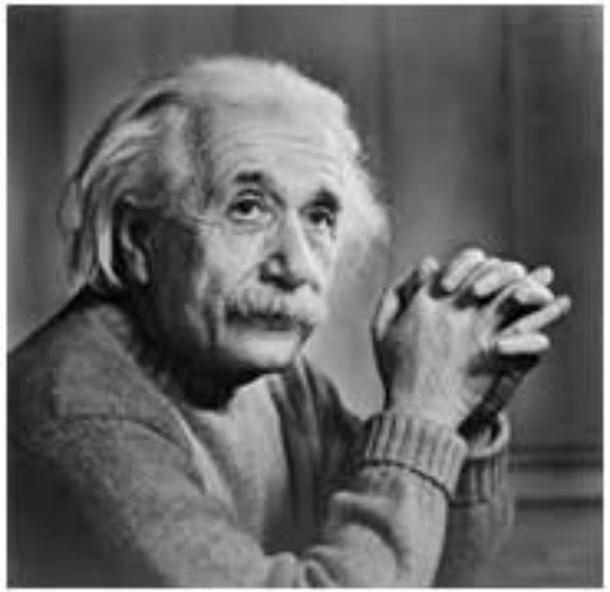
Histogram

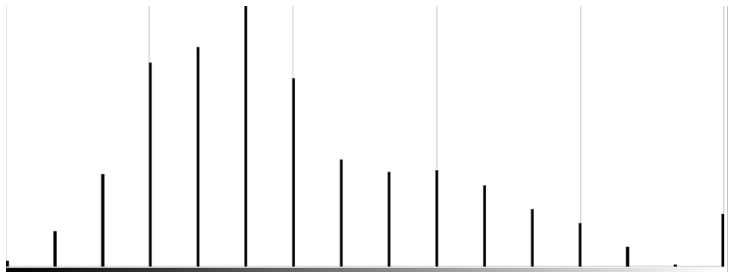
The Histogram of a digital image with the gray-levels in the range [0, L-1] is a discrete function $H(r_k)=n_k$, where the r_k is the k-th gray level and n_k is the number of pixels in the image having gray level r_k.

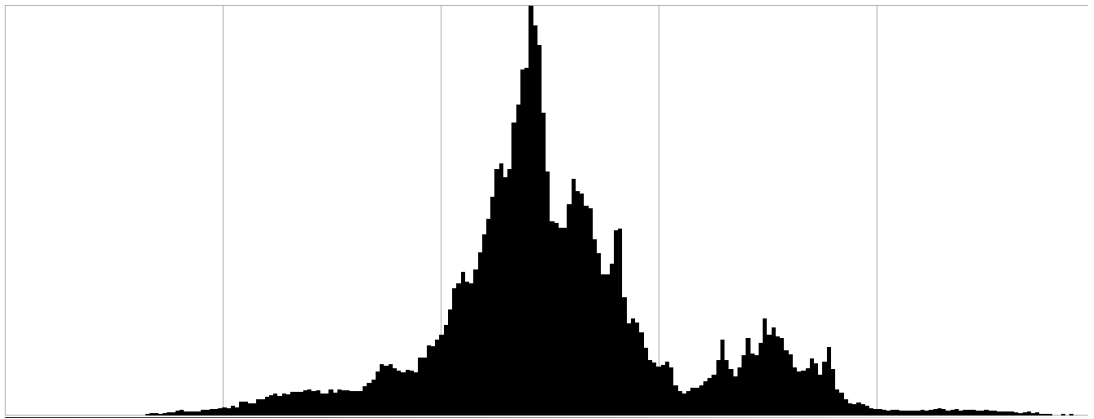
It is a common practice of normalizing the histogram in the range [0, 1], called as the normalized histogram.

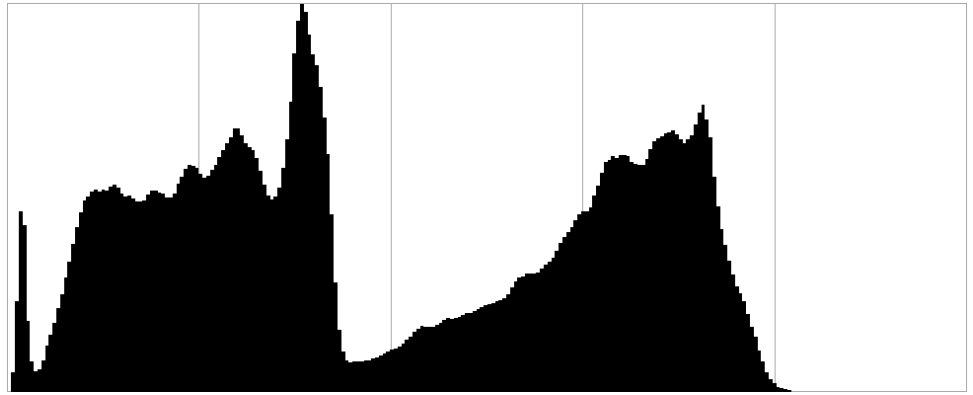
The normalized histogram can be given as $p(r_k)=n_k/n$.

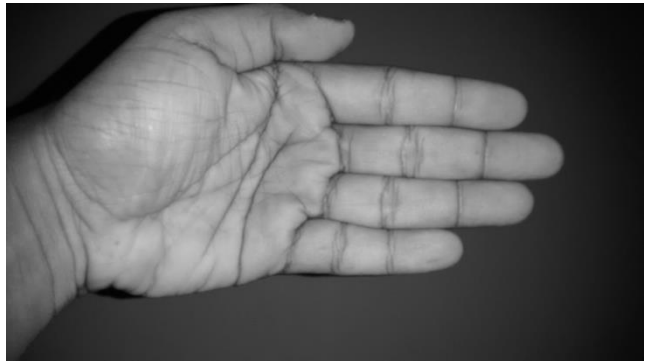
r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

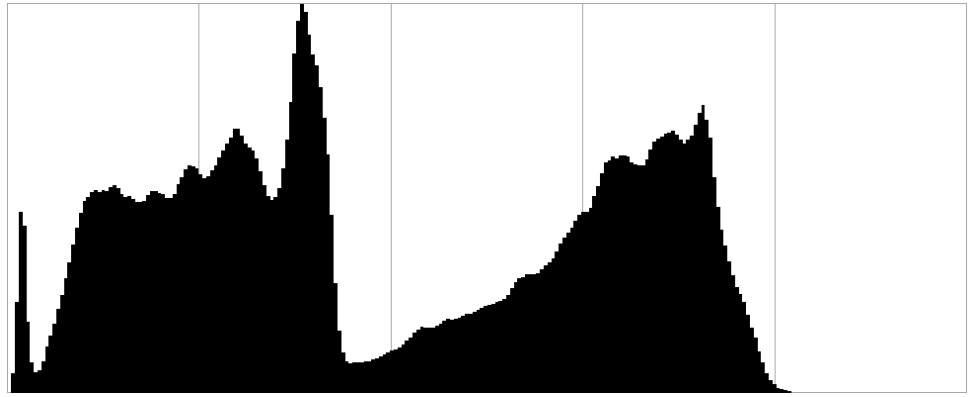


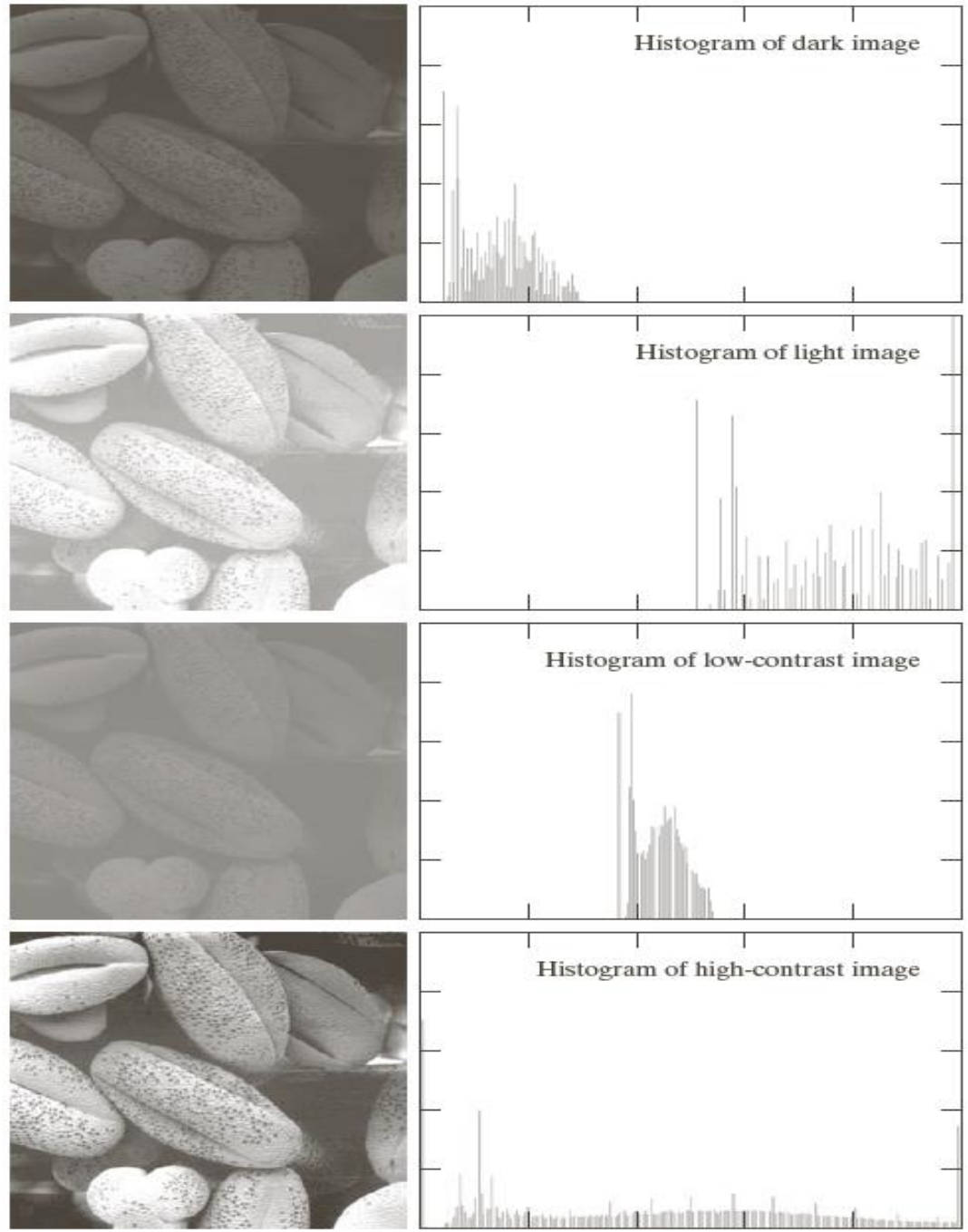




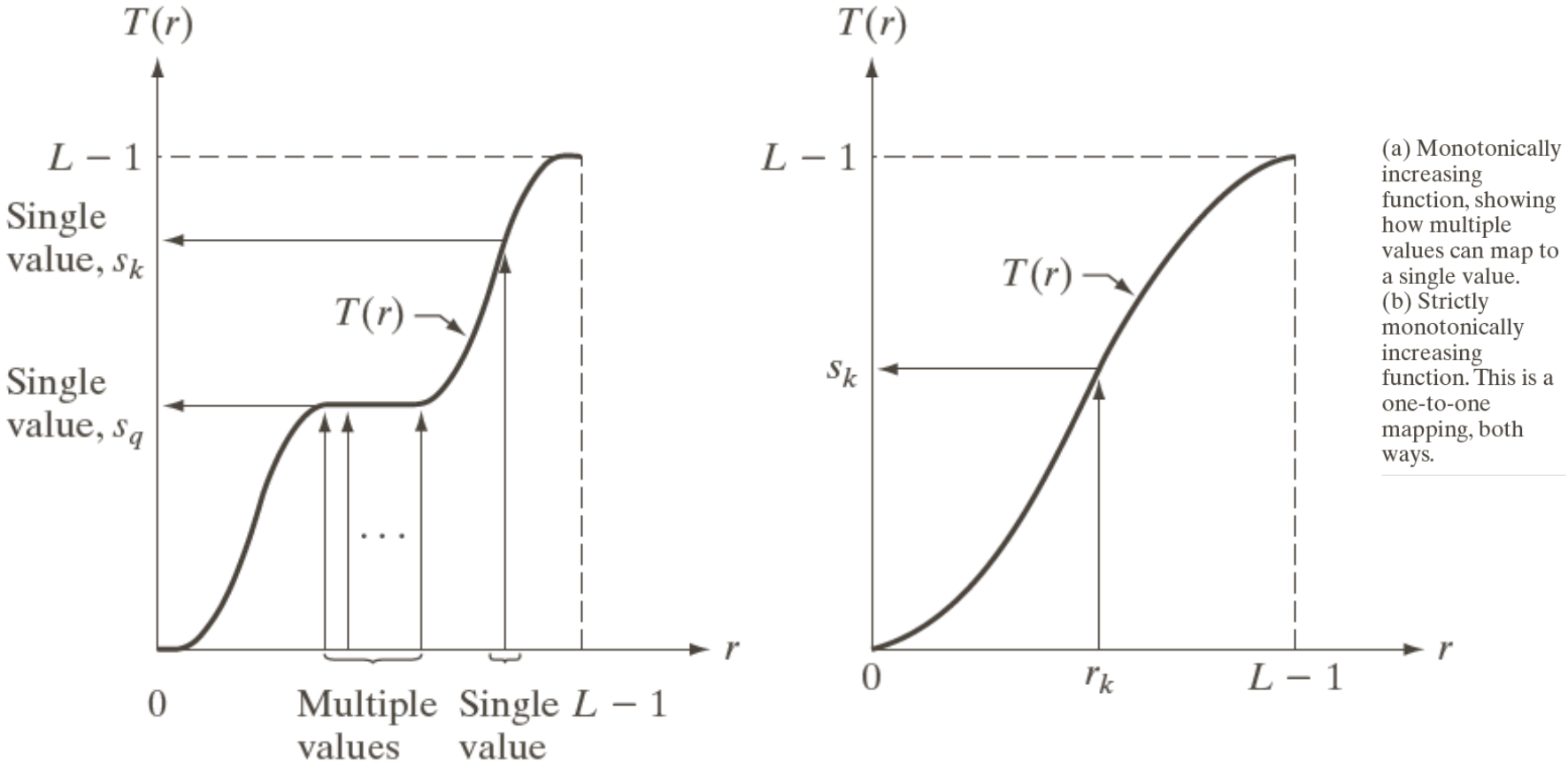




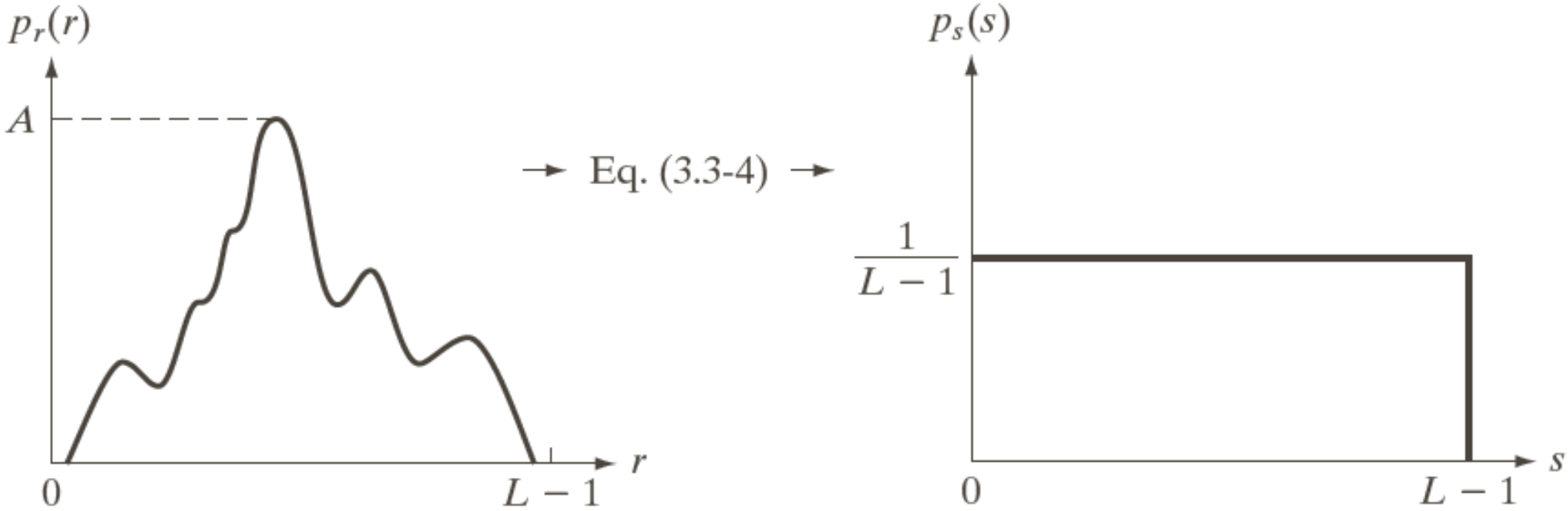



Histogram of dark image
Histogram of light image
Histogram of low-contrast image
Histogram of high-contrast image

Histogram Equalization



(a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.



a b

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0, 1, 2, \ldots, L - 1$$

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \ldots, L - 1.$$

Step1: Compute the histogram

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0

Step2: Compute the Cumulative distribution function (CDF)

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
CDF	0	0	0	6	20	25	25	25

Step3: Compute the normalized Cumulative distribution function (CDF)

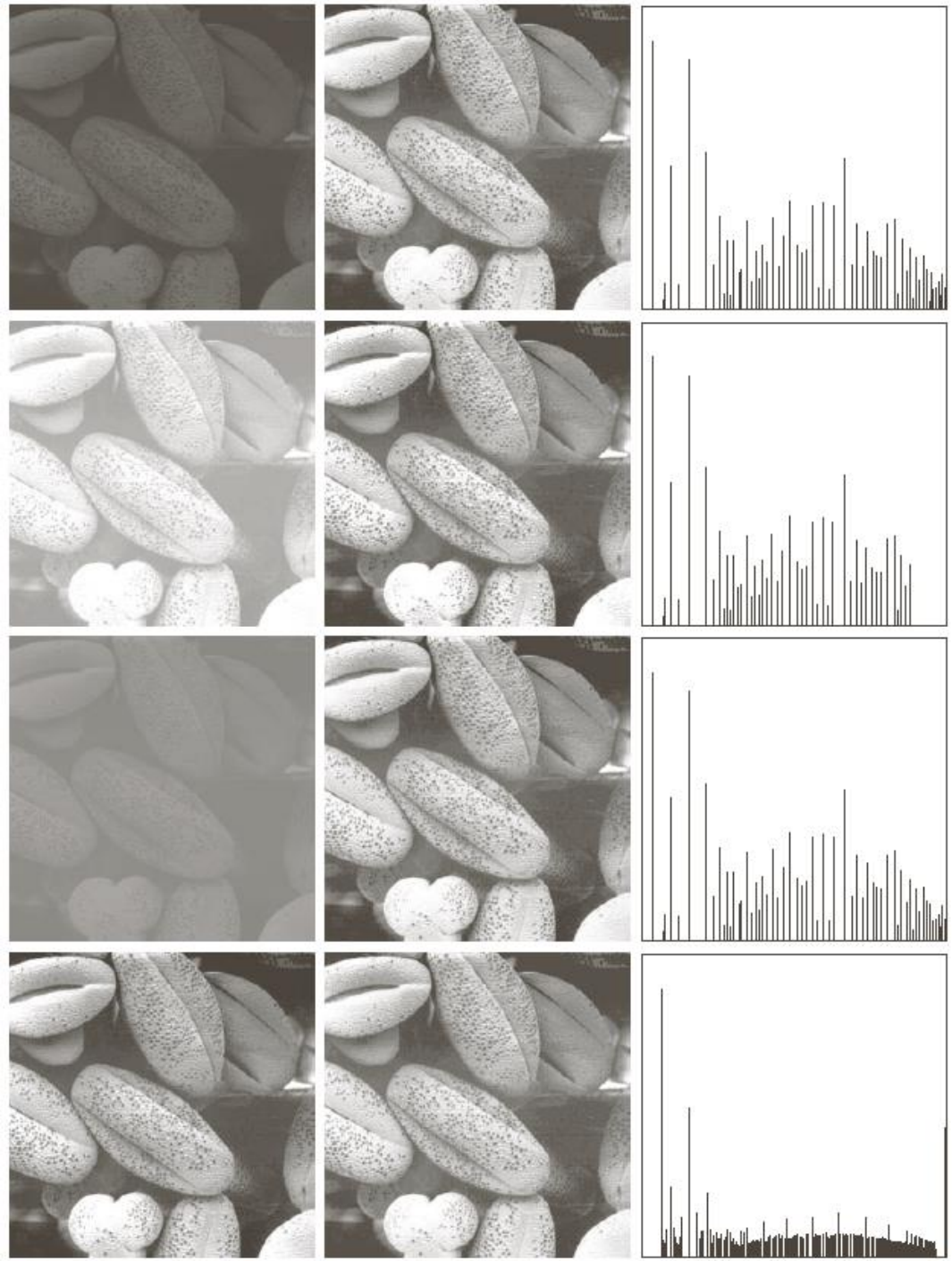
Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25
Normalized CDF	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25

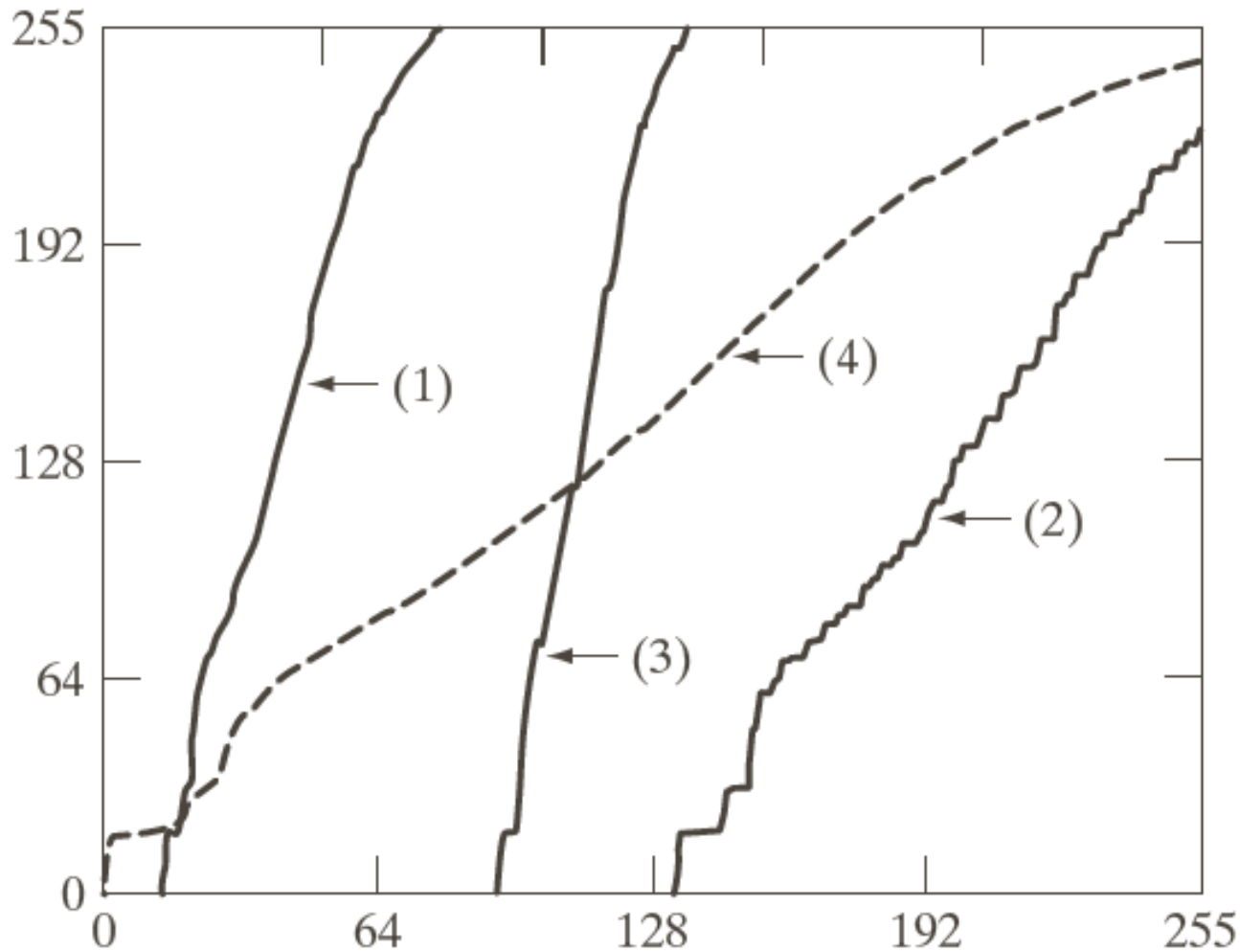
Step4: Compute Equalized image

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25
Normalized CDF	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25
Equalized image	0/25x7 =0	0/25x7 =0	0/25x7= 0	6/25x7= 2	20/25x7= 6	25/25x7= 7	25/25x7 =7	25/25x7=7

Step5: Input and output mapping

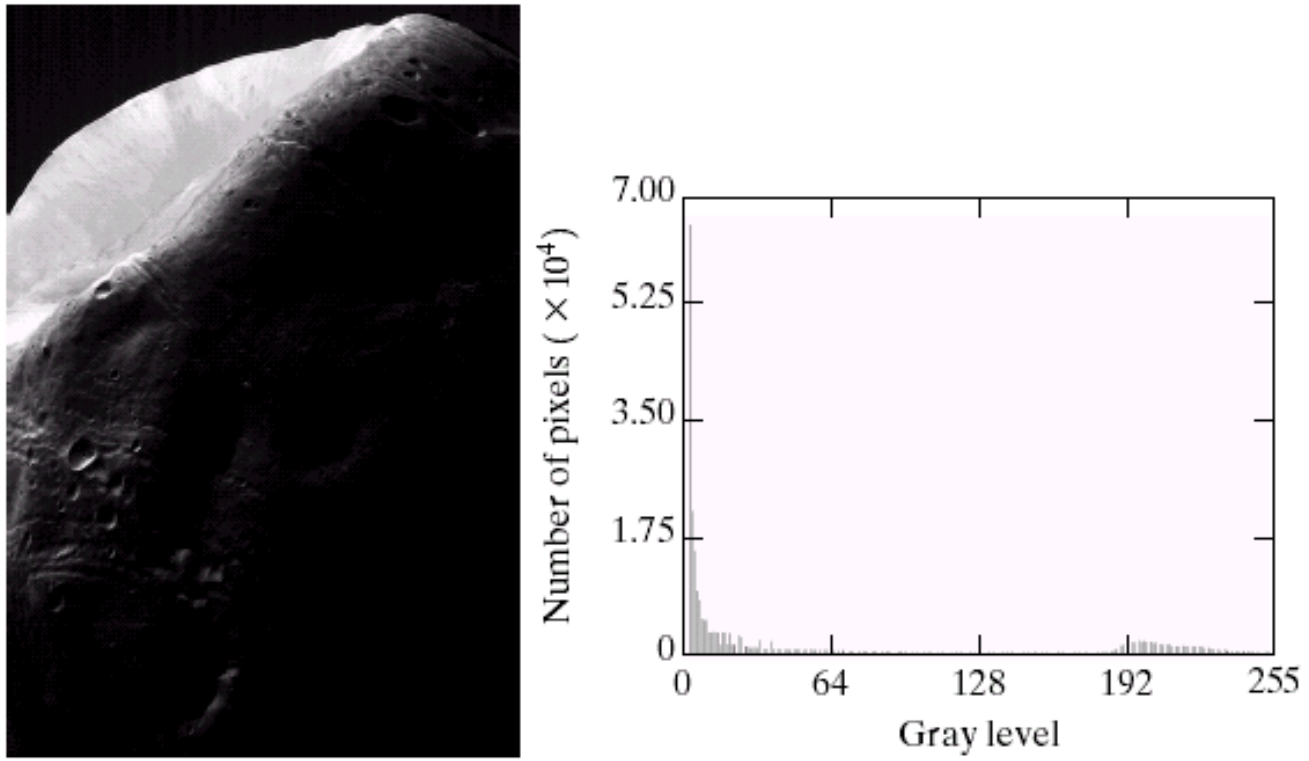
Input pixel value	Output pixel value
0	0
1	0
2	0
3	2
4	6
5	7
6	7
7	7




255
192
128
64
0
0
64
128
192
255
(1)
(2)
(3)
(4)

Histogram matching (specification)

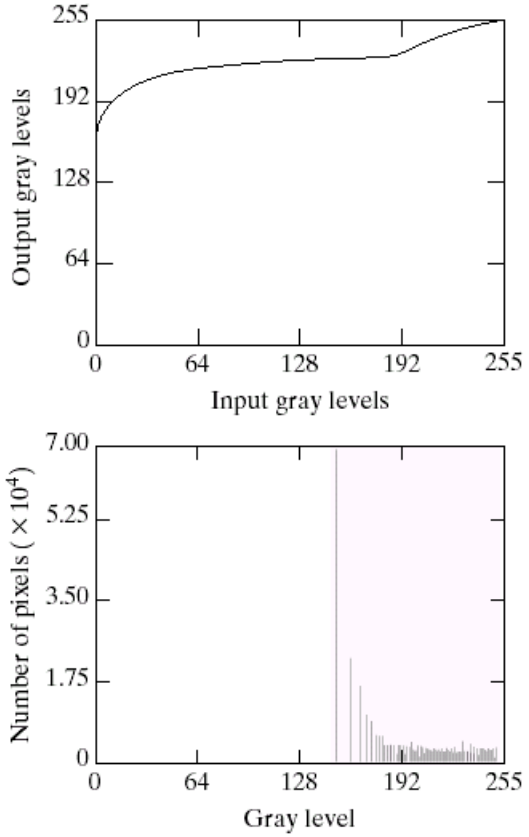
- Enhancement based on a uniform histogram is not the best approach
  - It is useful sometimes to specify the shape of the histogram that we wish to have
- Generate a processed image that has a specified histogram
- Suitable for interactive image enhancement
- Difficulty--build a meaningful histogram



a b

FIGURE 3.20 (a) Image of the Mars moon Photos taken by NASA's *Mars Global Surveyor.* (b) Histogram. (Original image courtesy of NASA.)

Image Enhancement in the Spatial Domain



a b
c

FIGURE 3.21
(a) Transformation function for histogram equalization. (b) Histogram-equalized image (note the washed-out appearance). (c) Histogram of (b).

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \ldots, L-1.$$

$$S_k=T(r_k)=(L-1)\sum_{j=0}^{k} n_j/n$$

$$S_k=T(r_k)=(L-1)\sum_{j=0}^{k} p_r(rj)$$

For any given distribution/histogram $P_z(z_i)$, where z_i is the input gray value.

$$G(z_q)=(L-1)\sum_{i=0}^{q} p_z(zi)$$

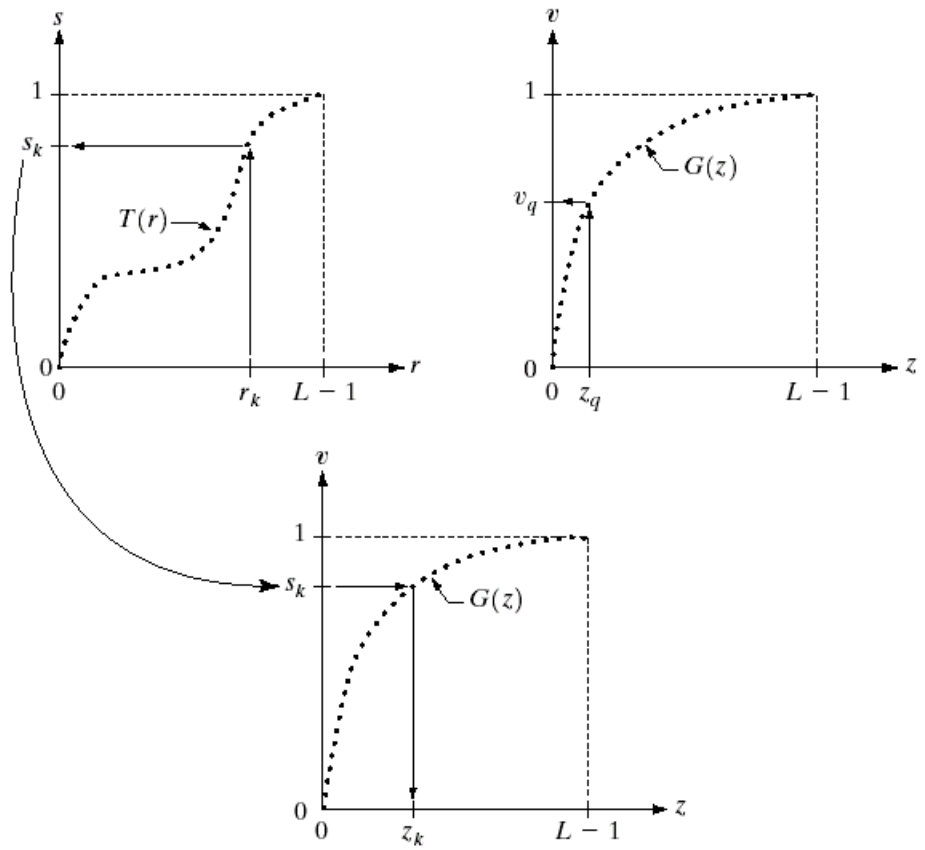
Mapping can be done by

$$G(z_q)=S_k \qquad z_q=G^{-1}(S_k)$$

Image Enhancement in the Spatial Domain

a b
c

FIGURE 3.19
(a) Graphical interpretation of mapping from r_k to s_k via $T(r)$. (b) Mapping of z_q to its corresponding value v_q via $G(z)$. (c) Inverse mapping from s_k to its corresponding value of z_k.



Original Image

Gray level	0	1	2	3	4	5	6	7
No. of Pixels	8	10	10	2	12	16	4	2

Desired Image

Gray level	0	1	2	3	4	5	6	7
No. of Pixels	0	0	0	0	20	20	16	8

Histogram Equalization (Input Image)

Gray level(r_k)	No. of Pixels(n_k)	PDF(n_k/N) $P_r(r_k)$	CDF	(L-1)*CDF	S
0	8	0.13	0.13	0.91	1
1	10	0.16	0.29	2.03	2
2	10	0.16	0.45	3.15	3
3	2	0.03	0.48	3.36	3
4	12	0.18	0.66	4.62	5
5	16	0.25	0.91	6.37	6
6	4	0.06	0.97	6.79	7
7	2	0.03	1.0	7	7
	64	1			

Histogram Equalization (Target Image)

Gray level(r_k)	No. of Pixels(n_k)	PDF(n_k/N) $P_r(r_k)$	CDF	(L-1)*CDF	G(z)
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	20	0.31	0.31	2.17	2
5	20	0.31	0.62	4.34	4
6	16	0.25	0.87	6.09	6
7	8	0.13	1.0	7	7
	64	1			

Mapping

Gray Scale	S	G(z)	Map
0	1	0	4
1	2	0	4
2	3	0	5
3	3	0	5
4	5	2	6
5	6	4	6
6	7	6	7
7	7	7	7

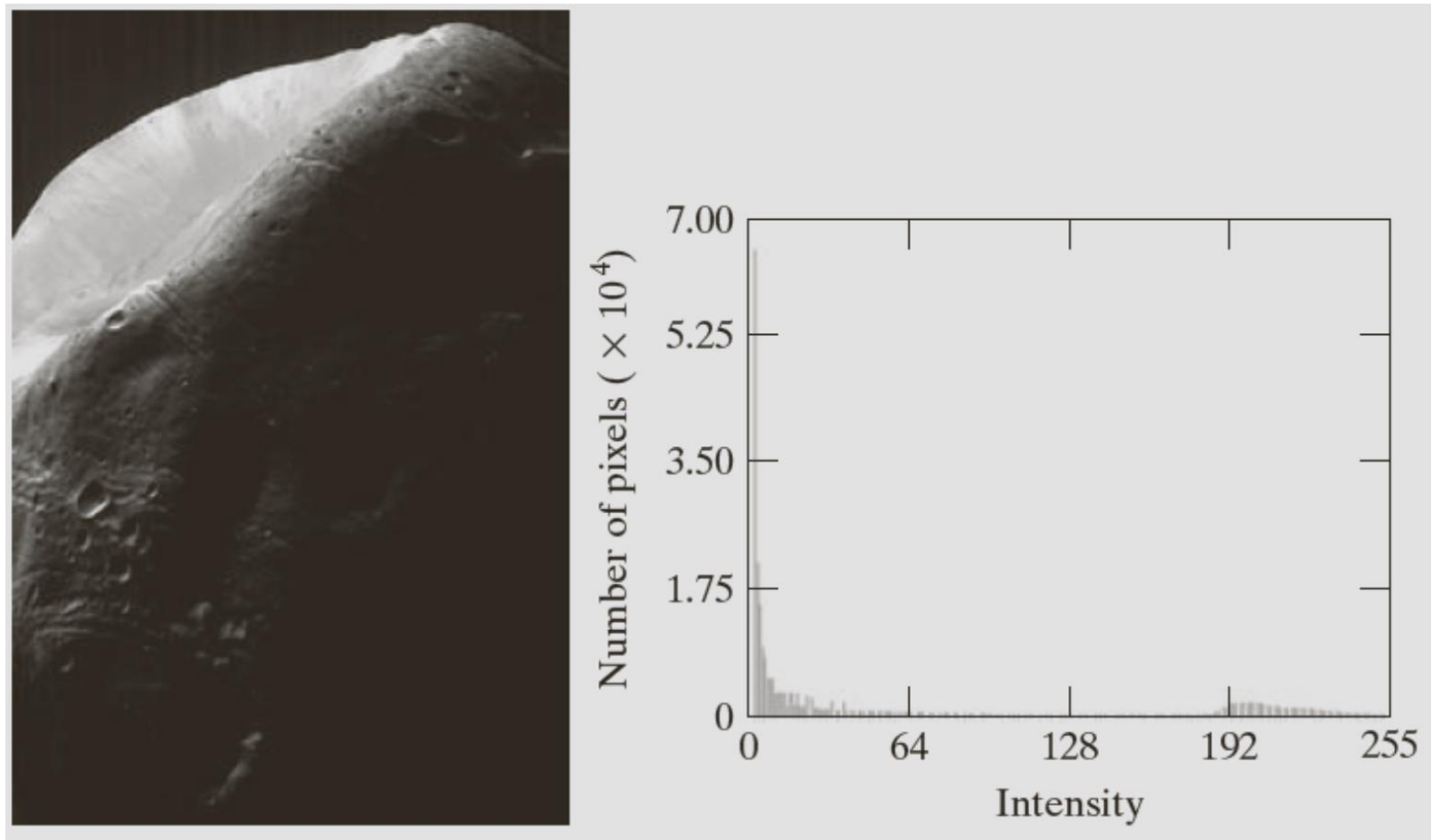
Gray Scale	S	G(z)	Map
0	1	0	4
1	2	0	4
2	3	0	5
3	3	0	5
4	5	2	6
5	6	4	6
6	7	6	7
7	7	7	7

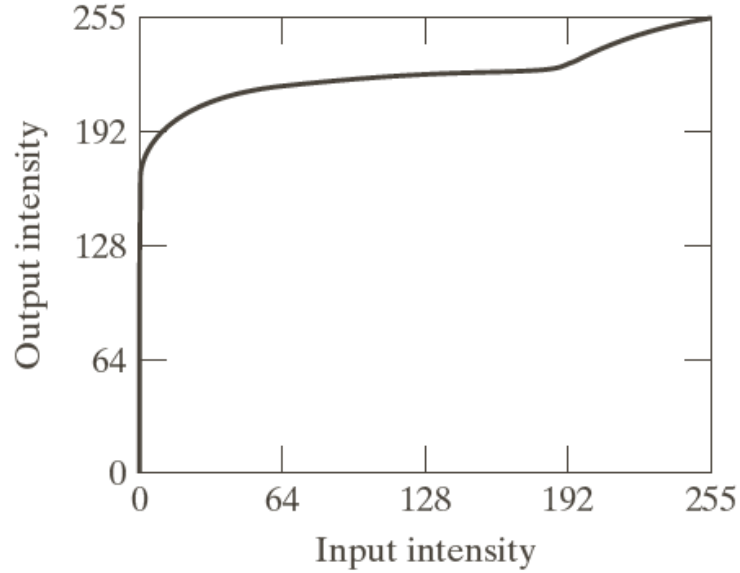
Original Image

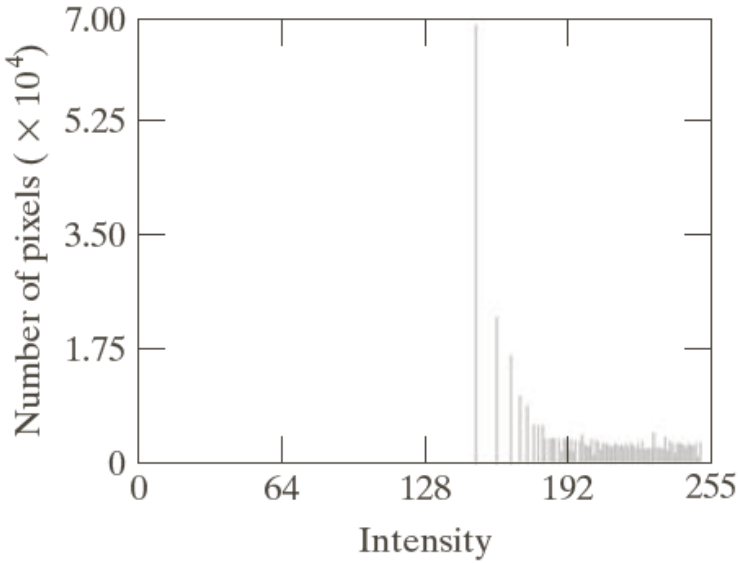
Gray level	0	1	2	3	4	5	6	7
No. of Pixels	8	10	10	2	12	16	4	2

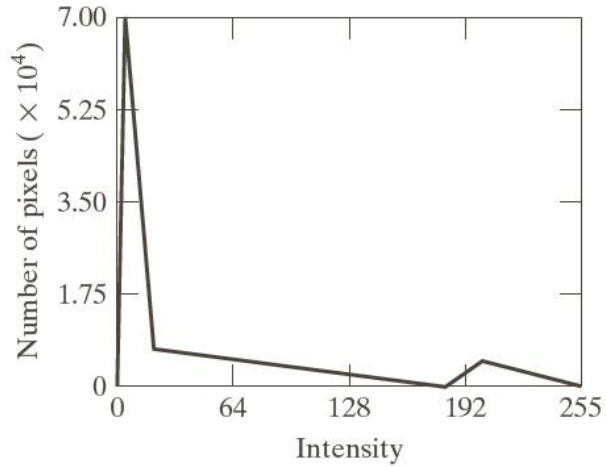
Modified Image

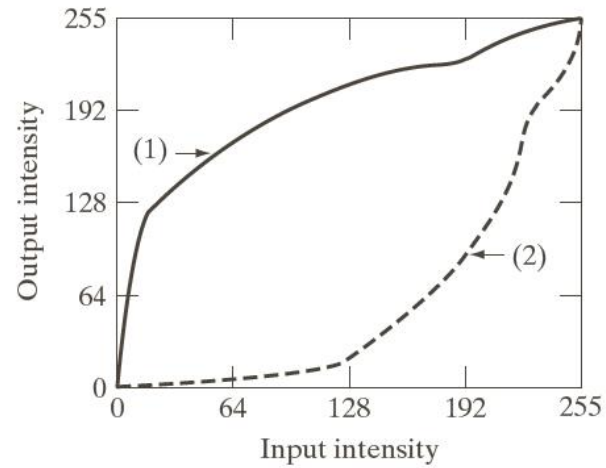
Gray level	0	1	2	3	4	5	6	7
No. of Pixels	0	0	0	0	18	12	28	6

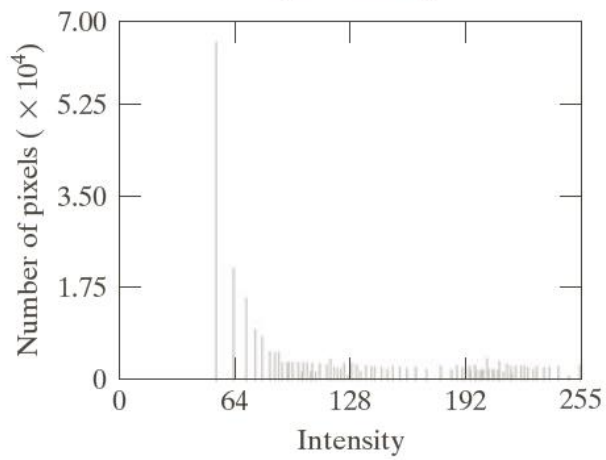

7.00
5.25
3.50
1.75
0
Number of pixels (× 10^4)
0
64
128
192
255
Intensity


255
192
128
64
0
Output intensity
0
64
128
192
255
Input intensity


7.00
5.25
3.50
1.75
0
Number of pixels (× 10^4)
0
64
128
192
255
Intensity


7.00
5.25
3.50
1.75
0
Number of pixels (× 10^4)
0
64
128
192
255
Intensity

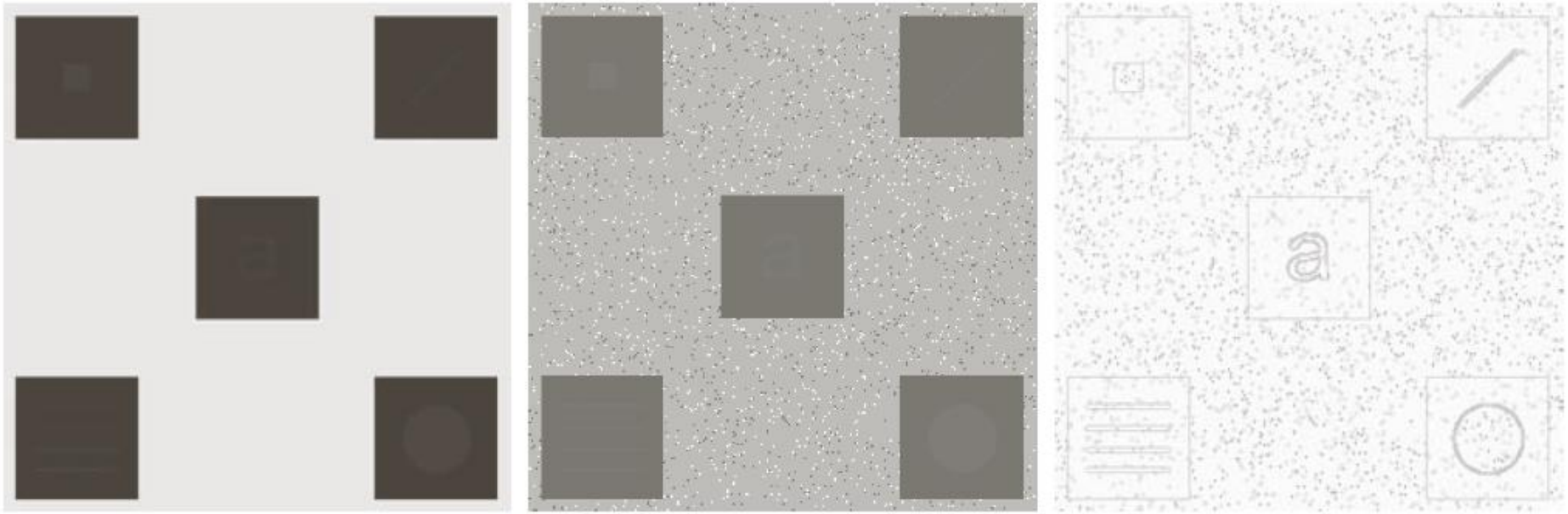

255
192
128
64
0
Output intensity
(1)
(2)
0
64
128
192
255
Input intensity


7.00
5.25
3.50
1.75
0
Number of pixels (× 10^4)
0
64
128
192
255
Intensity

Local Histogram Processing

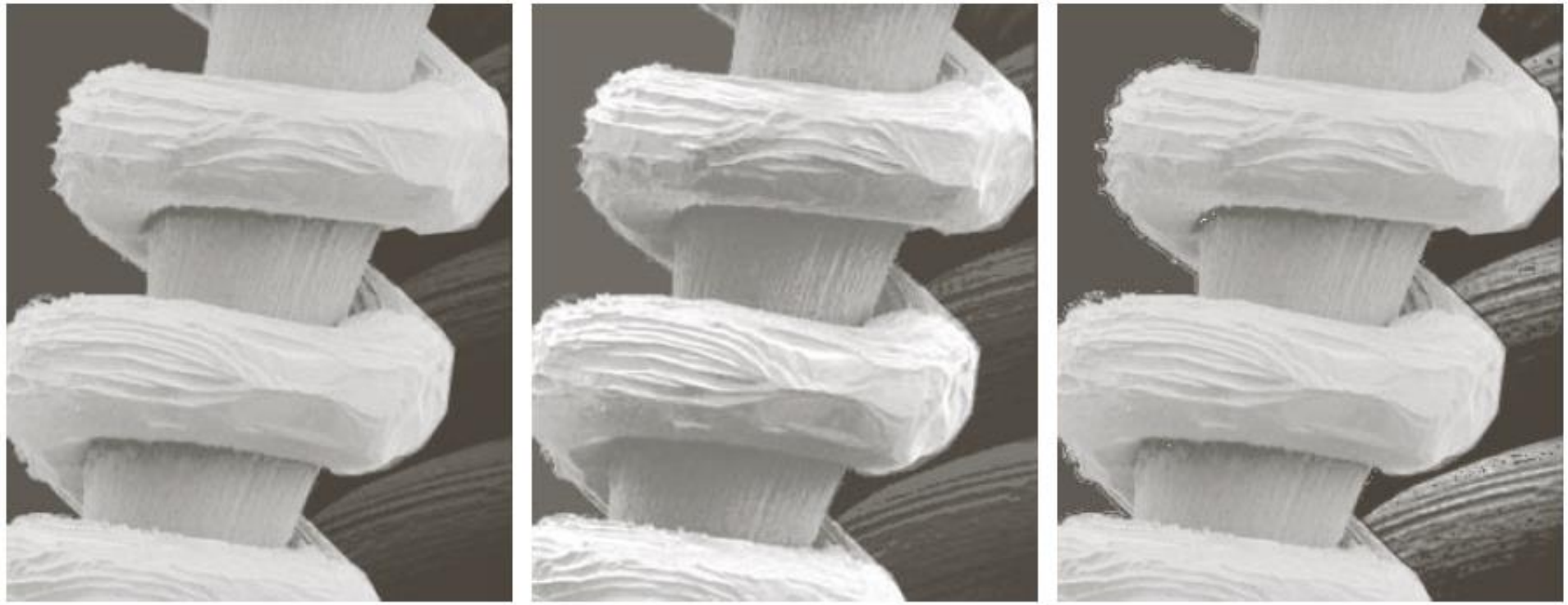
- Define a neighborhood and move its center from pixel to pixel
- At each location, the histogram of the points in the neighborhood is computed. Either histogram equalization or histogram specification transformation function is obtained
- Map the intensity of the pixel centered in the neighborhood
- Move to the next location and repeat the procedure

Local Histogram Processing



a b c

FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3×3.



a b c

FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Convolution and Correlation

Convolution and correlation operations are basically used to extract information from images.

Convolution and correlation are basically linear and shift-invariant operations

The term *linear* indicates that a pixel is replaced by the linear combination of its neighbors.

The term *shift invariant* indicates that the same operation is performed at every point in the image.

Convolution

Convolution is basically a mathematical operation where each value in the output is expressed as the sum of values in the input multiplied by a set of weighting coefficients.

Depending upon the weighting coefficients, convolution operation is used to perform spatial domain low-pass and high-pass filtering of the image.

An image can be either smoothened or sharpened by convolving the image with respect to low-pass and high-pass filter mask respectively.

Convolution has a multitude of applications including image filtering, image enhancement, image restoration, feature extraction and template matching.

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

The input matrix x(m, n) and h(m, n). Perform the linear convolution between these two matrices.

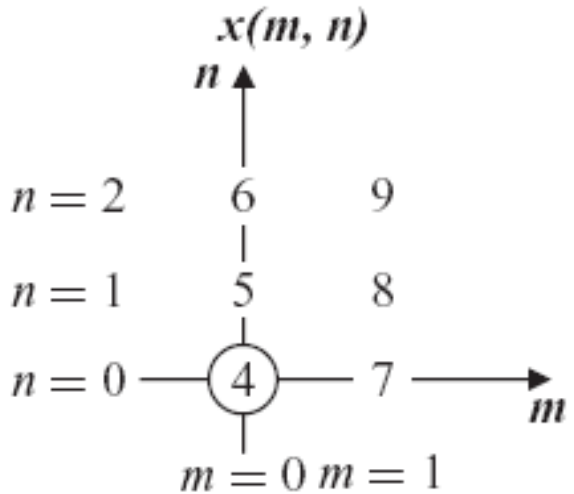
$$x(m,n)=\begin{pmatrix}4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$$

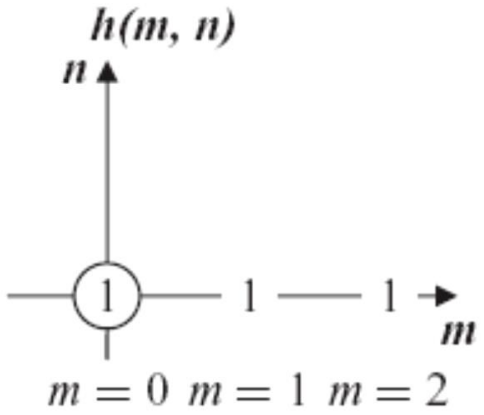
$$x(m,n)=\begin{pmatrix}\underset{4}{(0,0)} & \underset{5}{(0,1)} & \underset{6}{(0,2)}\\ \underset{7}{(1,0)} & \underset{8}{(1,1)} & \underset{9}{(1,2)}\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}\underset{1}{(0,0)}\\ \underset{1}{(1,0)}\\ \underset{1}{(2,0)}\end{pmatrix}$$

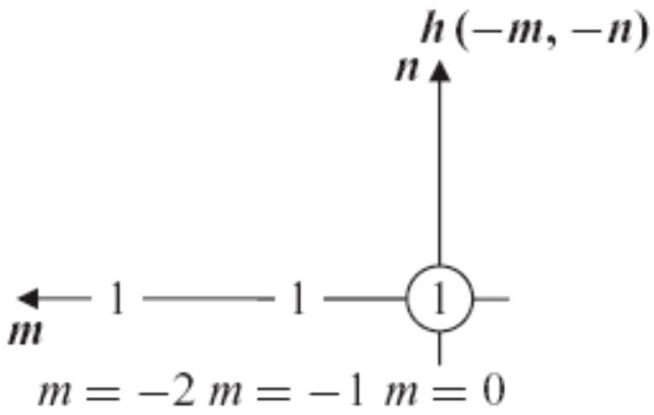
$$\text{Dimension of reslutant matrix}=\begin{cases}(\text{No. of rows of } x(m,n)+\text{No. of rows of } h(m,n)-1)\\ \times\\ (\text{No. of columns of } x(m,n)+\text{No. of columns of } h(m,n)-1)\end{cases}$$

Dimension of resultant matrix = (2 + 3 − 1) × (3 + 1 − 1) = 4 × 3

$$y(m,n) = \begin{pmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \\ y(2,0) & y(2,1) & y(2,2) \\ y(3,0) & y(3,1) & y(3,2) \end{pmatrix}$$

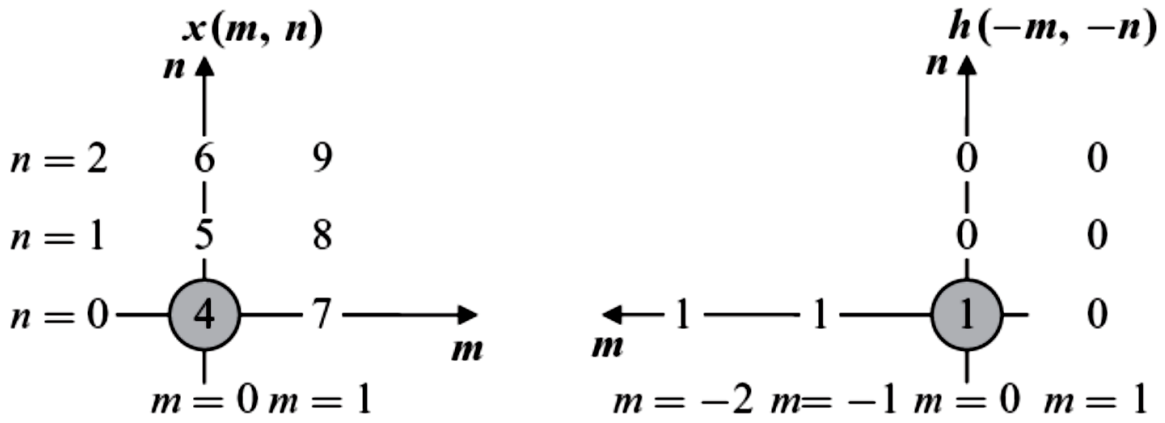






***To determine the value of y*(0, 0)**

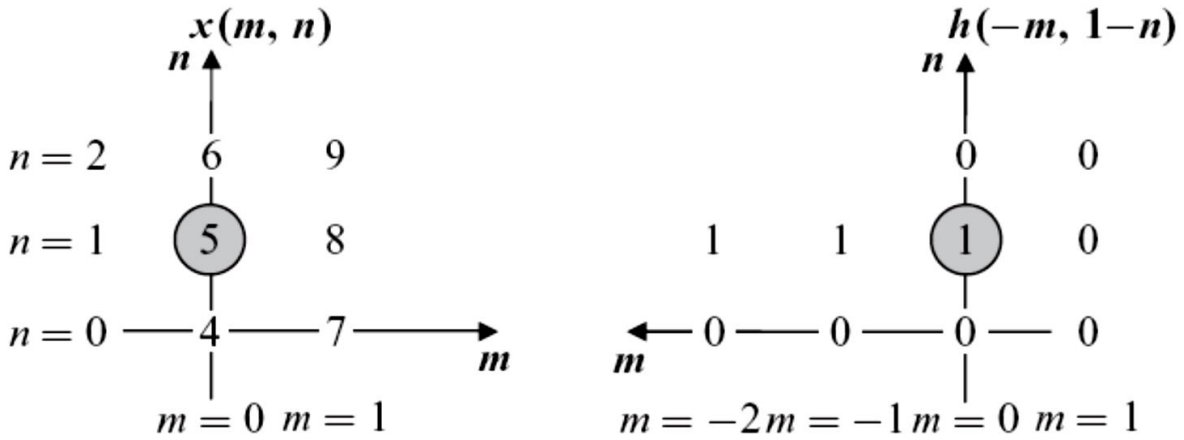
The common values of $x(m, n)$ and $h(-m, -n)$ are multiplied and then added to get the value of y (0, 0). The shaded circle indicates the common area between the two signals.



The value $y(0, 0)$ is obtained as $y(0, 0) = 4 \times 1 = 4$.

***To determine the value of y*(0,1)**

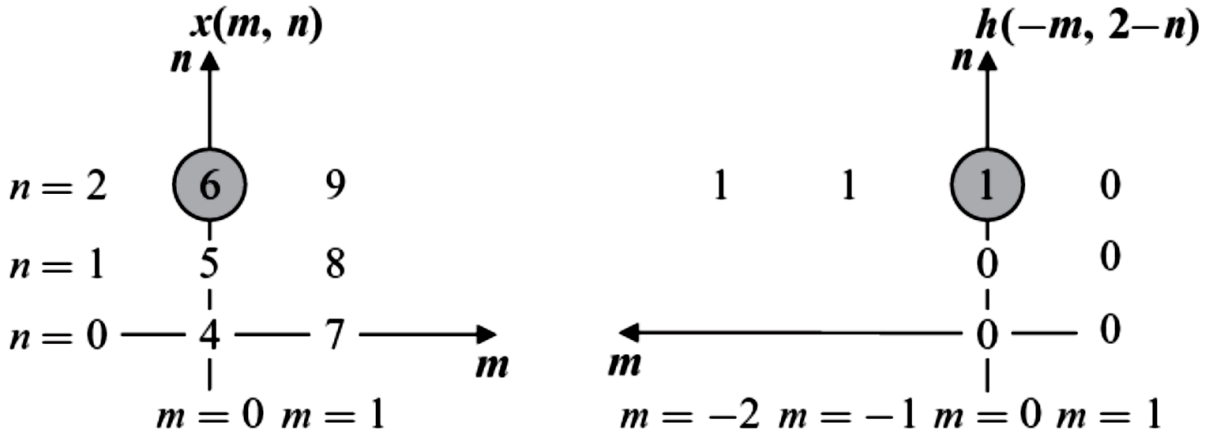
Now the signal $h(-m, -n)$ is shifted along the 'n' axis by one unit to get $h(-m, 1-n)$ and $x(m, n)$ is unaltered. The common value between the two signals is multiplied and then added to get y (0, 1).



The value of $y(0, 1)$ is given as $y(0, 1) = 5 \times 1 = 5$.

***To determine the value of y*(0, 2)**

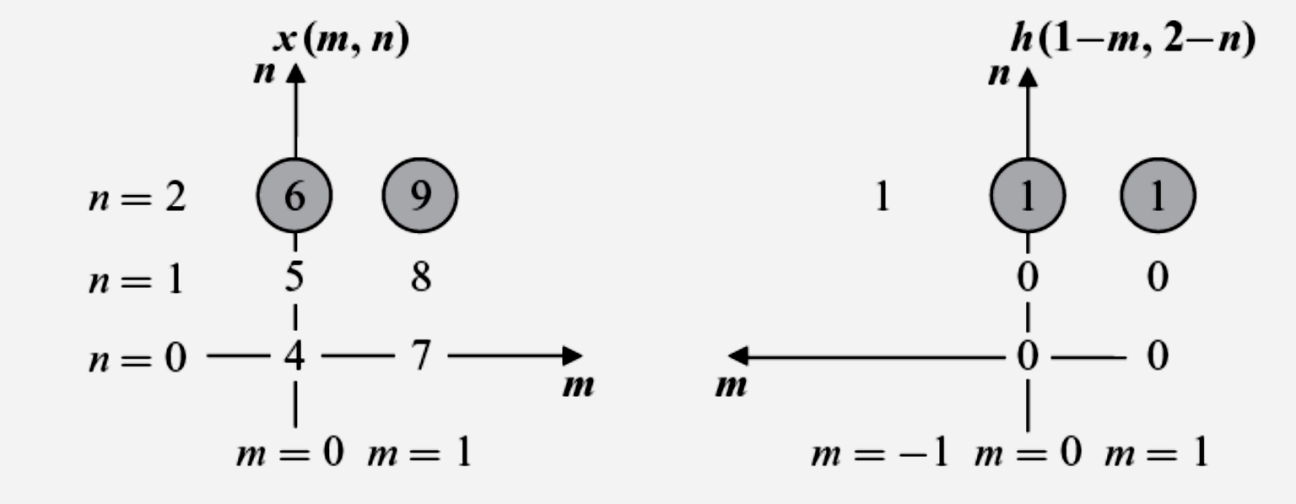
The value of $h(-m, -n)$ is shifted by two units along the 'n' axis to get $h(-m, 2-n)$. Then, the common values between $x(m, n)$ and $h(-m, 2-n)$ are multiplied and then added to get $y(0, 2)$.



The resultant value of $y(0, 2)$ is $y(0, 2) = 6 \times 1 = 6$.

***To determine the value of y*(1, 2)**

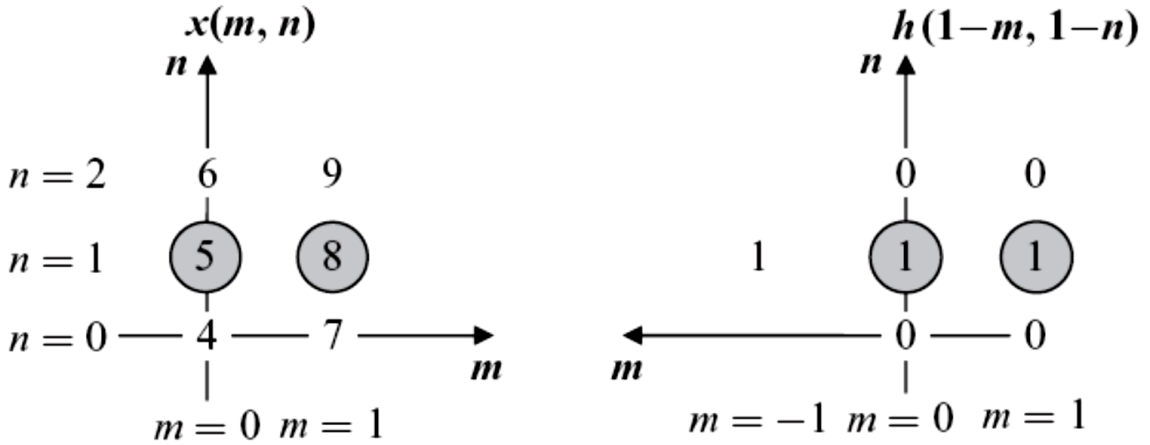
Here, $h(-m, 2-n)$ is shifted along the 'm' axis to one unit towards right to get $h(1-m, 2-n)$. Then the common values between $x(m, n)$ and $h(1-m, 2-n)$ are multiplied and added to get $y(1, 2)$.



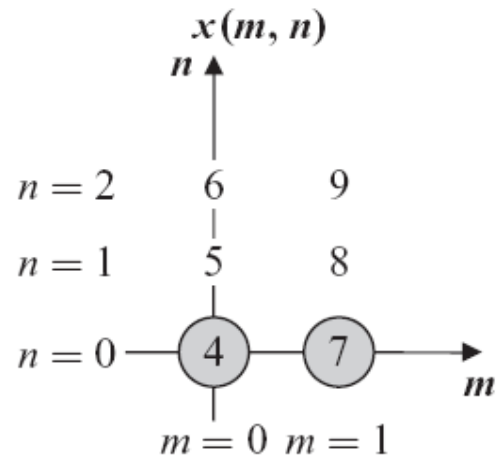
The final value of $y(1, 2) = 6 \times 1 + 9 \times 1 = 15$.

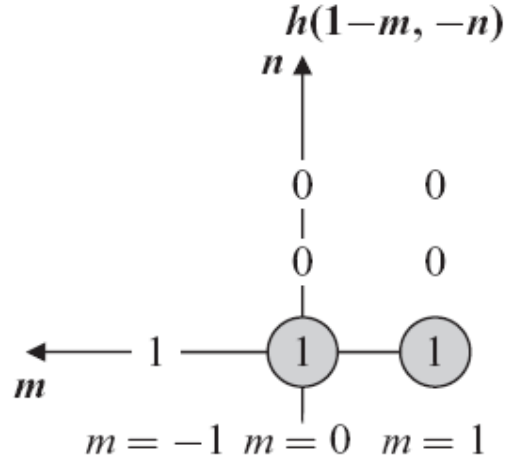
***To determine the value of y*(1, 1)**

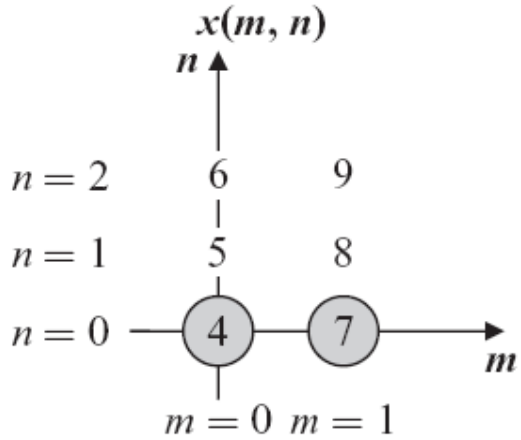
Now the value of $h(1-m, 2-n)$ is shifted down along the 'n' axis to get $h(1-m, 1-n)$. The common values between $x(m, n)$ and $h(1-m, 1-n)$ are multiplied and added to get $y(1, 1)$.

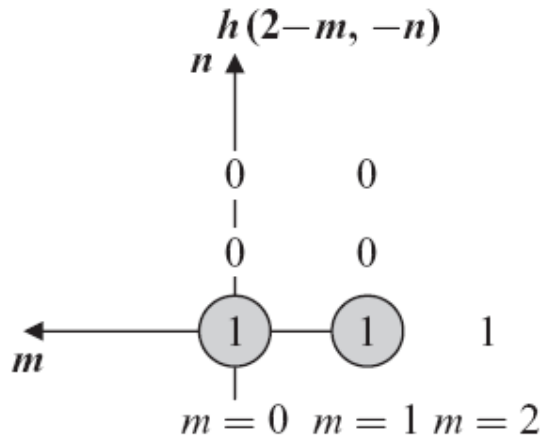


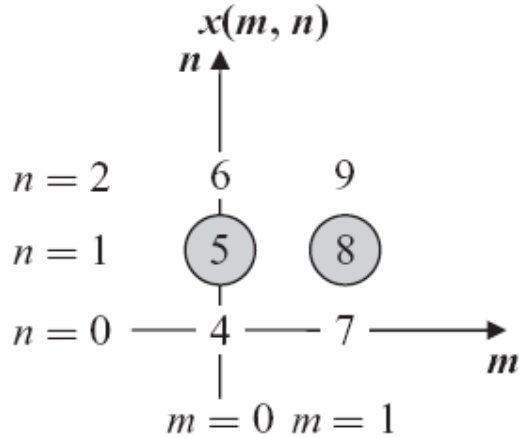
The value of $y(1, 1)$ is obtained as $y(1, 1) = 5 \times 1 + 8 \times 1 = 13$.

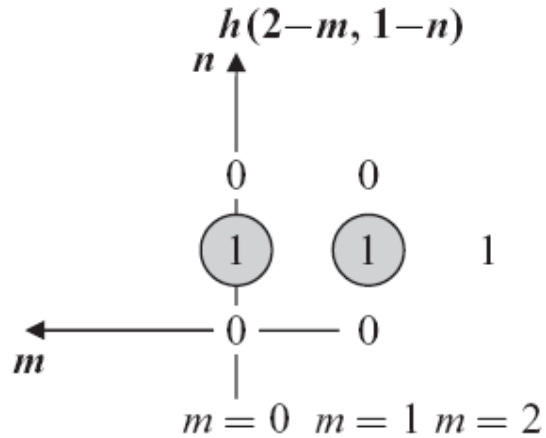

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0
m = 1

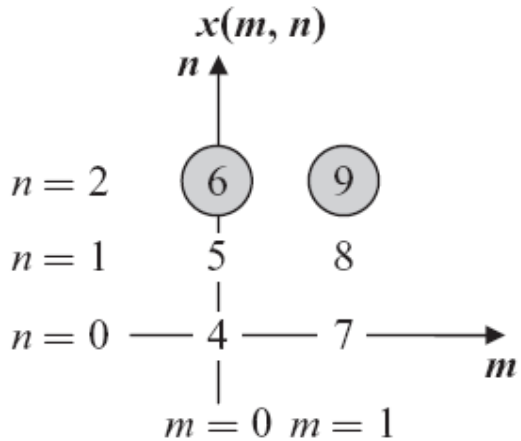

h(1−m, −n)
n
0
0
0
0
1
1
1
m
m = −1
m = 0
m = 1

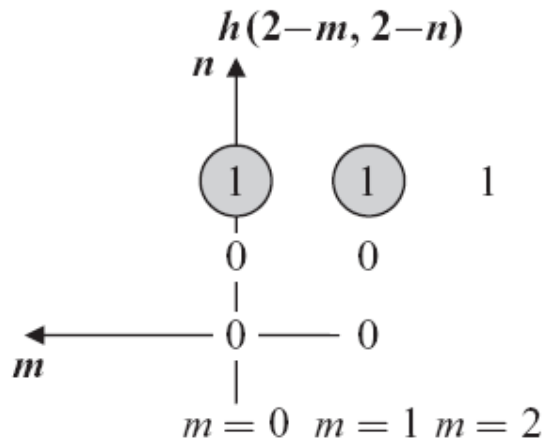

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0
m = 1

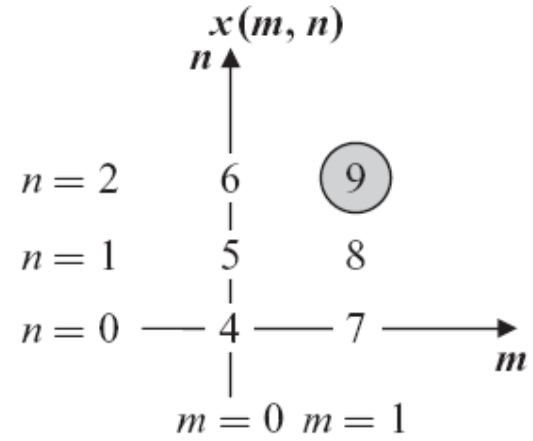

h (2−m, −n)
n
0
0
0
0
1
1
1
m
m = 0
m = 1
m = 2

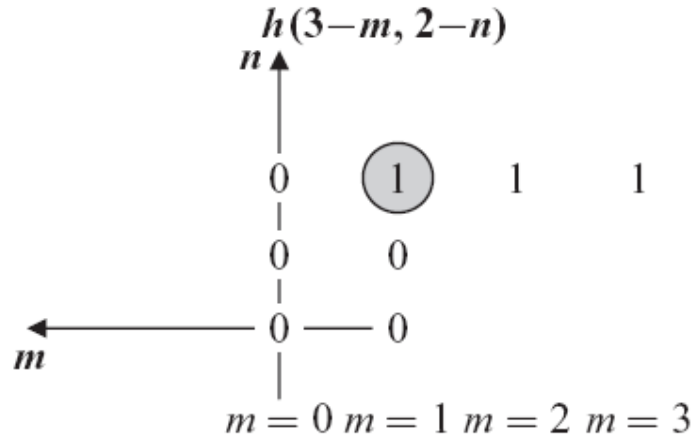

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1

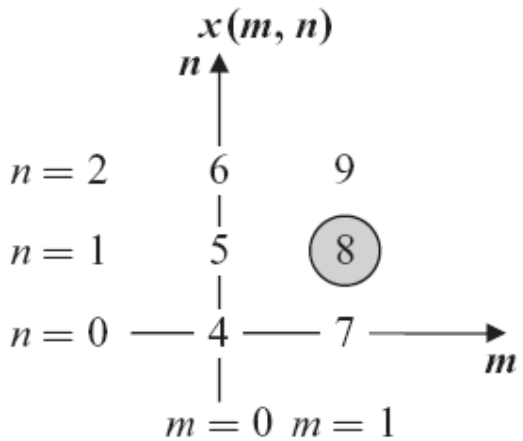

h(2−m, 1−n)
n
0
0
1
1
1
0
0
m
m = 0 m = 1 m = 2

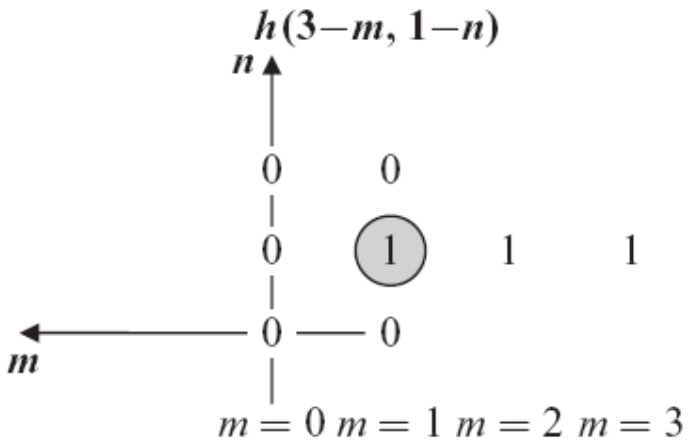

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1

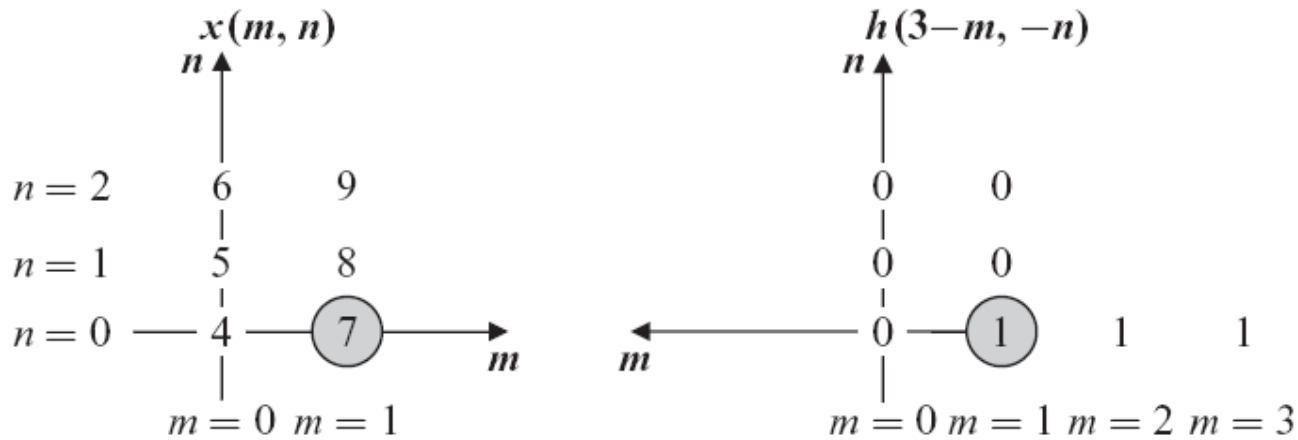

h(2−m, 2−n)
n
1
1
1
0
0
0
0
m
m = 0 m = 1 m = 2


x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1


h(3−m, 2−n)
n
0
1
1
1
0
0
0
0
m
m = 0 m = 1 m = 2 m = 3


x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1


h(3−m, 1−n)
n
0
0
0
1
1
1
0
0
m
m = 0 m = 1 m = 2 m = 3



$$y(m,n) = \begin{pmatrix} 4 & 5 & 6 \\ 11 & 13 & 15 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{pmatrix}$$

Matrix representation

Perform the linear convolution between the two matrices x(m, n) and h(m, n) given below

$$x(m,n)=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \quad h(m,n)=\begin{pmatrix}1 & 1\\ 1 & 1\\ 1 & 1\end{pmatrix}$$

$$y(m,n)=\begin{pmatrix}1 & 3 & 5 & 3\\ 5 & 12 & 16 & 9\\ 12 & 27 & 33 & 18\\ 1 & 24 & 28 & 15\\ 7 & 15 & 17 & 9\end{pmatrix}$$

Matrix representation

$$x(m,n)=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}3 & 4 & 5\end{pmatrix}$$

$$y(m,n)=\begin{pmatrix}3 & 10 & 22 & 22 & 15\\ 12 & 31 & 58 & 49 & 30\\ 21 & 52 & 94 & 76 & 45\end{pmatrix}$$

Matrix representation

Correlation

Correlation is a mathematical operation that is similar to convolution. Correlation is basically used to obtain similarity between two signals.

Autocorrelation is basically a signal correlated with itself.

The amplitude of each sample in the cross-correlation is a measure of how much one signal resembles the other.

$$x[n] * x^*[-n] = \sum_{k=-\infty}^{\infty} x[k]x^*[-(n-k)]$$

$$x[m,n] * x^*[-m,-n] = \sum_a \sum_b x(a,b)x^*[-(m-a),-(n-b)]$$

$$x[m,n] * x^*[-m,-n] = \sum_a \sum_b x(a,b)x^*[(a-m),(b-n)] = r_{xx}(m,n)$$

Determine the correlation between two matrices

$$x_1[m, n] = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 3 \end{bmatrix}.$$

Step 1
From the given $x_2[m, n]$ determine $x_2[-m, -n]$.

Step 1a To get the folded version of $x_2[m, n]$, first fold $x_2[m, n]$ column wise to get

$$x_2[m, -n] = \begin{bmatrix} 5 & 1 \\ 3 & 2 \end{bmatrix}$$

Step 1b Then fold $x_2[m, -n]$ along row-wise to get $x_2[-m, -n]$ which is given by

$$x_2[-m,-n]=\begin{bmatrix}3 & 2\\ 5 & 1\end{bmatrix}$$

Step 2

Now we have to perform linear convolution between $x_1[m, n]$ and $x_2[-m, -n]$.

Step 2a Formation of block matrix The number of block matrices depends on the number of columns of $x_1[m, n]$. In this case, the number of columns of $x_2[m, n]$ is two. Hence, two block matrices have to be formed. Let the two block matrices be H_0 and H_1 respectively.

Step 2b Formation of block matrix H_0 The block matrix H_0 formed from the first row of $x_1[m, n]$ is given below.

$$H_0 = \begin{bmatrix} 3 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix}$$

Step 2b Formation of block matrix H_1 The block matrix H_1 formed from the second row of $x_1[m, n]$ is given below.

$$H_1 = \begin{bmatrix} 2 & 0 \\ 4 & 2 \\ 0 & 4 \end{bmatrix}$$

Step 3 Formation of block Toeplitz matrix

Let the block Toeplitz matrix be denoted by A. The number of zeros to be appended in the block matrix A depends on the number of rows of $x_2[m, n]$. In this case, $x_2[m, n]$ has two rows; hence one zero has to be appended. The matrix A is given by

$$A = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}.$$

Then, substituting the values of H_0 and H_1 in the matrix A, we get

$$A = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 4 & 2 & 1 & 3 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

The resultant matrix $y[m, n]$ is obtained by multiplying the matrix A with a matrix whose elements are from $x_2[-m, -n]$.

$$y[m,n]=\begin{bmatrix}3&0&0&0\\1&3&0&0\\0&1&0&0\\2&0&3&0\\4&2&1&3\\0&4&0&1\\0&0&2&0\\0&0&4&2\\0&0&0&4\end{bmatrix}\times\begin{bmatrix}3\\2\\5\\1\end{bmatrix}=\begin{bmatrix}9\\9\\2\\21\\24\\9\\10\\22\\4\end{bmatrix}$$

The result $y[m, n]$ is a 3×3 matrix which is given by

$$y[m,n]=\begin{bmatrix}9&9&2\\21&24&9\\10&22&4\end{bmatrix}$$

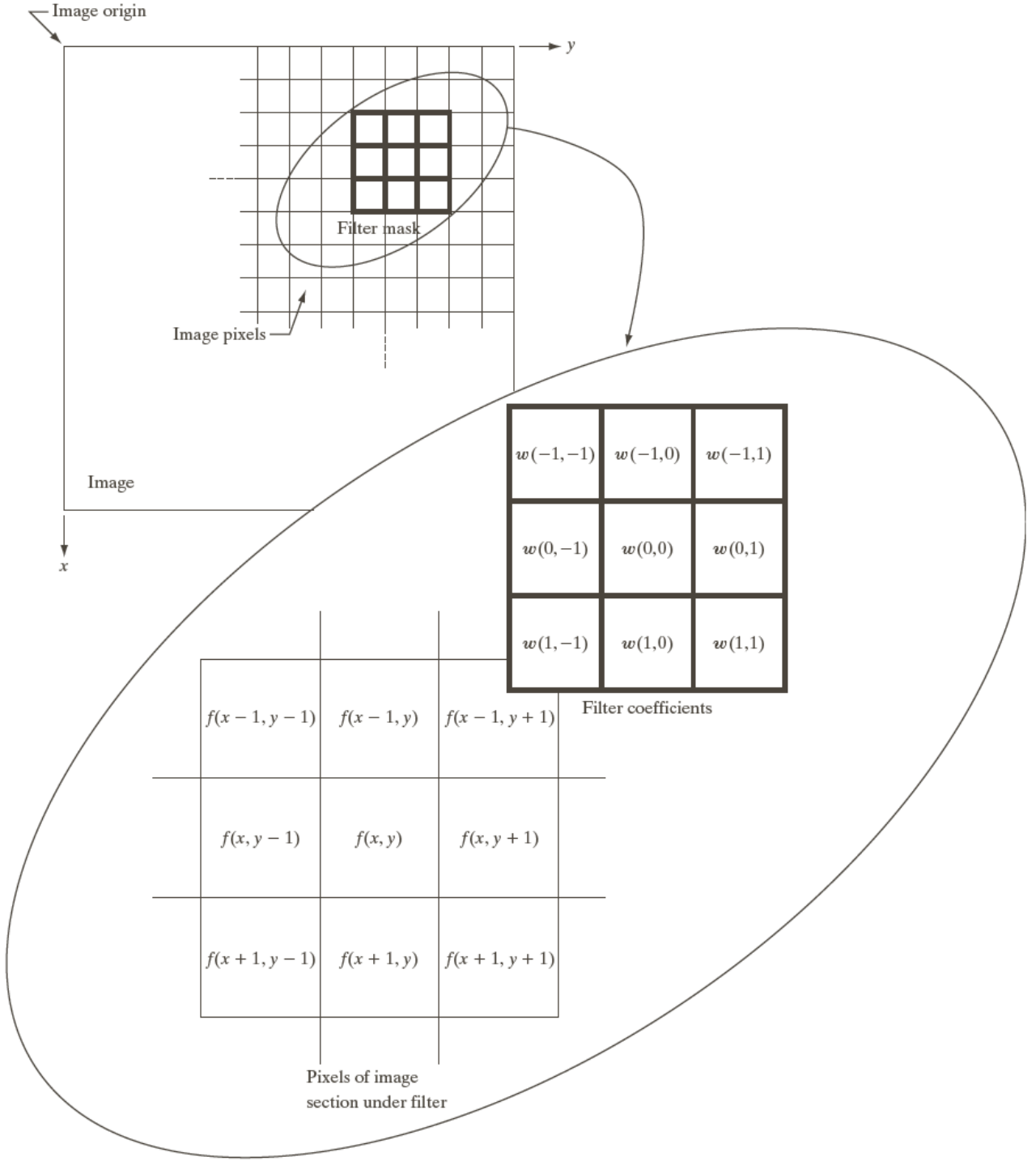
Determine the correlation between two matrices

$$x_1[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$$

Determine the correlation between two matrices

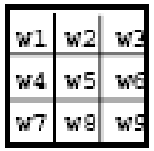
$$x_1[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix}$$

Neighborhood Processing



Linear vs Non-Linear Spatial Filtering Methods

- A filtering method is linear when the output is a weighted sum of the input pixels.



$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

- Methods that do not satisfy the above property are called non-linear.
  - e.g.,

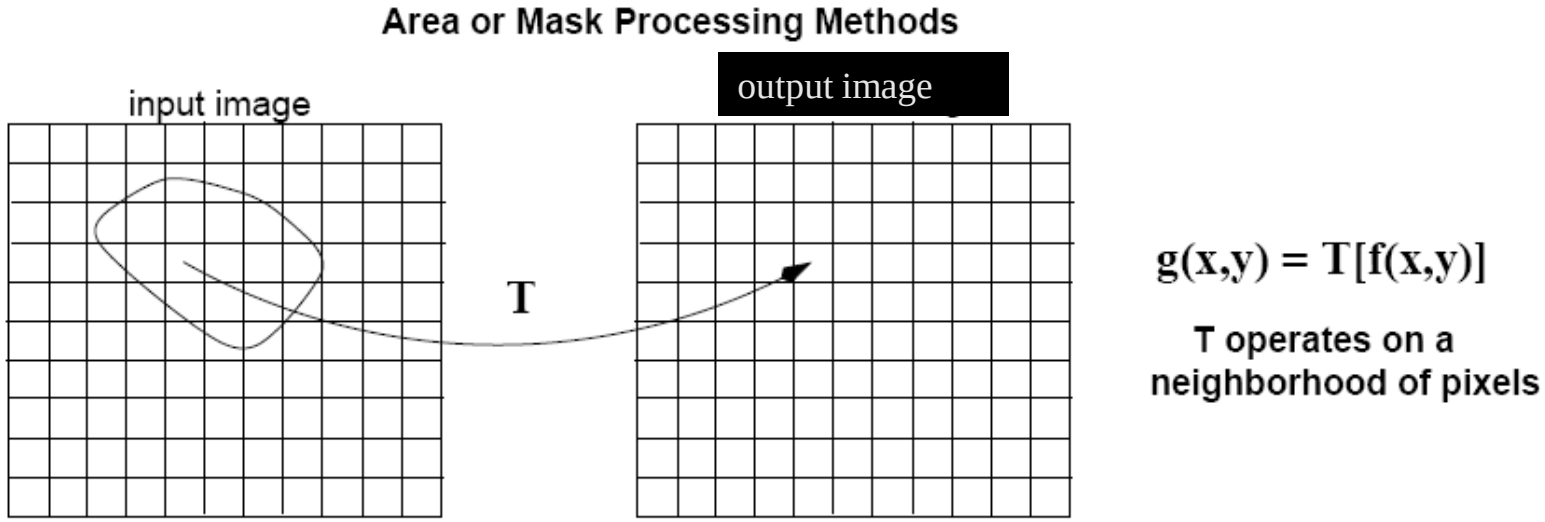
$$z_5' = max(z_k, k = 1, 2, \ldots, 9)$$

Spatial Filtering

- The word “filtering” has been borrowed from the frequency domain.
- Filters are classified as:
  - Low-pass (i.e., preserve low frequencies)
  - High-pass (i.e., preserve high frequencies)
  - Band-pass (i.e., preserve frequencies within a band)
  - Band-reject (i.e., reject frequencies within a band)

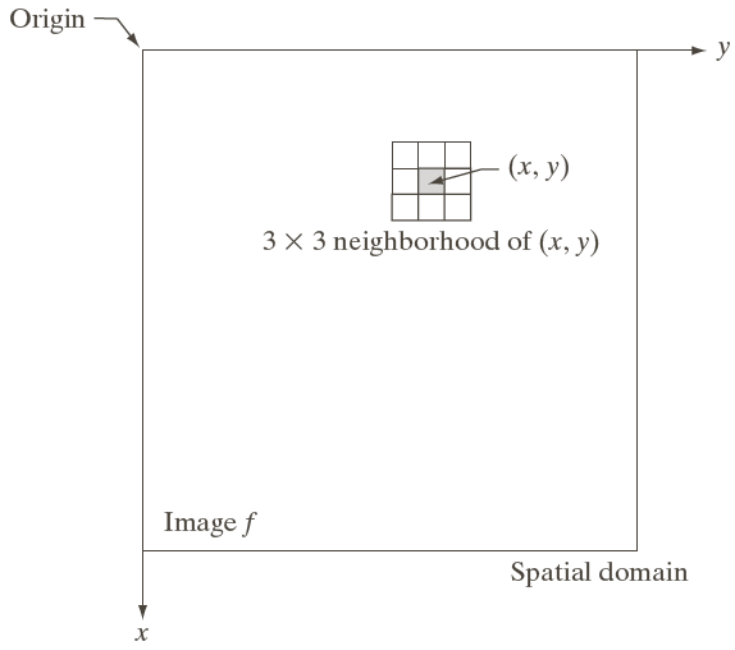
Spatial Filtering (cont'd)

- Spatial filtering is defined by:
  (1) A neighborhood
  (2) An operation that is performed on the pixels inside the neighborhood



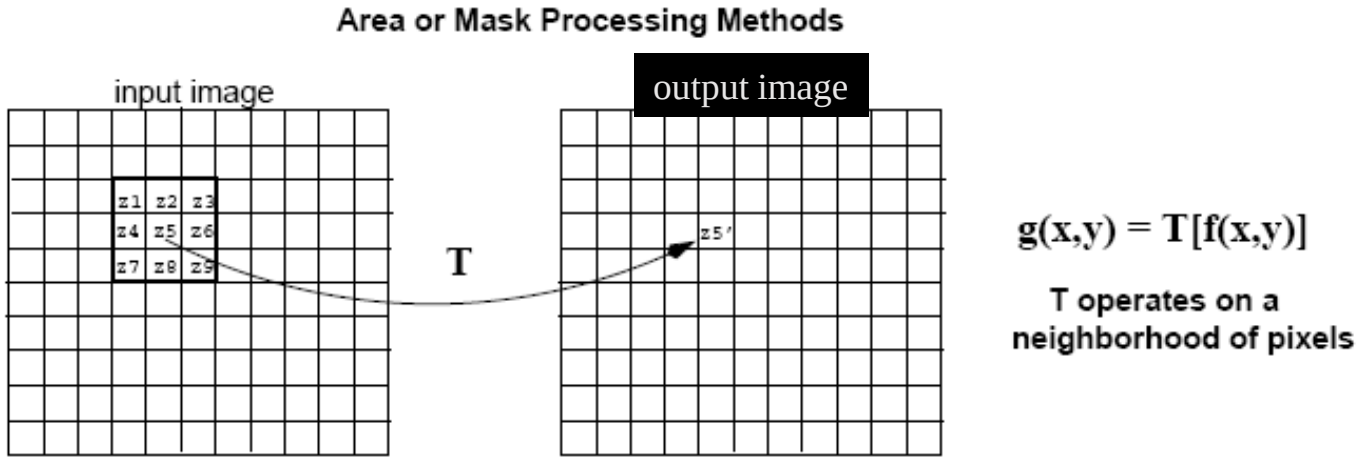
Spatial Filtering - Neighborhood

- Typically, the neighborhood is rectangular and its size is much smaller than that of *f(x,y)*
  - e.g., 3x3 or 5x5



Spatial filtering - Operation

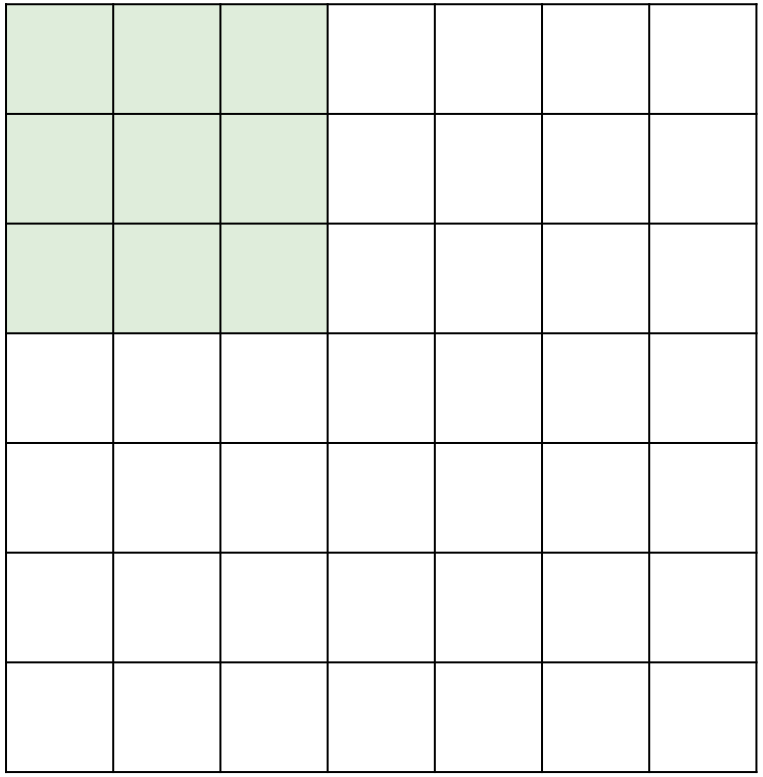
$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$



- A filtered image is generated as the center of the mask moves to every pixel in the input image.

A closer look at spatial dimensions:

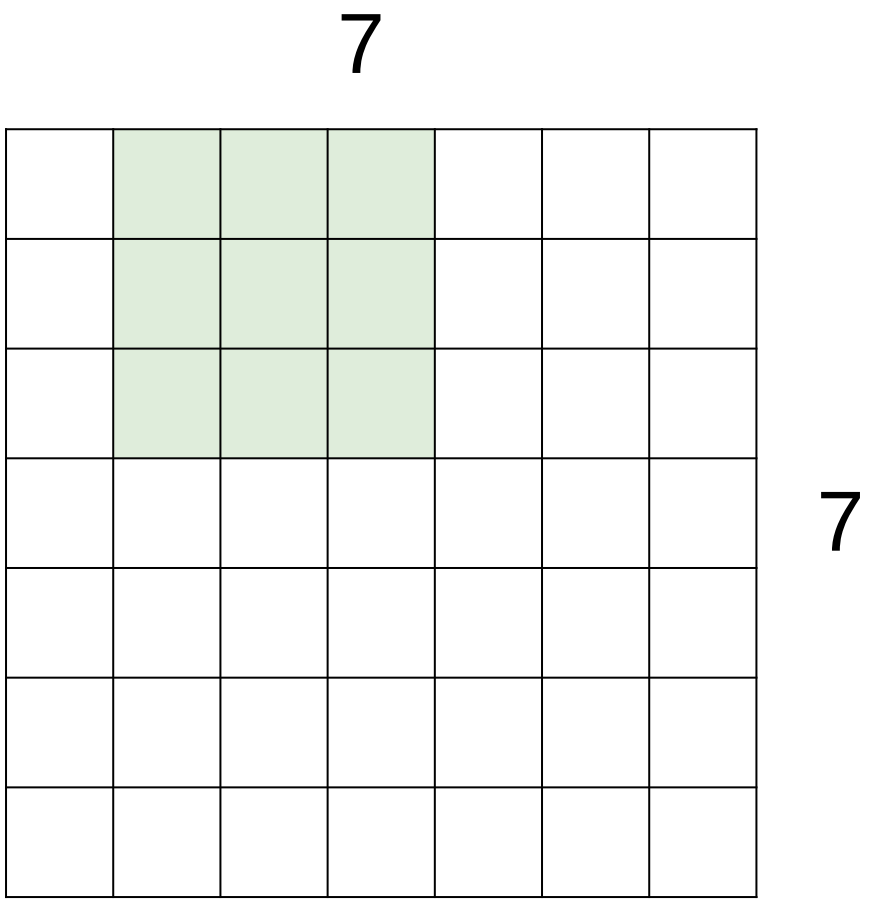
7



7

7x7 input (spatially) assume 3x3 filter

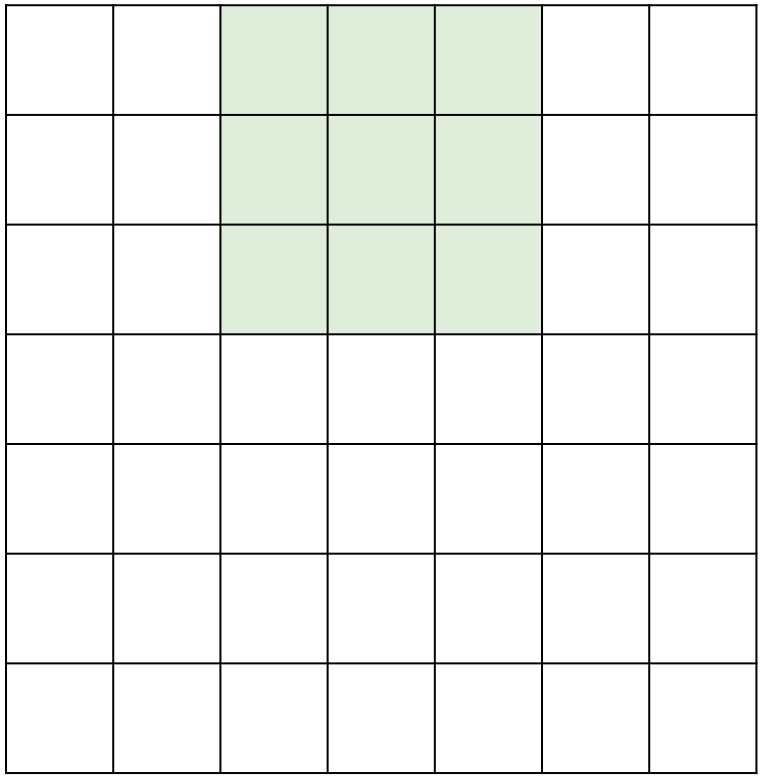
A closer look at spatial dimensions:



7x7 input (spatially) assume
3x3 filter

A closer look at spatial dimensions:

7

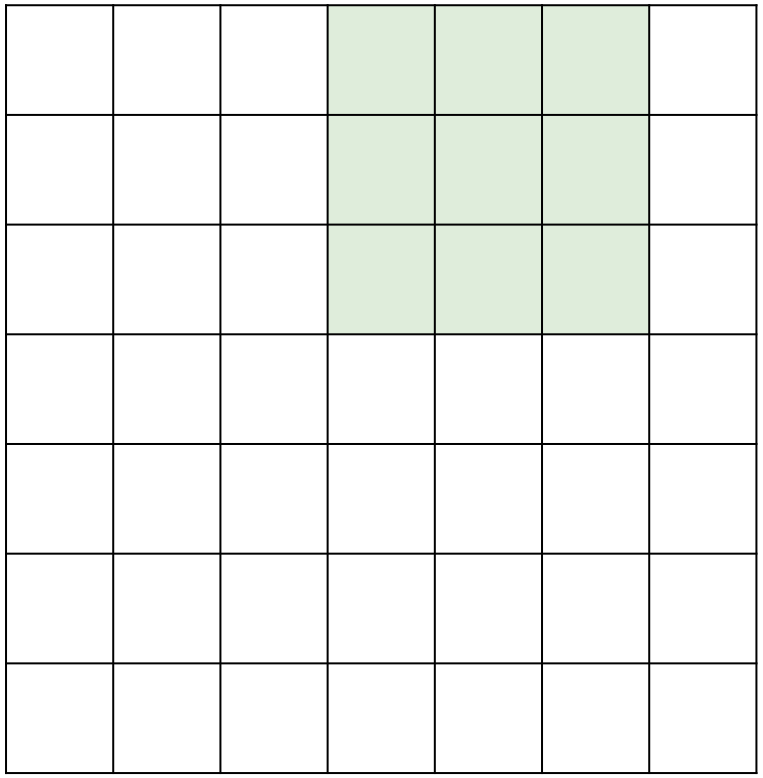


7

7x7 input (spatially) assume
3x3 filter

A closer look at spatial dimensions:

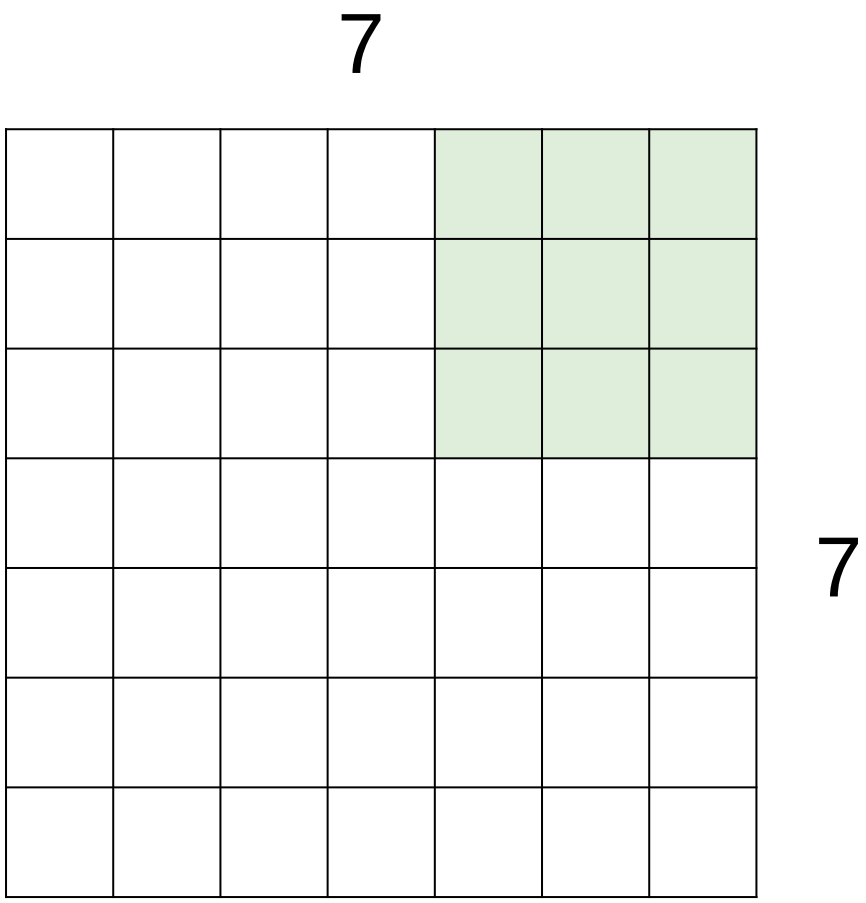
7



7

7x7 input (spatially) assume
3x3 filter

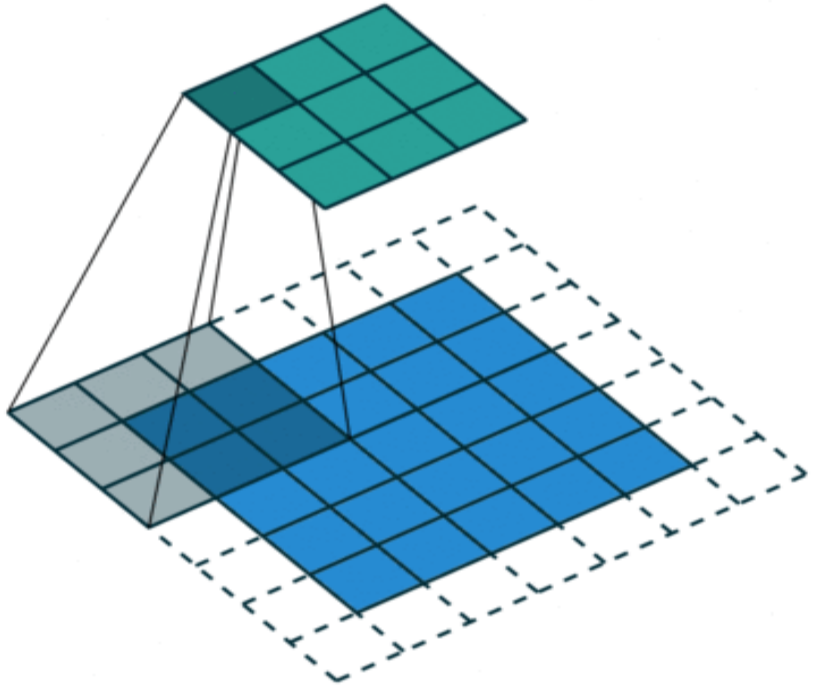
A closer look at spatial dimensions:

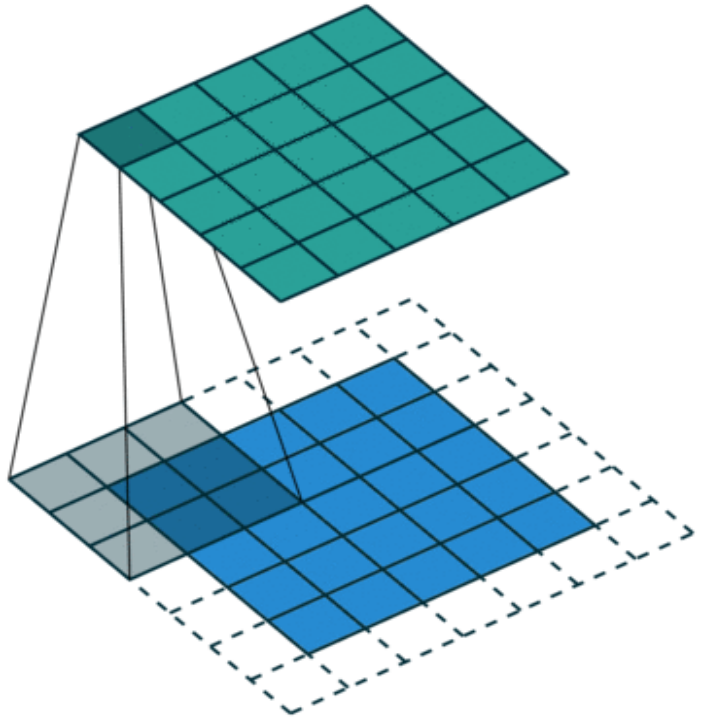


7x7 input (spatially) assume
3x3 filter

=> 5x5 output

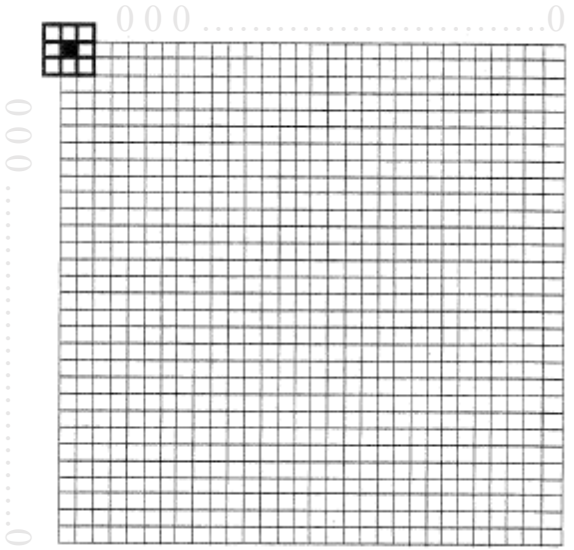
Filtering



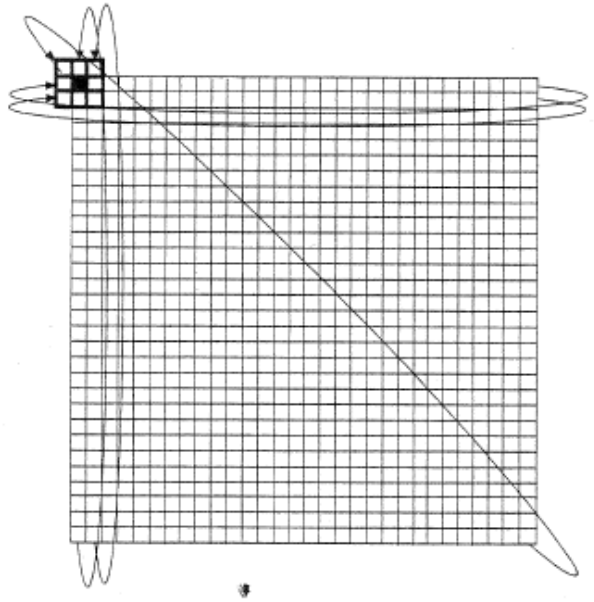


Handling Pixels Close to Boundaries

pad with zeroes

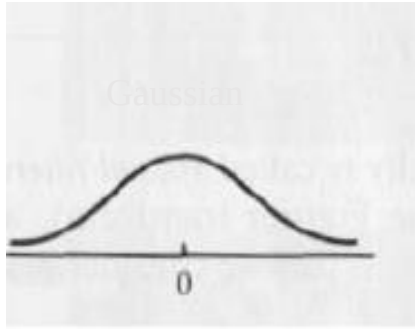


or

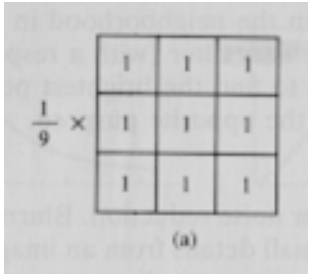


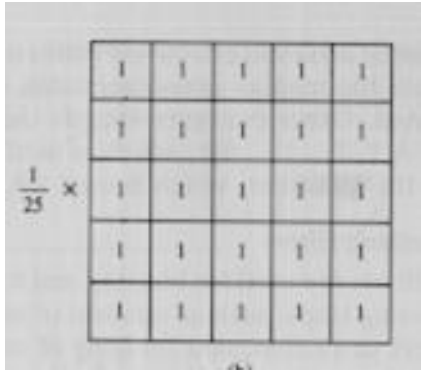
Filters

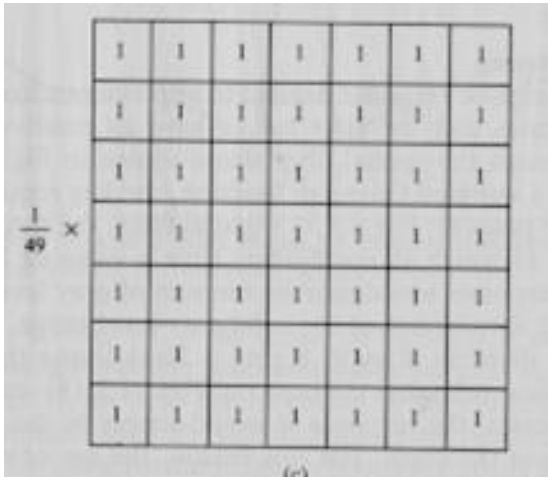
- Smoothing (i.e., low-pass filters)
  - Reduce noise and eliminate small details.
  - The elements of the mask must be **positive**.
  - Sum of mask elements is 1 (after normalization)



Smoothing Filters: Averaging (Low-pass filtering)



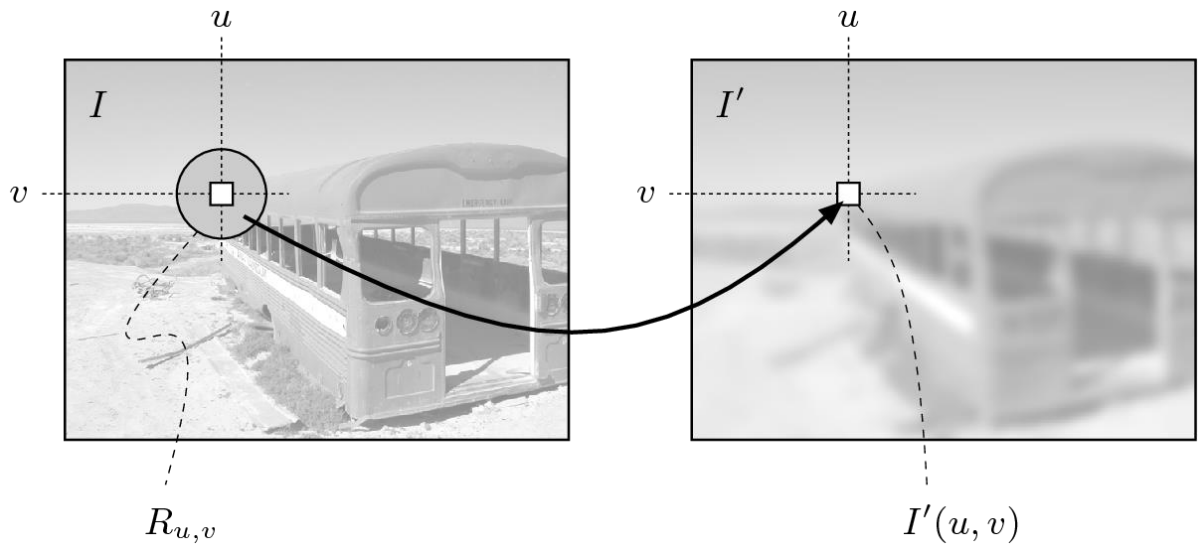


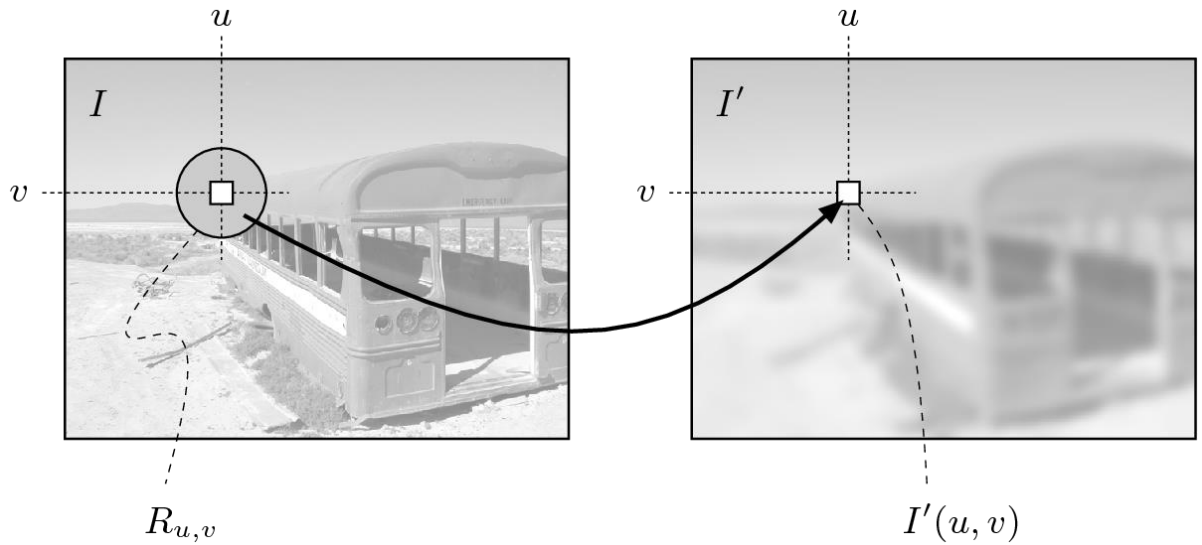


Smoothing an image by averaging

- Replace each pixel by the average of its neighboring pixels
- Assume a 3x3 neighborhood:

$$I'(u,v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

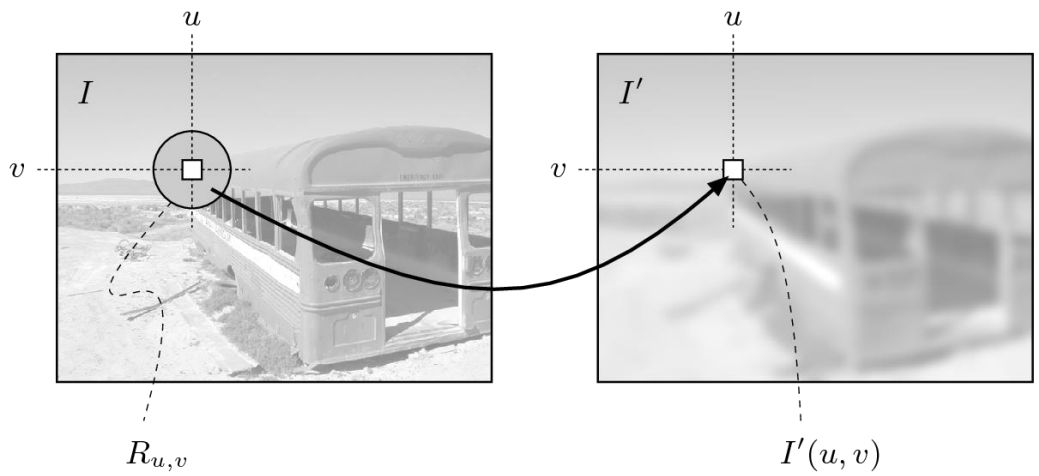




$$I'(u,v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

$$\begin{aligned} I'(u,v) \leftarrow \tfrac{1}{9} \cdot [\, & I(u-1,v-1) + I(u,v-1) + I(u+1,v-1) + \\ & I(u-1,v) + I(u,v) + I(u+1,v) + \\ & I(u-1,v+1) + I(u,v+1) + I(u+1,v+1) \,] \end{aligned}$$

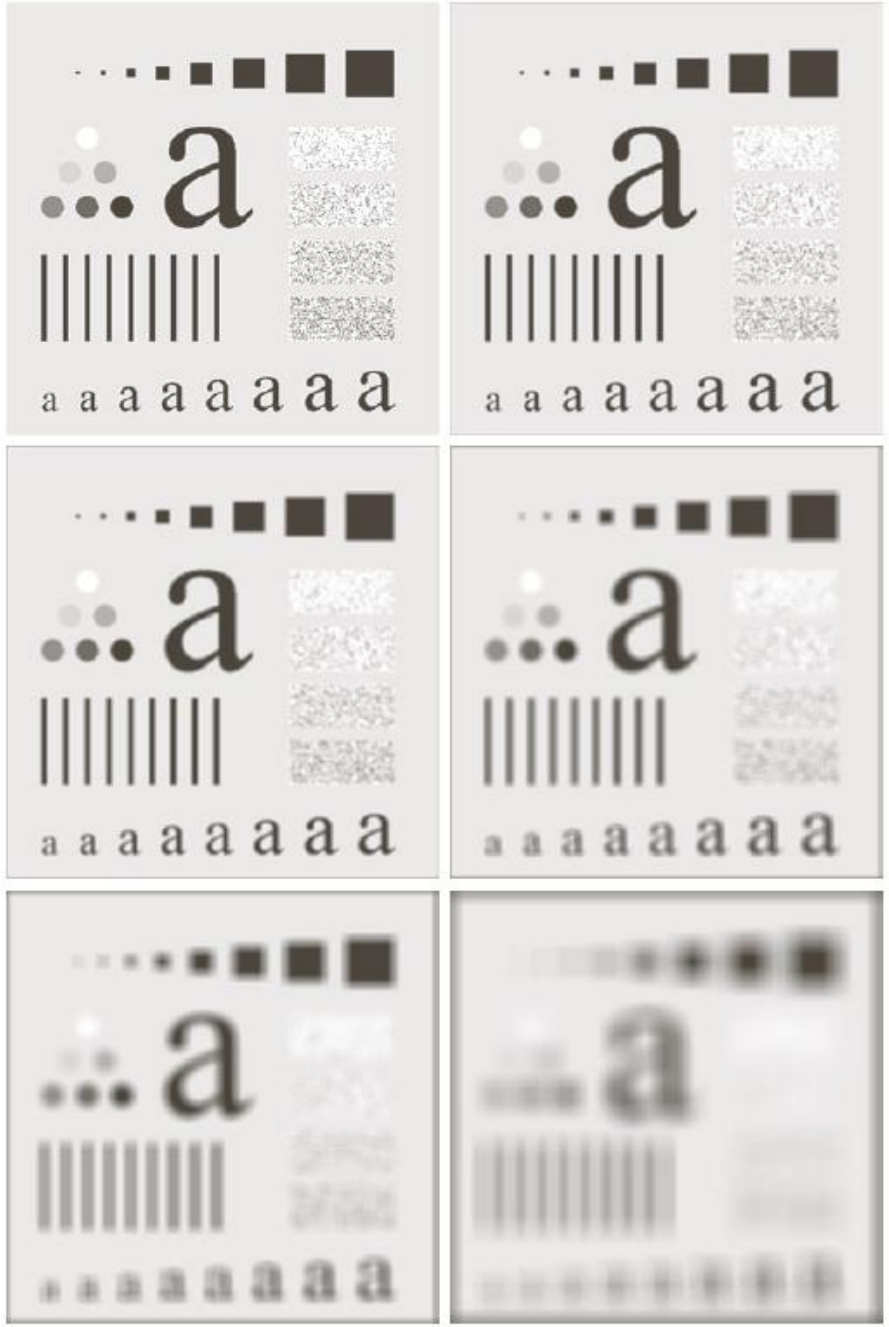
$$I'(u,v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

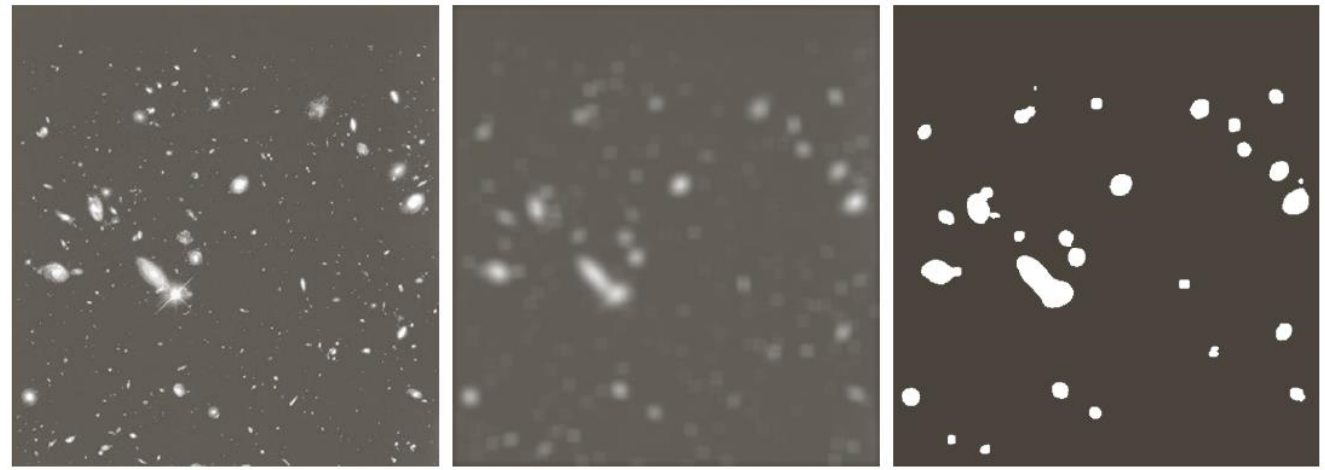


- In general a filter applies a function over the values of a small neighborhood of pixels to compute the result
- The size of the filter = the size of the neighborhood: 3x3, 5x5, 7x7, …, 21x21,..
- The shape of the filter region is not necessarily square, can be a rectangle, a circle…

$$R = w_1 z_1 + w_2 z_2 + \ldots\ldots = \sum_{k=1}^{mn} w_k z_k$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

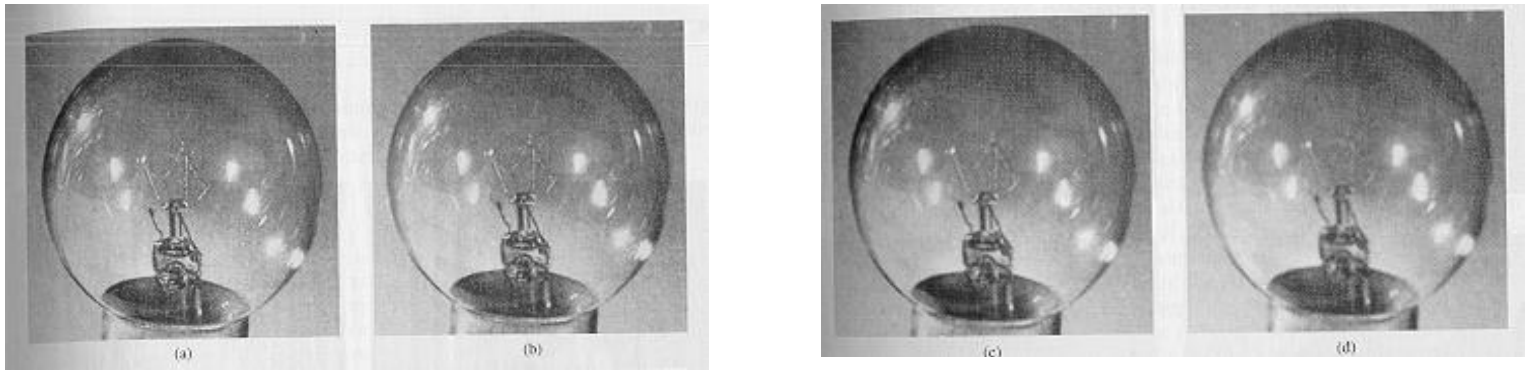




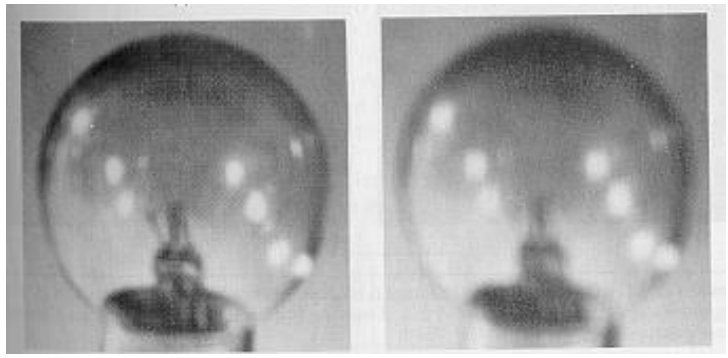
Smoothing Filters: Averaging (cont'd)

- Mask size determines the degree of smoothing and loss of detail.

original 3x3 5x5 7x7



15x15 25x25

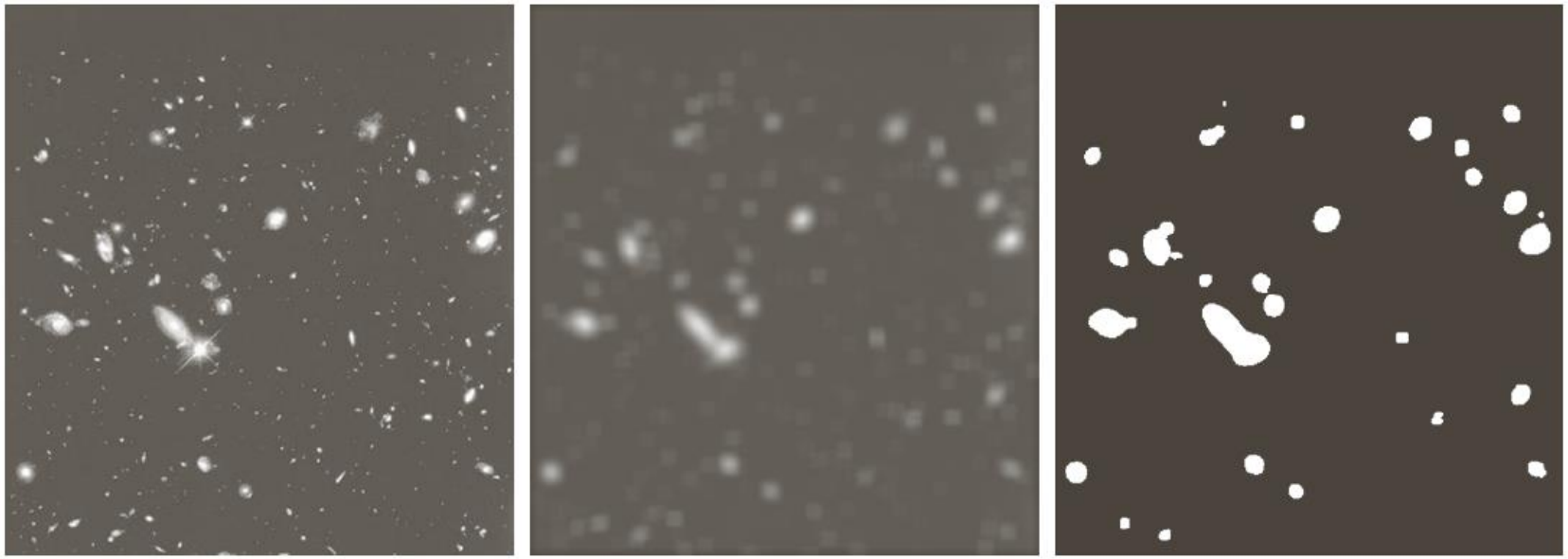


Smoothing Filters: Averaging (cont'd)

Example: extract, largest, brightest objects

15 x 15 averaging

image thresholding



$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

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

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	25	51	72	90	85	35	61	0
0	21	21	23	35	34	36	37	0
0	38	41	44	45	47	48	31	0
0	21	79	90	91	81	85	90	0
0	88	21	34	56	76	88	90	0
0	51	51	56	57	67	88	90	0
0	91	95	93	92	66	67	65	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

1	1	1
1	1	1
1	1	1

Mask size=3x3

Zero padded=row=3-1=2

col=3-1=2

0	0	0	0	0	0	0	0	0
0	25	51	72	90	85	35	61	0
0	21	21	23	35	34	36	37	0
0	38	41	44	45	47	48	31	0
0	21	79	90	91	81	85	90	0
0	88	21	34	56	76	88	90	0
0	51	51	56	57	67	88	90	0
0	91	95	93	92	66	67	65	0
0	0	0	0	0	0	0	0	0

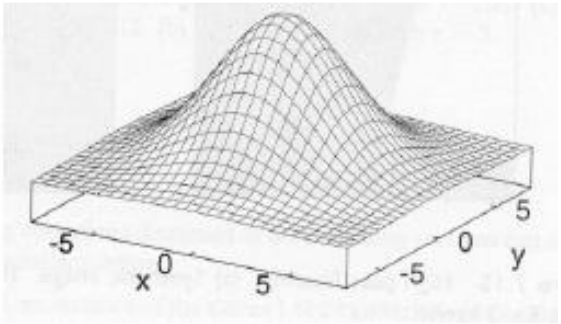
1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0
0	13	24	32	38	35	32	19	0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Smoothing filters: Gaussian

- The weights are samples of the Gaussian function

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



7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

σ = 1.4

mask size is
a function of **σ** :

$$height = width = 5\sigma \text{ (subtends 98.76\% of the area)}$$

Smoothing filters: Gaussian (cont'd)

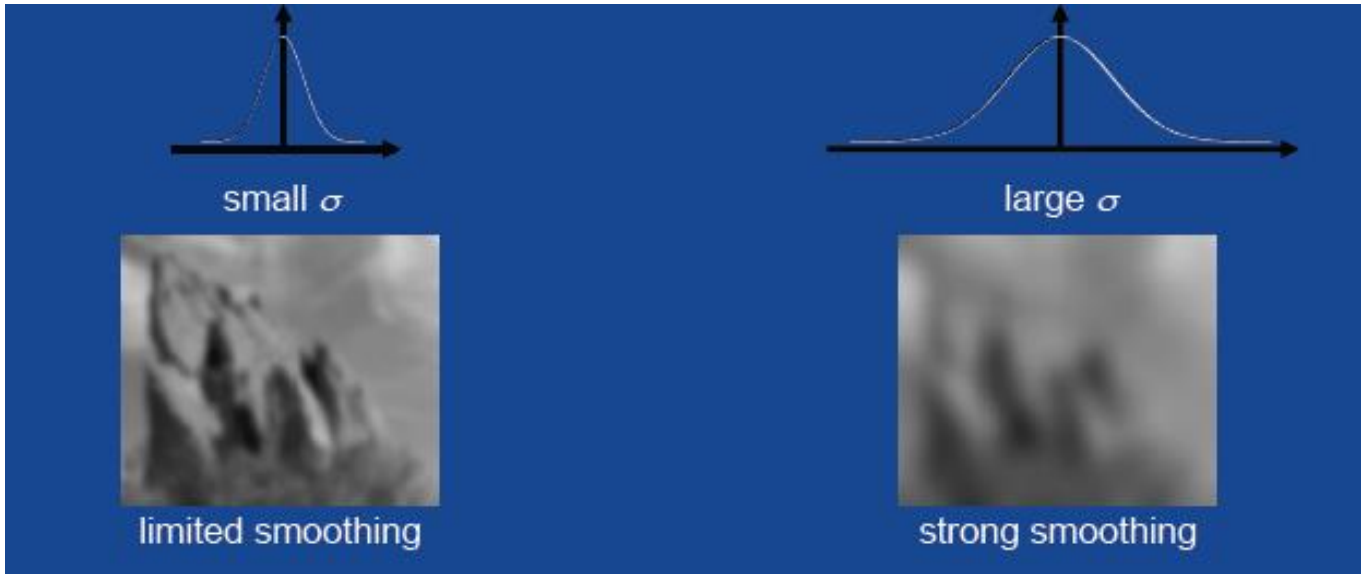
- **σ** controls the amount of smoothing
- As **σ** increases, more samples must be obtained to represent the Gaussian function accurately.

σ = 3

15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

Smoothing filters: Gaussian (cont'd)



Averaging vs Gaussian Smoothing

Averaging

Gaussian

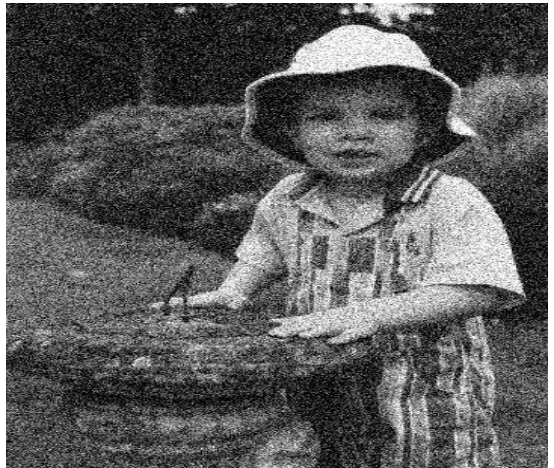
Rank filtering

Order filter are implemented by arranging the neighborhood pixels in order from smallest to largest gray level value and using this ordering to select the "correct" value. The placement of the value within this ordered set is referred as the *rank* .

1. **Given an N x N window W, the pixel values can be ordered , as follows**

   $I_1 < I_2 < \ldots . I_N^2$ **where** $I_1 \ldots . I_3 \ldots I_{N^2}$ **are intensity values.**

2. **Select a value from a particular position in the list to use as the new value for the pixel.**



From noise image Rank filtering mask (7 x 7) rank 4

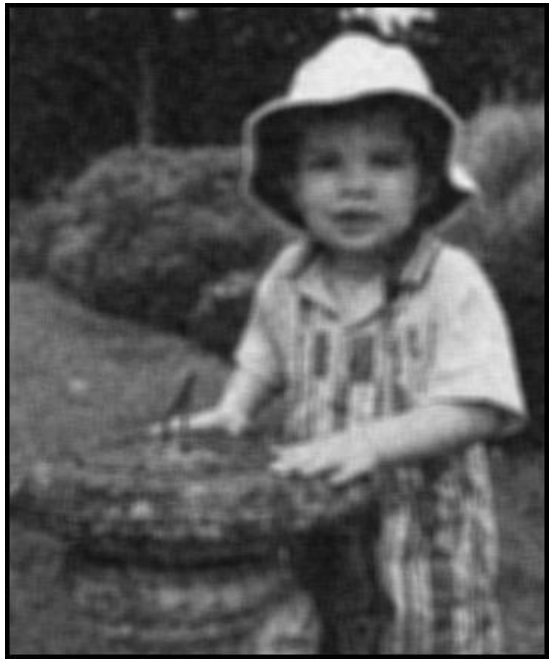
Median Filtering

- We have seen that smoothing (low pass) filters reduce noise.
- However, the underlying assumption is that the neighboring pixels represent additional samples of the same value as the reference pixel, i.e. they represent the same feature.
- At edges, this is clearly not true, and blurring of features results

Median filtering

In order to perform median filtering in a neghborhod of a pixel [i.j]:

1. **Sort the pixels into ascending order by gray level.**
2. **Select the value of the middle pixel as the new value for pixel [i.j]**

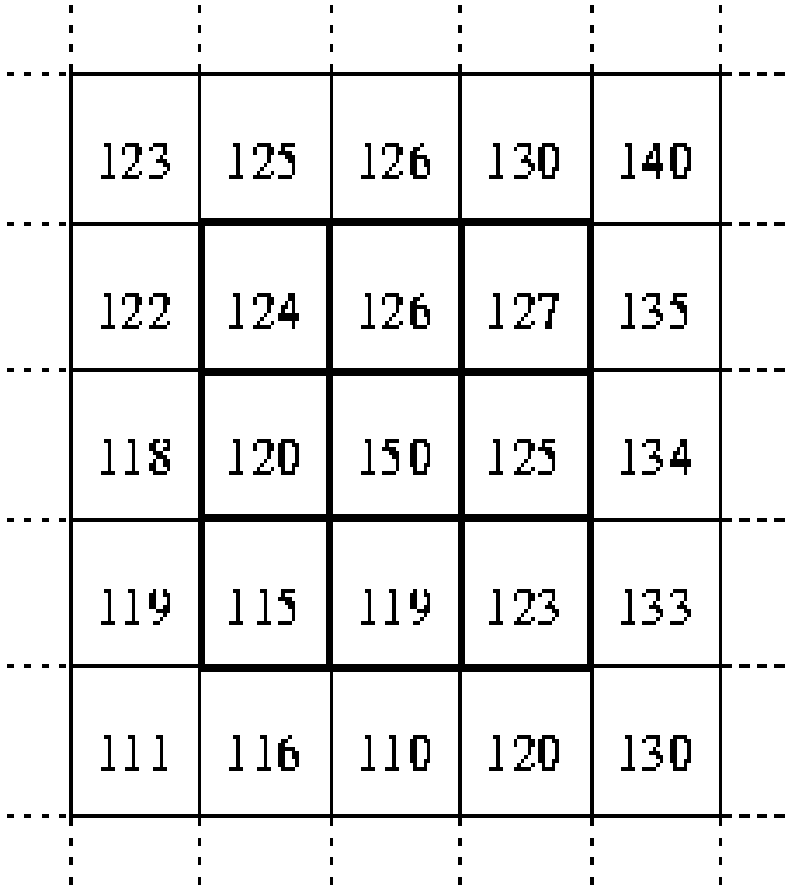


Mean Filter size =7 x 7

Median Filter size =7 x 7

This filters are excellent for impulse type of noise

Median filtering



Neighbourhood values:

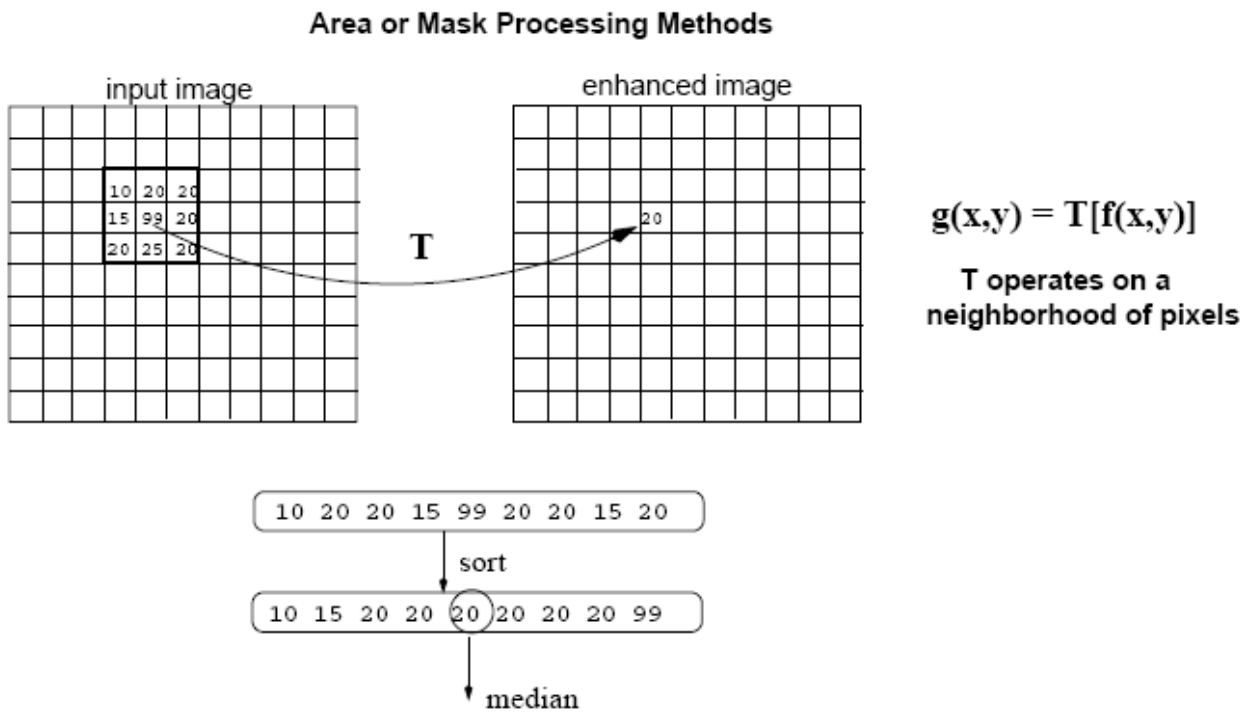
115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Figure 1 Calculating the median value of a pixel neighborhood. As can be seen, the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value: 124. A 3×3 square neighborhood is used here --- larger neighborhoods will produce more severe smoothing.

Smoothing Filters: Median Filtering (cont'd)

- Replace each pixel by the median in a neighborhood around the pixel.



Median filtering - Applications

The image has been corrupted with higher levels (*i.e.* p=5% that a bit is flipped) of salt and pepper noise

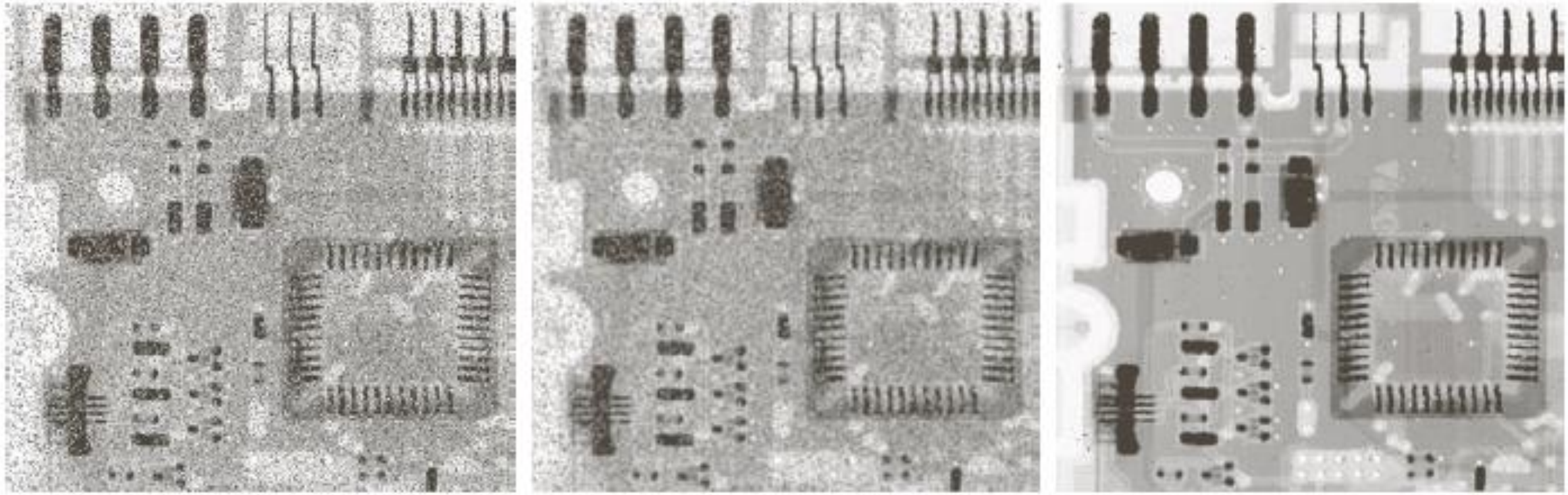
Median filtering - Applications

After smoothing with a 3×3 filter, most of the noise has been eliminated

If we smooth the noisy image with a larger median filter, *e.g.* 7×7, all the noisy pixels disappear, as shown in this image

Median filtering - Applications

Note that the image is beginning to look a bit `blotchy', as graylevel regions are mapped together. Alternatively, we can pass a 3×3 median filter over the image three times in order to remove all the noise with less loss of detail



a b c

Smoothing Filters: Median Filtering (non-linear)

- Very effective for removing “salt and pepper” noise (i.e., random occurrences of black and white pixels).

averaging

median filtering

Advantages of median filter

By calculating the median value of a neighborhood rather than the mean filter, the median filter has two main advantages over the mean filter:

1.The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.

2. Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles a edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.

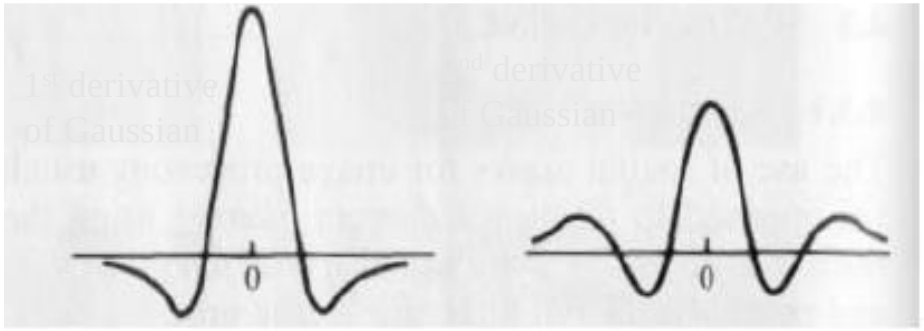
Disadvantages of median filter

1. One of the major problems with the median filter is that it is relatively expensive and complex to compute. To find the median it is necessary to sort all the values in the neighborhood into numerical order and this is relatively slow, even with fast sorting algorithms such as quick sort.

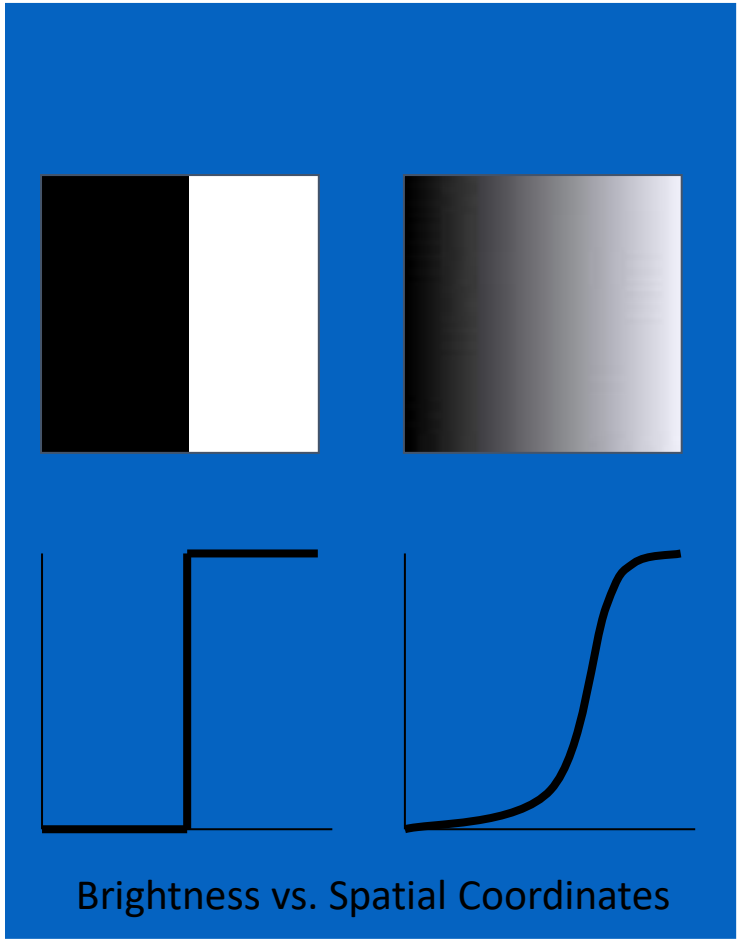
2. **Any structure that occupies less than half of the filter's neighborhood will tend to be eliminated**

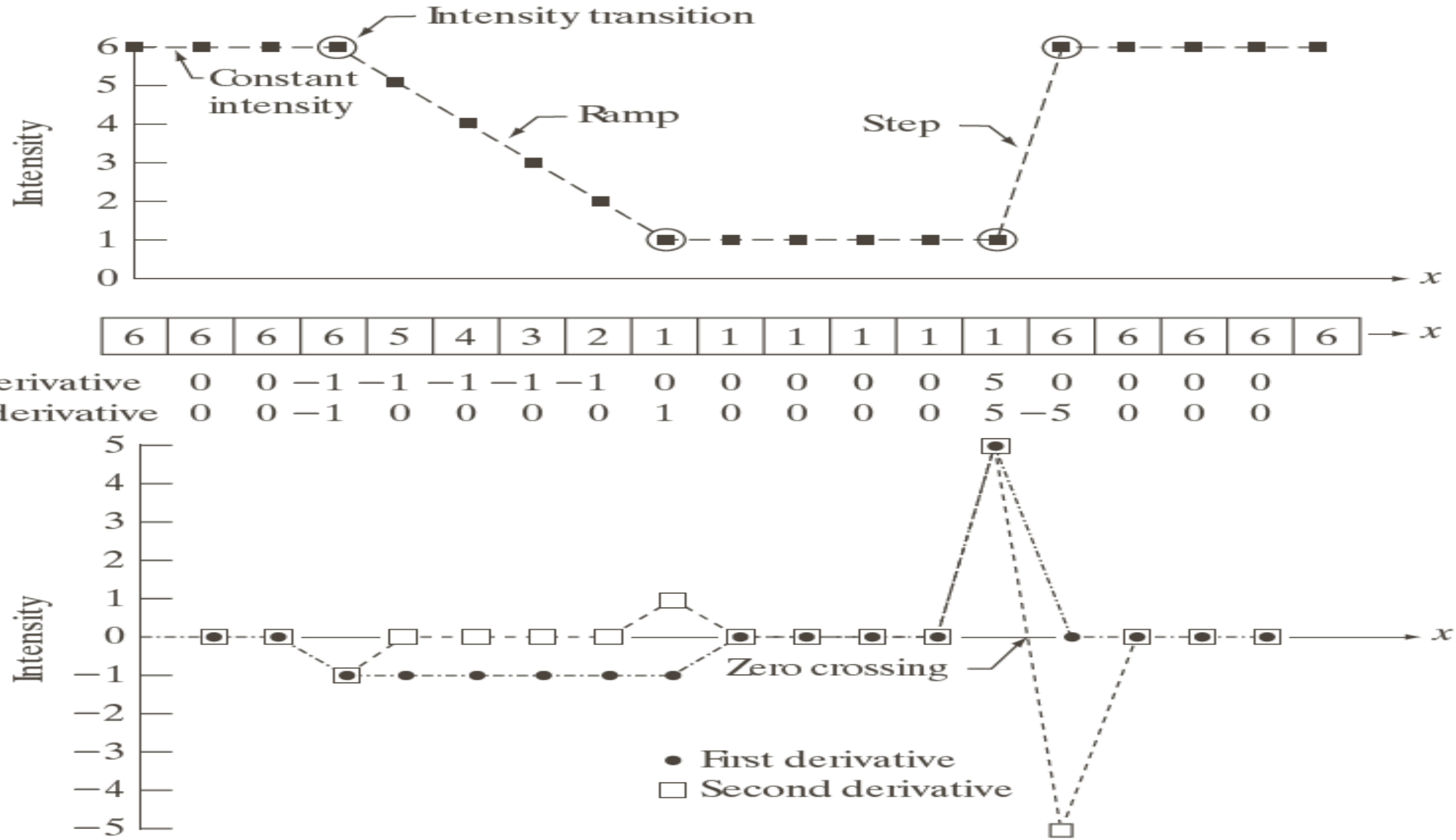
Filters (cont'd)

- Sharpening (i.e., high-pass filters)
  - Highlight fine detail or enhance detail that has been blurred.
  - The elements of the mask contain both **positive** and **negative** weights.
  - Sum of the mask weights is 0 (after normalization)



- Edges are those places in an image that correspond to object boundaries.
- Edges are pixels where image brightness changes abruptly.



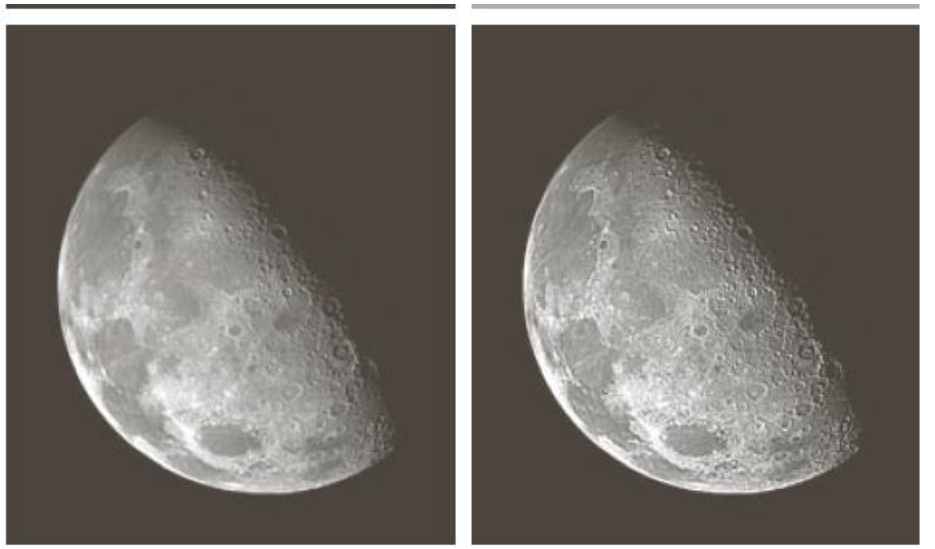

Intensity transition
Constant intensity
Ramp
Step
Intensity
x
Scan line
6 6 6 6 5 4 3 2 1 1 1 1 1 1 6 6 6 6 6
1st derivative 0 0 −1 −1 −1 −1 −1 0 0 0 0 0 5 0 0 0 0
2nd derivative 0 0 −1 0 0 0 0 1 0 0 0 0 5 −5 0 0 0
Zero crossing
First derivative
Second derivative

Sharpening Filters (cont’d)

- Note that the response of high-pass filtering might be negative.
- Values must be re-mapped to [0, 255]

original image

sharpened images



Sharpening Filters: Derivatives

- Taking the derivative of an image results in sharpening the image.
- The derivative of an image can be computed using the gradient.

$$grad(f) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Sharpening Filters: Derivatives (cont'd)

- The gradient is a **vector** which has magnitude and direction:

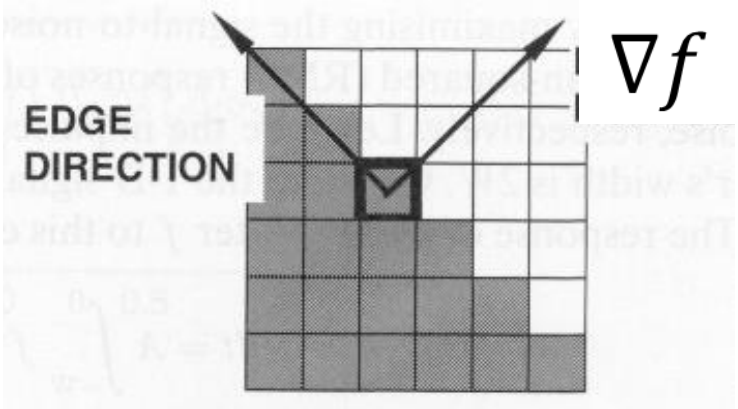
$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

$$direction(grad(f)) = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

Sharpening Filters: Derivatives (cont'd)

- **Magnitude:** provides information about edge strength.

- **Direction:** perpendicular to the direction of the edge.



Sharpening Filters: Gradient Computation

- Approximate gradient using finite differences:

$$f'(x) = \lim_{h->0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

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

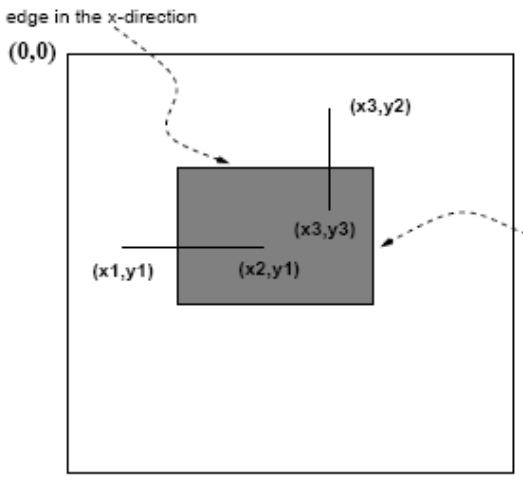
$$\frac{\partial f}{\partial y} = \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y} = f(x,y) - f(x, y+1), \quad (\Delta y=1)$$

Sharpening Filters: Gradient Computation (cont'd)

$$\frac{f(x3,y2) - f(x3,y3)}{y2 - y3}$$

or $$\frac{f(x,y+Dy) - f(x,y)}{-Dy} = \boxed{f(x,y) - f(x,y+1)}$$ **(grad in the y-direction)**

(y3=y2+Dy, y2=y, x3=x, Dy=1)



edge in the y-direction

$$\frac{f(x2,y1) - f(x1,y1)}{x2 - x1}$$

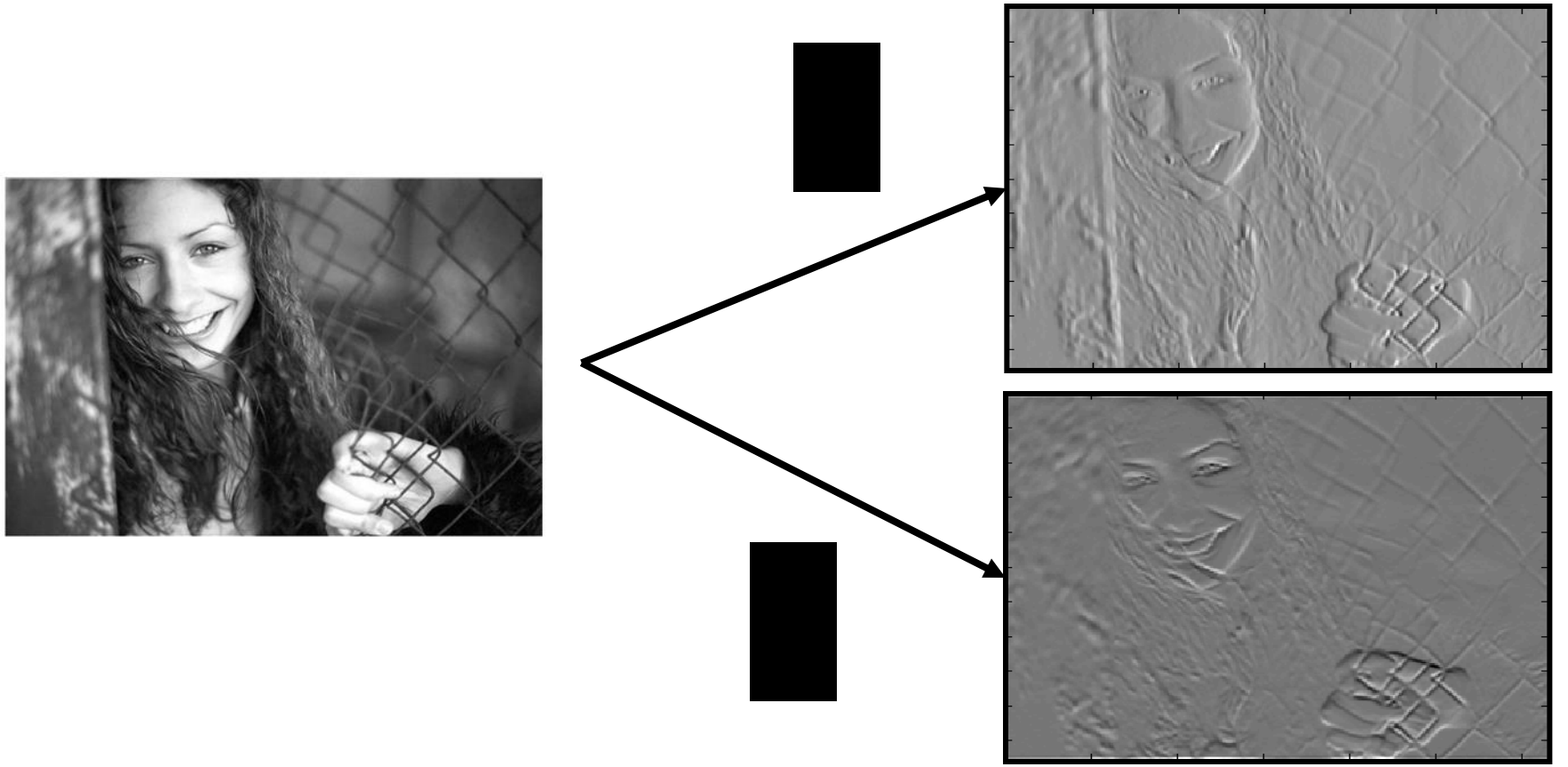
or $$\frac{f(x+Dx,y) - f(x,y)}{Dx} =$$

$$\boxed{f(x+1,y) - f(x,y)}$$

(grad in the x-direction)

(x2=x+Dx, x1=x, y1=y2=y, Dx=1)

Example



Sharpening Filters: Gradient Computation (cont'd)

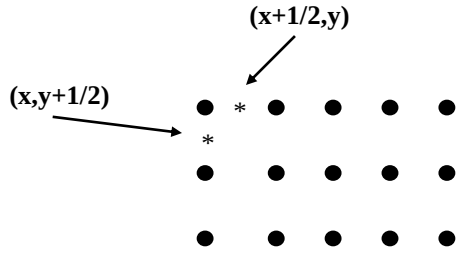
- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using masks:

-1	1

1
-1

$\frac{\partial f}{\partial x}$ good approximation at (x+1/2,y)

$\frac{\partial f}{\partial y}$ good approximation at (x,y+1/2)



- <u>Example:</u> approximate gradient at z_5

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial f}{\partial x} = z_6 - z_5$$

$$\frac{\partial f}{\partial y} = z_5 - z_8$$

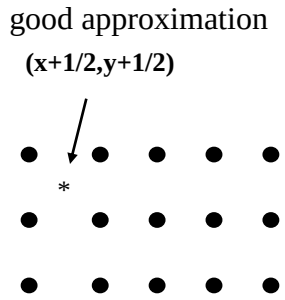
$$mag(grad(f)) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

Sharpening Filters: Gradient Computation (cont'd)

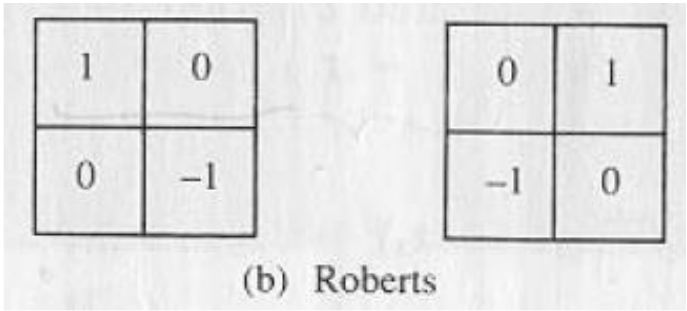
- A different approximation of the gradient:

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

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



- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:



Sharpening Filters: Gradient Computation (cont'd)

- Example: approximate gradient at z_5

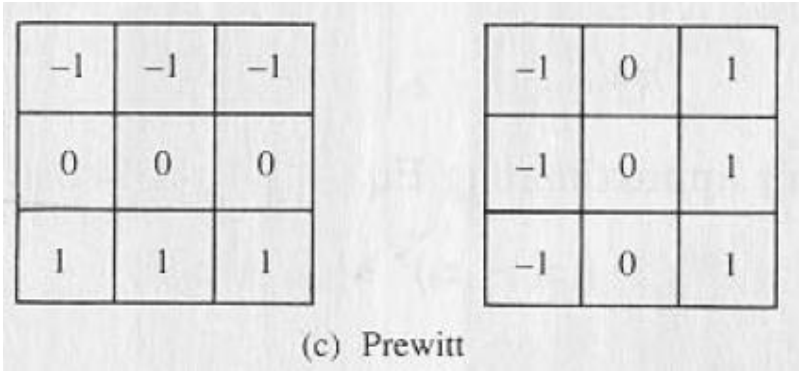
Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial f}{\partial x} = z_5 - z_9$$

$$\frac{\partial f}{\partial y} = z_6 - z_8$$

$$mag(grad(f)) = \sqrt{(z_5 - z_9)^2 + (z_6 - z_8)^2}$$

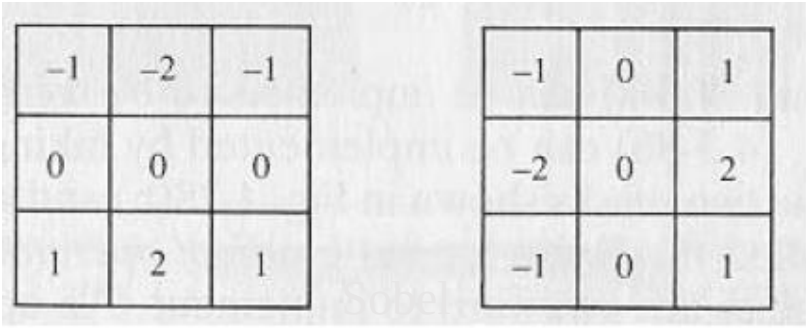
- Other approximations



−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

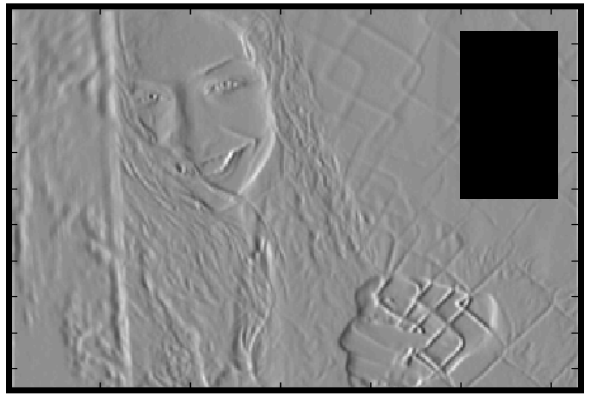
(c) Prewitt

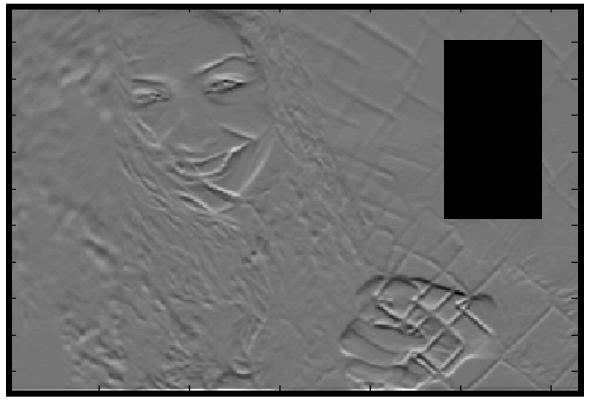


−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

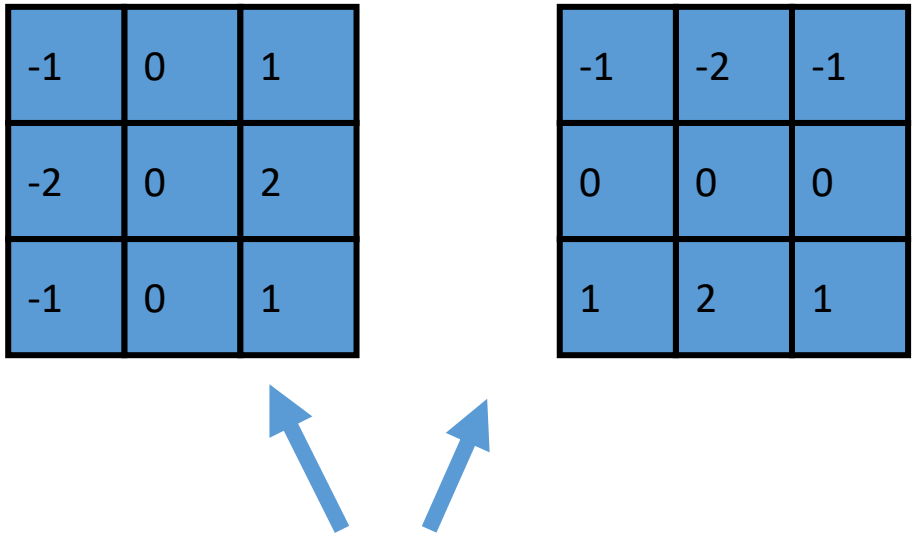
Example



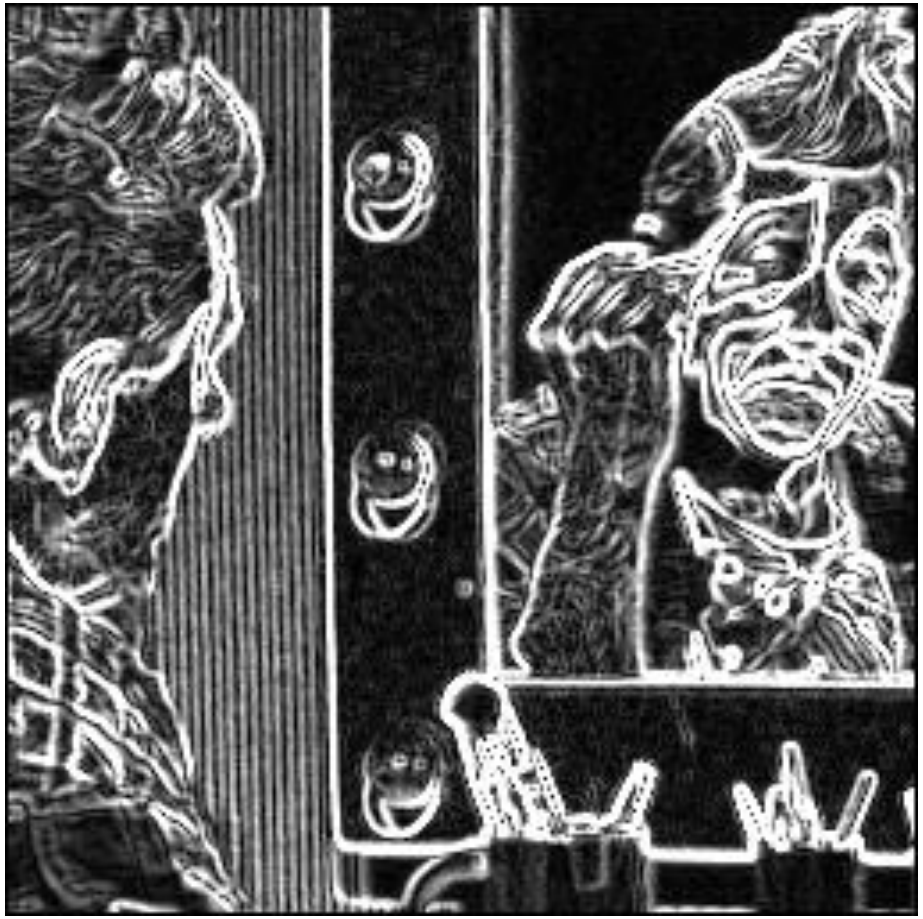


$$\sqrt{\frac{\partial f^2}{\partial x} + \frac{\partial f^2}{\partial y}}$$

$$Mag \sim abs(G_x) + abs(G_y)$$



“Sobel Operators”



Sharpening Filters: Laplacian

The Laplacian (2nd derivative) is defined as:

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

(dot product)

Approximate derivatives:

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

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

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

Sharpening Filters: Laplacian (cont'd)

Laplacian Mask

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

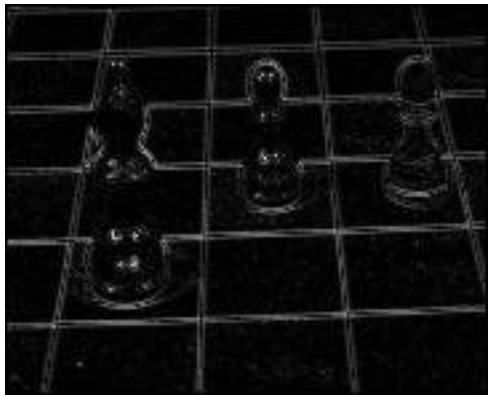
5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

detect zero-crossings

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

Sharpening Filters: Laplacian (cont’d)

Laplacian



Sobel

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

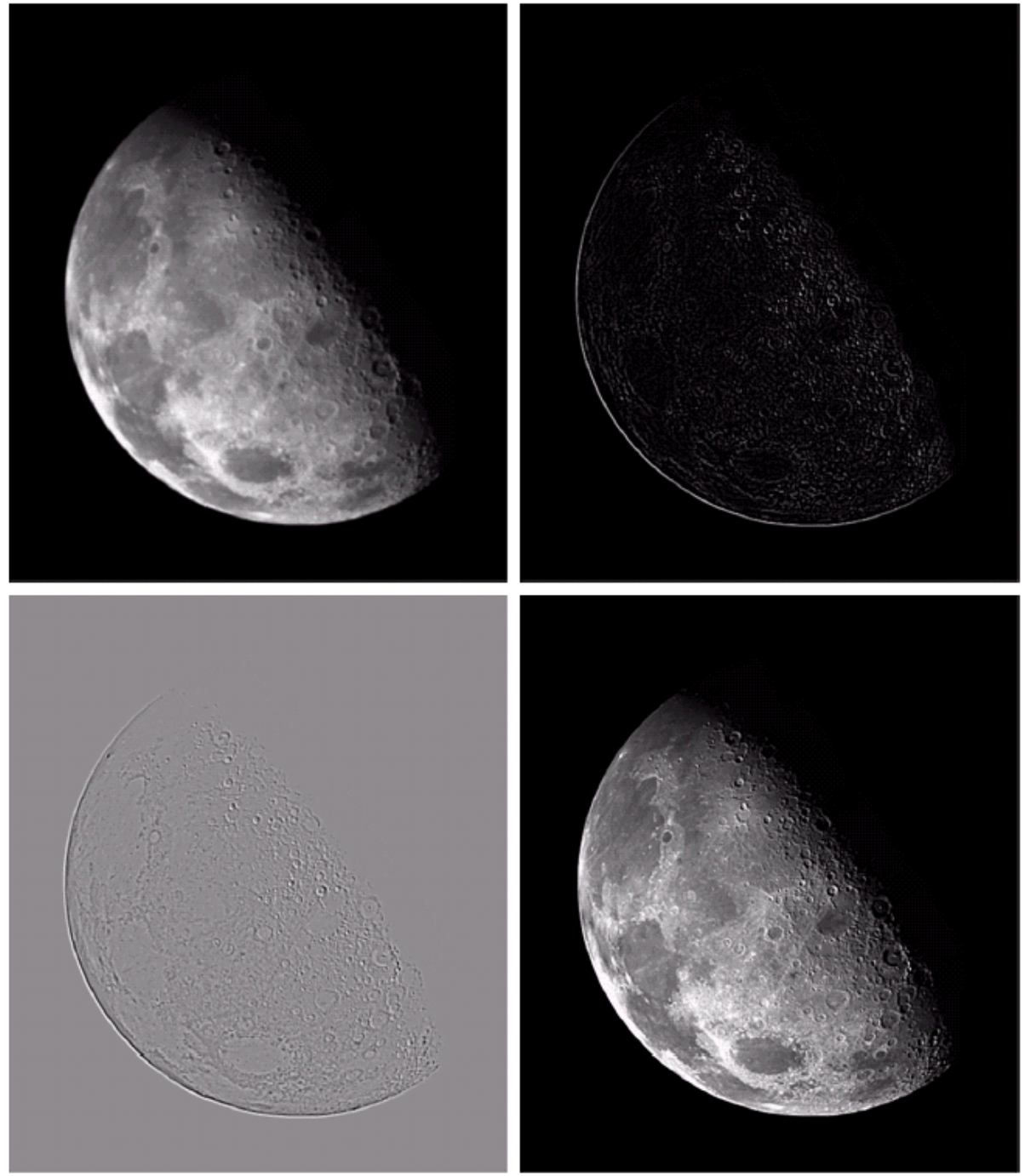


FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)

Sharpening Filters: Unsharp Masking

- Obtain a sharp image by subtracting a lowpass filtered (i.e., smoothed) image from the original image:

$$Highpass = Original - Lowpass$$

Sharpening Filters: High Boost

- Image sharpening emphasizes edges but details (i.e., low frequency components) might be lost.
- **High boost filter**: amplify input image, then subtract a lowpass image.

$$Highboost = A\ Original - Lowpass$$

$$= (A-1)\ Original + Original - Lowpass$$

$$= (A-1)\ Original + Highpass$$

(A-1)

Sharpening Filters: High Boost (cont'd)

- If **A=1**, we get a high pass filter

- If **A>1**, part of the original image is added back to the high pass filtered image.

A>=1
w = 9A-1

-1	-1	-1
-1	w	-1
-1	-1	-1

A=2
w = 17

-1	-1	-1
-1	17	-1
-1	-1	-1

a b
c d

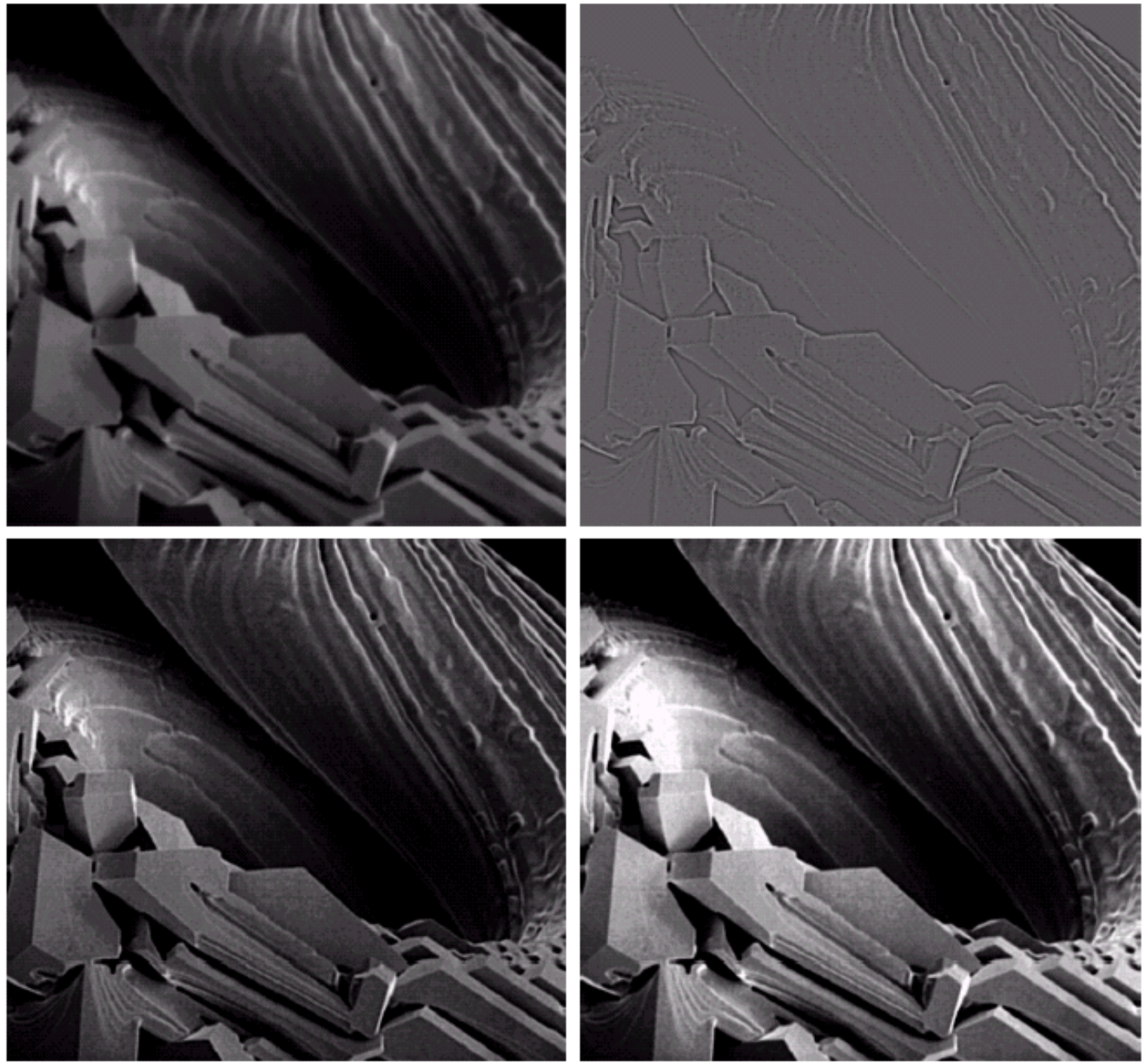


FIGURE 3.43
(a) Same as Fig. 3.41(c), but darker.
(a) Laplacian of (a) computed with the mask in Fig. 3.42(b) using $A = 0$.
(c) Laplacian enhanced image using the mask in Fig. 3.42(b) with $A = 1$. (d) Same as (c), but using $A = 1.7$.