Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 03
Intensity Transformation and Spatial Filtering

1. Introduction

• All the image processing techniques discussed in this topic are implemented in the spatial domain, which is simply the plane containing the pixels of an image.

• Later we will discuss techniques implemented in the frequency domain.

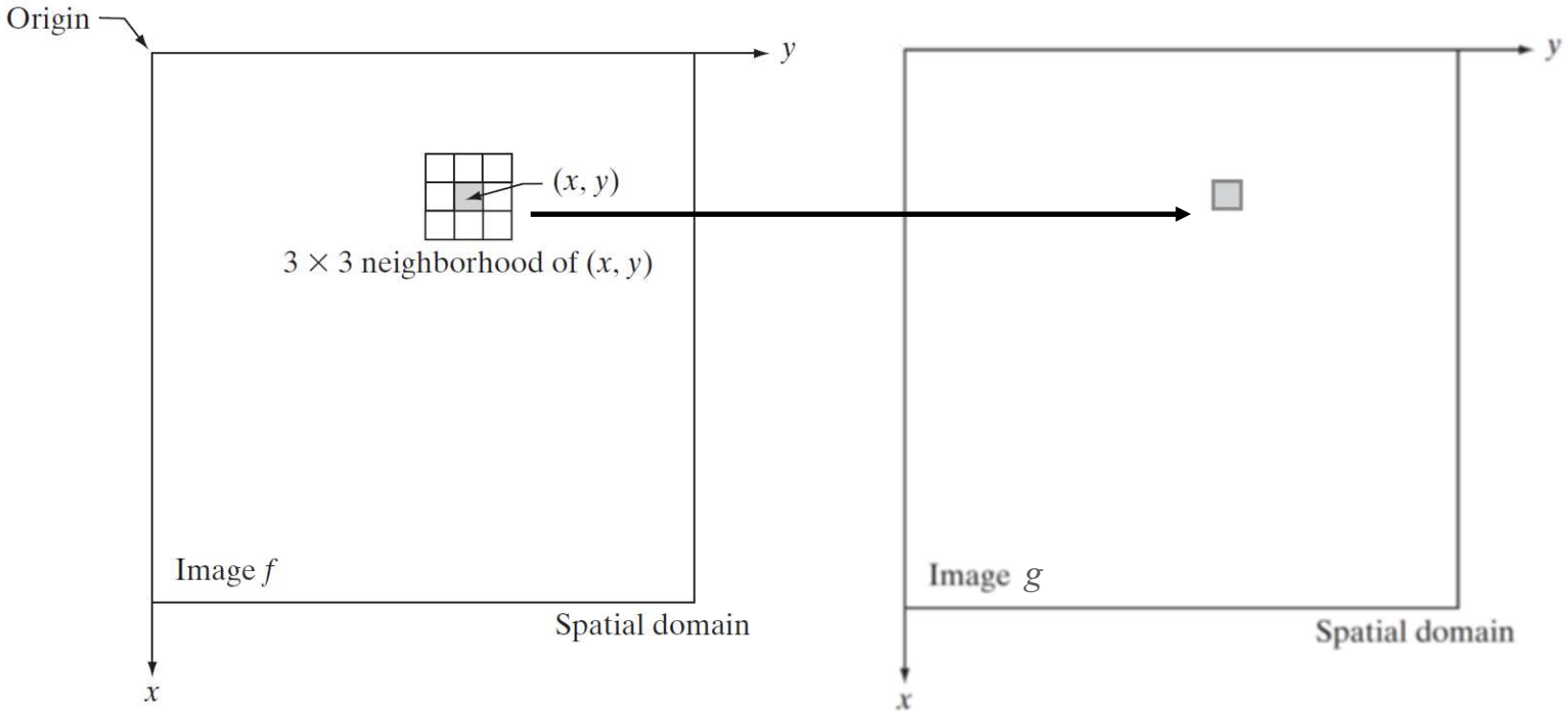
• The spatial domain processes we discuss in this chapter can be denoted by the expression:

$$g(x,y) = T[f(x,y)]$$

where f(x,y) is the input image, g(x,y) is the output image, and T is an operator on *f* defined over a neighborhood of point (x, y).

1. Introduction

• A 3 x 3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

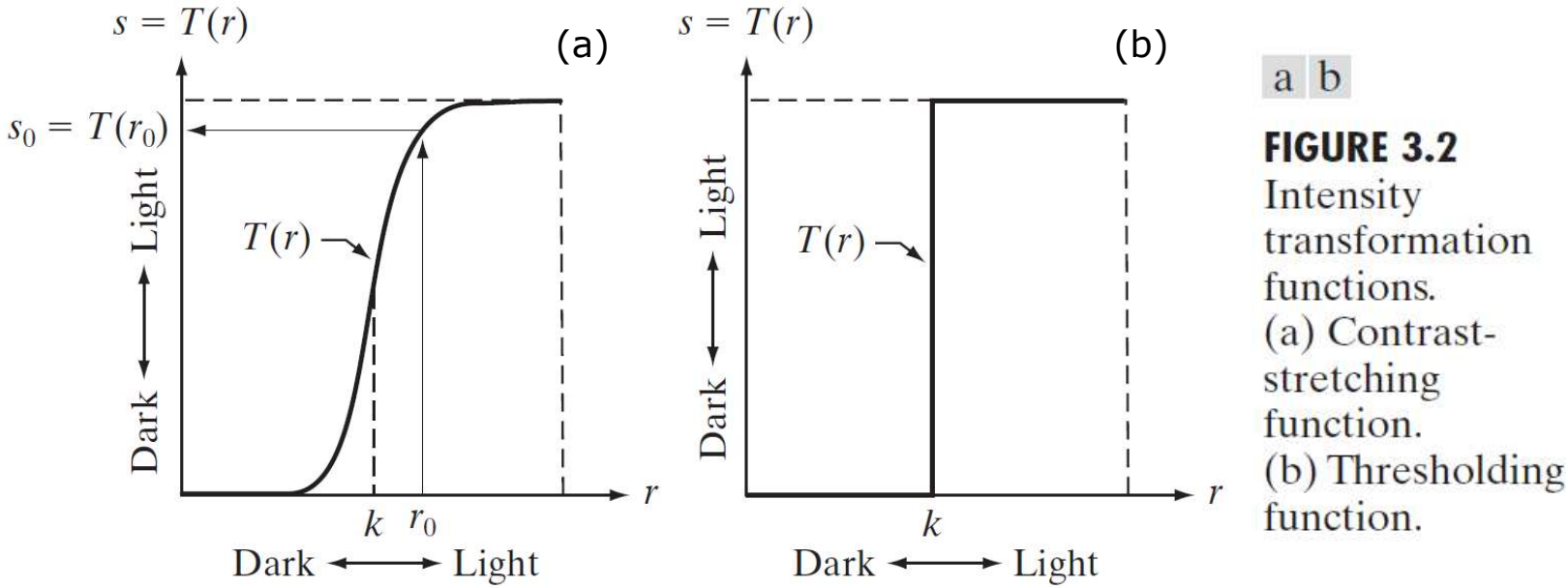


1. Introduction

• The smallest possible neighborhood is of size 1 x 1. In this case, T becomes an intensity transformation function of the form,

$$s = T(r)$$

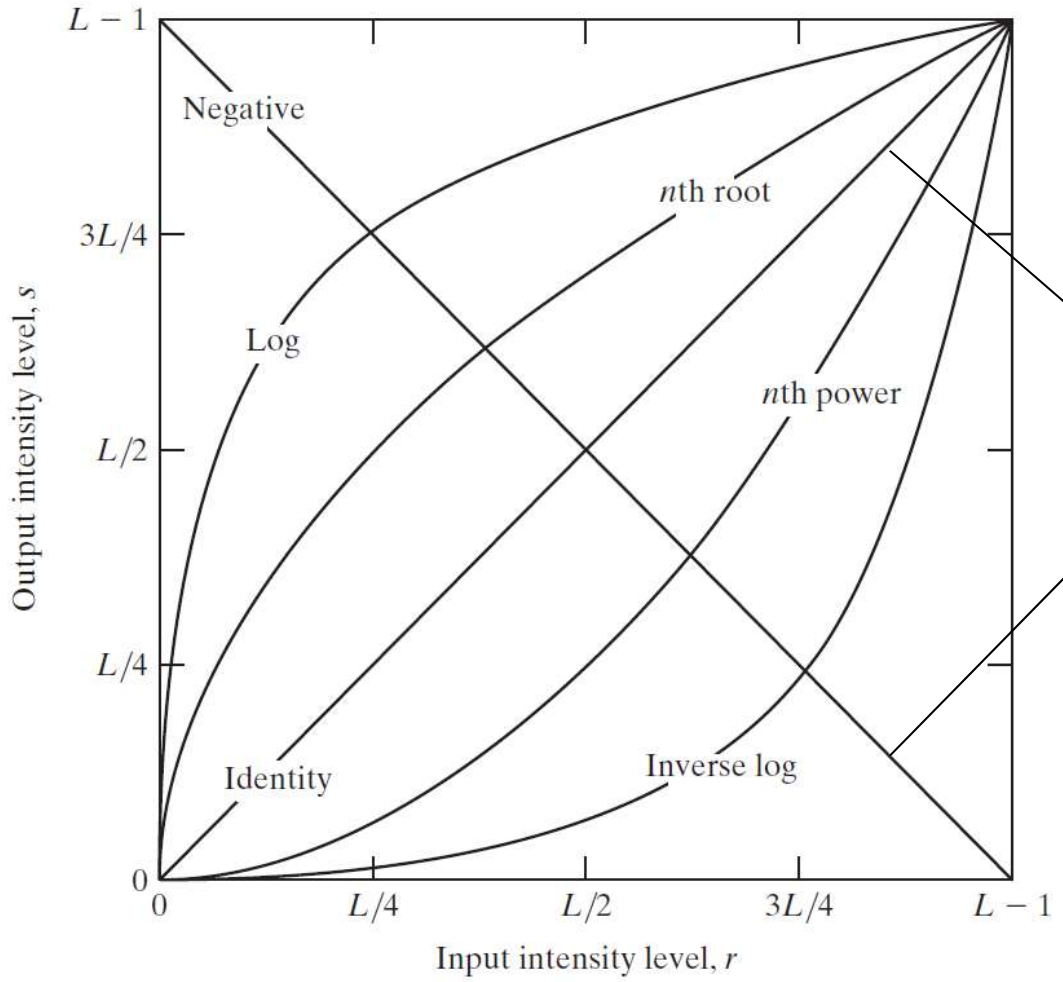
• Examples: *contrast stretching* e *thresholding.*



a b

FIGURE 3.2 Intensity transformation functions. (a) Contrast-stretching function. (b) Thresholding function.

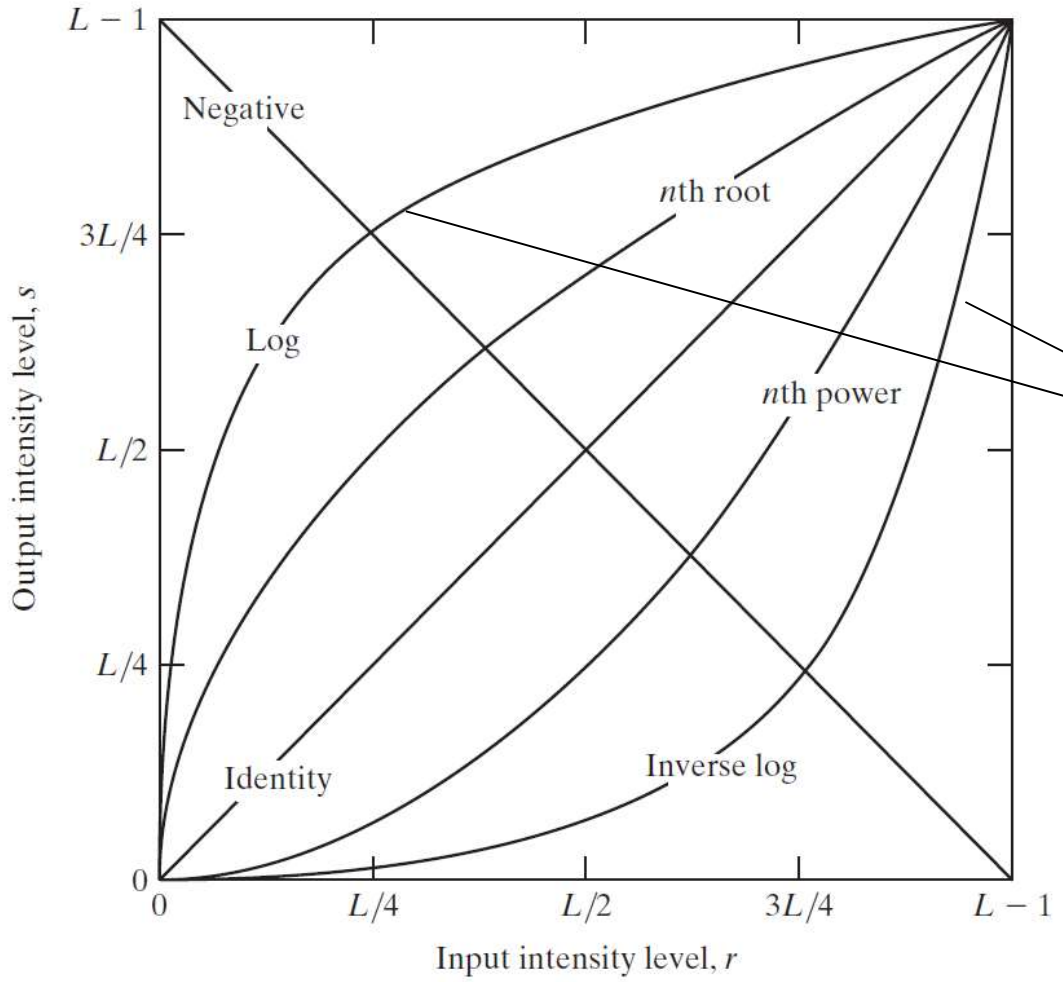
2. Some Basic Intensity Transformation Functions



linear

FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

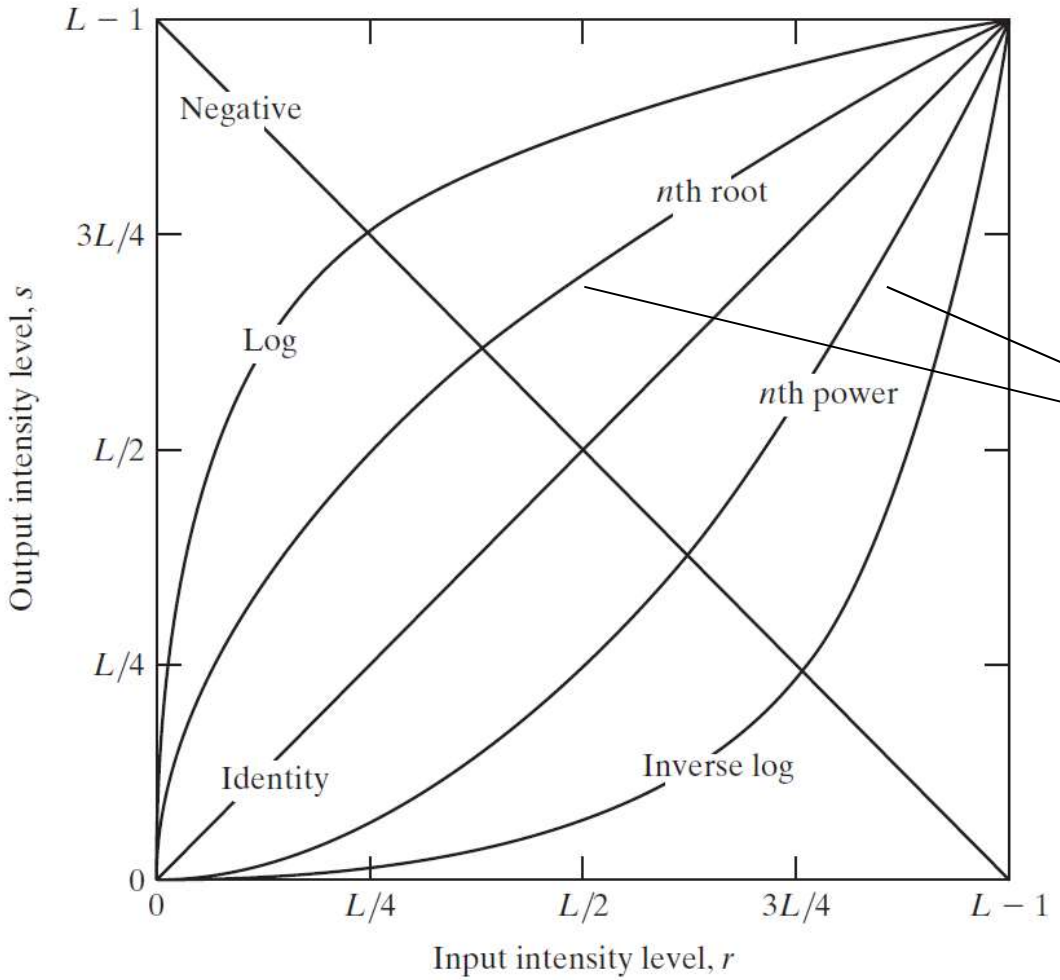
2. Some Basic Intensity Transformation Functions



logarithmic

FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

2. Some Basic Intensity Transformation Functions

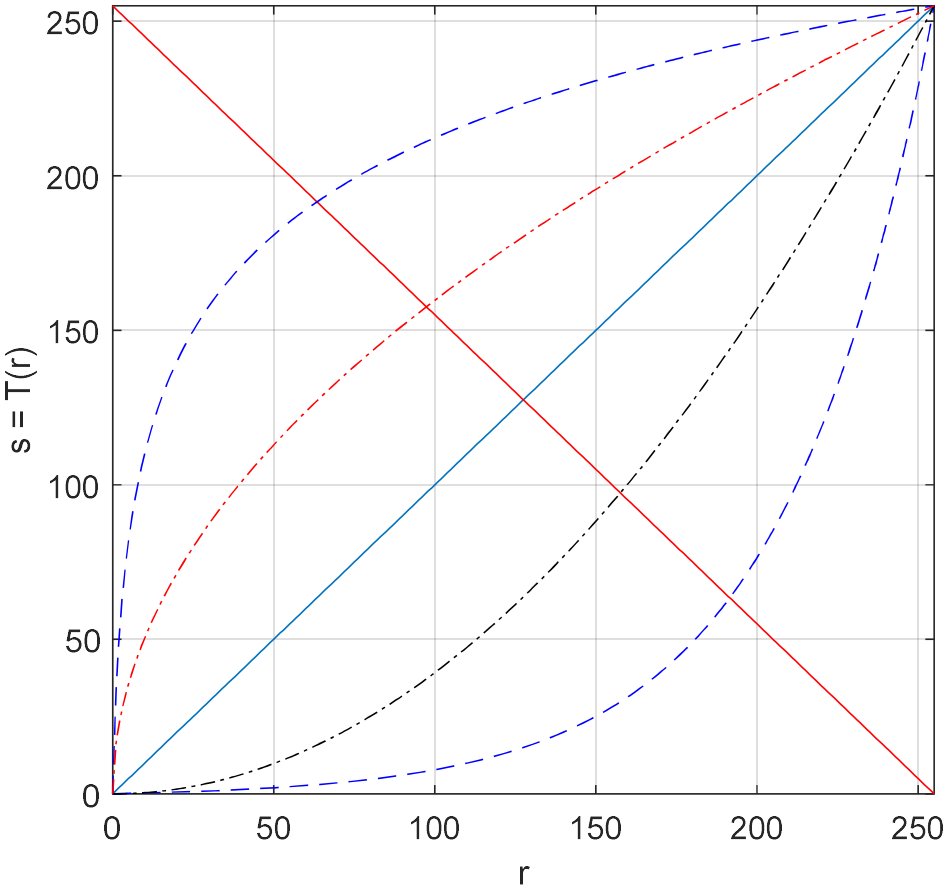


power-law

FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

2. Some Basic Intensity Transformation Functions

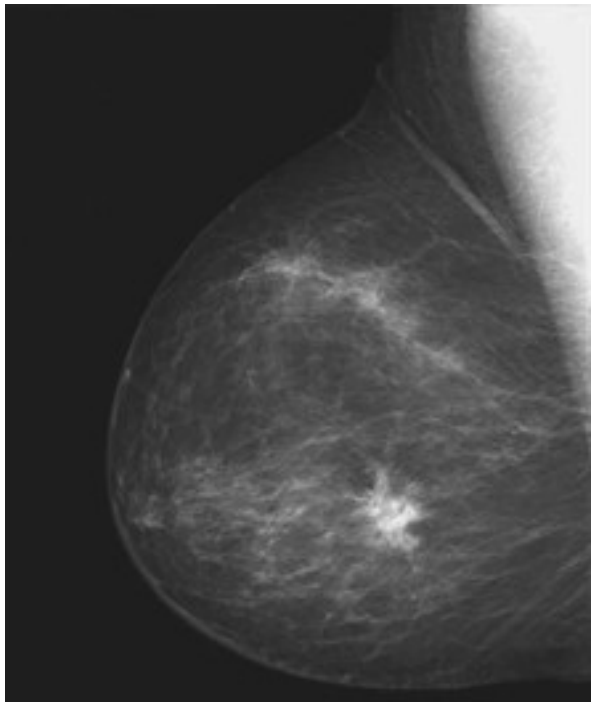
- MATLAB: s09intensity_transformations.m

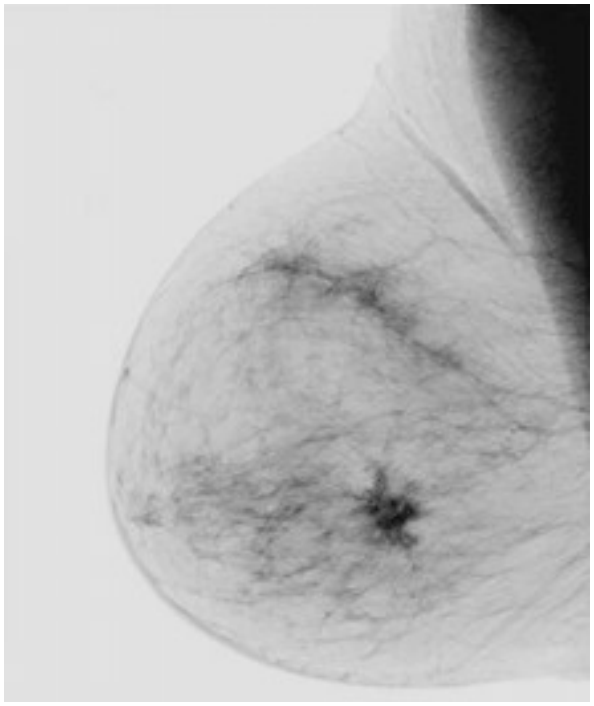


2. Some Basic Intensity Transformation Functions

• The **negative** of an image with intensity levels in the range [0 L-1] is obtained by using the negative transformation,

$$s = L - 1 - r$$





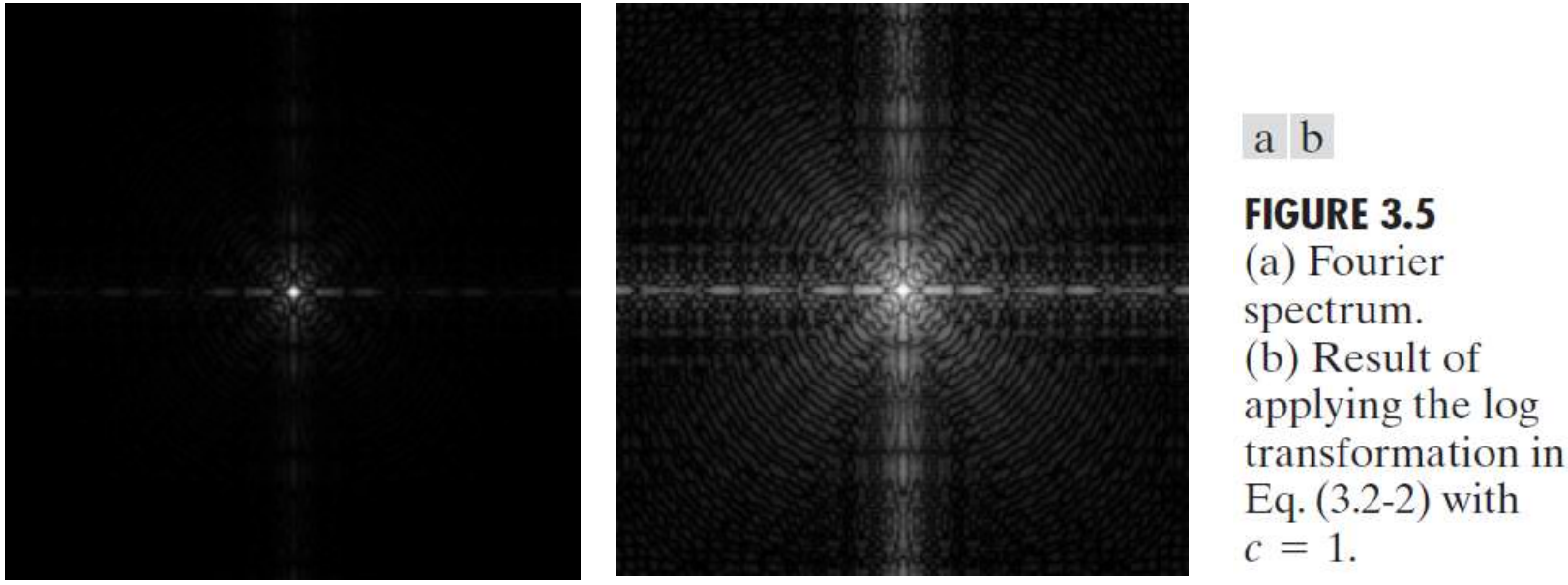
a b

FIGURE 3.4
(a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

2. Some Basic Intensity Transformation Functions

- **Log Transformations:** We use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true of the inverse log transformation.

$$s = c\log(1+r)$$



a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation in Eq. (3.2-2) with $c = 1$.

2. Some Basic Intensity Transformation Functions

- **Power-Law (Gamma) Transformations:** $s = cr^{\gamma}$

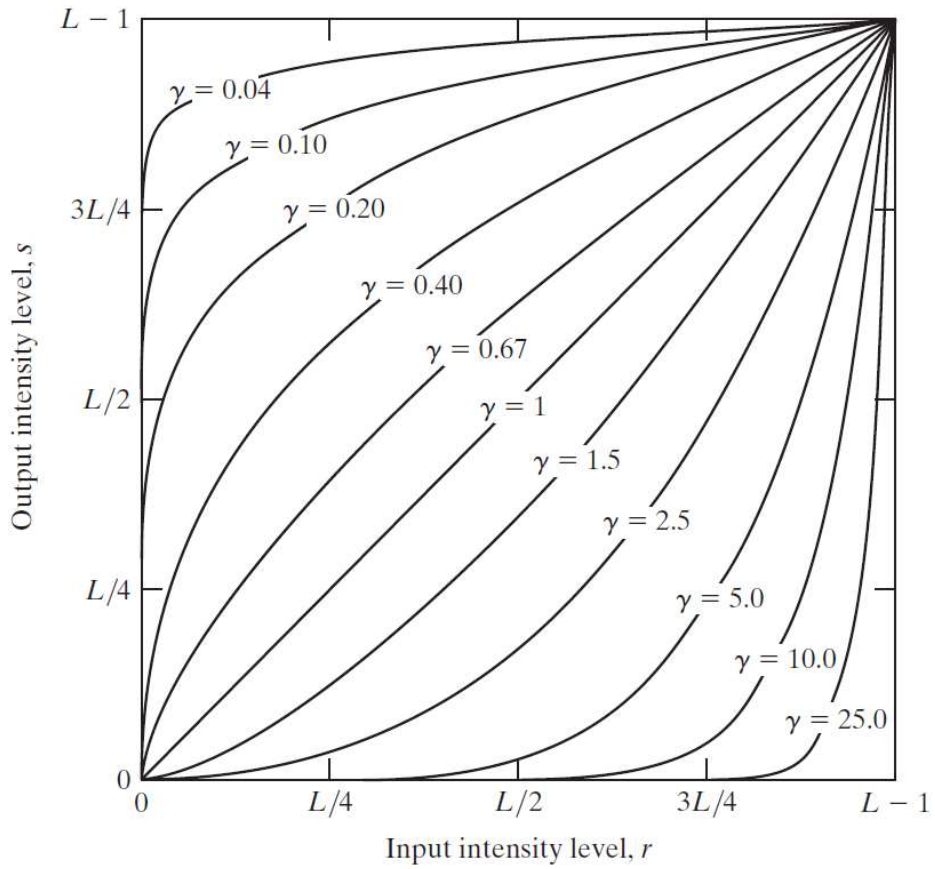


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

2. Some Basic Intensity Transformation Functions

• A variety of **devices** used for image capture, printing, and display **respond according to a power-law**.

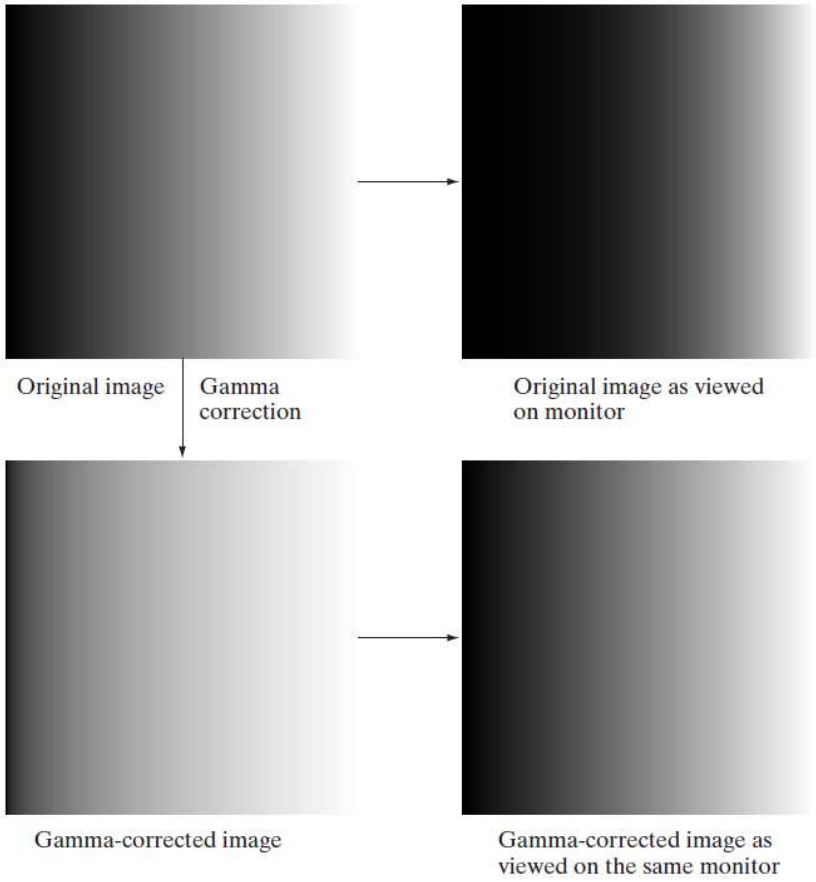
• By convention, the exponent in the power-law equation is referred to as **gamma**.

• The process used to correct these power-law response phenomena is called **gamma correction**.

• Gamma correction is important if displaying an image accurately on a computer screen is of concern.

2. Some Basic Intensity Transformation Functions

- Example of gamma correction:

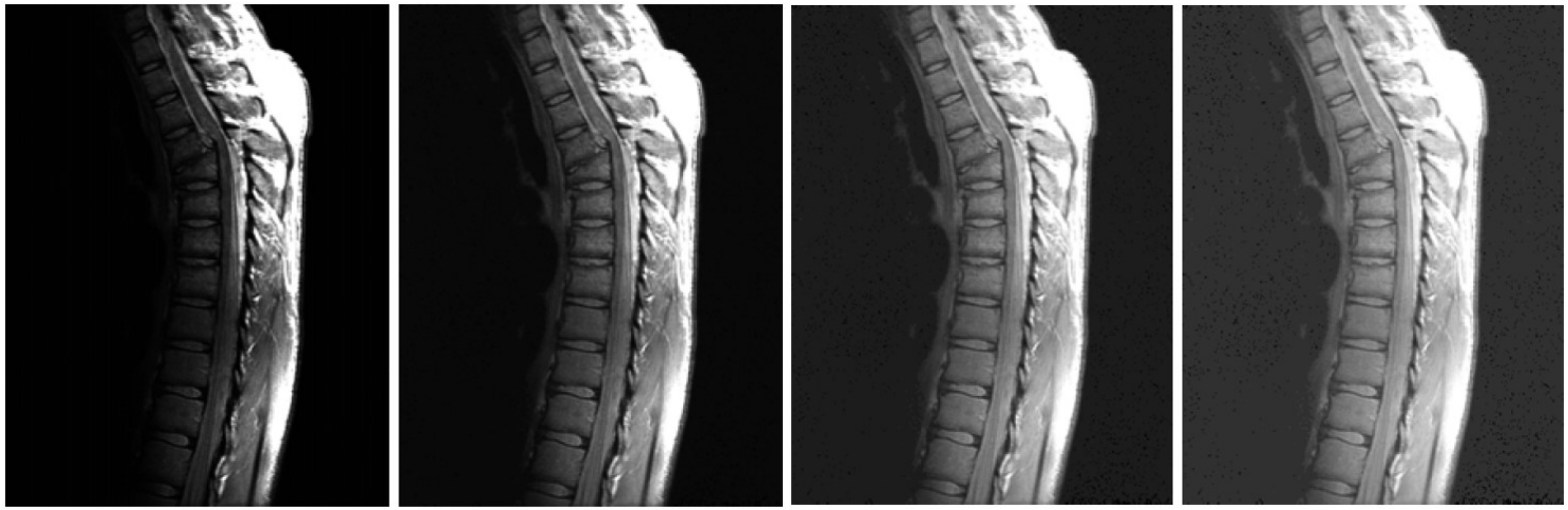


a b
c d

FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).

2. Some Basic Intensity Transformation Functions

- Power-law general-purpose contrast manipulation:



a b c d

FIGURE 3.8 (a) Magnetic resonance image (MRI) of a fractured human spine. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

2. Some Basic Intensity Transformation Functions

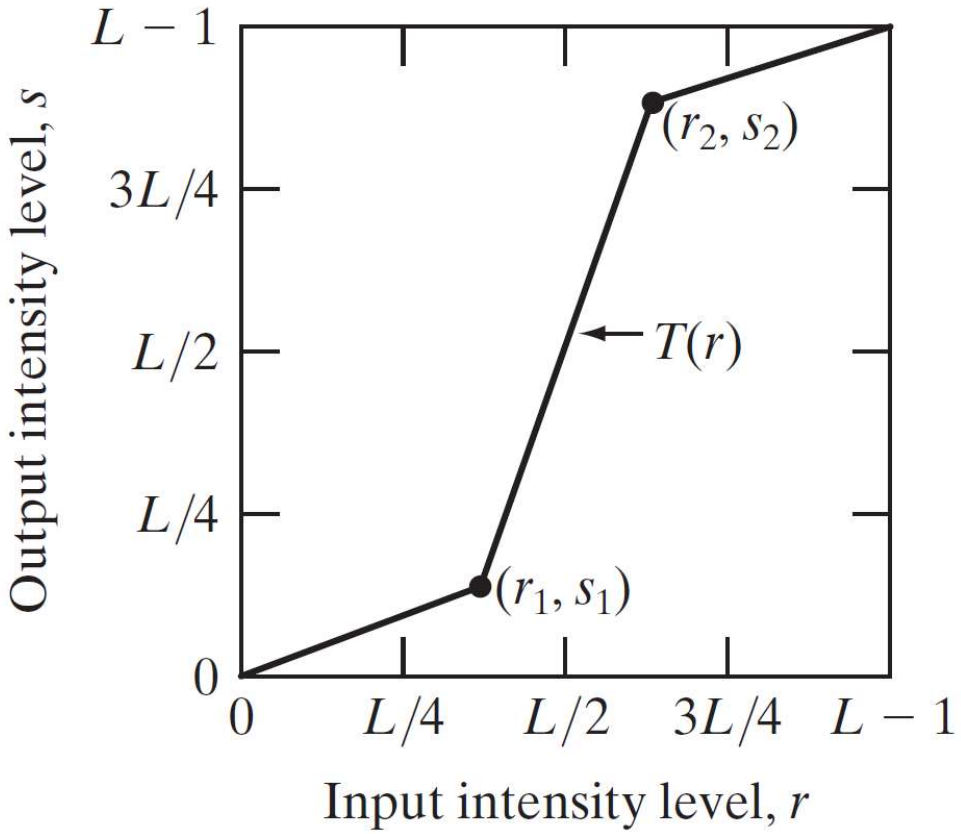
- Power-law general-purpose contrast manipulation:

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0$, 4.0, and 5.0, respectively. (Original image for this example courtesy of NASA.)

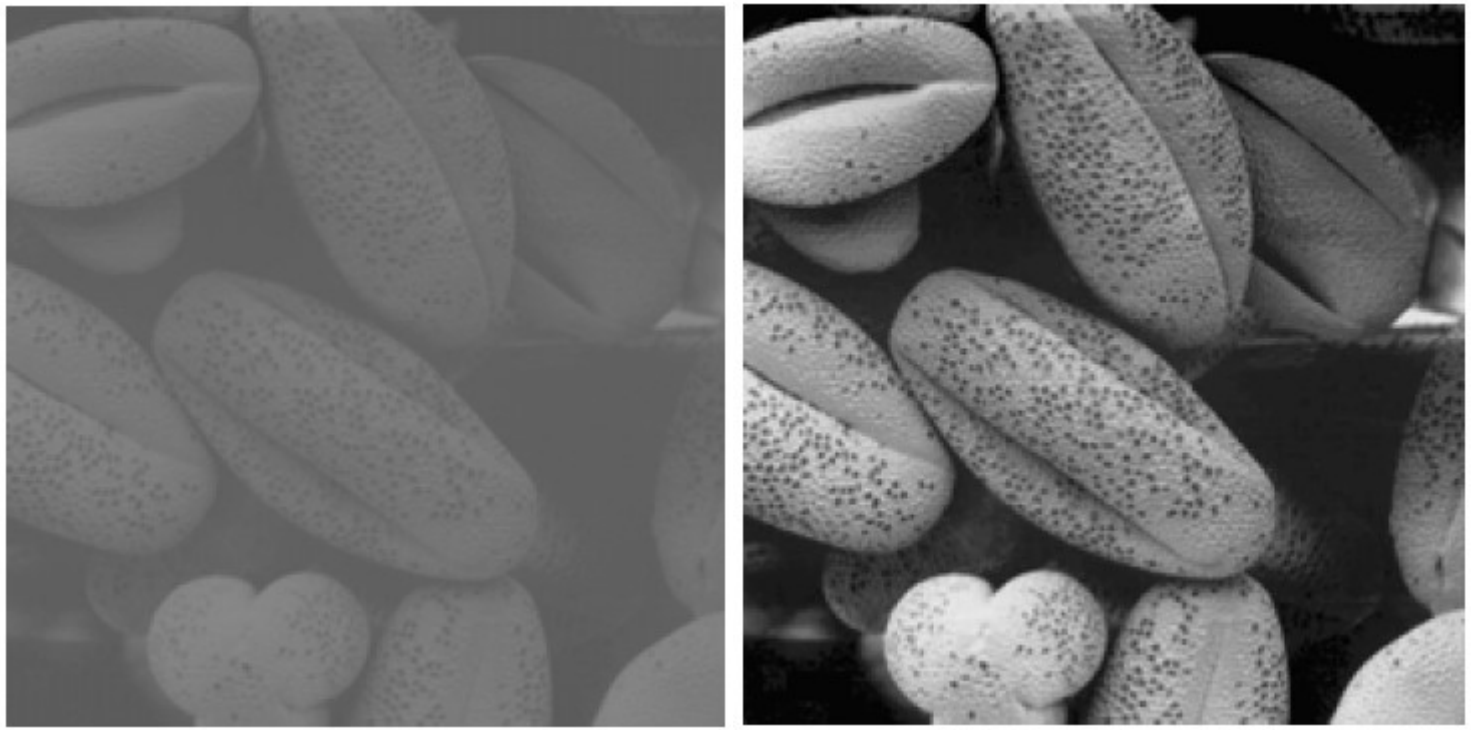
2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **contrast stretching**



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **contrast stretching**



2. Some Basic Intensity Transformation Functions

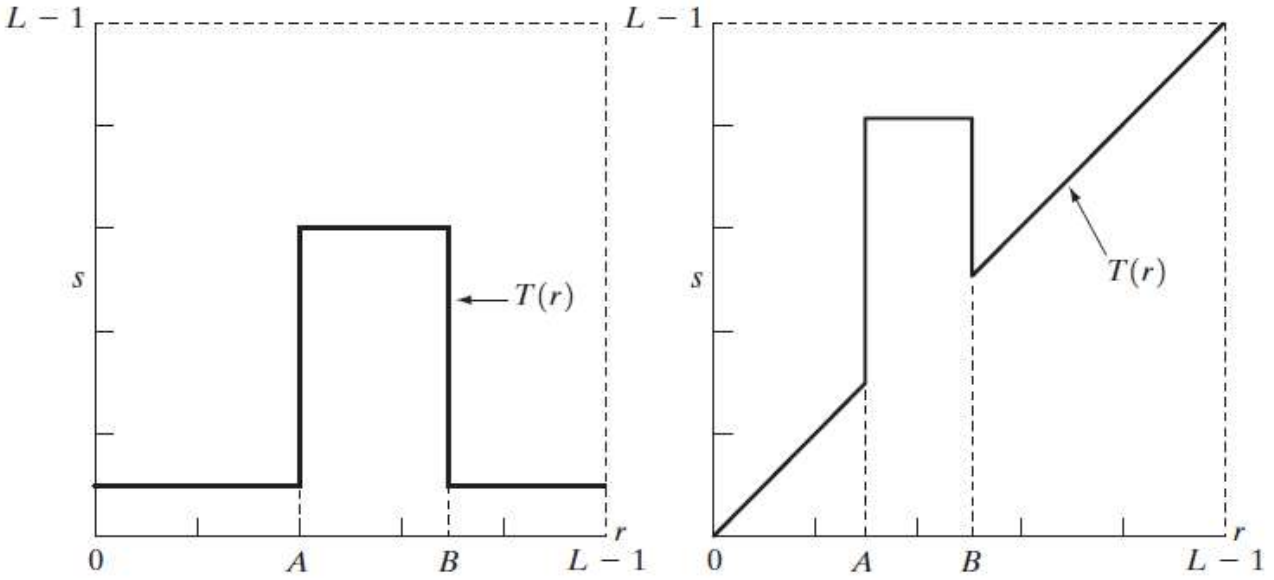
- Piecewise-Linear Transformation Functions:

thresholding

$$s = \begin{cases} 0, & \text{se r} \le m \\ 255, & \text{se r} > m \end{cases}$$

2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **Intensity-level slicing**

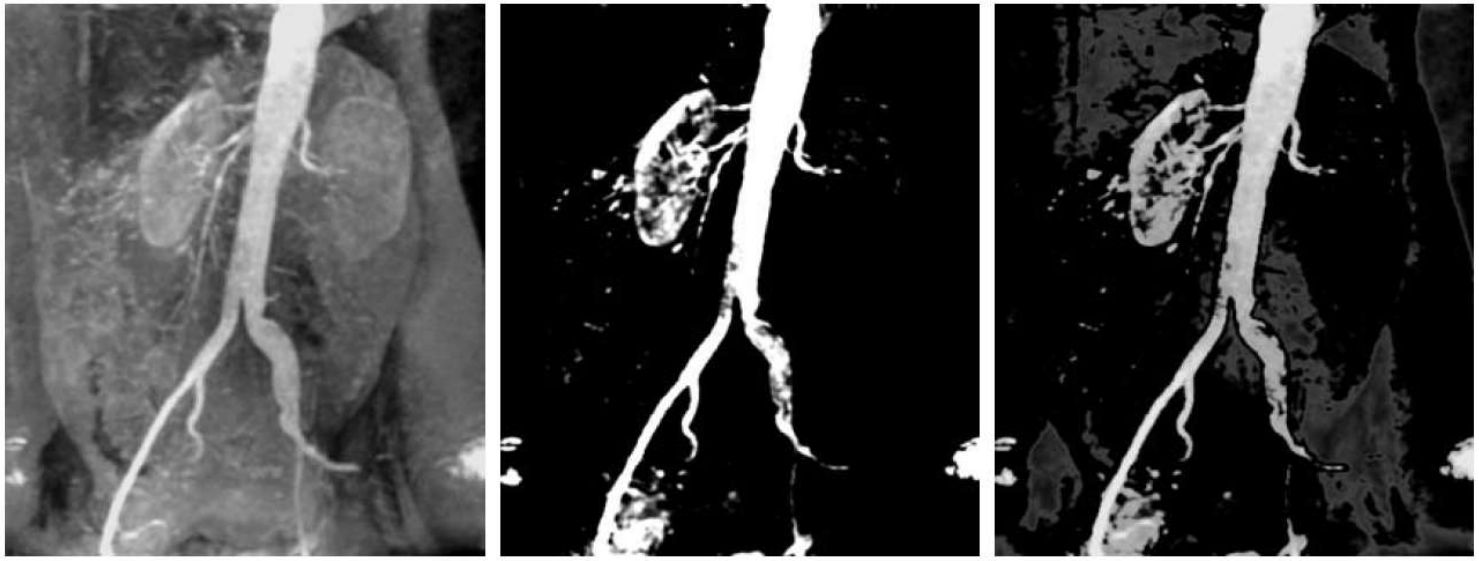


a b

FIGURE 3.11 (a) This transformation highlights intensity range $[A, B]$ and reduces all other intensities to a lower level. (b) This transformation highlights range $[A, B]$ and preserves all other intensity levels.

2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **Intensity-level slicing**

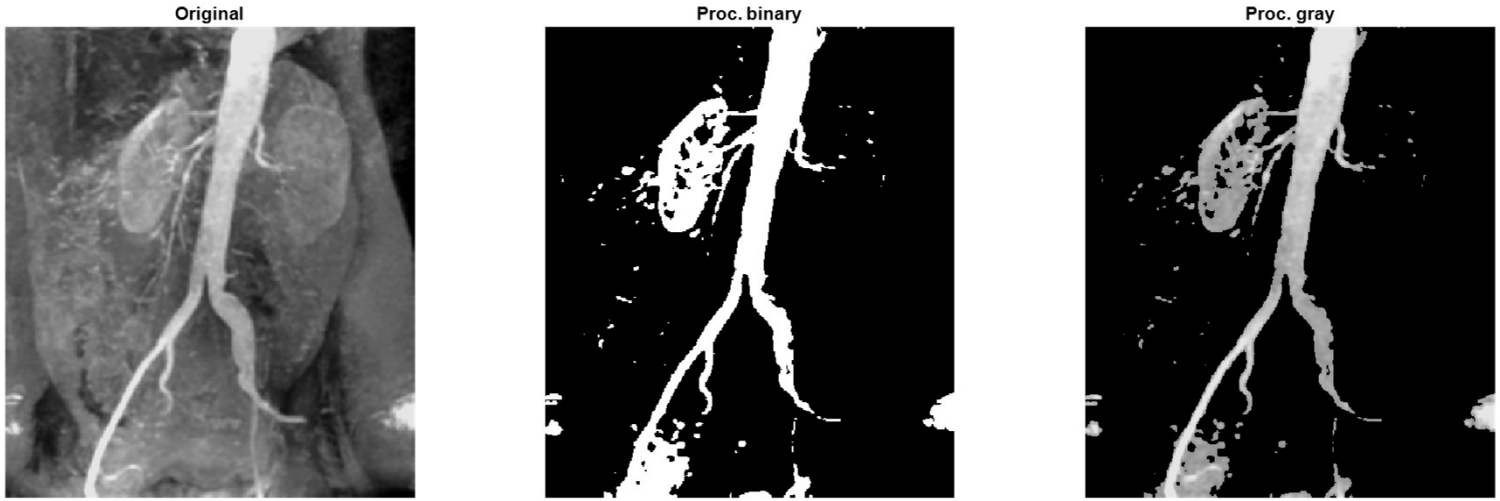


a b c

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.)

2. Some Basic Intensity Transformation Functions

- **MATLAB:** s24kidney.m



2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**

• Instead of highlighting **intensity-level ranges**, we could highlight the contribution made to total image appearance by **specific bits**.

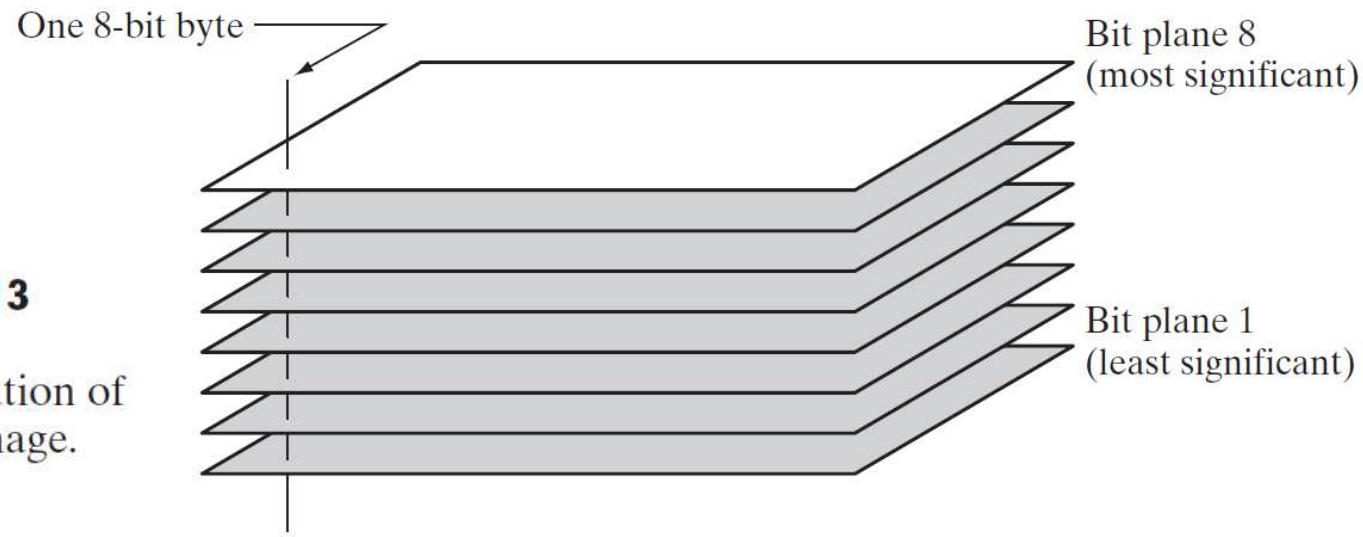
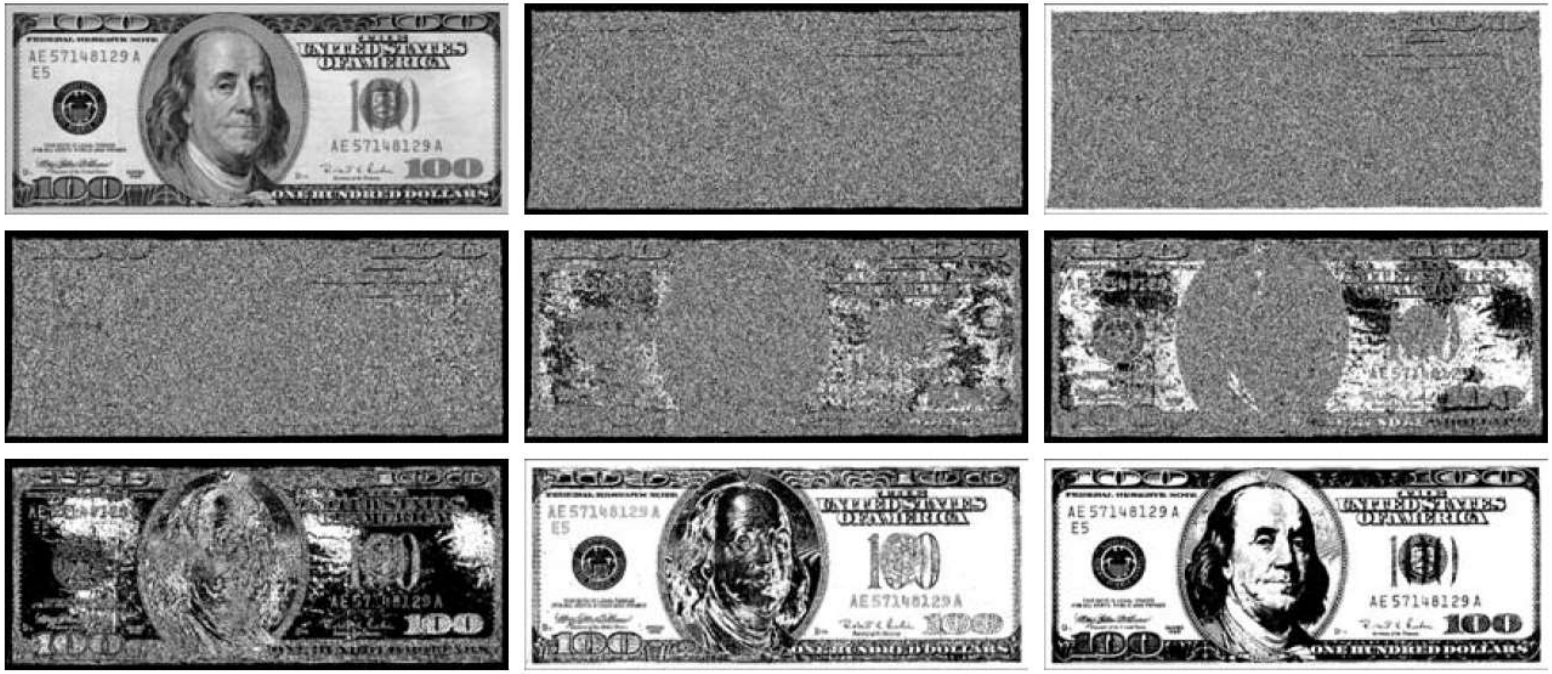


FIGURE 3.13
Bit-plane representation of an 8-bit image.

2. Some Basic Intensity Transformation Functions

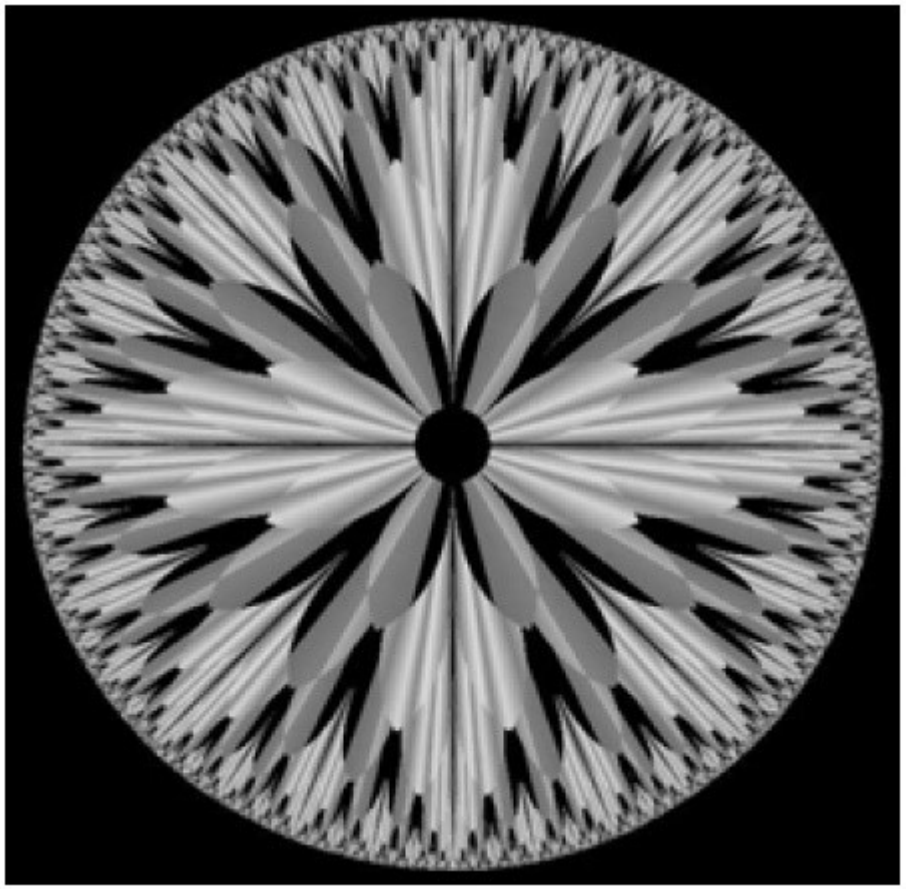


a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

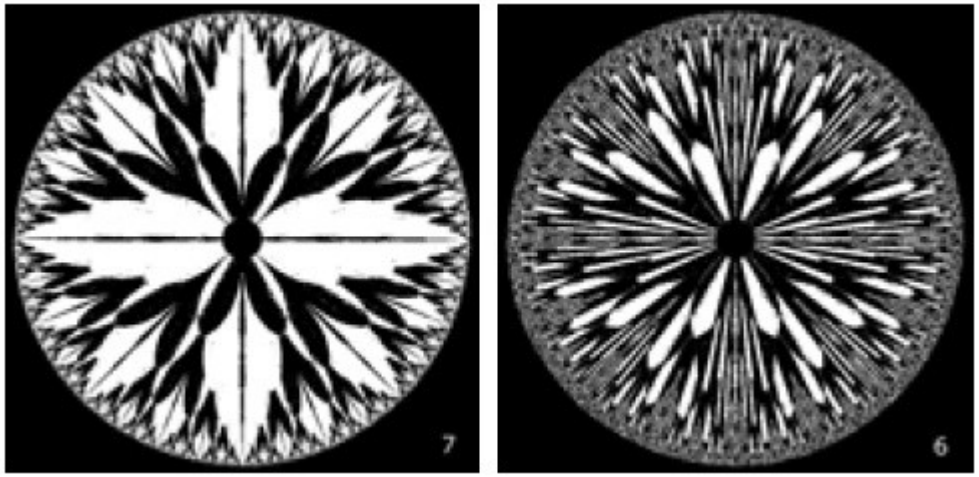
2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**



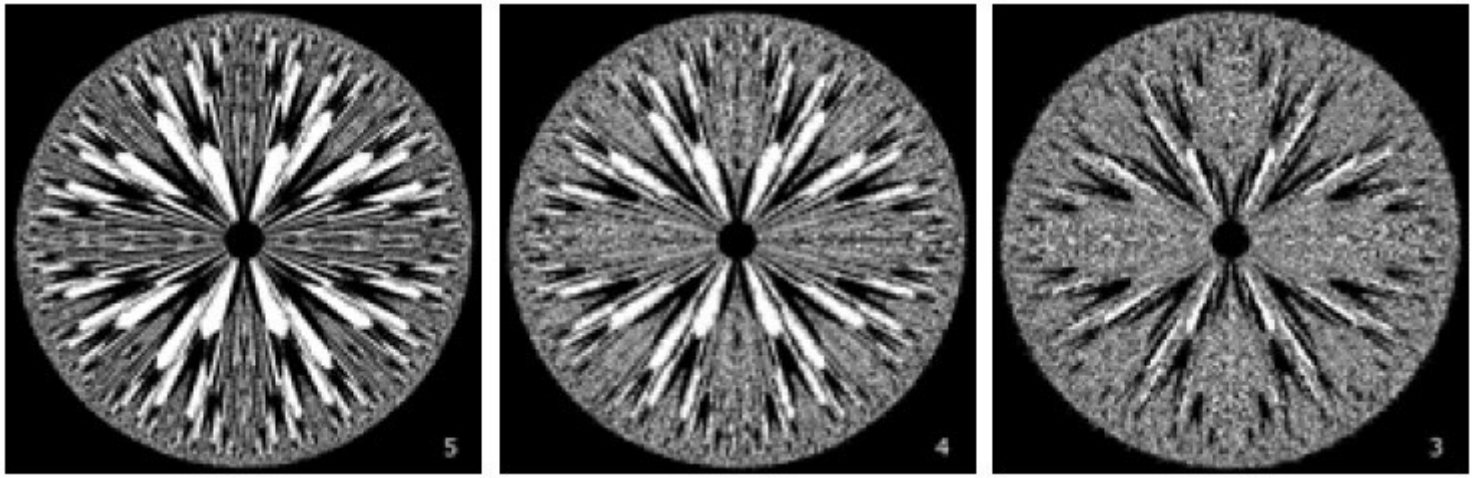
2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**



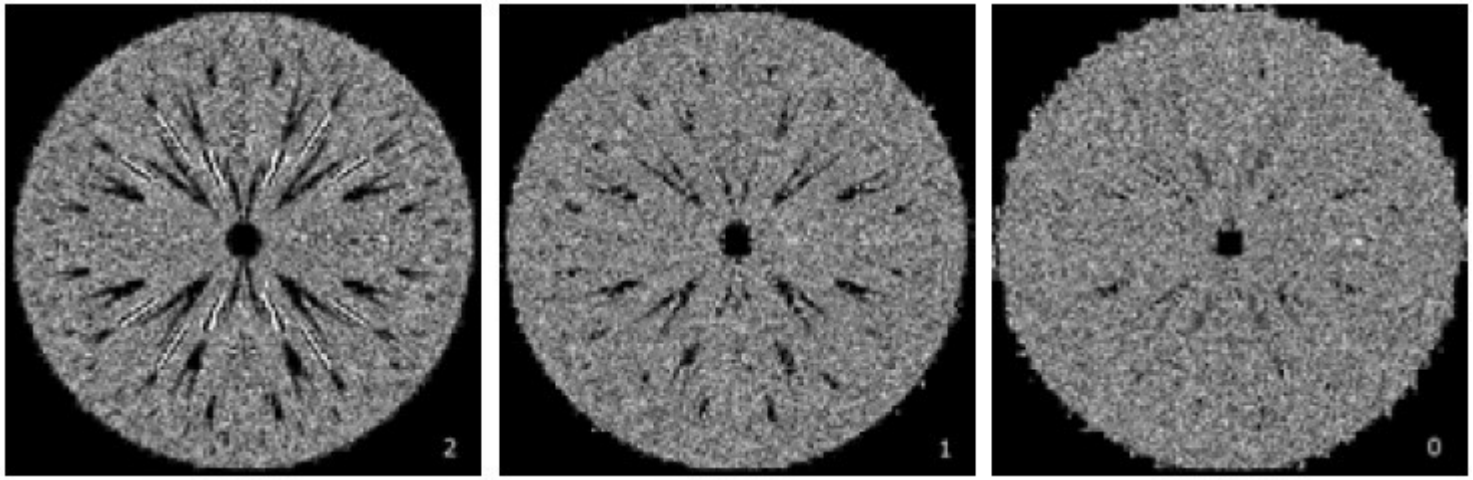
2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**



2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**



2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**

• Decomposing an image into its bit planes is useful for analyzing the relative **importance of each bit-plane** in the image.

• This type of decomposition is useful for image compression, in which fewer than all planes are used in reconstructing an image.

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).

2. Some Basic Intensity Transformation Functions

• Piecewise-Linear Transformation Functions: **bit-plane slicing.**

3. Histogram Processing

• The histogram of a digital image with intensity levels in the range [0 L-1] is a discrete function,

$$h(r_k) = n_k,$$

where r_k is the kth intensity value and n_k the number of pixels in the image with intensity r_k.

• It is common practice to normalize a histogram by dividing each of its components by the total number of pixels N in the image,

$$p(r_k) = n_k / N.$$

• Loosely speaking, $p(r_k)$ is an estimate of the probability of occurrence of intensity level in an image.

3. Histogram Processing

- Example:

2	8	4	8	7	5	6	6
7	5	1	9	4	1	7	2
4	5	2	4	2	2	6	3
8	1	1	4	4	3	4	6
2	8	2	3	1	7	5	7
2	6	2	8	1	0	3	1
1	3	4	3	8	0	7	8
1	5	0	1	9	2	2	7

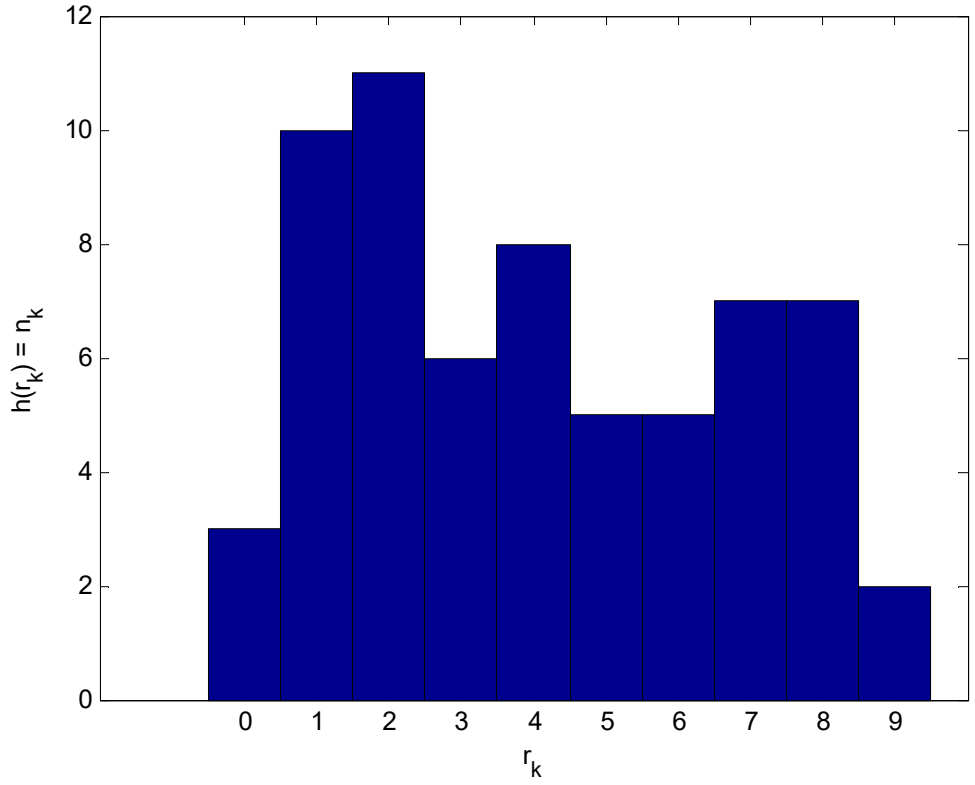
$r_k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$

$h(r_k) = 3, 10, 11, 6, 8, 5, 5, 7, 7, 2 = n_k$

$p(r_k) = n_k/64$

3. Histogram Processing

- Example:

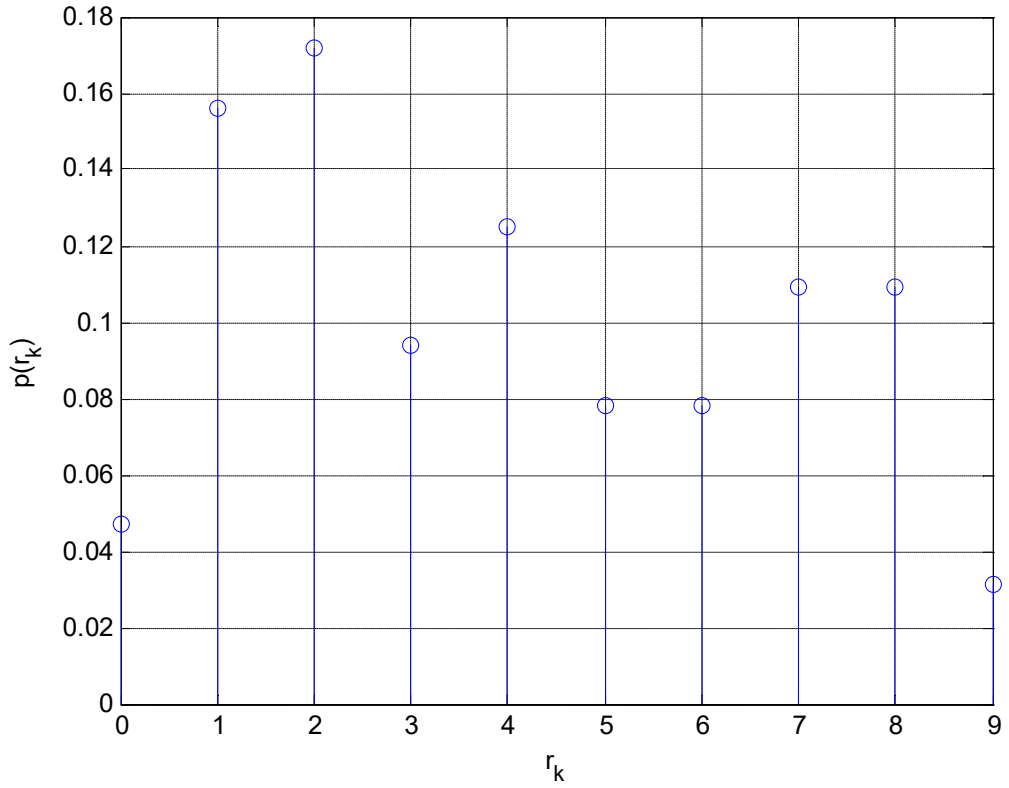


- MATLAB: hist()

3. Histogram Processing

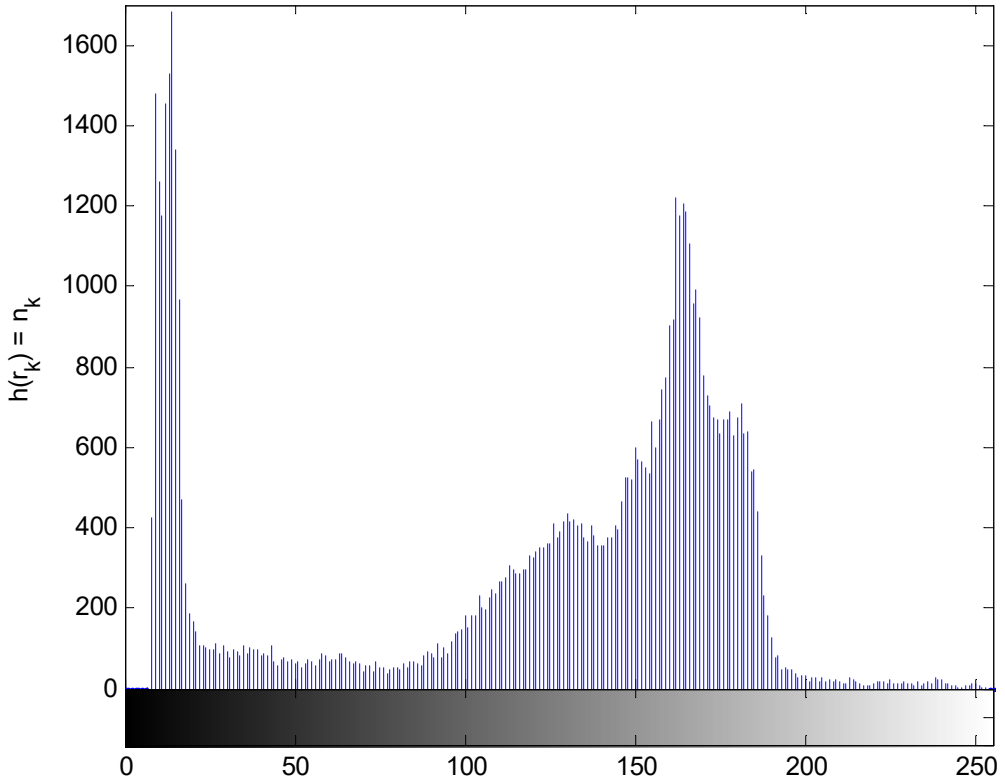
- Example:

$p(r_k) = n_k/64$



3. Histogram Processing

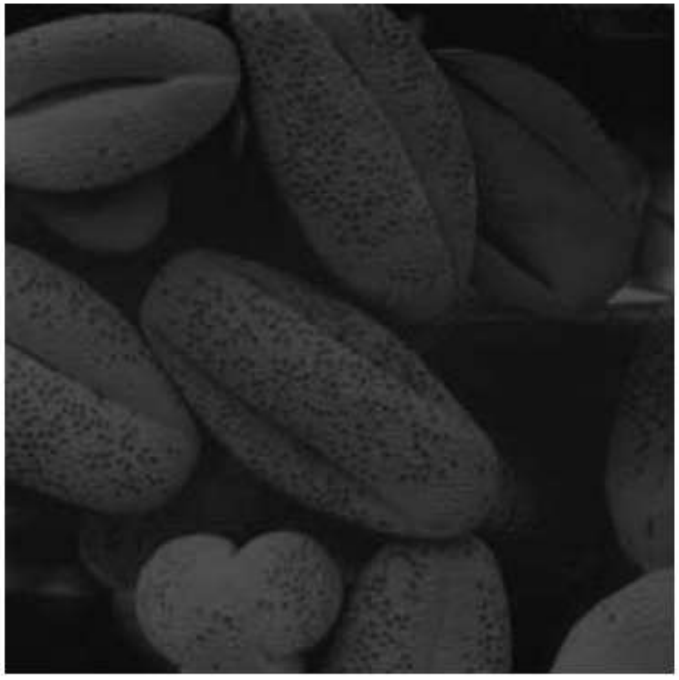
- Example:

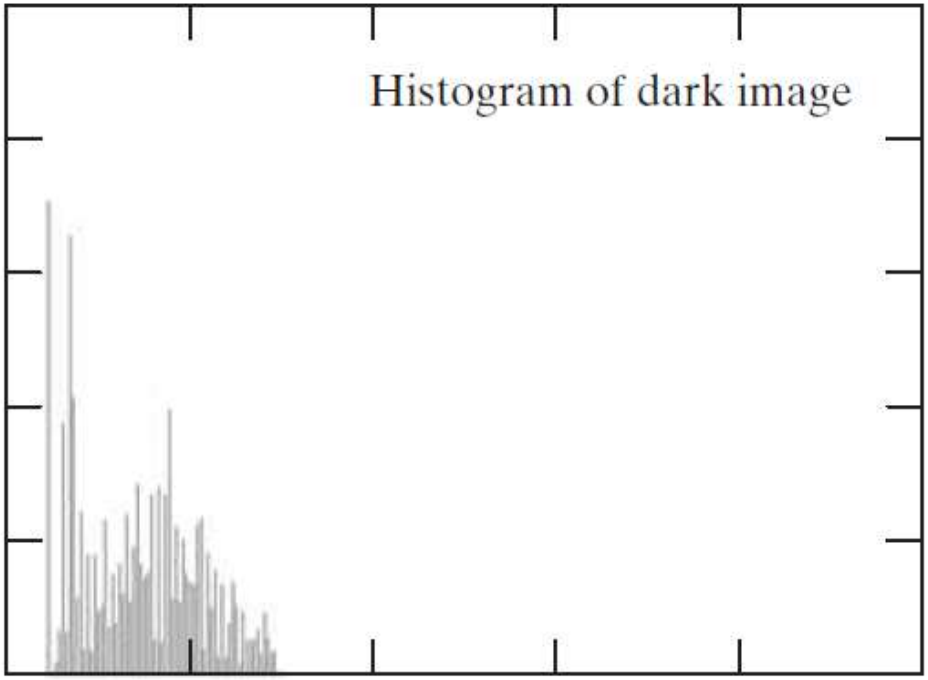


- MATLAB imhist();

3. Histogram Processing

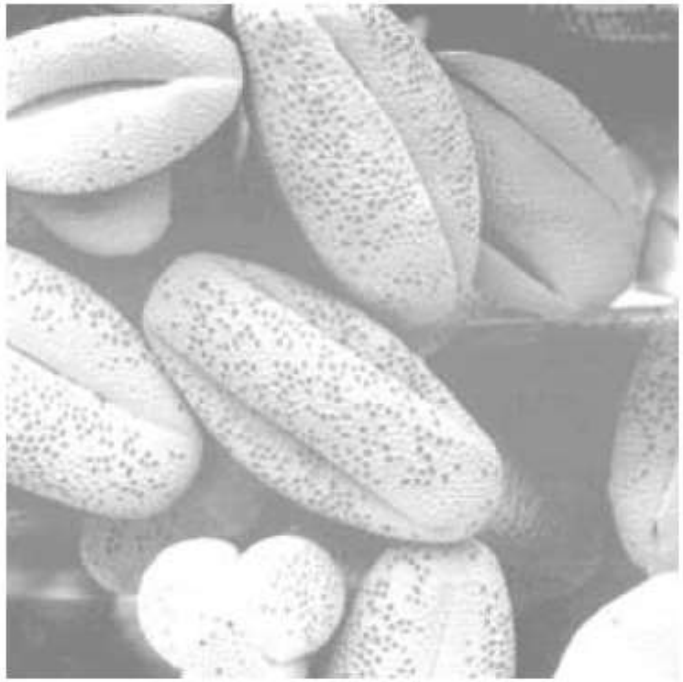
- Example:

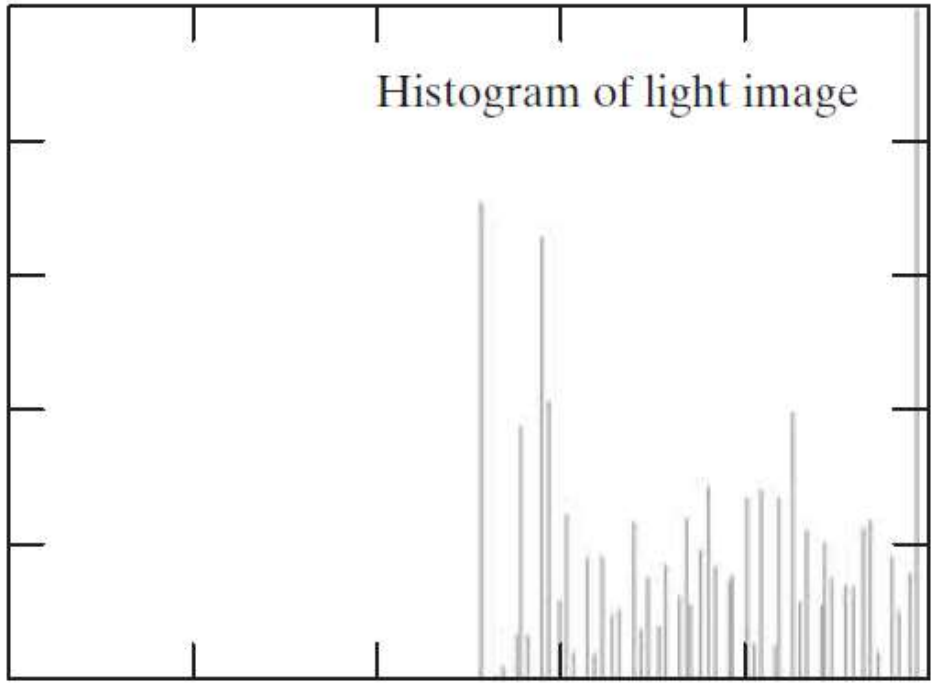




3. Histogram Processing

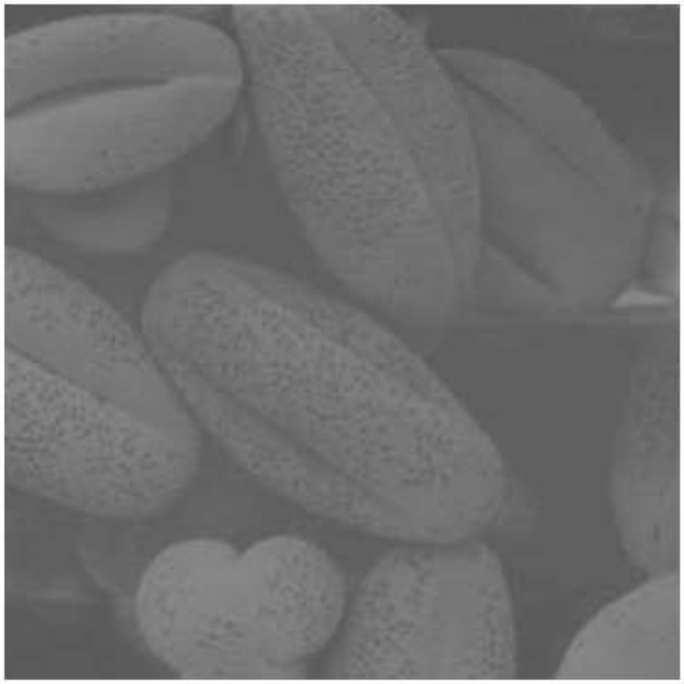
- Example:

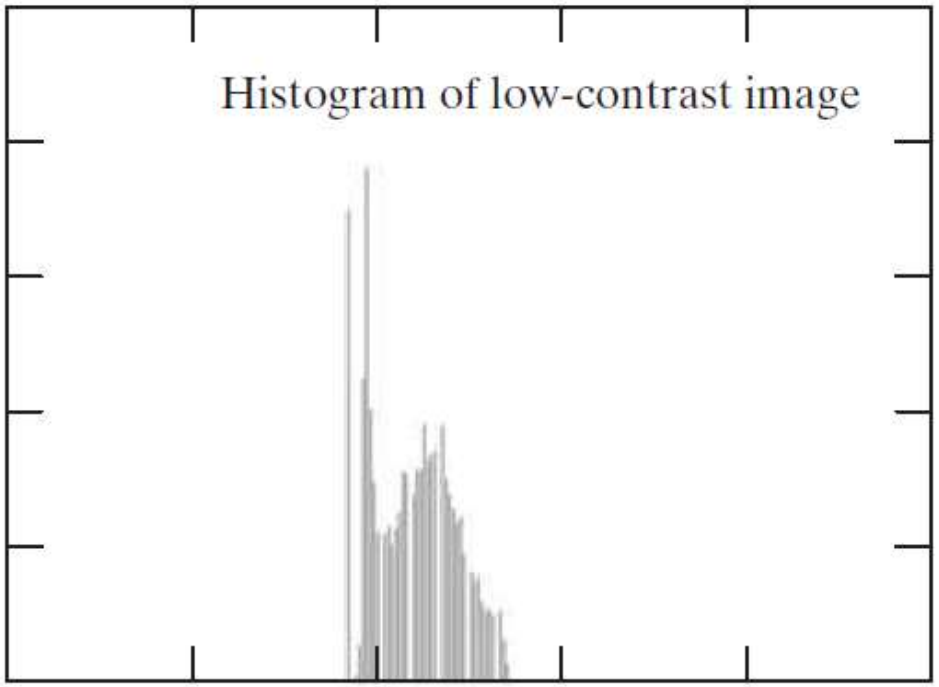




3. Histogram Processing

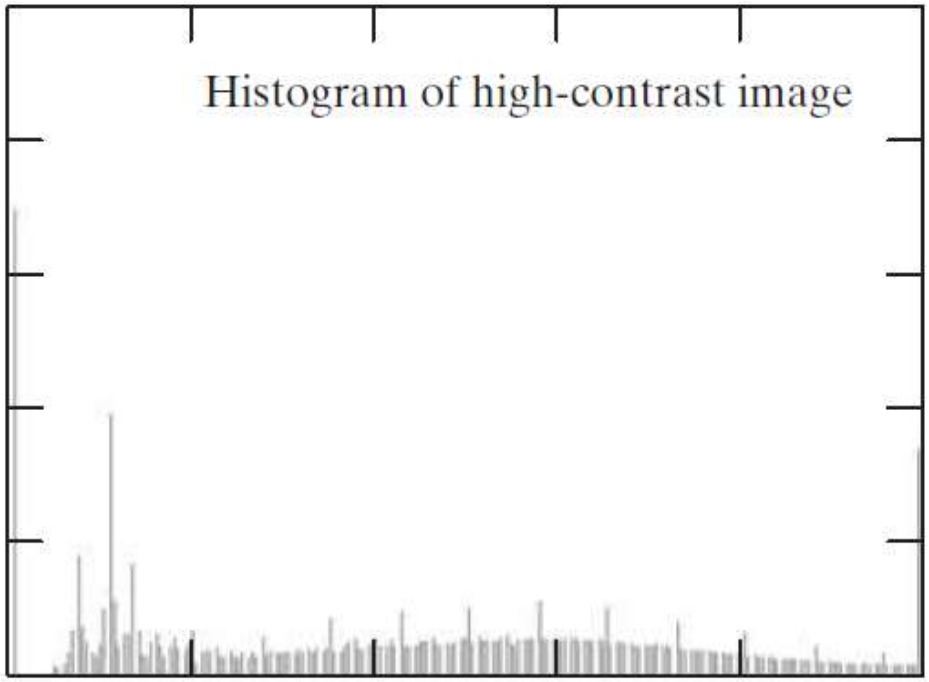
- Example:





3. Histogram Processing

- Example:



3. Histogram Processing

- Histogram Equalization:

  - ✓ Let the variable *r* denote the intensities of an image to be processed.

  - ✓ We assume that *r* is in the range [0, L-1], with *r*=0 representing black and *r* = L-1 representing white.

  - ✓ For *r* satisfying these conditions, we focus attention on transformations of the form

$$s = T(r) \quad 0 \leq r \leq L - 1$$

that produces an output intensity level *s* for every pixel in the input image having intensity *r*.

3. Histogram Processing

- Histogram Equalization

  ✓ We assume that,

  a) *T(r)* is a monotonically increasing function in the interval $0 \leq r \leq L-1$ and

  b) $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$.

  ✓ In some formulations to be discussed later, we use the inverse

$$r = T^{-1}(s) \quad 0 \leq s \leq L-1$$

  in which case we change condition *a)* to

  a') *T(r)* is a strictly monotonically increasing function in the interval $0 \leq r \leq L-1$.

3. Histogram Processing

- Histogram Equalization:

  a. $T(r)$ is a monotonically increasing function:

  ✓ Guarantees that output intensity values will never be less than corresponding input values.

  b. $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$:

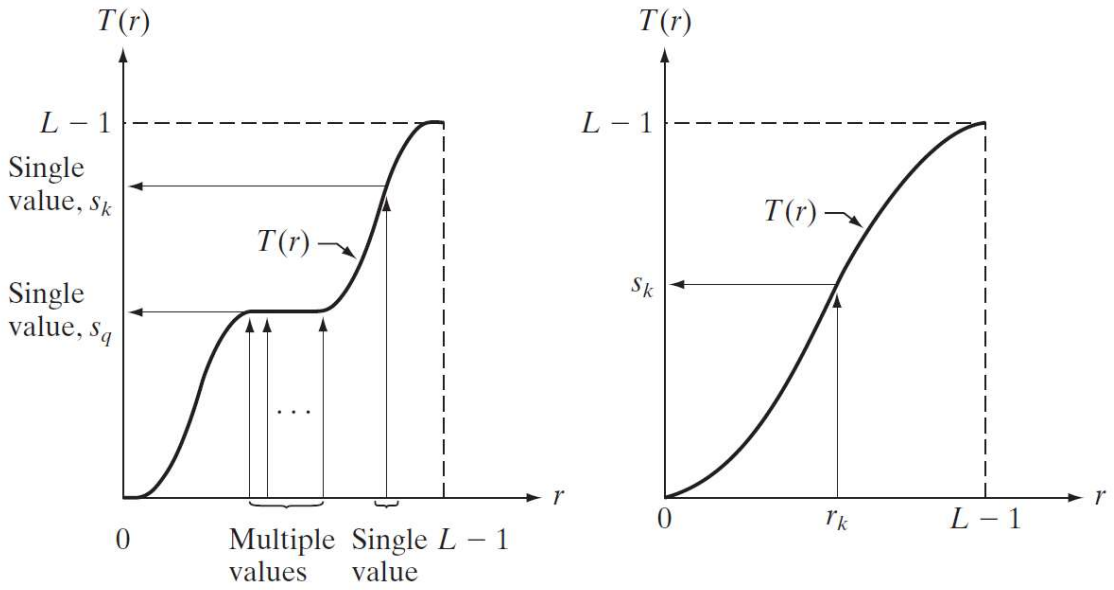
  ✓ Guarantees that the range of output intensities is the same as the input.

  a'. $T(r)$ is a strictly monotonically increasing function:

  ✓ Guarantees that the mappings from s back to r will be one-to-one, thus preventing ambiguities.

3. Histogram Processing

- Histogram Equalization:



a b

FIGURE 3.17 (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.

[†]Recall that a function $T(r)$ is *monotonically increasing* if $T(r_2) \geq T(r_1)$ for $r_2 > r_1$. $T(r)$ is a *strictly monotonically increasing* function if $T(r_2) > T(r_1)$ for $r_2 > r_1$. Similar definitions apply to monotonically decreasing functions.

3. Histogram Processing

- Histogram Equalization:

  - ✓ The intensity levels in an image may be viewed as random variables in the interval [0, L-1].

  - ✓ A fundamental descriptor of a random variable is its probability density function (PDF).

  - ✓ Let $p_r(r)$ and $p_s(s)$ denote the PDFs of r and s, respectively.

  - ✓ A fundamental result from basic probability theory (https://goo.gl/HChKEI):

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

3. Histogram Processing

- Histogram Equalization:

  ✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

  ✓ The right side of this equation is recognized as a scaled version of the cumulative distribution function (CDF) of random variable *r*.

  ✓ To find the corresponding to the transformation, we use,

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

3. Histogram Processing

- Histogram Equalization:

  ✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

$$\frac{ds}{dr} = \frac{dT(r)}{dr}$$

$$= (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right]$$

$$= (L-1)p_r(r)$$

3. Histogram Processing

- Histogram Equalization:

  ✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

$$= p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right|$$

$$= \frac{1}{L-1} \qquad 0 \le s \le L-1$$

3. Histogram Processing

- Histogram Equalization:

  - ✓ We recognize the form of $p_s(s)$ as a uniform probability density function.

  - ✓ Simply stated, we have demonstrated that performing the suggested intensity transformation yields a random variable, *s*, characterized by a uniform PDF.

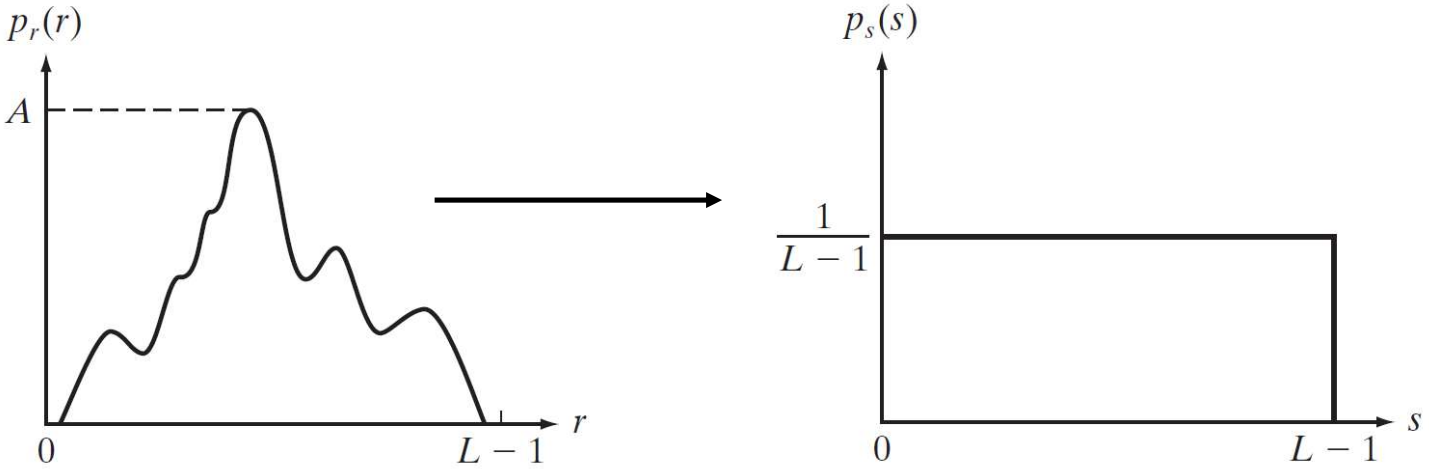
$$p_s(s) = \frac{1}{L-1} \qquad 0 \le s \le L-1$$

  - ✓ It is important to note that $p_s(s)$ is always uniform, independently of the form of $p_r(r)$.

3. Histogram Processing

- Histogram Equalization:

  ✓ We recognize the form of $p_s(s)$ as a uniform probability density function.



a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

3. Histogram Processing

• Example: Suppose that the (continuous) intensity values in an image have the PDF,

$$p_r(r) = \begin{cases} \dfrac{2r}{(L-1)^2} & \text{for } 0 \le r \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

then,

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw = \frac{2}{L-1}\int_0^r w\,dw = \frac{r^2}{L-1}$$

• Suppose next that we form a new image with intensities, *s*, obtained using this transformation.

• Suppose L = 10 and the particular case *r* = 3.

3. Histogram Processing

- Suppose L = 10 and the particular case $r = 3$.
- Then, $s = T(r) = r^2/9 = 1$
- We can verify that the PDF of the intensities in the new image is uniform.

$$p_r(r) = \frac{2r}{(L-1)^2} \qquad p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = \frac{2r}{(L-1)^2}\left|\left[\frac{ds}{dr}\right]^{-1}\right|$$

$$= \frac{2r}{(L-1)^2}\left|\left[\frac{d}{dr}\frac{r^2}{L-1}\right]^{-1}\right|$$

$$= \frac{2r}{(L-1)^2}\left|\frac{(L-1)}{2r}\right| = \frac{1}{L-1}$$

3. Histogram Processing

- Histogram Equalization (for discrete values):

  ✓ For discrete values, we deal with probabilities (histogram values) and summations instead of PDFs and integrals.

  ✓ As mentioned earlier, the probability of occurrence of intensity level in a digital image is approximated by,

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

where *MN* is the total number of pixels in the image, n_k is the number of pixels that have intensity r_k, and L is the number of possible intensity levels in the image.

3. Histogram Processing

- Histogram Equalization (for discrete values):

  ✓ The discrete form of the previously proposed transformation is,

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

$$= \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \qquad k = 0, 1, 2, \ldots, L-1$$

3. Histogram Processing

- Example:

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

TABLE 3.1
Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

3. Histogram Processing

- Example:

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

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33$$

3. Histogram Processing

- Example:

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

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

3. Histogram Processing

- Example:

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

$$s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86, s_7 = 7.00$$

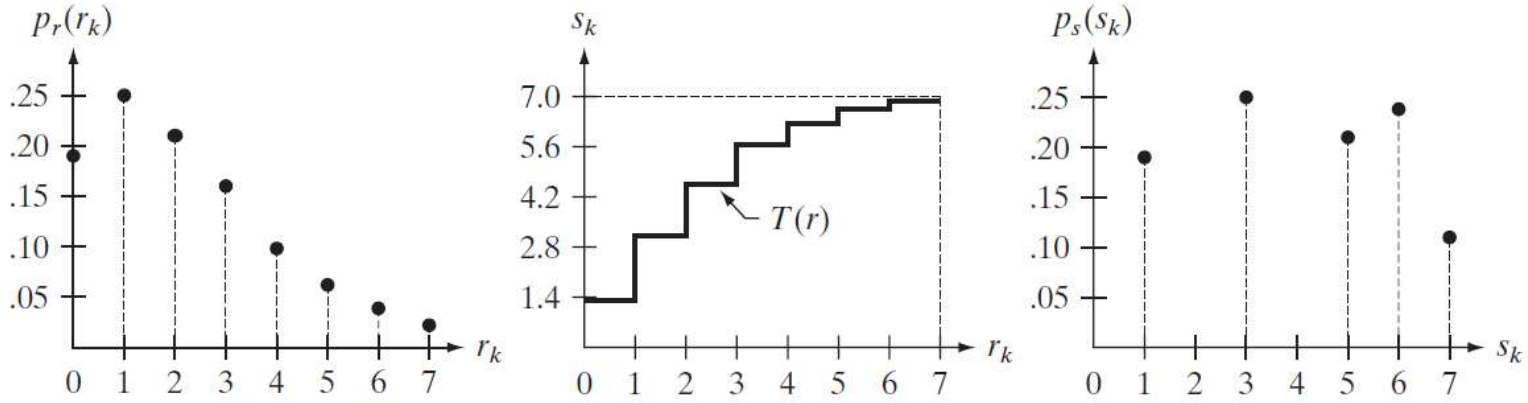
3. Histogram Processing

- Example:

r_k	n_k	$p_r(r_k) = n_k/MN$	
$r_0 = 0$	790	0.19	$s_0 = 1.33 \rightarrow 1$
$r_1 = 1$	1023	0.25	$s_1 = 3.08 \rightarrow 3$
$r_2 = 2$	850	0.21	$s_2 = 4.55 \rightarrow 5$
$r_3 = 3$	656	0.16	$s_3 = 5.67 \rightarrow 6$
$r_4 = 4$	329	0.08	$s_4 = 6.23 \rightarrow 6$
$r_5 = 5$	245	0.06	$s_5 = 6.65 \rightarrow 7$
$r_6 = 6$	122	0.03	$s_6 = 6.86 \rightarrow 7$
$r_7 = 7$	81	0.02	$s_7 = 7.00 \rightarrow 7$

3. Histogram Processing

- Example:



a b c

FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

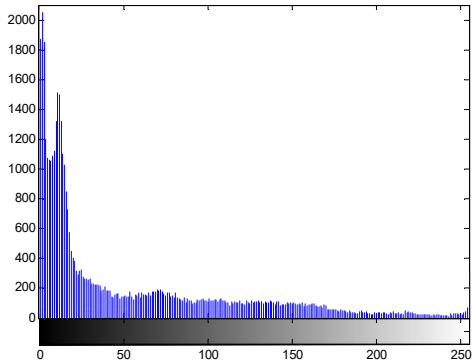
3. Histogram Processing

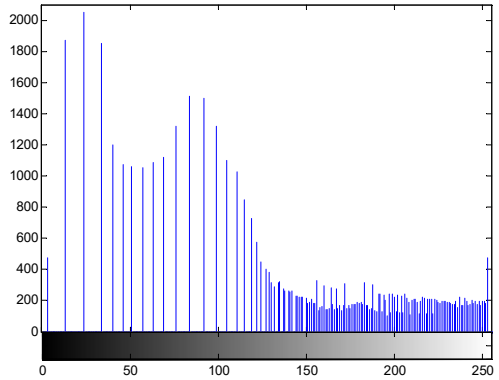
- MATLAB

```
% Histogram equalization
clc
clear all
close all

I = imread('tire.tif');
[h, w] = size(I);
n = length(I(:));

% Probability Mass Function (PMF)
[nk rk] = hist(I(:), [0:255]);
stem(rk, nk,'.'); grid on; title('Original Image Histogram (PMF)');

% Building the transformation function (CDF)
ps = nk/n;
sk = cumsum(ps);
figure; stem(rk,255*sk,'.'); grid on; title('Transformation Function (CDF)');
xlabel('r_k'); ylabel('s_k = T(r_k)')
figure; imshow(I)

% Equalization
for i = 1:h
    for j = 1:w
        Ih(i,j) = round(255*sk(I(i,j)+1));
    end
end
figure; imshow(uint8(Ih))
```

3. Histogram Processing

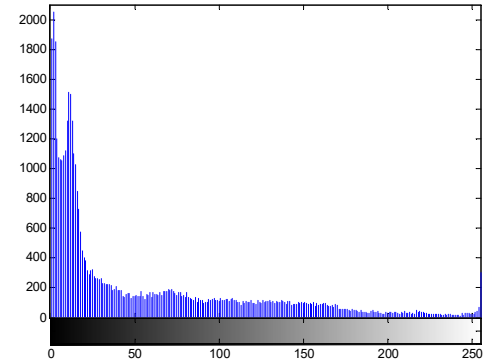
- Example:

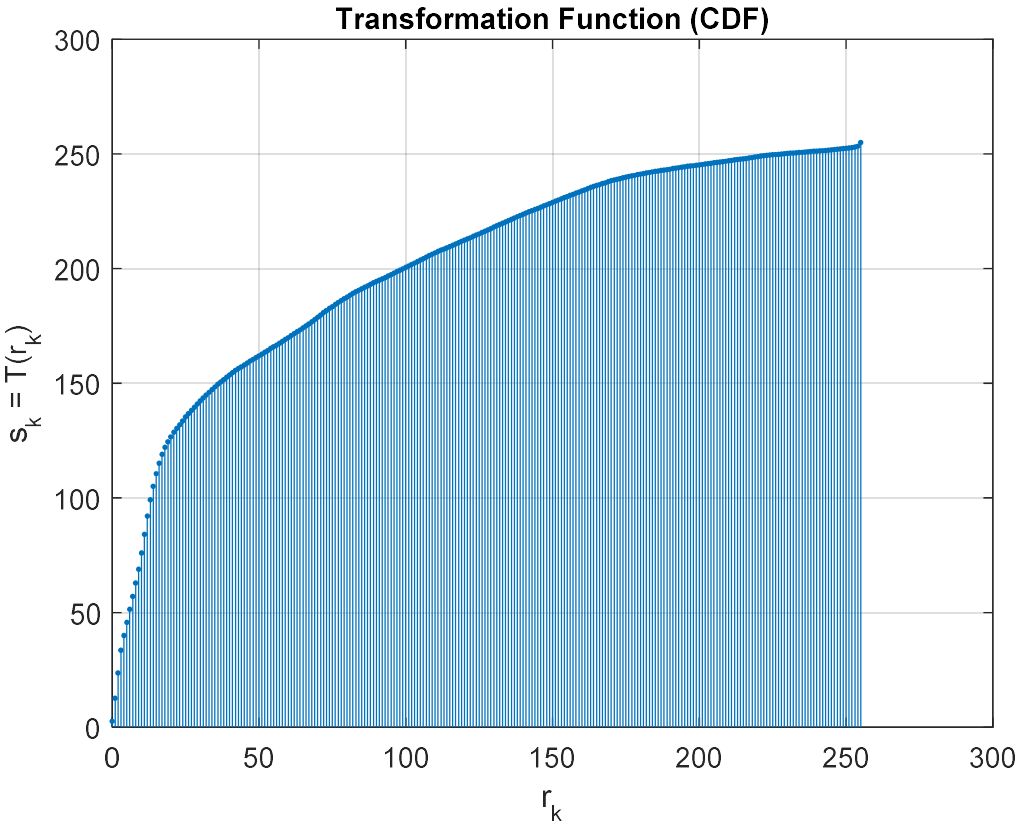




3. Histogram Processing

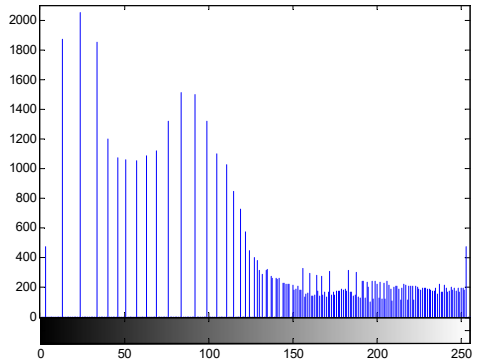
- Example:

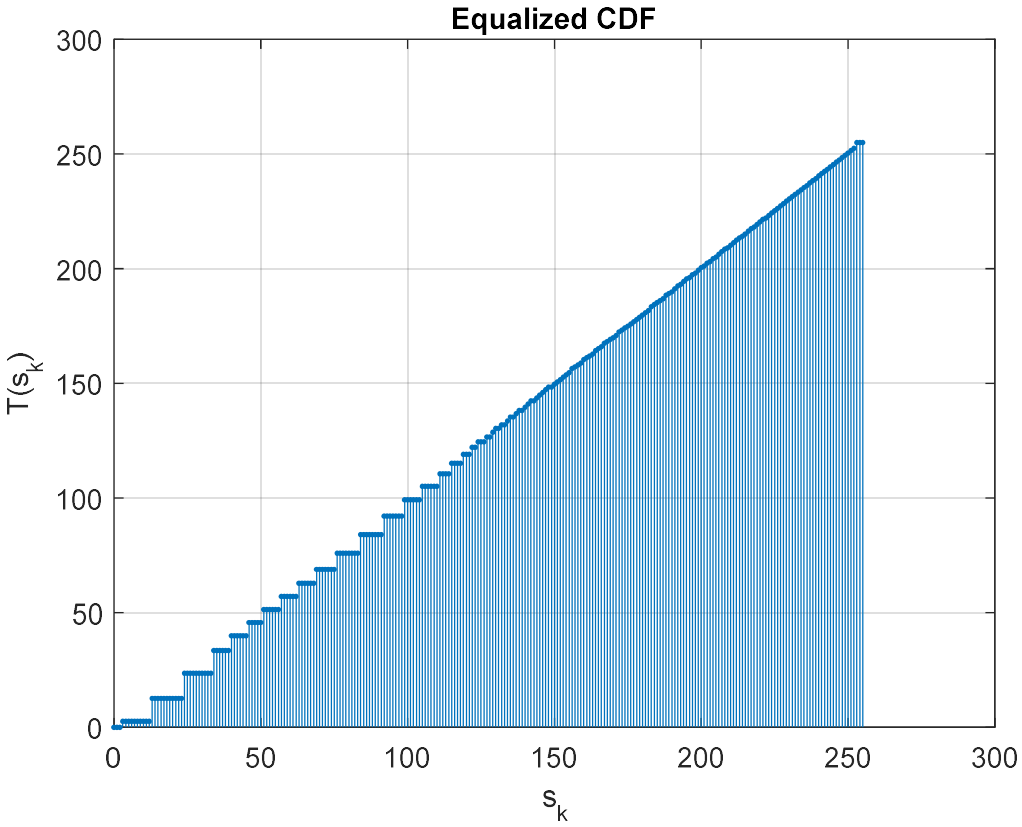




3. Histogram Processing

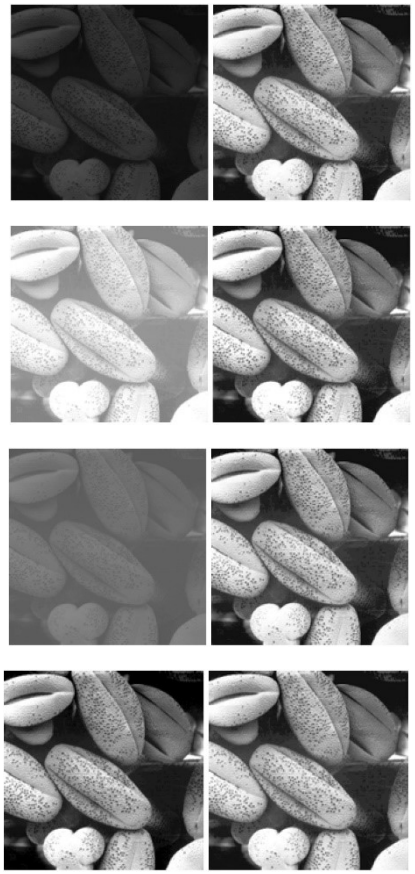
- Example:

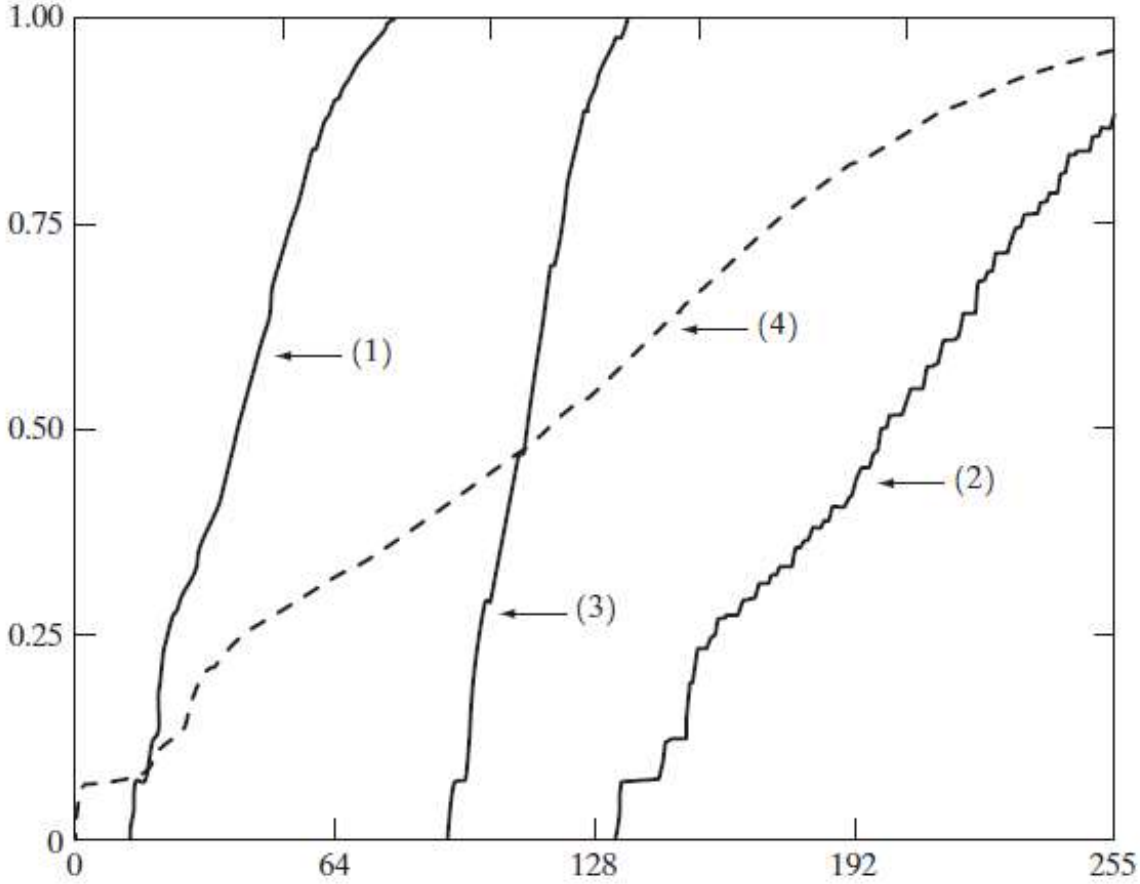




3. Processamento de Histograma

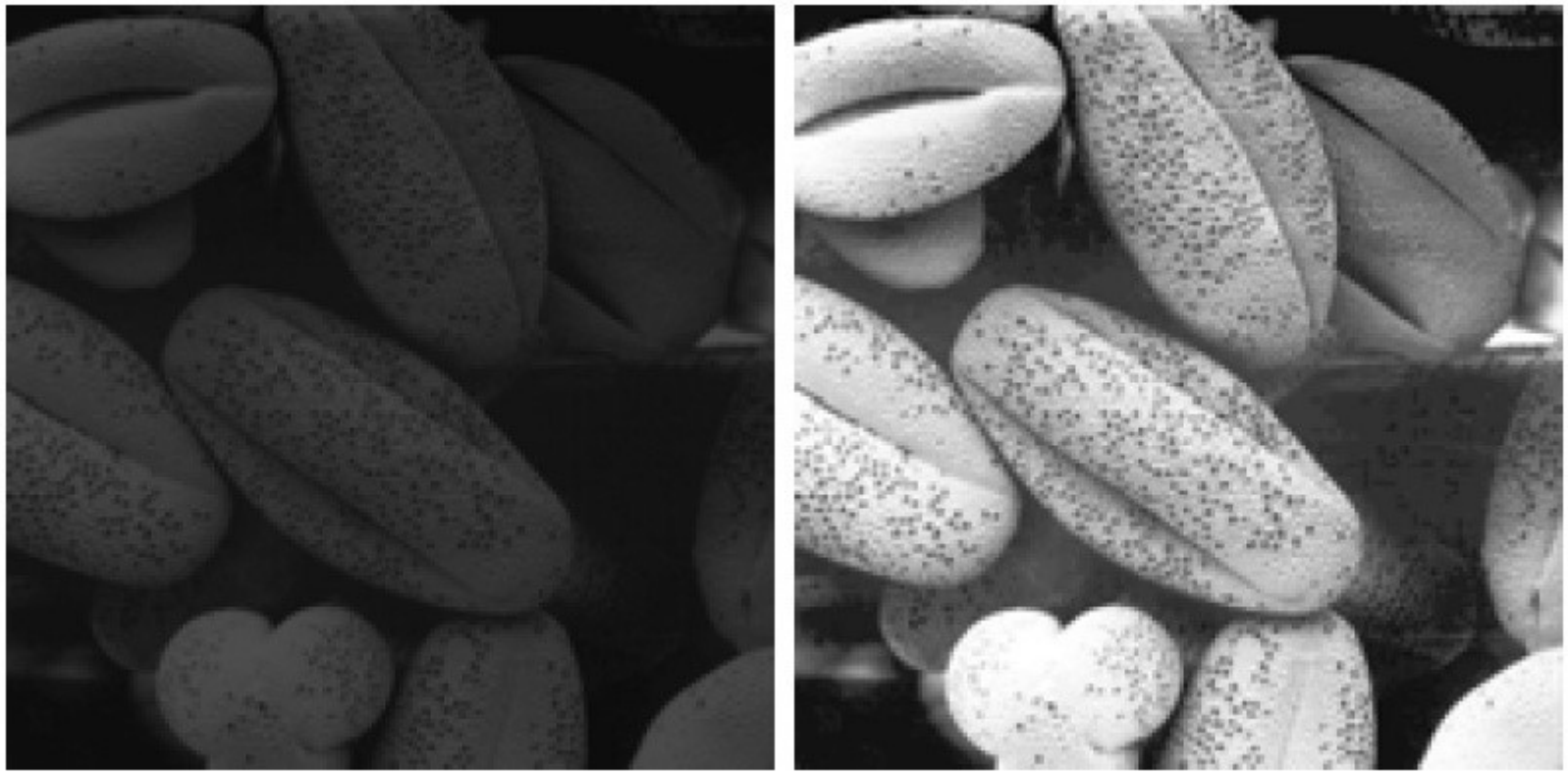
- Other examples (1-4):





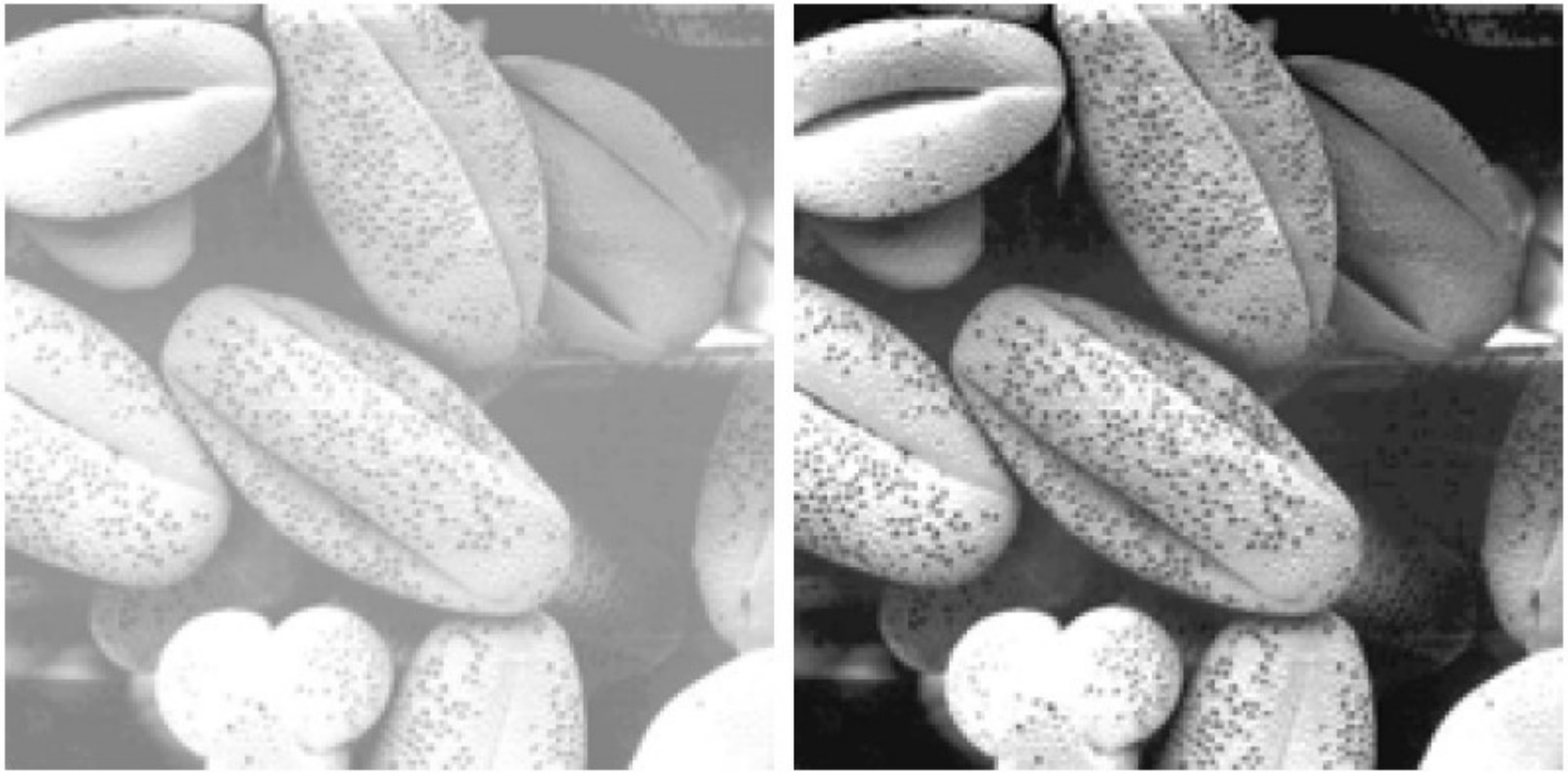
3. Histogram Processing

- Other examples (1):



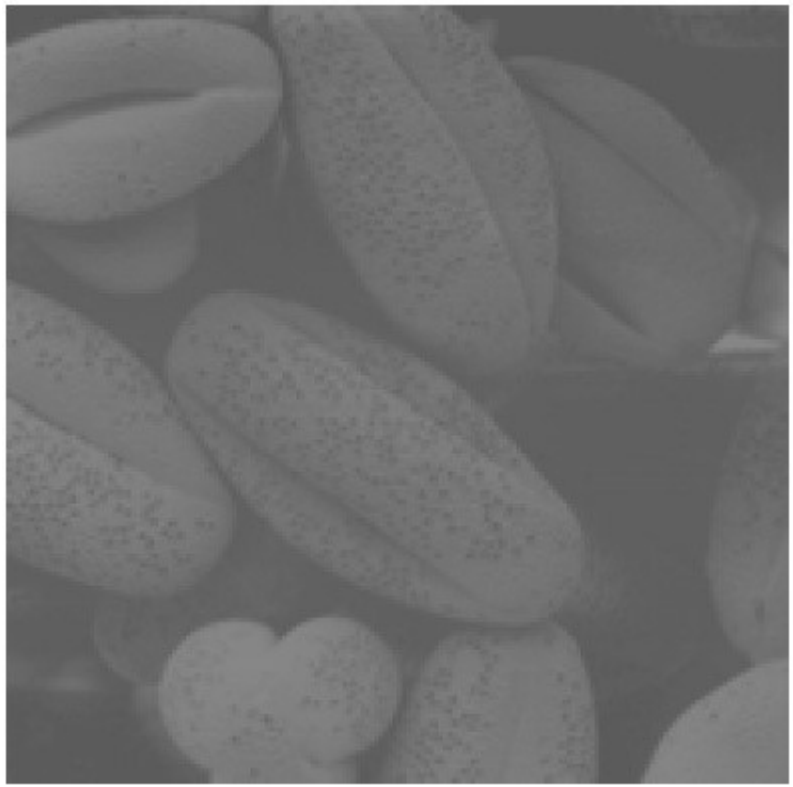
3. Histogram Processing

- Other examples (2):



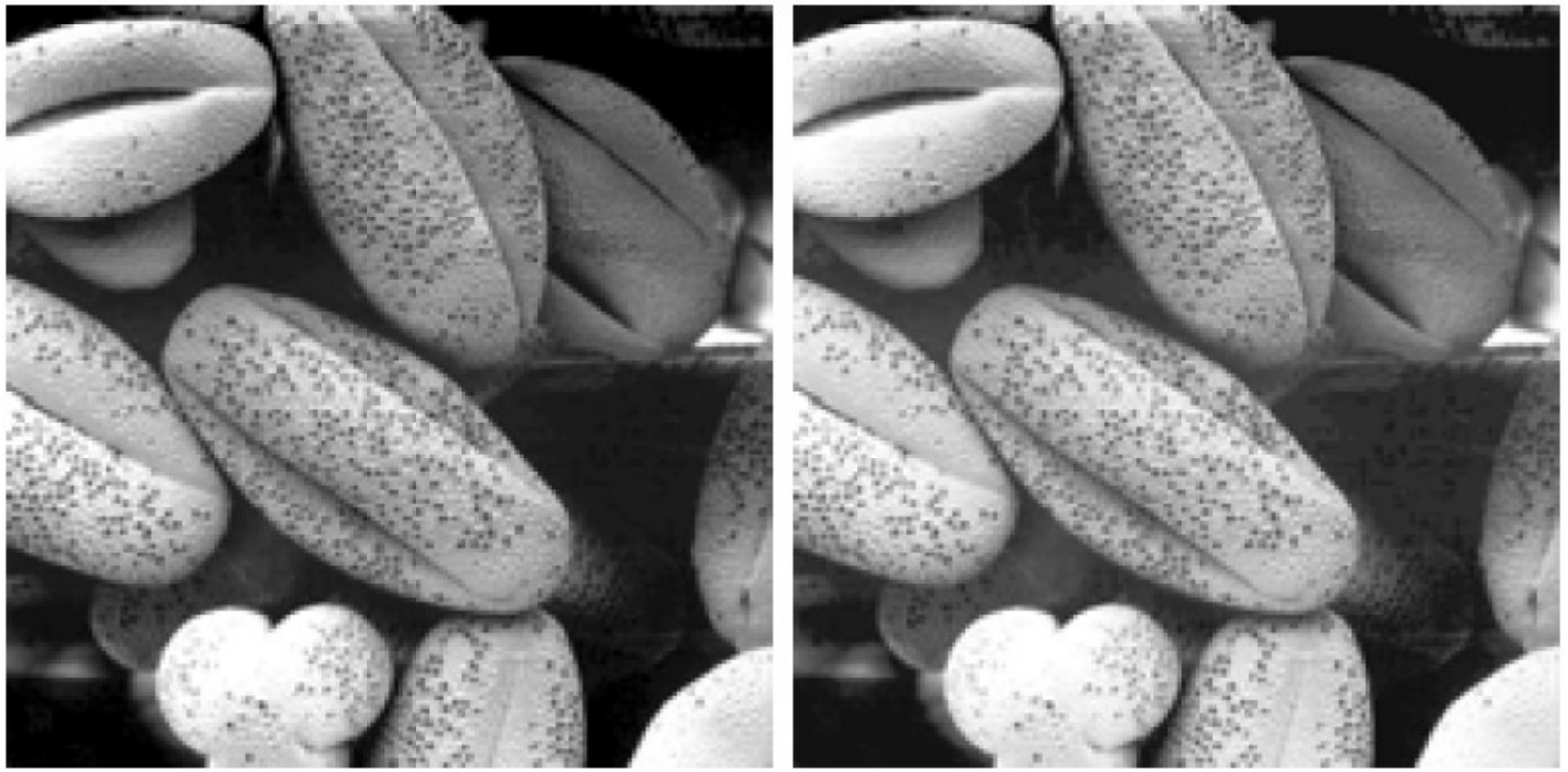
3. Histogram Processing

- Other examples (3):



3. Histogram Processing

- Other examples (4):



3. Histogram Processing

- Histogram Matching (Specification):

  - ✓ It is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have.

  - ✓ The method used to generate a processed image that has a specified histogram is called **histogram matching** or **histogram specification**.

  - ✓ Let *s* be a random variable s with the property,

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

  - ✓ Suppose next that we define a random variable *z* with the property,

$$G(z) = (L-1)\int_0^z p_z(t)\,dt = s$$

3. Histogram Processing

- Histogram Matching (Specification):

  ✓ It then follows from these two equations that,

$$s = G(z) = T(r)$$

  and, therefore, that z must satisfy the condition,

$$z = G^{-1}[T(r)] = G^{-1}(s)$$

  ✓ $T(r)$ can be obtained, since $p_r(r)$ has been estimated from the input image.

  ✓ Similarly, the transformation function $G(z)$ can be obtained because $p_z(z)$ is given.

3. Histogram Processing

- Histogram Matching (Specification):

  ✓ Discrete formulation

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$
$$k = 0, 1, 2, \ldots, L-1$$

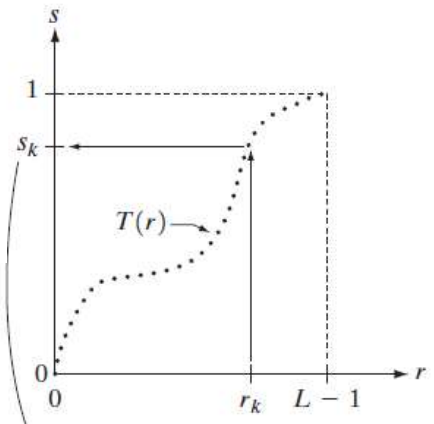
$$G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i)$$

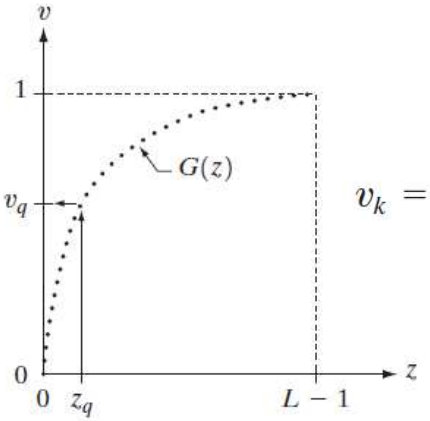
$$G(z_q) = s_k \longrightarrow z_q = G^{-1}(s_k)$$

3. Histogram Processing

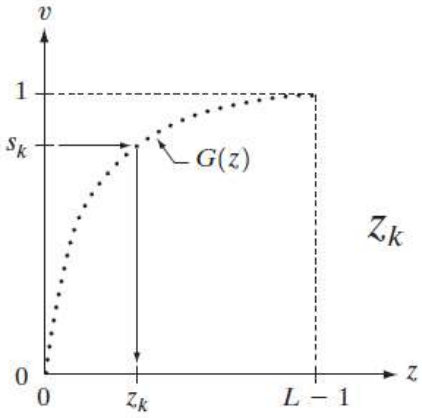
- Histogram Matching (Specification):

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





$$v_k = G(z_k) = \sum_{i=0}^{k} p_z(z_i)$$



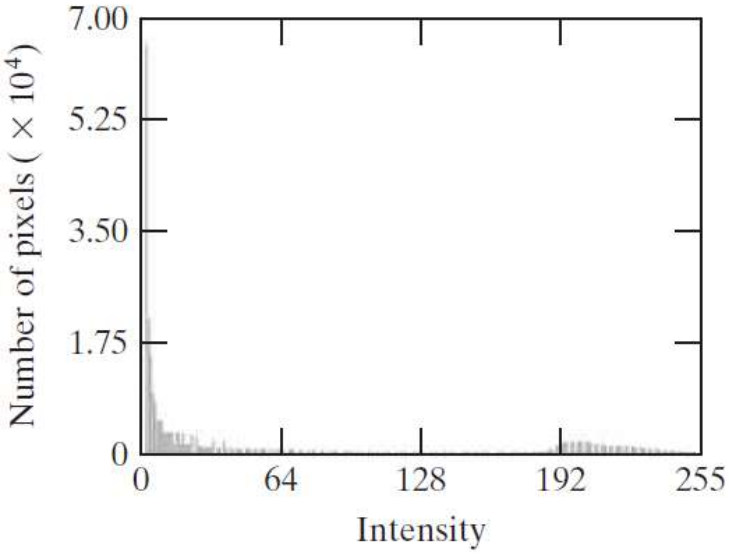
$$z_k = G^{-1}(s_k)$$

3. Histogram Processing

- Histogram Matching (Specification):

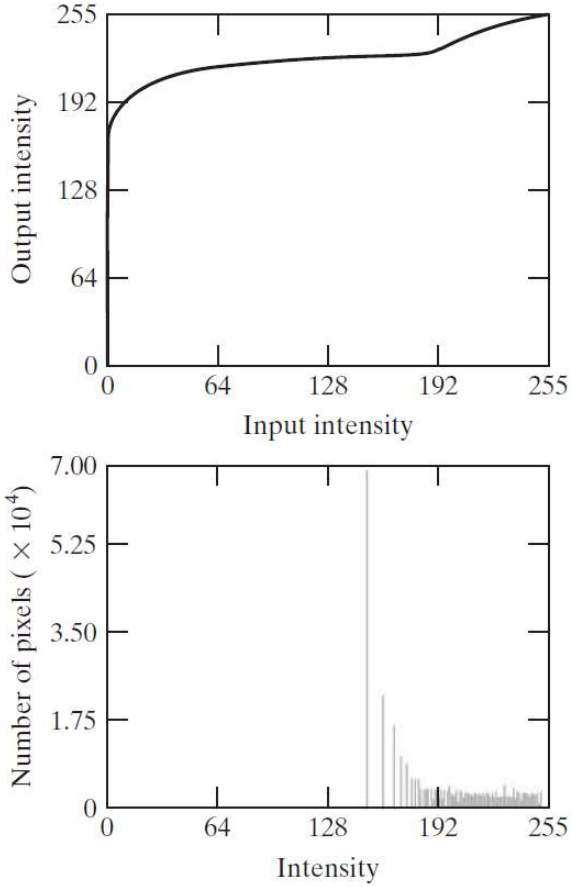
a b

FIGURE 3.23
(a) Image of the Mars moon Phobos taken by NASA's *Mars Global Surveyor*. (b) Histogram. (Original image courtesy of NASA.)



3. Histogram Processing

- Histogram Matching (Specification):

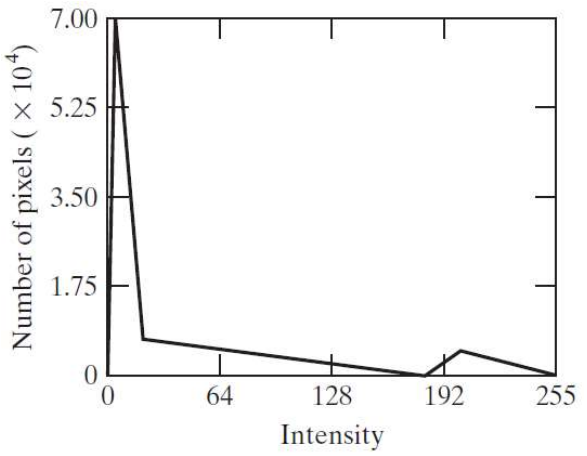


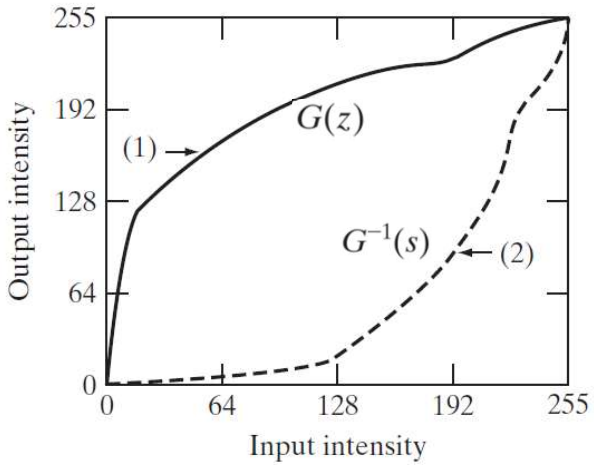
a b
c

FIGURE 3.24
(a) Transformation function for histogram equalization. (b) Histogram-equalized image (note the washed-out appearance). (c) Histogram of (b).

3. Histogram Processing

- Histogram Matching (Specification):





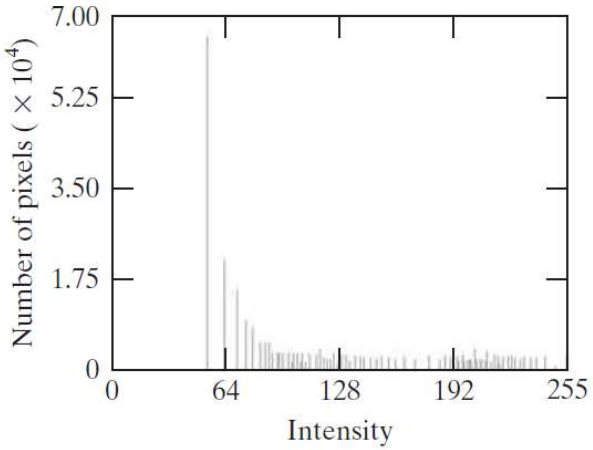
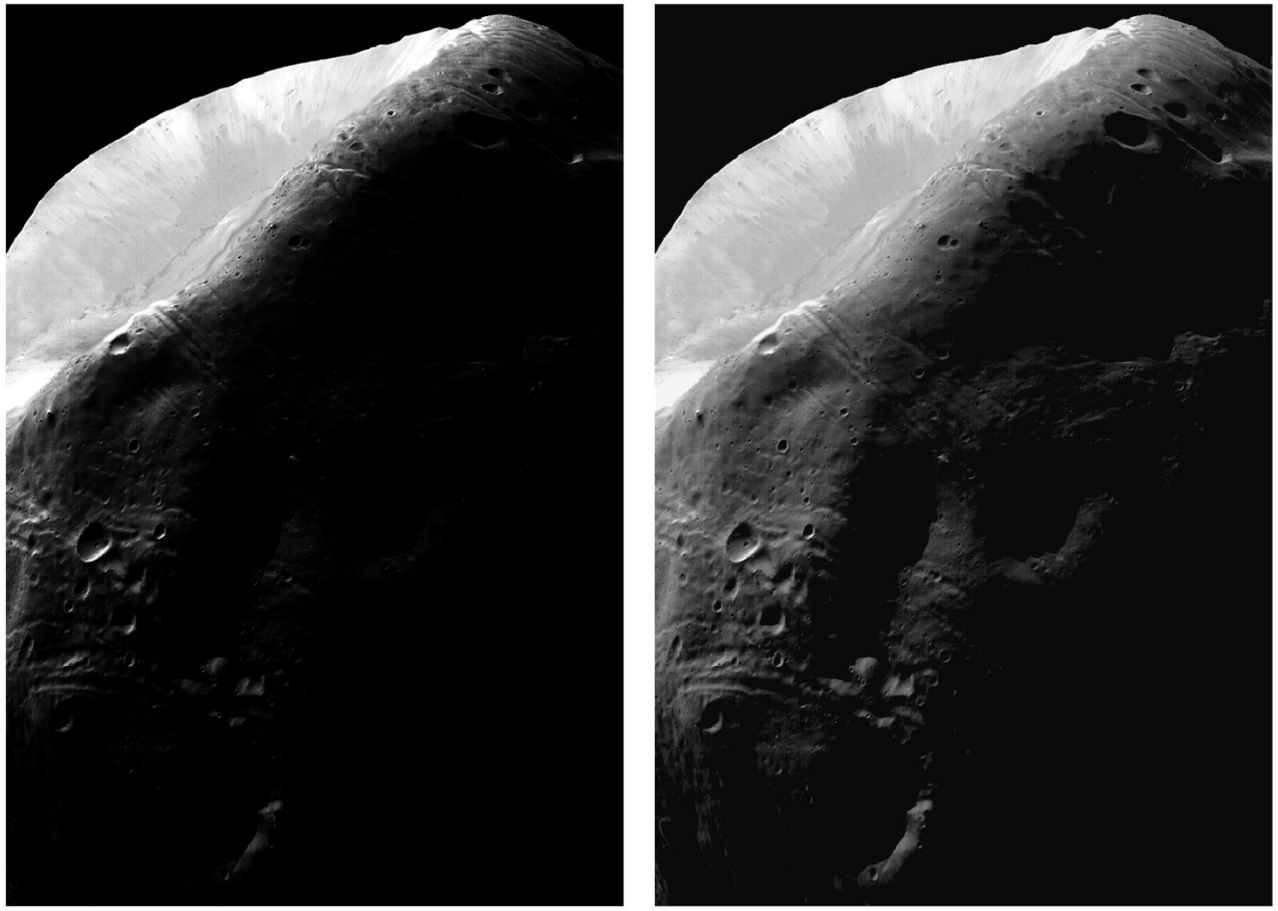


FIGURE 3.25
(a) Specified histogram.
(b) Transformations.
(c) Enhanced image using mappings from curve (2).
(d) Histogram of (c).

3. Histogram Processing

- MATLAB: s076histspec.m

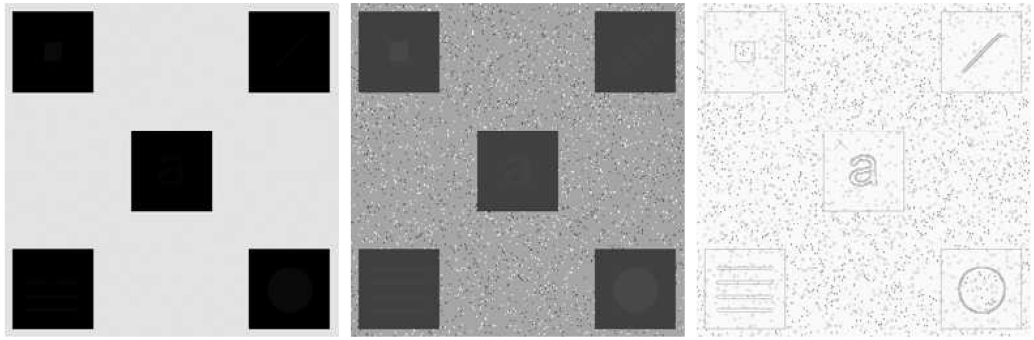


3. Histogram Processing

- Local Histogram Processing:

  ✓ The histogram processing techniques previously described are easily adapted to local enhancement.

  ✓ The procedure is to define a neighborhood and move its center from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed.



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.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

  ✓ The *nth* moment of *r* about its mean *m* is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$$

  where

$$m = \sum_{i=0}^{L-1} r_i p(r_i)$$

  ✓ The second moment (variance) is particularly important

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i)$$

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

  ✓ Example:

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

L = 4 ➔ r_k = [0 1 2 3]

$$p(r_k) = [0.24 \quad 0.28 \quad 0.28 \quad 0.20]$$

$$m = 0 \cdot 0.24 + 1 \cdot 0.28 + 2 \cdot 0.28 + 3 \cdot 0.20 = 1.44$$

$$\mu_2 = (0-1.44)^2 \cdot 0.24 + (1-1.44)^2 \cdot 0.28 +$$

$$(2-1.44)^2 \cdot 0.28 + (3-1.44)^2 \cdot 0.20 = 1.1264$$

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

  ✓ It is common practice to estimate the mean and the variance sample values, without computing the histogram:

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

$$\sigma^2 = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f(x, y) - m \right]^2$$

  ✓ Global mean and variance may be computed over an entire image

  ✓ However, a more powerful use of these parameters is in local enhancement.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

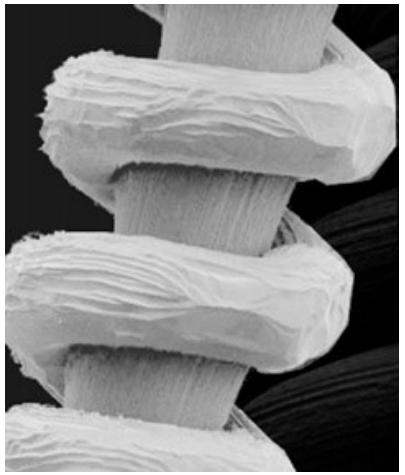
  ➢ In this particular case, the problem is to enhance dark areas while leaving the light area as unchanged as possible because it does not require enhancement.

  ➢ A measure of whether an area is relatively light or dark at a point (x, y) is to compare the average local intensity m_{Sxy} to the average image intensity M_G.

  ➢ Candidate for processing:

$$m_{S_{xy}} \leq k_0 M_G$$

  k_0 is positive, less than 1.0.



3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

  ➢ Enhancing areas that have low contrast. Candidates:

$$\sigma_{S_{xy}} \leq k_2 D_G$$

  where $\sigma_{S_{xy}}$ is the local standard deviation, D_G is the global standard deviation and *k2* is a positive constant, less than o 1.

  ➢ Finally, we need to restrict the lowest values of contrast we are willing to accept,

$$k_1 D_G \leq \sigma_{S_{xy}}$$

  with *k1* < *k2*.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

  ➢ A pixel that meets all the conditions for local enhancement is processed simply by multiplying it by a specified constant, E, to increase the value of its intensity level relative to the rest of the image:

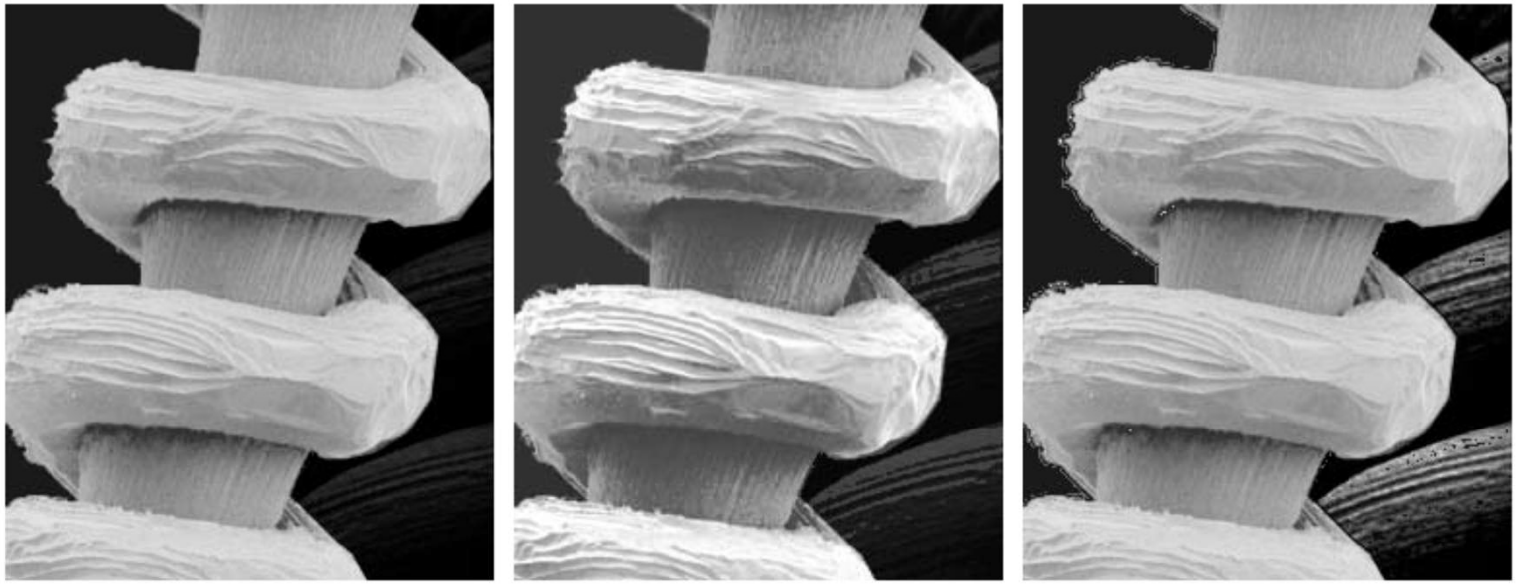
$$g(x,y) = \begin{cases} E \cdot f(x,y) & \text{if } m_{S_{xy}} \leq k_0 m_G \text{ AND } k_1\sigma_G \leq \sigma_{S_{xy}} \leq k_2\sigma_G \\ f(x,y) & \text{otherwise} \end{cases}$$

  ➢ Choosing these parameters generally requires a bit of experimentation.

  ➢ In this example, $E = 4.0$, $k_0 = 0.4$, $k_1 = 0.02$ and $k_2 = 0.4$. A 3x3 window was used.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:



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.)

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:
  - The name *filter* is borrowed from frequency domain processing.
  - "Filtering" refers to accepting (passing) or rejecting certain frequency components.
  - For example, a filter that passes low frequencies is called a **lowpass filter**.
  - If the operation performed on the image pixels is **linear**, then the filter is called a **linear spatial filter**. Otherwise, the filter is **nonlinear**.

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:

  - For a mask (filter) of size *m* x *n,* we assume that *m* = *2a* + *1* and *n* = *2b* + *1,* where *a* e *b* are positive integers.

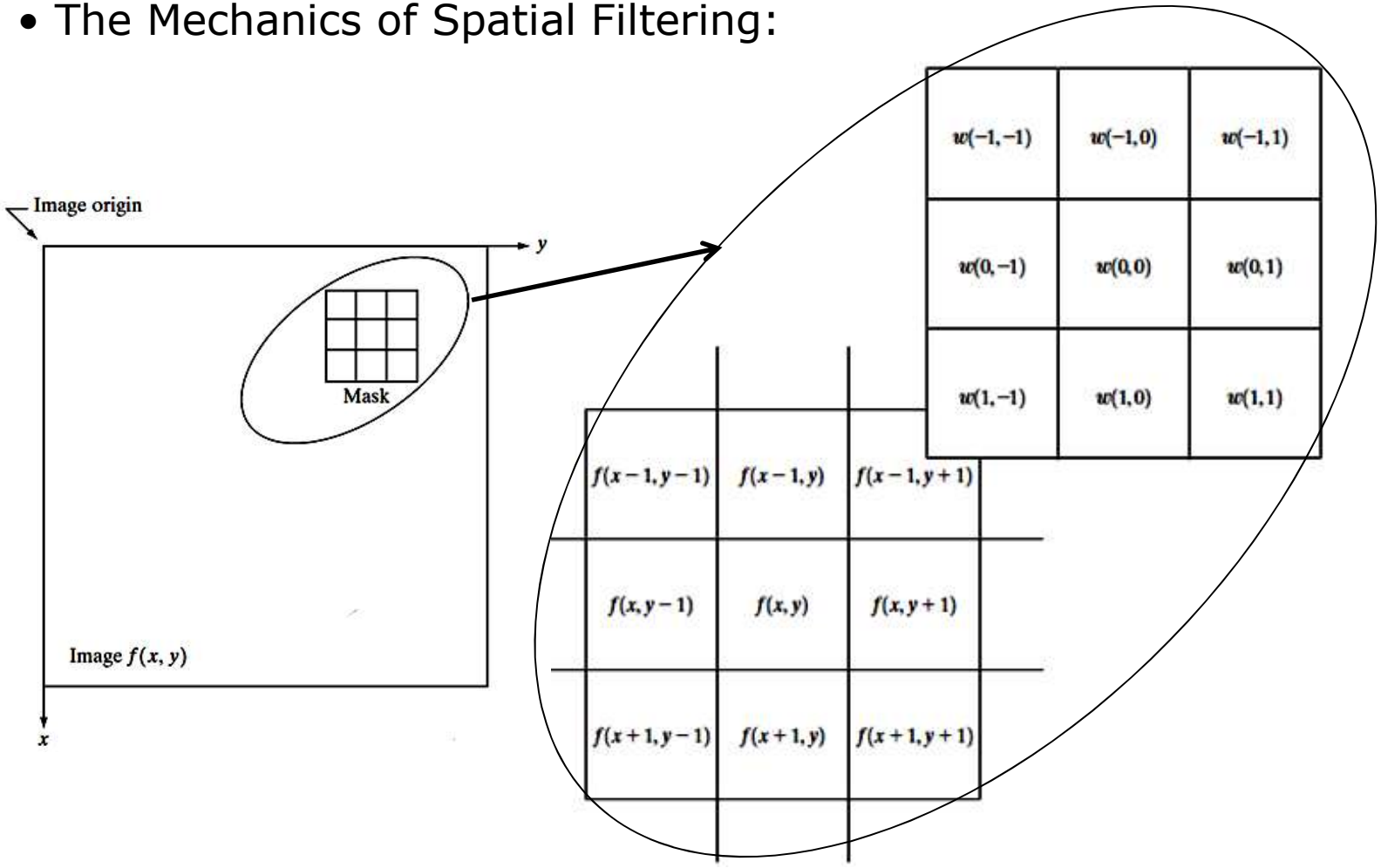
  - At any point (x, y) in the image, the response, *g(x,y)*, of the filter is the sum of products of the filter coefficients and the image pixels encompassed by the filter:

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

where x and y are varied so that each pixel in *w* visits every pixel in *f*.

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:



4. Fundamentals of Spatial Filtering

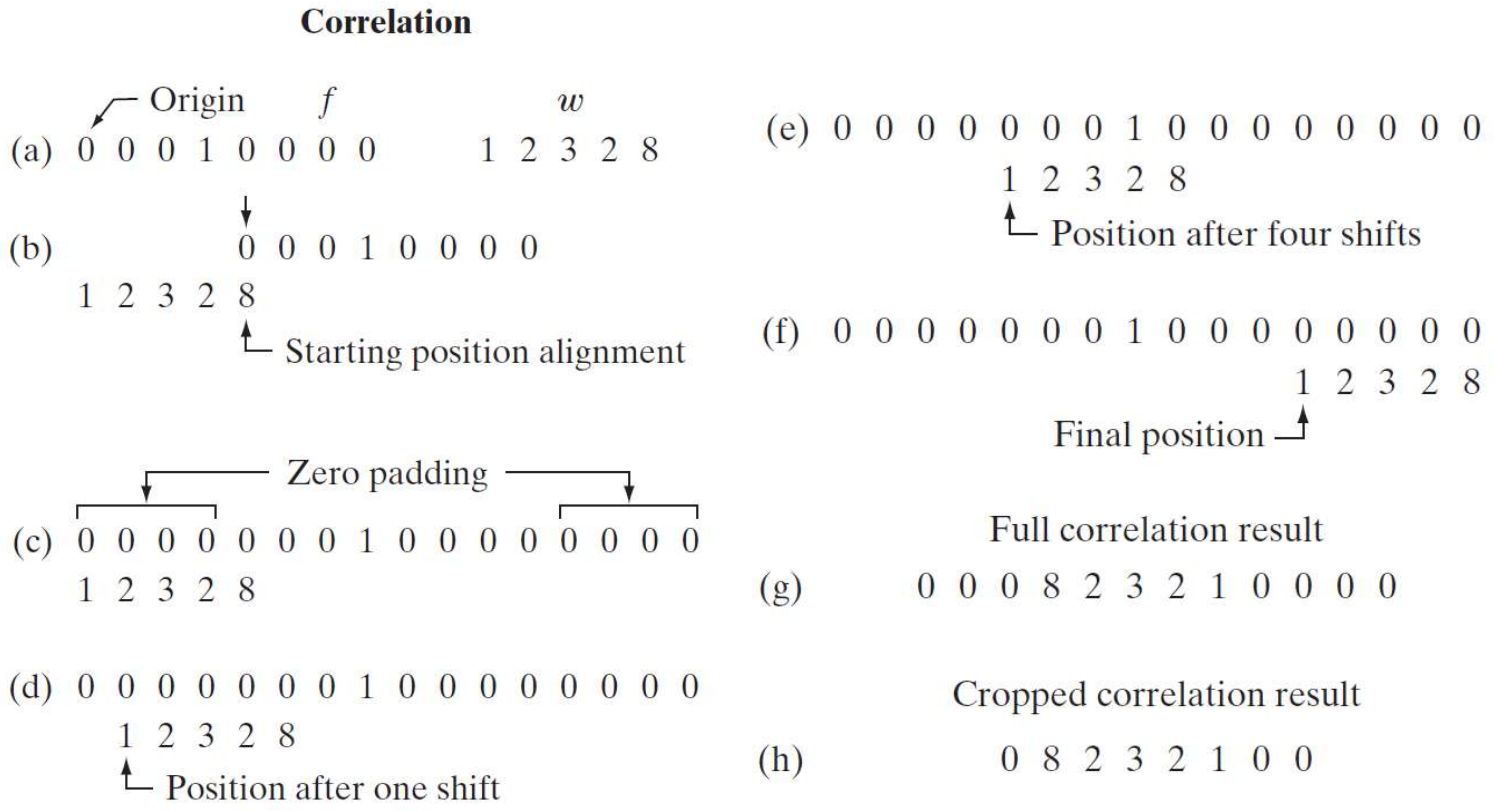
- Spatial Correlation and Convolution:

  - **Correlation** is the process of moving a filter mask over the image and computing the sum of products at each location, exactly as explained in the previous slides.

  - The mechanics of **convolution** are the same, except that the filter is first rotated by 180°.

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:



4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

Convolution

Origin f w rotated 180°

0 0 0 1 0 0 0 0 8 2 3 2 1 (i)

0 0 0 1 0 0 0 0 (j)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (k)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (l)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (m)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (n)
8 2 3 2 1

Full convolution result

0 0 0 1 2 3 2 8 0 0 0 0 (o)

Cropped convolution result

0 1 2 3 2 8 0 0 (p)

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

  - We call a function that contains a single 1 with the rest being 0s a **discrete unit impulse**.

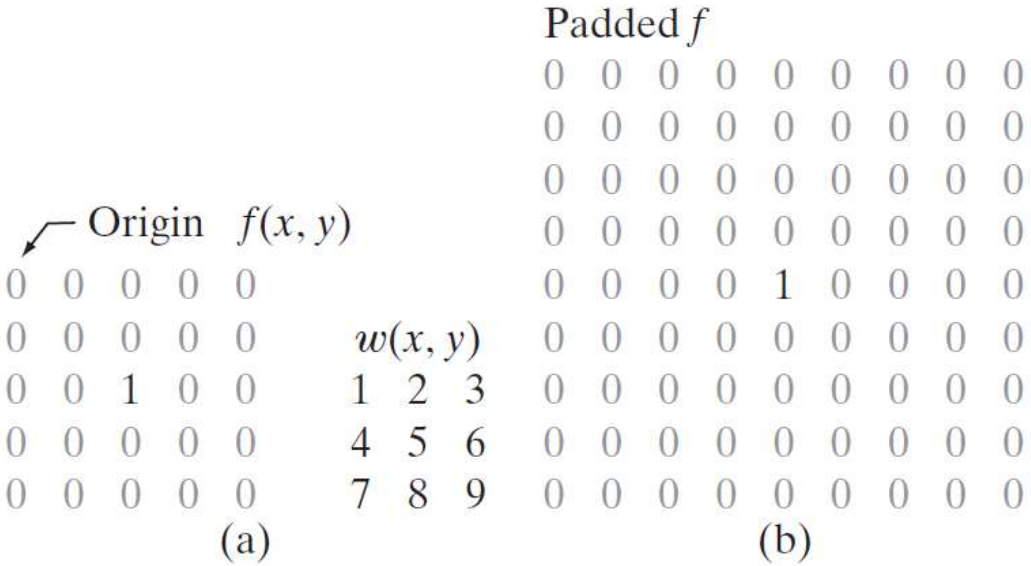
  - So we conclude that **correlation** of a function with a discrete unit impulse yields a 180° **rotated** version of the function at the location of the impulse.

  - **Convolution** is a cornerstone of **linear system theory** and will be addressed in more detail later.

  - Convolving a function with a unit impulse yields a **copy** of the function at the location of the impulse.

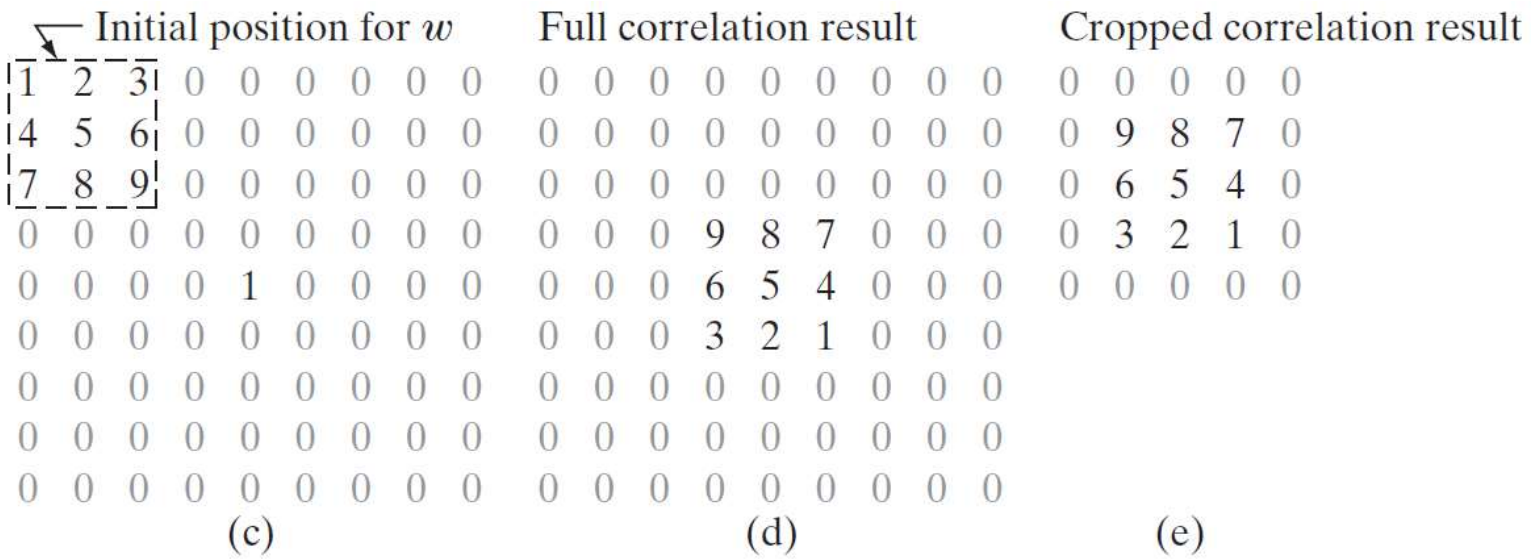
4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:



4. Fundamentals of Spatial Filtering

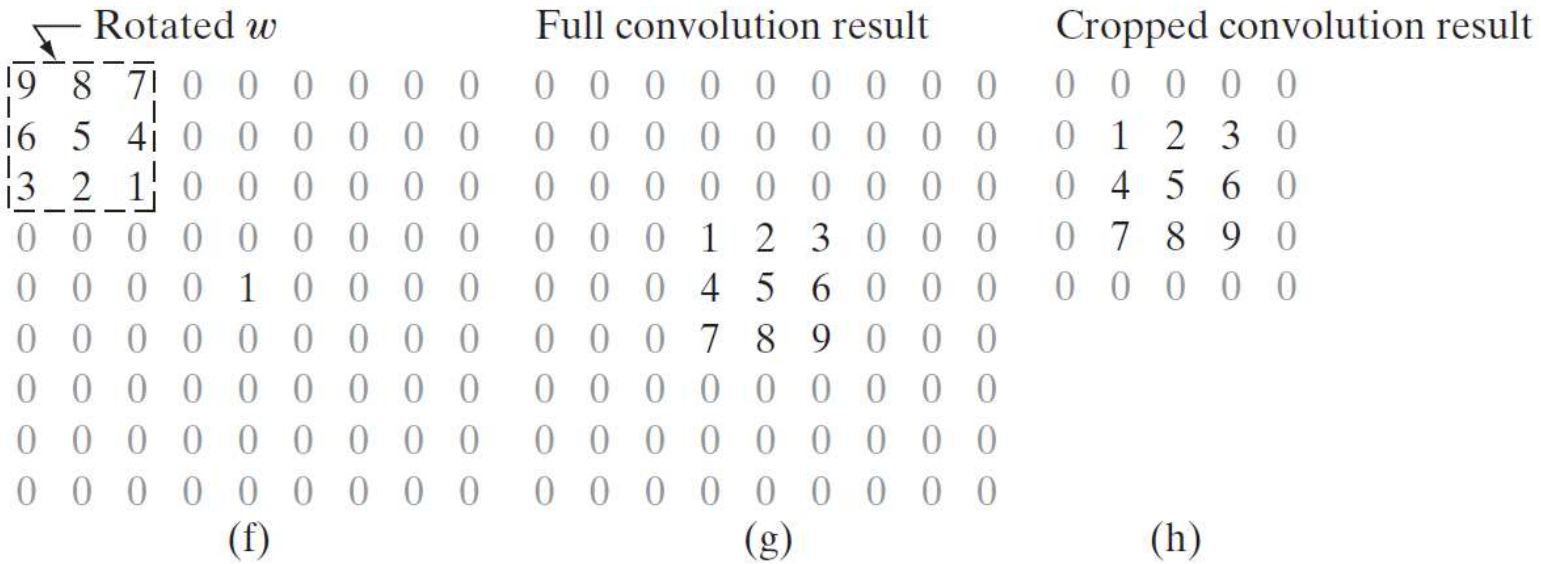
- Spatial Correlation and Convolution:



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

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:



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

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

  - If, instead of containing a single 1, image f had contained a region equal to *w*, the value of the correlation function (after normalization) would have been maximum when was centered on that region.

  - Thus, correlation can be used also to find matches between images.

  - Using correlation or convolution to perform spatial filtering is a matter of preference.

  - What is important is that the filter mask used in a given filtering task corresponds to the intended operation.

4. Fundamentals of Spatial Filtering

- Vector Representation of Linear Filtering

  - Response, R, of a mask either for correlation or convolution,

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn}$$

$$= \sum_{k=1}^{mn} w_k z_k$$

$$= \mathbf{w}^T \mathbf{z}$$

where the *w*s are the coefficients of an m x n filter and the *z*s are the corresponding image intensities encompassed by the filter.

4. Fundamentals of Spatial Filtering

- Vector Representation of Linear Filtering
  - Example:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 3.31 Another representation of a general 3×3 filter mask.

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$

$$= \sum_{k=1}^{9} w_k z_k$$

$$= \mathbf{w}^T \mathbf{z}$$

4. Fundamentals of Spatial Filtering

- Generating Spatial Filter Masks

  - The filter coefficients are selected based on what the filter is supposed to do.

  - Keep in mind that all we can do with linear filtering is to implement a sum of products.

  - For example, suppose that we want to replace the pixels in an image by the average intensity of a neighborhood centered on those pixels.

$$w_i = 1/9 \qquad R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

4. Fundamentals of Spatial Filtering

- Generating Spatial Filter Masks

  - In some applications, we have a continuous function of two variables, and the objective is o obtain a spatial filter mask based on that function.

  - For example, a Gaussian function of two variables

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

where σ is the standard deviation and, as usual, we assume that coordinates x and y are integers.

  - Then, to generate the filter mask from this function, we sample it about its center.

5. Smoothing Spatial Filters

• Smoothing spatial filters are used for blurring and for noise reduction.

• The output of a **smoothing**, **linear spatial filter** is simply the **average** of the pixels contained in the neighborhood of the filter mask.

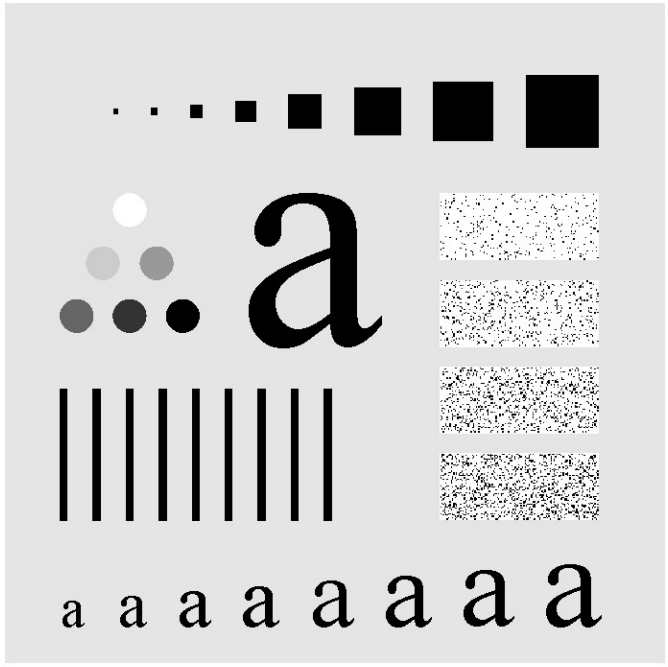
• They also are referred to **lowpass filters**.

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$\frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

5. Smoothing Spatial Filters

- Example:



Original

3 x 3

5. Smoothing Spatial Filters

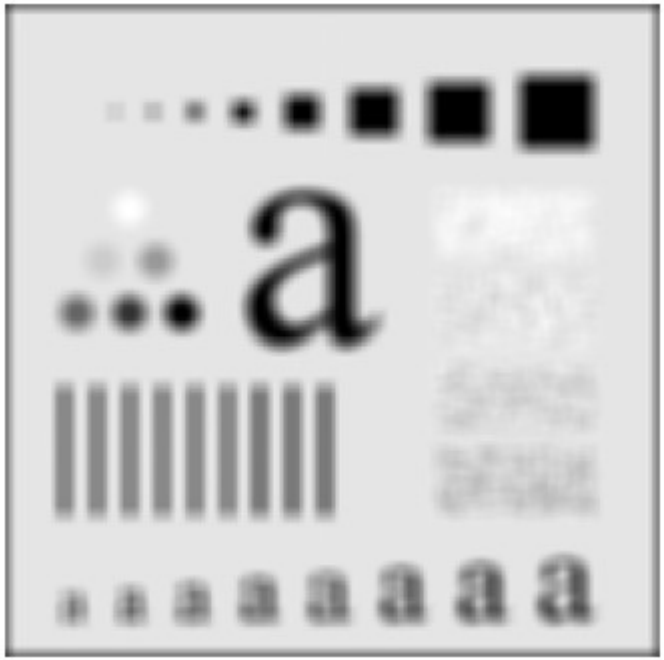
- Example:

5 x 5

9 x 9

5. Smoothing Spatial Filters

- Example:

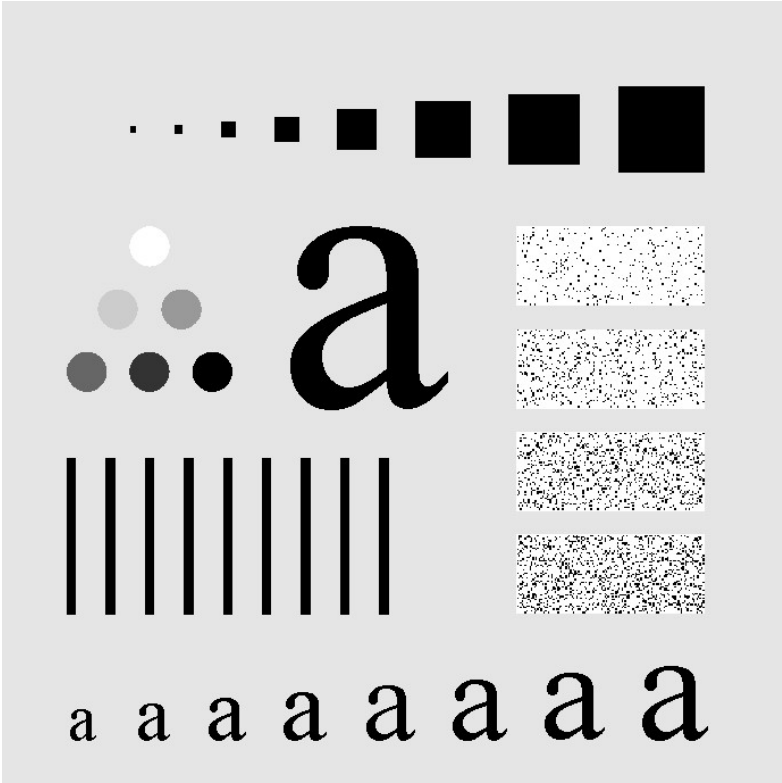


15 x 15

35 x 35

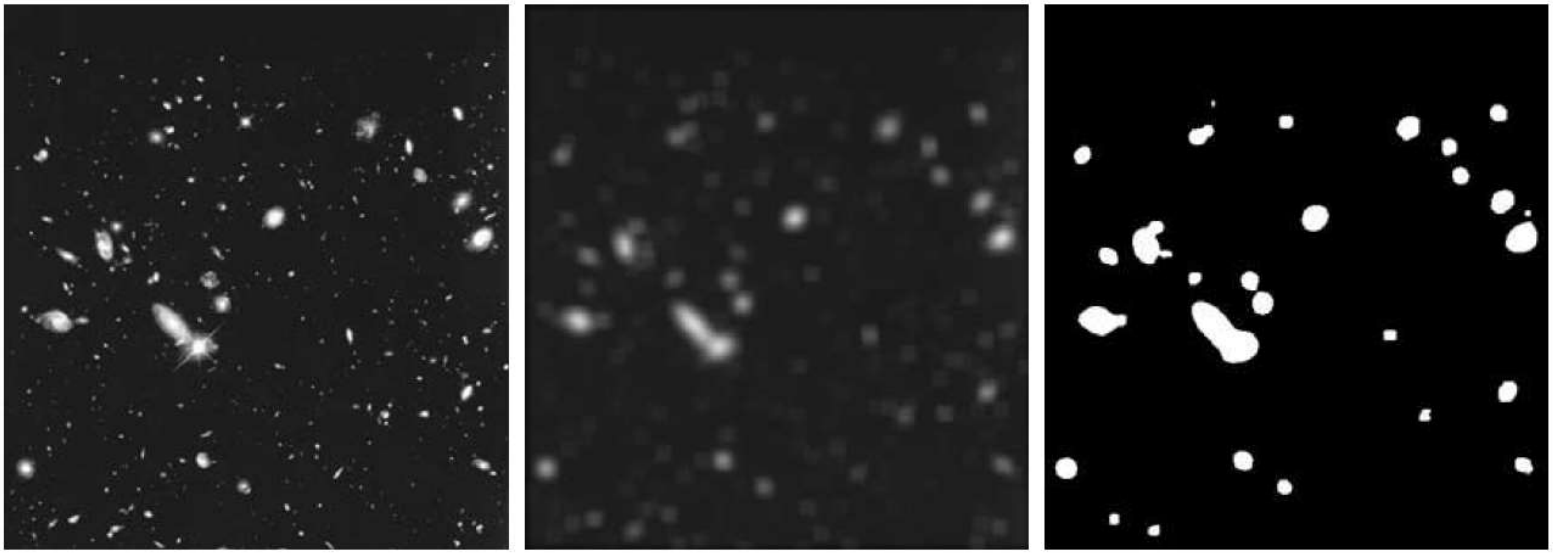
5. Smoothing Spatial Filters

- MATLAB: s104characters.m



5. Smoothing Spatial Filters

- An important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest.



a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

5. Smoothing Spatial Filters

- MATLAB: s106hubble.m

5. Smoothing Spatial Filters

- Order-Statistic (Nonlinear) Filters

  - **Median filter**: effective in the presence of **impulse noise**, also called **salt-and-pepper** noise because of its appearance as white and black dots superimposed on an image.

  - Causes less blurring than linear smoothing filters of similar size.

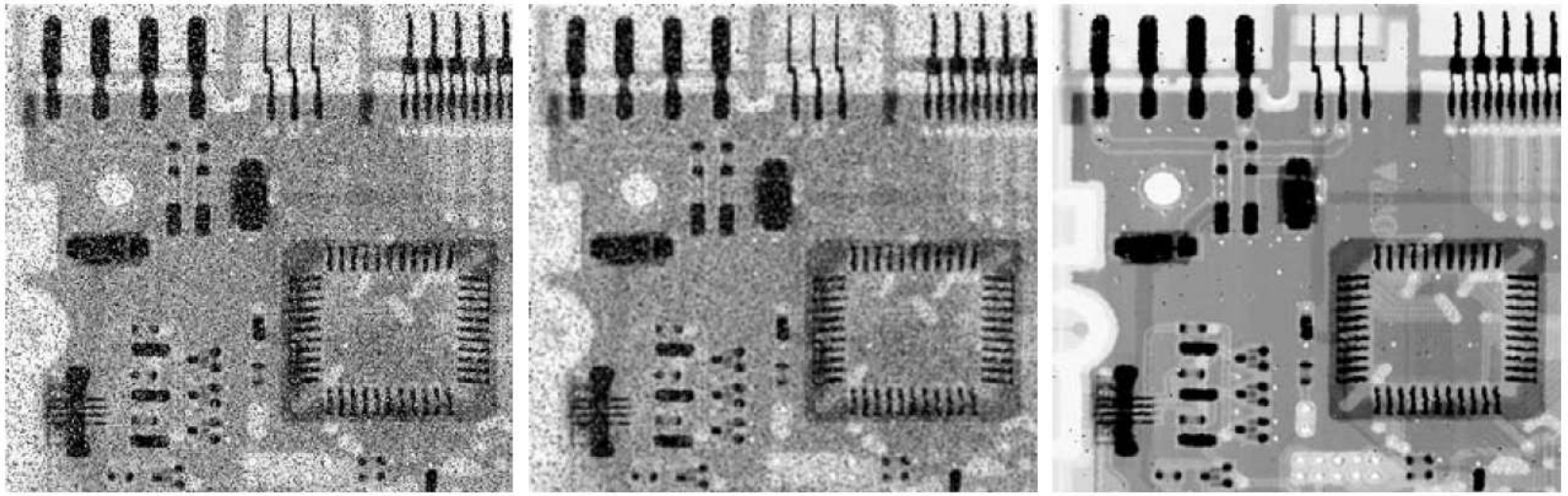
    - ✓ Example:

10	20	20
20	15	20
20	25	100

    - ✓ Sorted pixels: 10, 15, 20, 20, 20, 20, 20, 25, 100

5. Smoothing Spatial Filters

- Example:

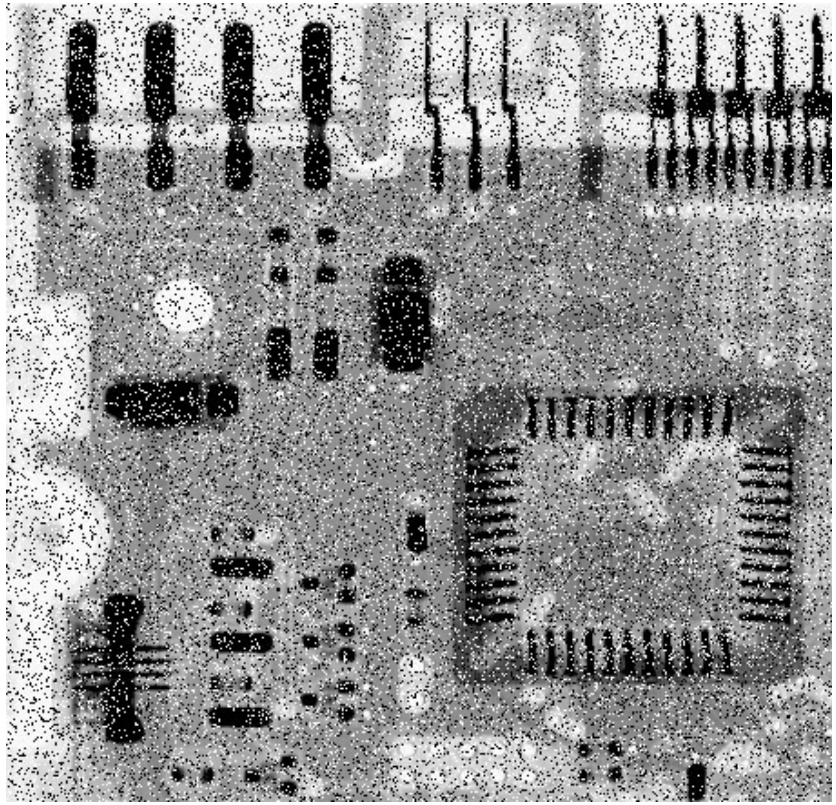


a b c

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.)

5. Smoothing Spatial Filters

- MATLAB: s109circuit.m



5. Smoothing Spatial Filters

• The median represents the 50th percentile of a ranked set of numbers.

• Using the 100th percentile results in the so-called **max filter**, useful for finding the brightest points in an image.

$$R = \max\{z_k | k = 1, 2, \ldots, 9\}$$

• The 0th percentile filter is the **min filter**, used for the opposite purpose.

$$R = \min\{z_k | k = 1, 2, \ldots, 9\}$$

6. Sharpening Spatial Filters

• Because **smoothing** is analogous to integration, it is logical to conclude that **sharpening** can be accomplished by spatial differentiation.

• Image differentiation enhances edges and other discontinuities and deemphasizes areas with slowly varying intensities.

• Thus, the principal objective of sharpening is to highlight transitions in intensity.

6. Sharpening Spatial Filters

• The derivatives of a digital function are defined in terms of 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; and
3. must be nonzero along ramps.

• A basic definition of the first-order derivative of a one-dimensional function *f* is the difference

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

6. Sharpening Spatial Filters

- Similarly, any definition of a **second derivativ**e must:

  1. must be zero in constant areas;
  2. must be nonzero at the onset and end of an intensity step or ramp; and
  3. must be zero along ramps of constant slope.

- We define the second-order derivative of as the difference

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

6. Sharpening Spatial Filters

- Second-order derivative

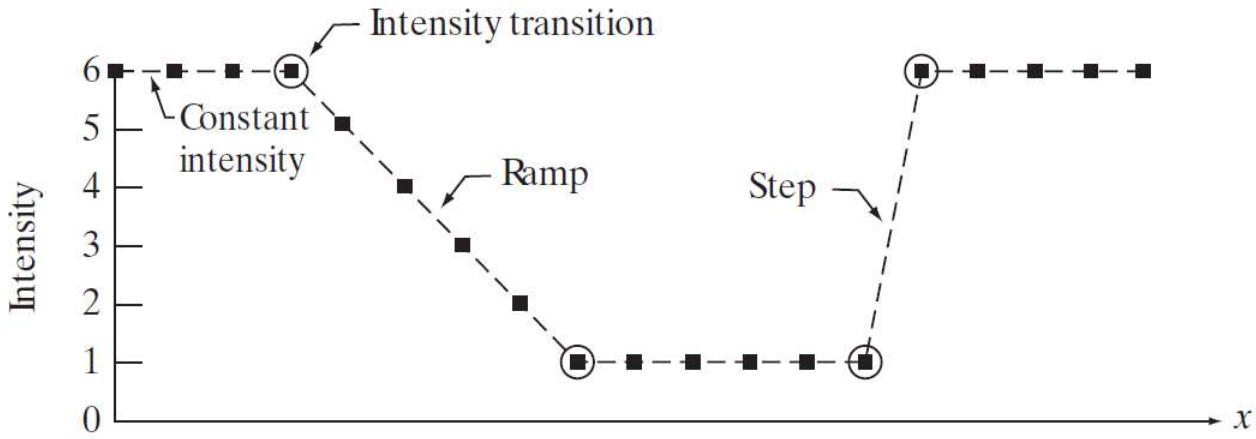
$$\boxed{\frac{\partial f}{\partial x} = f(x+1) - f(x)}$$

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

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

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

6. Sharpening Spatial Filters

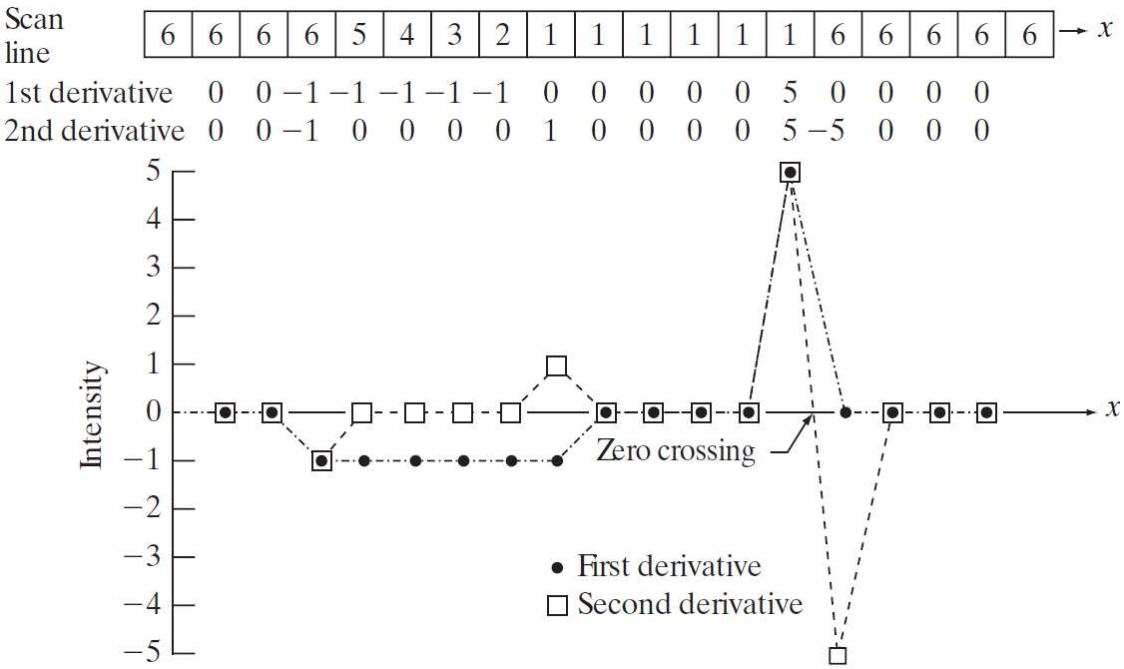


Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6	x
1st derivative	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0			
2nd derivative	0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0			

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

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

6. Sharpening Spatial Filters



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

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

6. Sharpening Spatial Filters

- Using the second derivative – **The Laplacian**:

  - The approach basically consists of defining a **discrete** formulation of the **second-order derivative** and then constructing a **filter mask** based on that formulation.

  - We are interested in **isotropic filters**, whose response is independent of the direction of the discontinuities in the image to which the filter is applied.

  - The simplest isotropic derivative operator is the Laplacian, which, for a function of two variables (image), is defined as

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:
  - We must express this equation in discrete form.
    - ✓ In x-direction we have:

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

    - ✓ In y-direction we have:

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

  - It follows that the discrete Laplacian of two variables is

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:
  - The previous equation can be implemented using following filter masks

0	1	0
1	−4	1
0	1	0

0	−1	0
−1	4	−1
0	−1	0

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:

  - The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms

1	1	1
1	−8	1
1	1	1

−1	−1	−1
−1	8	−1
−1	−1	−1

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:

  - Because the Laplacian is a derivative operator, its use **highlights** intensity discontinuities in an image and **deemphasizes** regions with slowly varying intensity levels.

  - The basic way in which we use the Laplacian for image sharpening is

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

where *f* and *g* are the input and sharpened images, respectively.

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:

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

- c = -1 if

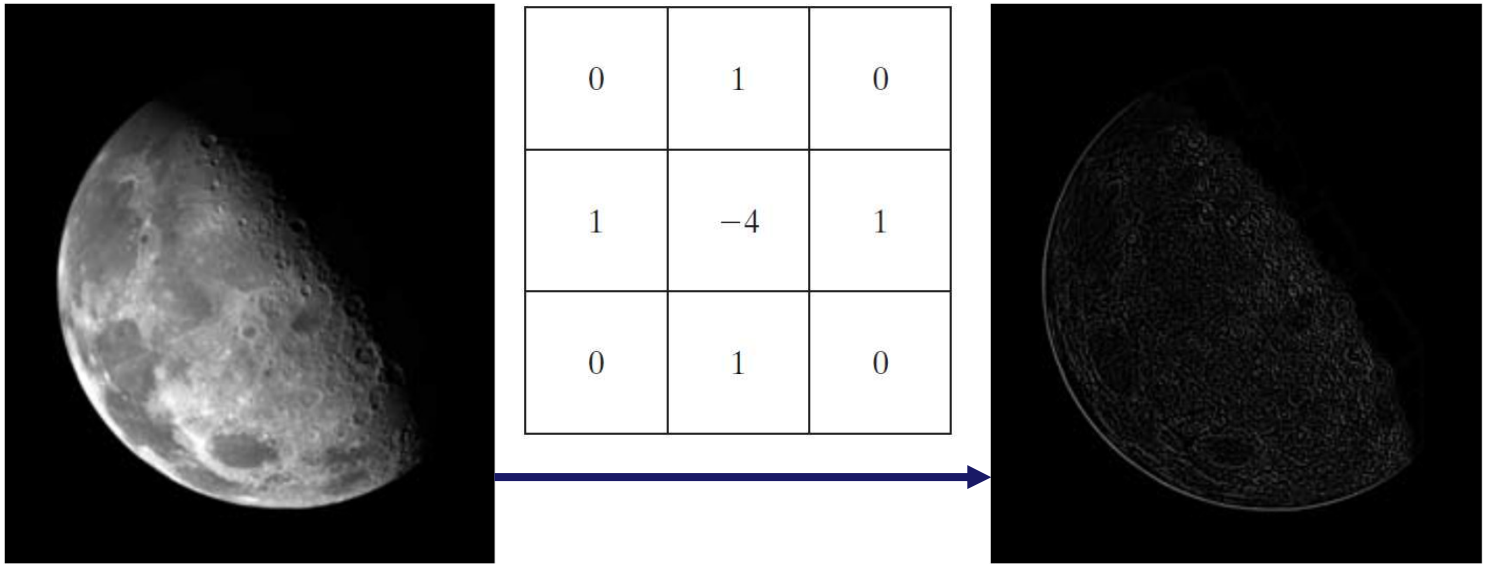
0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1

- c = 1 if

0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

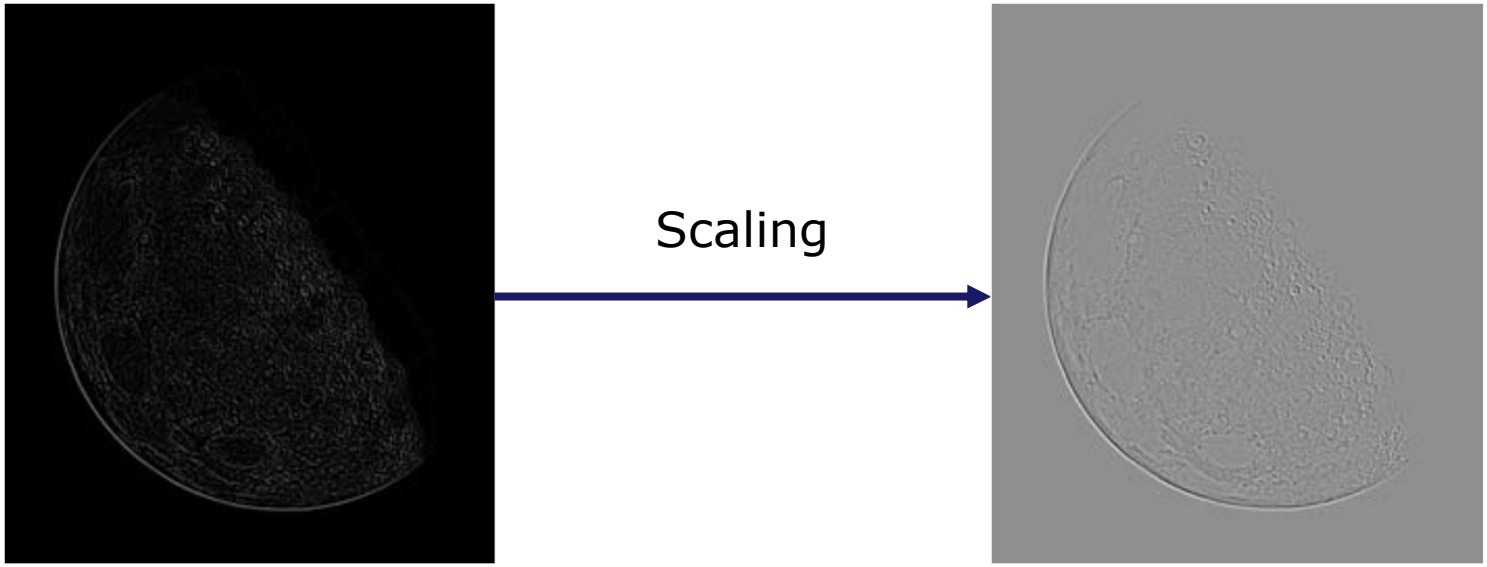
6. Sharpening Spatial Filters

- Example:



6. Sharpening Spatial Filters

- Example:



6. Sharpening Spatial Filters

- Example:

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

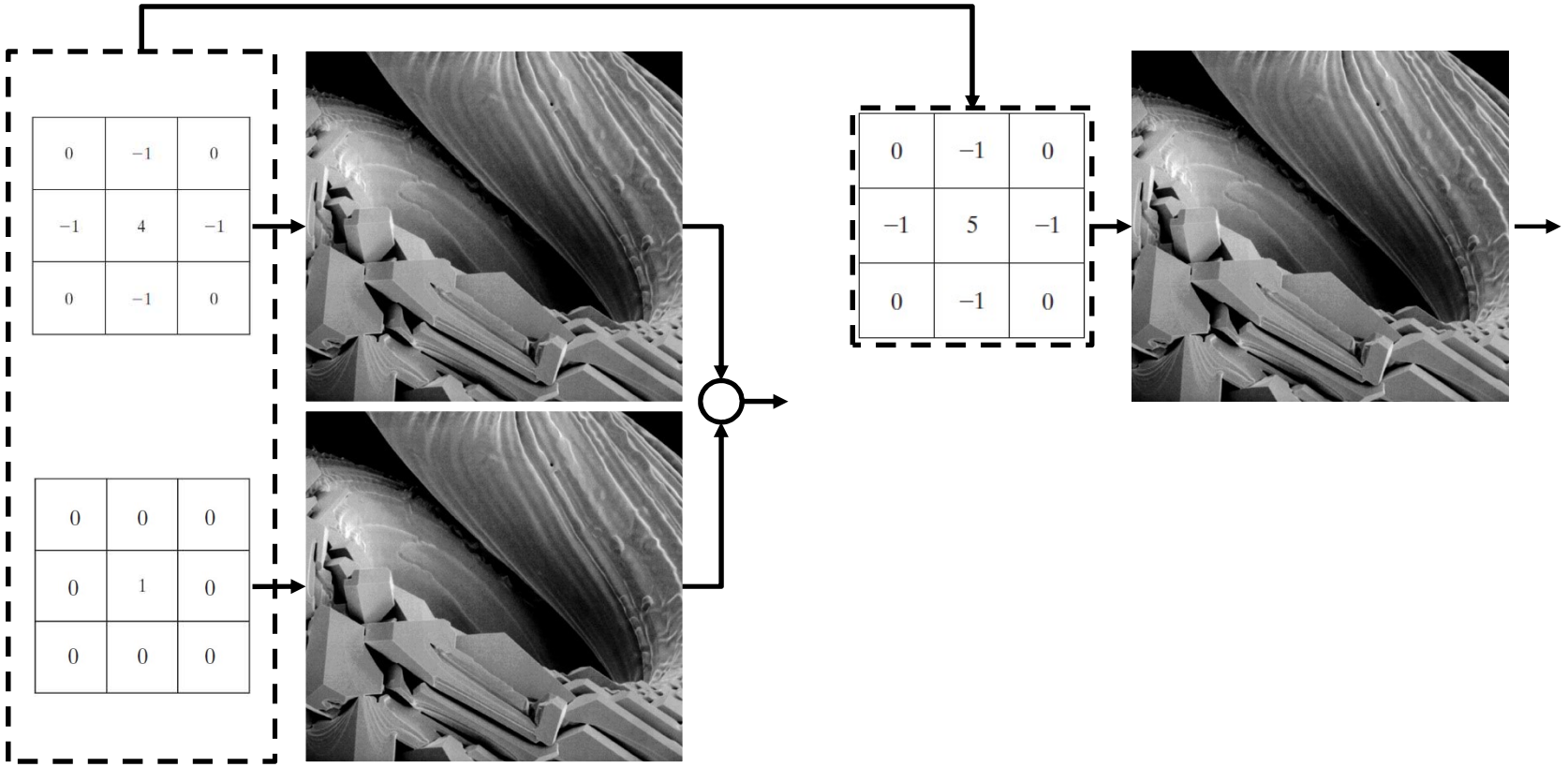
$g(x, y)$ $f(x, y)$ $c\left[\nabla^2 f(x, y)\right]$

6. Sharpening Spatial Filters

- MATLAB: s126Laplacian.m

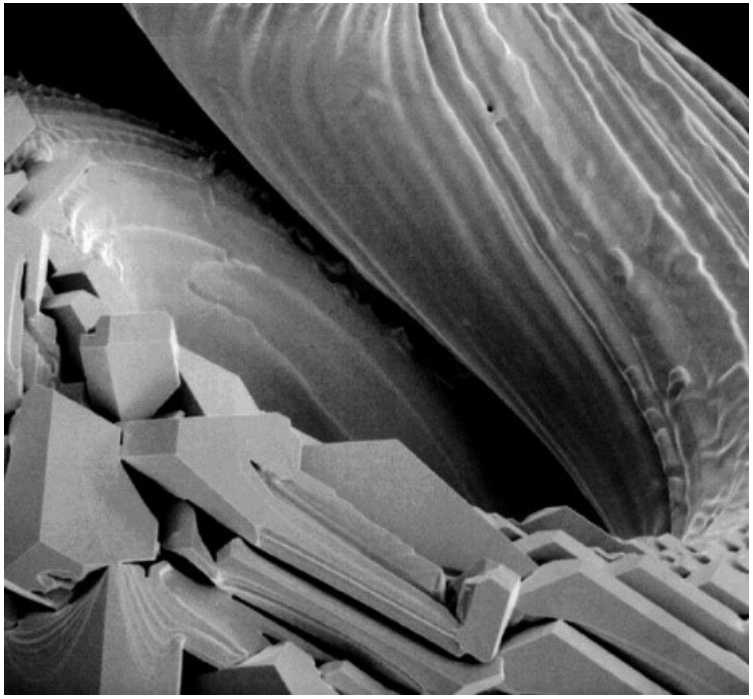
6. Sharpening Spatial Filters

- Simplification



6. Sharpening Spatial Filters

- MATLAB: s128LaplacianSimpl.m

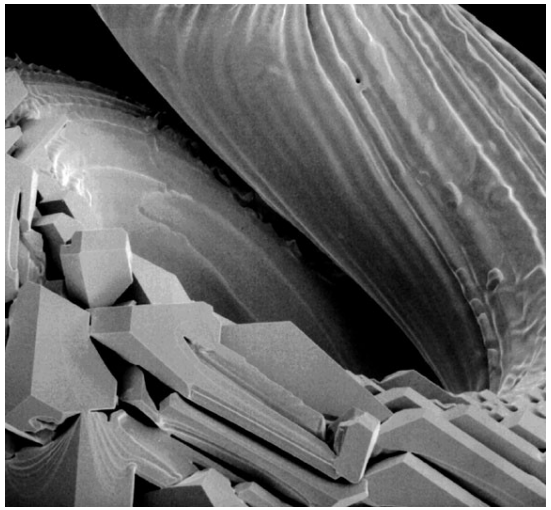


6. Sharpening Spatial Filters

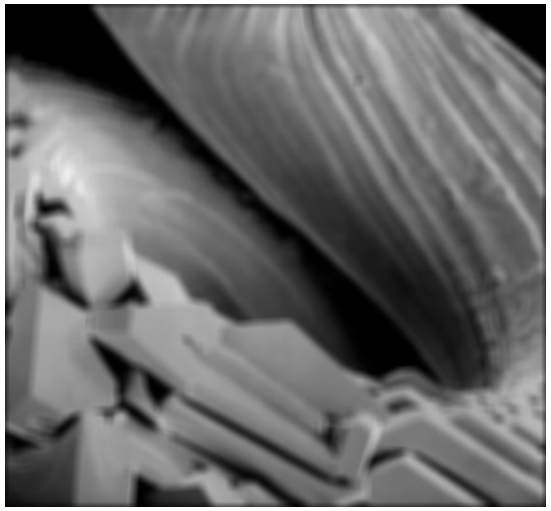
• **Unsharp Masking** Filtering: cconsists of subtracting an unsharp (smoothed) version of an image from the original image.

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

$g_{\text{mask}}(x, y)$



$f(x, y)$

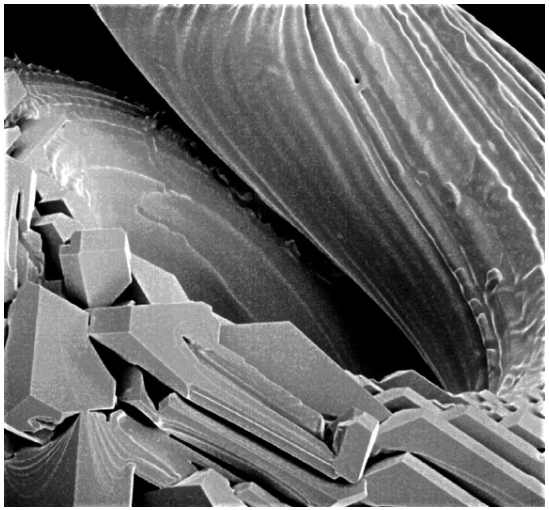


$\overline{f}(x, y)$

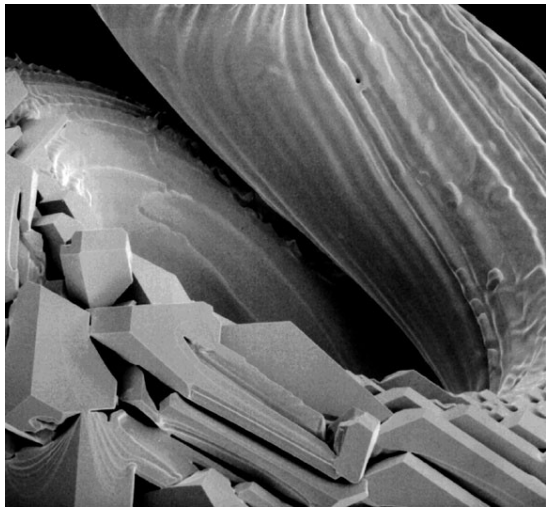
6. Sharpening Spatial Filters

- Unsharp Masking Filtering ($0 > k \geq 1$)

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$



$g(x, y)$



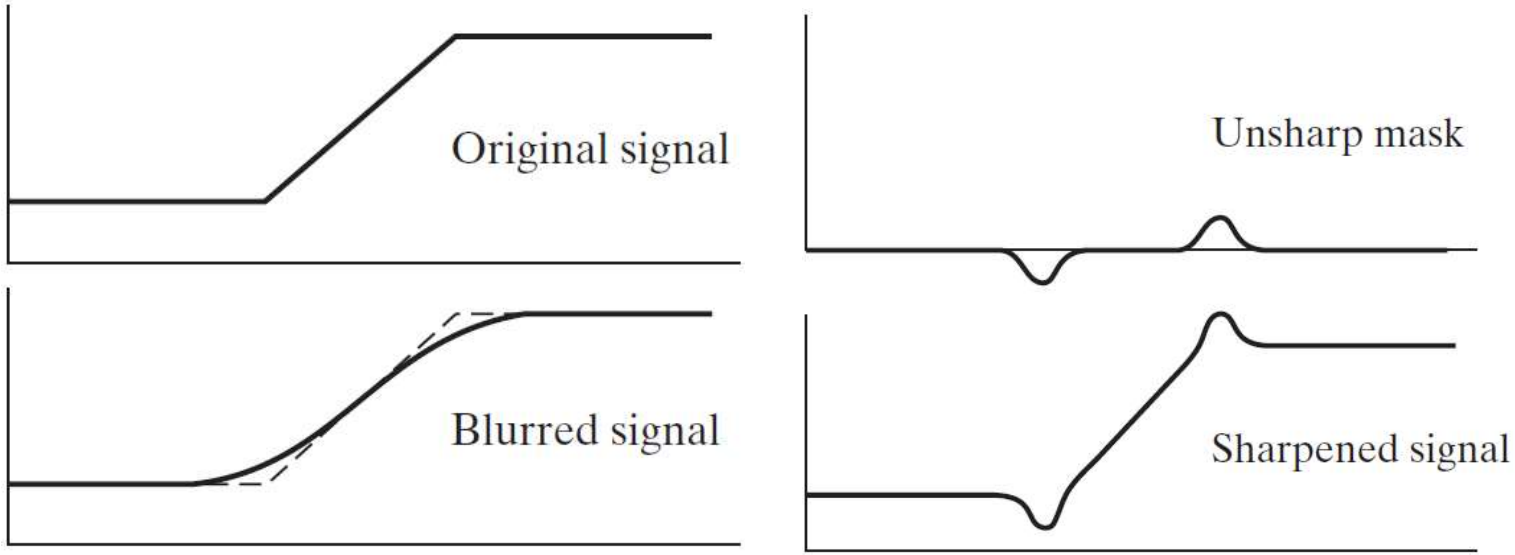
$f(x, y)$

$g_{\text{mask}}(x, y)$

6. Sharpening Spatial Filters

- Unsharp Masking Filtering ($0 > k \geq 1$)

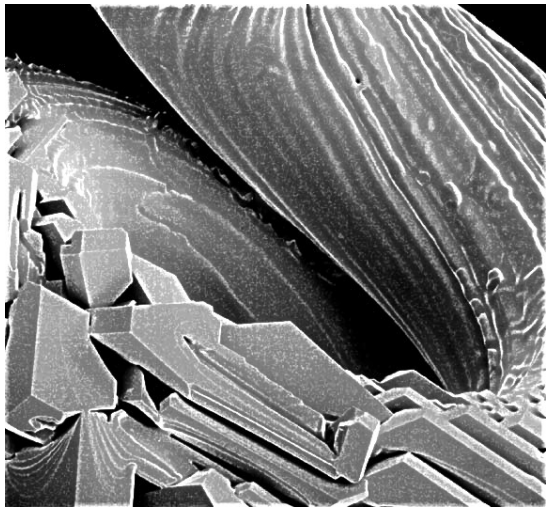
$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$



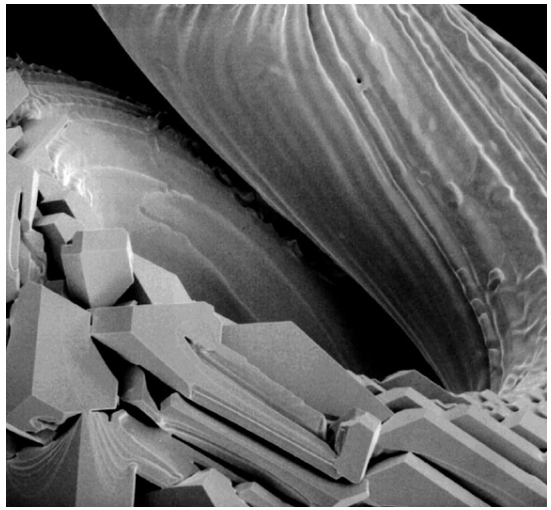
6. Sharpening Spatial Filters

- **High-Boost** Filtering (k>1)

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$



$g(x, y)$



$f(x, y)$

$k * g_{\text{mask}}(x, y)$

6. Sharpening Spatial Filters

- MATLAB: s133HighBoost.m

6. Sharpening Spatial Filters

- Using First-Order Derivatives - **The Gradient**

  - First derivatives in image processing are implemented using the *magnitude of the gradient*.

  - The gradient of a function *f* at coordinates *(x, y)* is defined as the two-dimensional column *vector*

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

  - This vector points in the direction of the greatest rate of change of *f* at location *(x, y)*.

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient

  - The **magnitude** of vector ∇f, denoted as *M(x, y),* is

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

  - It is common practice to refer to M as the *gradient image*.

  - In some implementations, it is more suitable computationally to approximate the squares and square root operations by absolute values:

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

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient

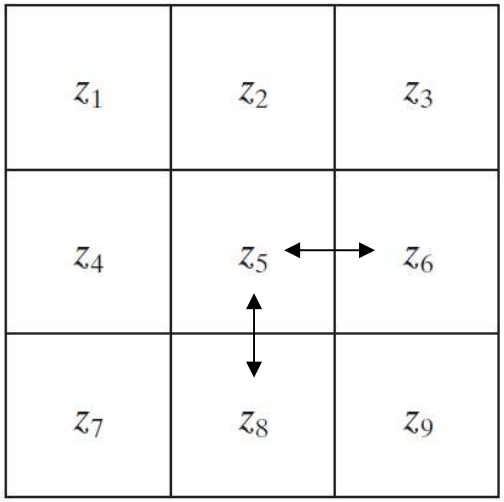
  - In order to simplify the discussion that follows, we will use the following notation:

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient

  - The simplest approximations to a first-order derivative are

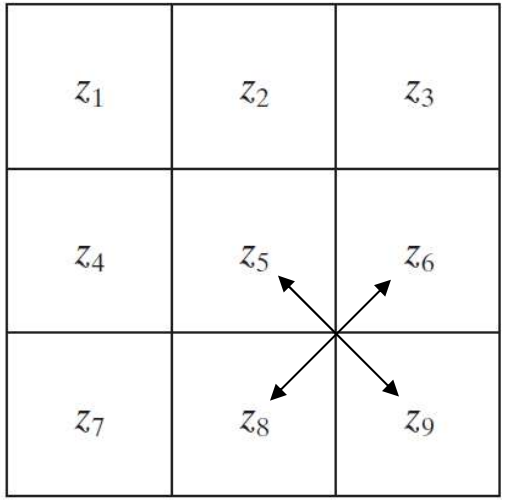


$$g_y = (z_6 - z_5)$$

$$g_x = (z_8 - z_5)$$

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient
  - Two other definitions proposed by Roberts [1965]
    - ✓ Roberts cross gradient operators



$$g_x = (z_9 - z_5) \qquad g_y = (z_8 - z_6)$$

−1	0
0	1

0	−1
1	0

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient

  - Approximations using a 3x3 neighborhood centered on z_5 are as follows:

    - ✓ Sobel operators

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

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

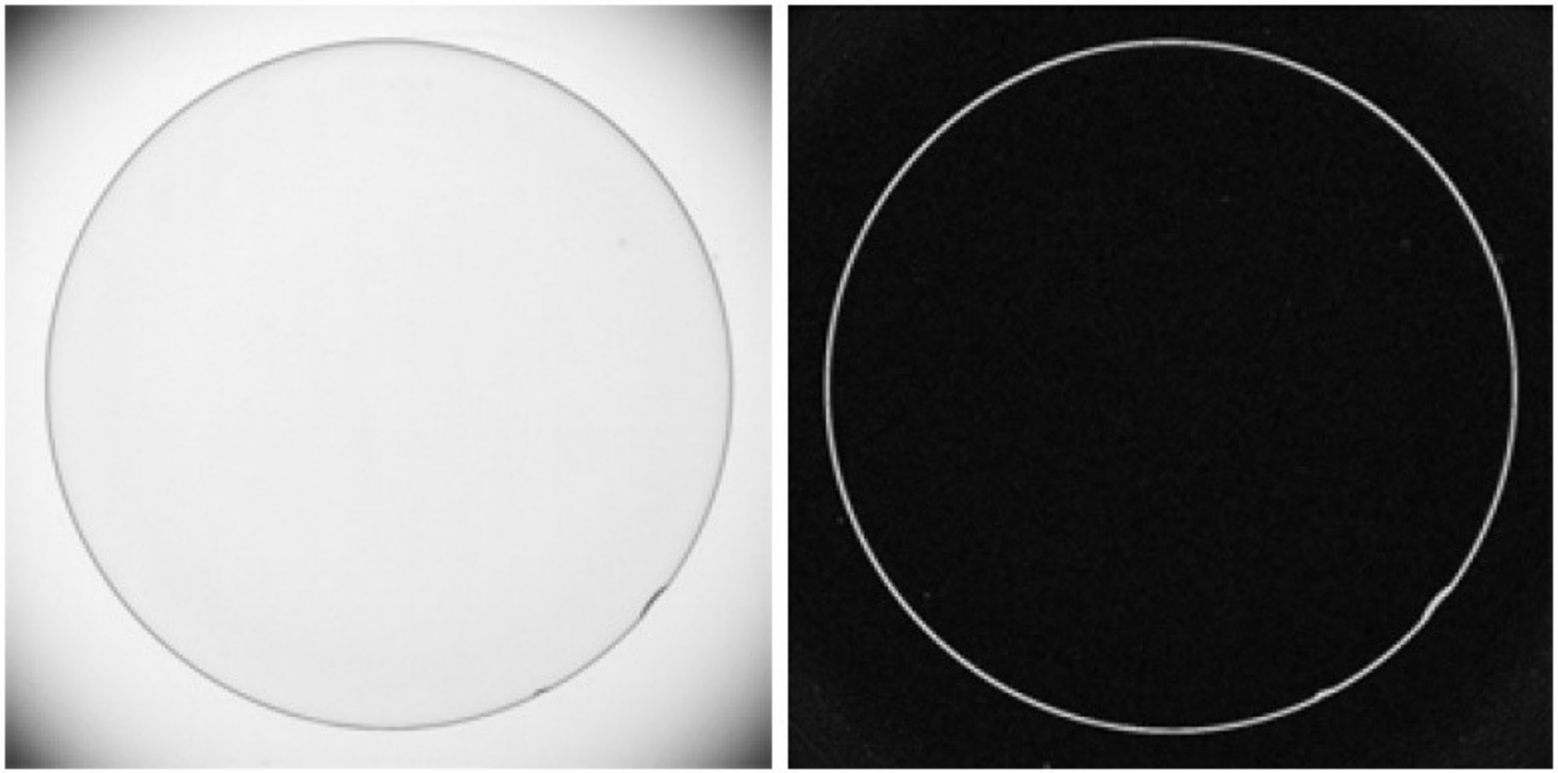
z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient



6. Sharpening Spatial Filters

- MATLAB: s141Sobel.m