

Lecture 7: Basic Mathematical Operations and Intensity Transformations

Dr. Muhammad Abeer Irfan
Department of Computer Systems Engineering
UET Peshawar

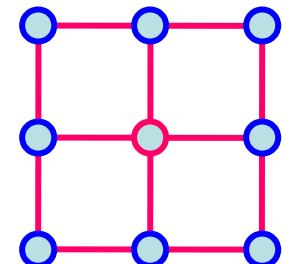
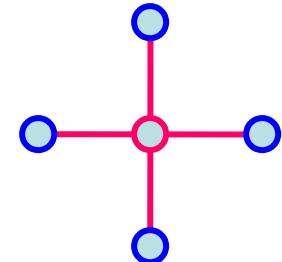
Readings: Chapter 2 (sections 2.5 and 2.6), Chapter 3 (sections 3.1 and 3.2)

Table of Content

- Relationships Between Pixels
- Mathematical Tools for DIP
- Basics of Intensity Transformations and Spatial Filtering
- Basic Intensity Transformation Functions

Neighbors of a Pixel

- The **4-neighborhood** of pixel $p(x,y)$ is the set:
 $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$
or $\{(i, j): |x-i| + |y-j| = 1\}$
- The two pixels p and q are **4-adjacent** (or **connected**) if q is in $N_4(p)$ or p is in $N_4(q)$.
- The **8-neighborhood** of pixel $p(x,y)$ is the set:
 $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1),$
 $(x-1, y-1), (x+1, y-1), (x-1, y+1), (x+1, y+1)\}$
or $\{(i, j): \max(|x-i|, |y-j|) = 1\}$
- The two pixels p and q are **8-adjacent** (or **connected**) if q is in $N_8(p)$ or p is in $N_8(q)$.



Paths Between Pixels

- A **digital path** from the pixel $p(x,y)$ to the pixel $q(s,t)$ is a sequence of pixels with coordinates:

$$(x_0, y_0), (x_1, y_1) \dots (x_n, y_n)$$

where

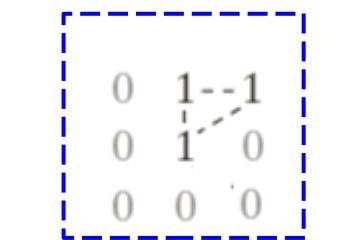
$$(x, y) = (x_0, y_0), (s, t) = (x_n, y_n)$$

and pixels

(x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$

- The adjacency can be defined using 4- or 8-neighborhood
- **Length** of a path: number of edges on the path
- **Closed** paths: $(x_0, y_0) = (x_n, y_n)$.
- Ambiguity of a path:

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



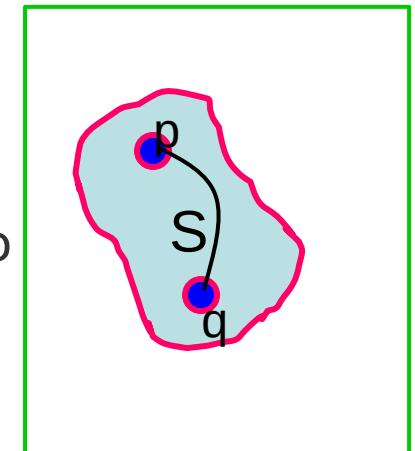
8-neighborhood

$$\begin{matrix} 0 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

4-neighborhood

Paths Between Pixels

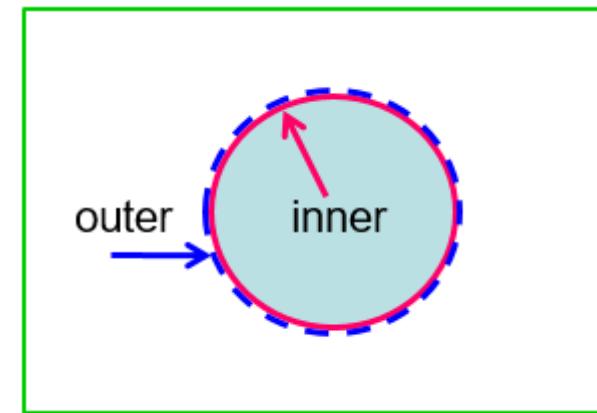
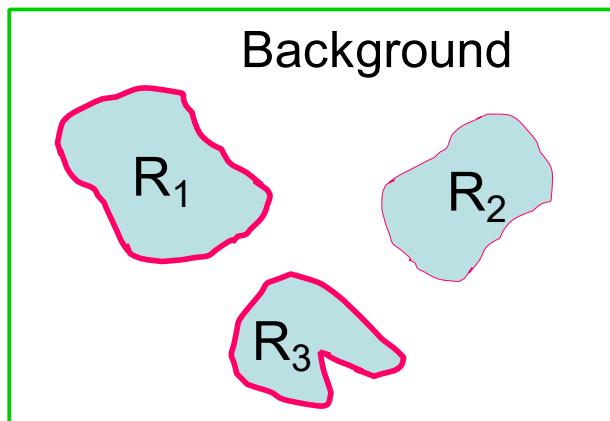
- Assuming \mathbf{S} a subset of pixels in an image, pixels p and q are connected in \mathbf{S} if there exists a path between them consisting entirely of pixels in \mathbf{S} .
- For any pixel p in \mathbf{S} , the set of pixels that are connected to it in \mathbf{S} is called a connected component of \mathbf{S} .
- If it only has one component, and that component is connected, the \mathbf{S} is called a connected set.
- Assuming \mathbf{R} a subset of pixels in an image, \mathbf{R} is called a region of the image if \mathbf{R} is a connected set. Two regions R_i and R_j are adjacent if their union forms a connected set. Otherwise they are disjoint.
- The definitions are dependent on the how the neighborhood is defined.



$$\begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \left. \begin{array}{l} \\ \\ \\ \\ R_i \\ R_j \end{array} \right\}$$

Paths Between Pixels

- Suppose an image contains K disjoint regions, R_k , $k=1,2,3,\dots,K$. Let R_u denote the union of all of the K regions, and let $(R_u)^c$ denote its complement. All the points in R_u are called the **foreground**, while all the points in the $(R_u)^c$ are called the **background**.
- The inner boundary (border, or contour) of a region R is the set of pixels in R that are *adjacent* to pixels in the complement of R .
- The outer boundary of a region is the set of pixels in the background that have at least one neighbor in R .



Distance Measures

- For pixels $p(x,y)$, $q(u,v)$ and $s(w,z)$, D is a distance function or metric if:

- (a) $D(p,q) \geq 0$ ($D(p,q) = 0$ iff $p = q$),
- (b) $D(p,q) = D(q,p)$, and
- (c) $D(p,s) \leq D(p,q) + D(q,s)$.

- The Euclidean distance between p and q is defined as:

$$D_e(p,q) = \sqrt{(x-u)^2 + (y-v)^2}$$

- The D_4 distance (city-block distance) between p and q is defined as:

$$D_4(p,q) = |x - u| + |y - v|$$

2					
2	1	2			
2	1	0	1	2	
2	1	2			
			2		

- The D_8 distance (chessboard distance) between p and q is defined as:

$$D_8(p,q) = \max(|x - u|, |y - v|)$$

2	2	2	2	2	2
2	1	1	1	2	
2	1	0	1	2	
2	1	1	1	2	
2	2	2	2	2	2

Linear vs. Non-Linear Operations

- Assume a general operator that produces an output image, $\mathbf{g(x,y)}$, from a given input image $\mathbf{f(x,y)}$:

$$\mathcal{H}[f(x, y)] = g(x, y)$$

- The operator is said to be linear, if it satisfies two properties:
 - Homogeneity
 - Additivity

$$\begin{aligned}\mathcal{H}[af_1(x, y) + bf_2(x, y)] &= a\mathcal{H}[f_1(x, y)] + b\mathcal{H}[f_2(x, y)] \\ &= ag_1(x, y) + bg_2(x, y)\end{aligned}$$

- For example mean operator is a linear operator, while max operator is not.

Arithmetic Operations

- Arithmetic operations between two images are defined as:

$$s(x, y) = f(x, y) + g(x, y)$$

$$d(x, y) = f(x, y) - g(x, y)$$

$$p(x, y) = f(x, y) \times g(x, y)$$

$$v(x, y) = f(x, y) \div g(x, y)$$

- All are element-wise operations, performed between corresponding pixel pairs in **f** and **g** for **x=0,1,2,...,M-1** and **y=0,1,2,..., N-1**, resulting in images of the same size as inputs.
- They are defined for input images of the same size.

Arithmetic Operations: Examples

- Assume that in image acquisition, zero-mean uncorrelated noise is added to the noiseless image:

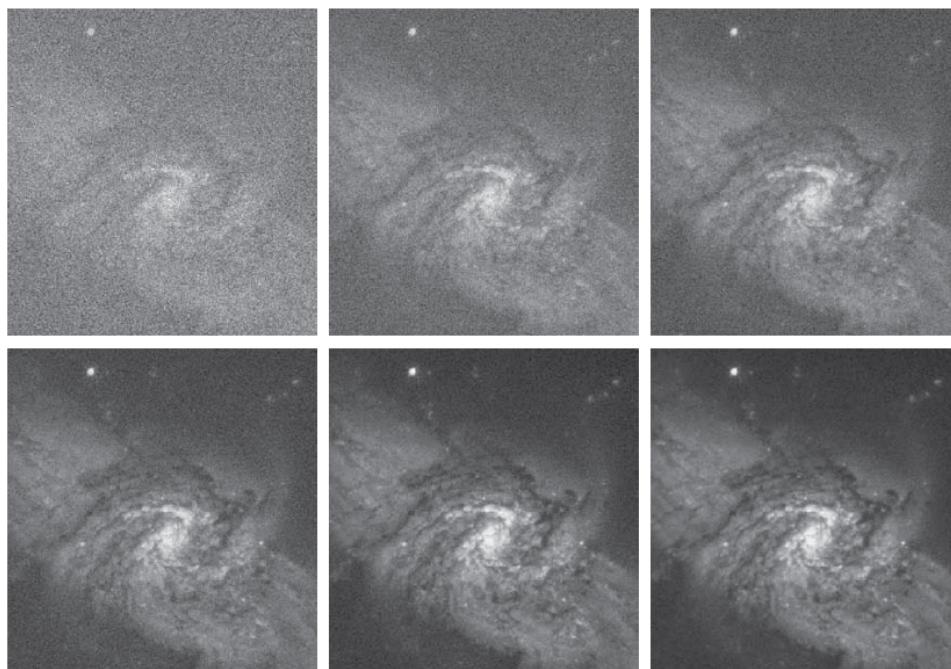
$$g(x, y) = f(x, y) + \eta(x, y)$$

- If noise is zero-mean and uncorrelated with the noise-less image, averaging **K** different noisy images can help in reducing the noise:

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y)$$

$$E\{\bar{g}(x, y)\} = f(x, y)$$

$$\sigma_{\bar{g}(x, y)}^2 = \frac{1}{K} \sigma_{\eta(x, y)}^2$$



Arithmetic Operations: Examples

subtraction is used for comparing of images

- Another example of arithmetic operations is by using subtraction to compare images.



a b c

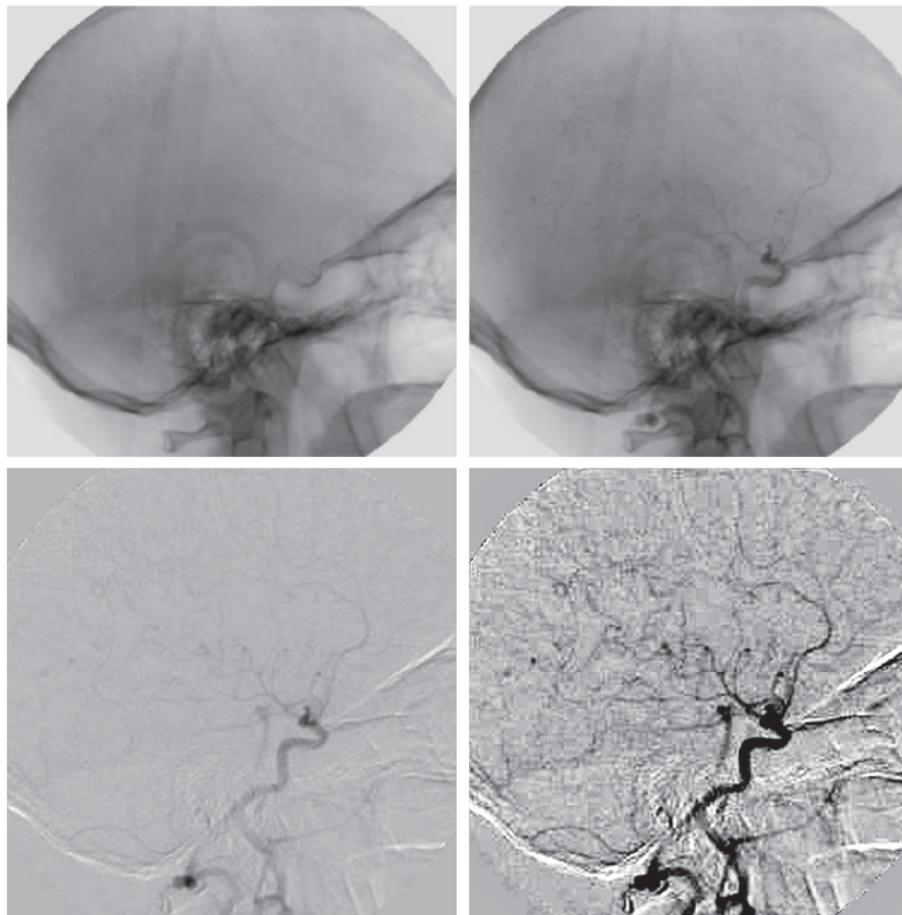
FIGURE 2.31 (a) Difference between the 930 dpi and 72 dpi images in Fig. 2.23. (b) Difference between the 930 dpi and 150 dpi images. (c) Difference between the 930 dpi and 300 dpi images.

Arithmetic Operations: Examples

- Digital Subtraction Angiography (DSA)

a b
c d

FIGURE 2.32
Digital subtraction angiography.
(a) Mask image.
(b) A live image.
(c) Difference between (a) and (b). (d) Enhanced difference image.
(Figures (a) and (b) courtesy of the Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)



Arithmetic Operations: Examples

- Image multiplication and division can be used for shading correction.
- Assume an imaging sensor produces images that are modeled as the product of a perfect image $f(x,y)$, times a shading function $h(x,y)$:

$$g(x, y) = f(x, y)h(x, y)$$

- If $h(x,y)$ is known, or could be estimated, then:

$$f(x, y) = g(x, y)/h(x, y)$$



a b c

FIGURE 2.33 Shading correction. (a) Shaded test pattern. (b) Estimated shading pattern. (c) Product of (a) by the reciprocal of (b). (See Section 3.5 for a discussion of how (b) was estimated.)

Technical Note

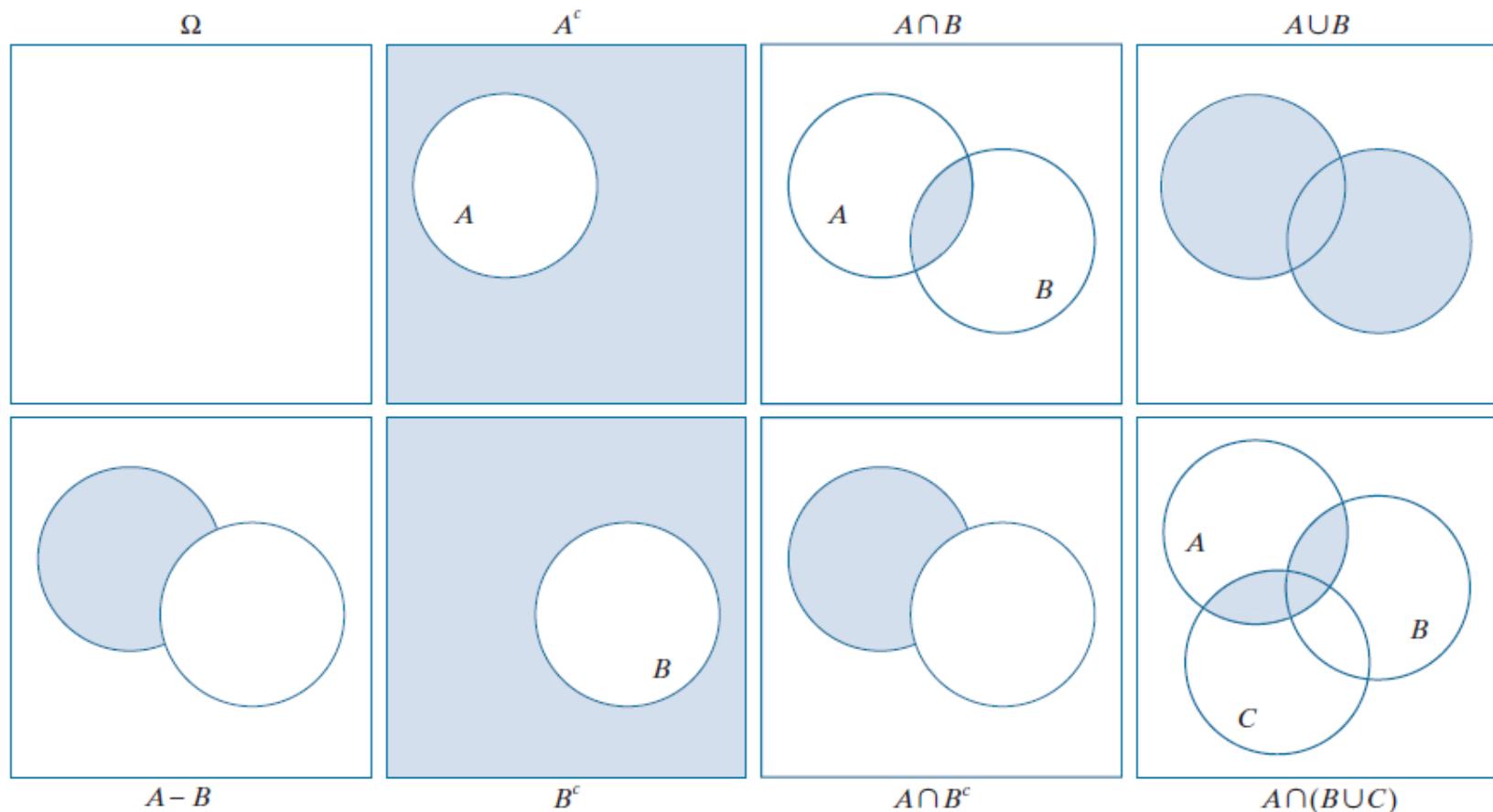
- Since there is a range of values that can be shown using any specific number of bits, it is likely that image processing procedures result in values that are out of this range.
- For example, subtraction of two 8-bit images can range from -255 to +255, which is out of bound for an 8-bit image.
- Clipping or scaling is necessary, especially when saving the images.
- To map the processed image to the desired range:

$$g_s(x, y) = K \frac{g(x, y) - g_{min}}{g_{max} - g_{min}}$$

- For an 8-bit image, $K=255$.
- g_{min} and g_{max} are the smallest and largest intensity values of the image $g(x, y)$ respectively.

Set Operations

- Set: a collection of distinct objects.



Set Operations

- How set operations can be applied to images? Let's see an example.
- Complement of an image can be defined using set operations as:

$$A^c = \{(x, y, K - z) \mid (x, y, z) \in A\}$$

- **K** is equal to the maximum intensity values in the image. In an 8-bit image, **K=255**.
- The union of two grayscale sets **A** and **B** with the same number of elements is defined as:

$$A \cup B = \left\{ \max_z(a, b) \mid a \in A, b \in B \right\}$$

Set Operations

a b c

FIGURE 2.36

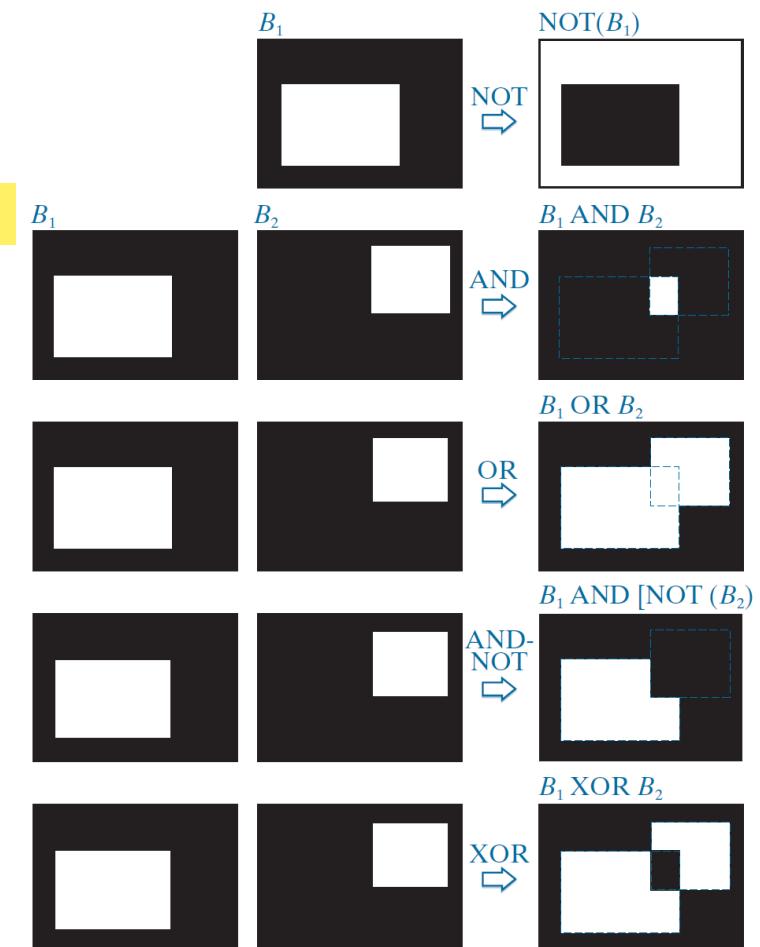
Set operations involving grayscale images. (a) Original image. (b) Image negative obtained using grayscale set complementation. (c) The union of image (a) and a constant image. (Original image courtesy of G.E. Medical Systems.)



Logical Operations

- Logical operations deal with **TRUE (1)** and **FALSE (0)** variables and expressions, which means that we can apply them to binary images with foreground (1-valued) pixels and background (0-valued) pixels.
- A binary image can be considered as a Venn diagram with regions of 1-valued pixels considered as sets.

a	b	$a \text{ AND } b$	$a \text{ OR } b$	$\text{NOT}(a)$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0



Spatial Operations

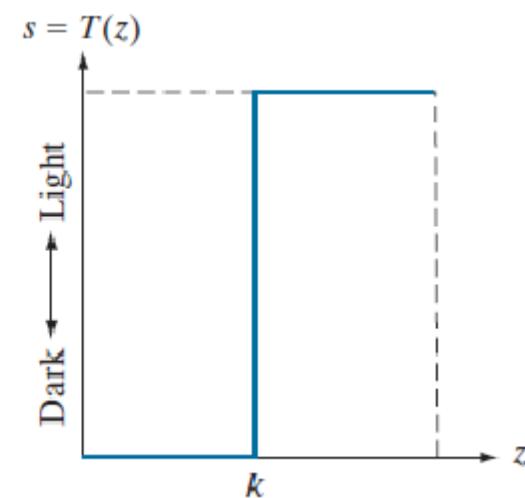
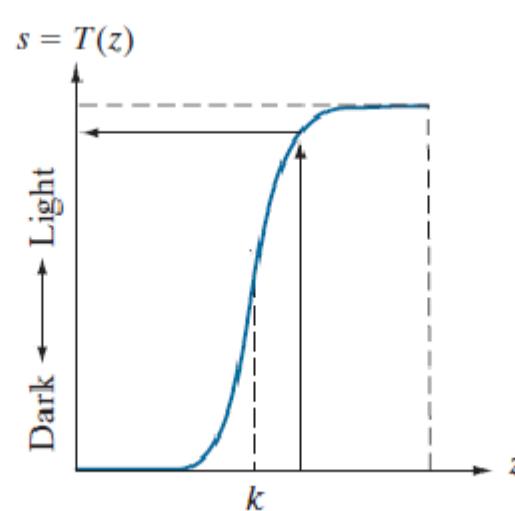
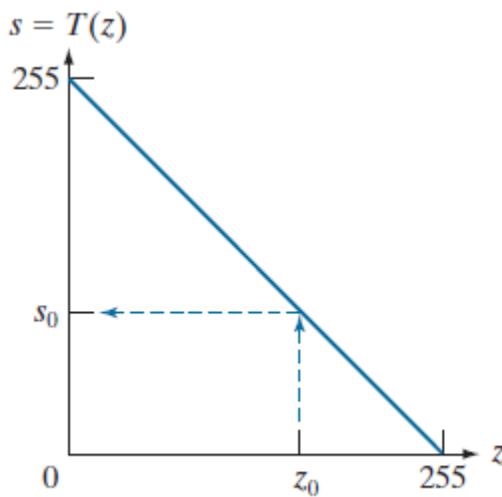
- Spatial operations work directly on the pixels of an image.
- Three types of spatial operations:
 - Single-pixel operations
 - Neighborhood operations
 - Geometric spatial transformations

SOs: Single-Pixel Operations

- To alter the intensity of pixels individually using a transformation function T , where z is the intensity of a pixel in the original image and s is the mapped (transformed) intensity of the corresponding pixel in the processed image.

$$s = T(z)$$

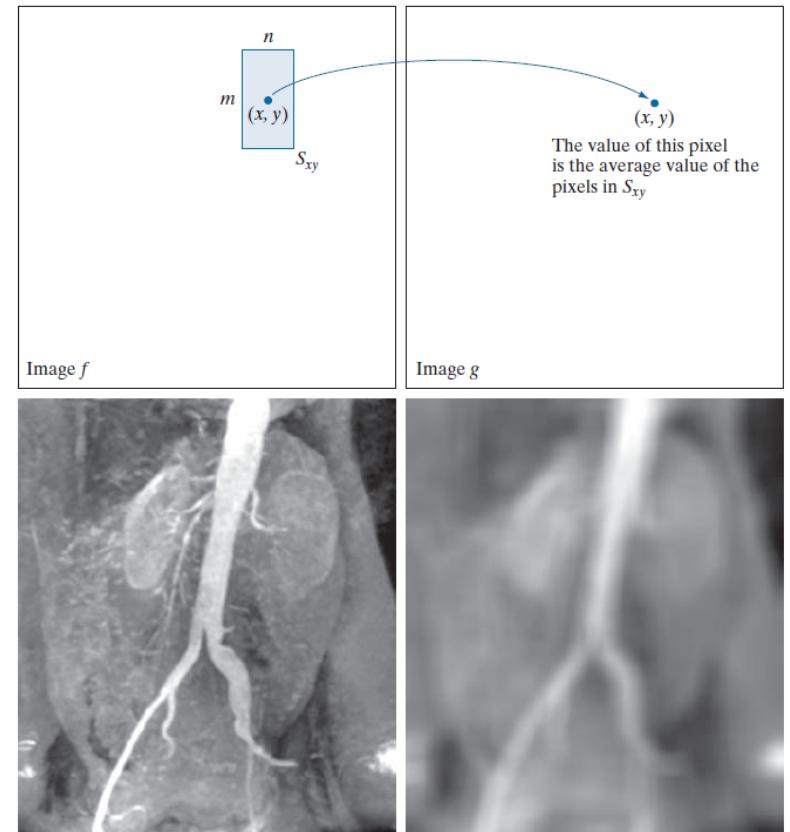
- Examples for negative (complement), contrast stretching and thresholding of images, respectively.



SOs: Neighborhood Operations

- Assume S_{xy} as a set of neighborhood pixels around a center (x,y) in image f .
- Neighborhood operations generate an intensity value for the center pixel (x,y) such that its value is determined by a specified operation on the neighborhood of the pixels in the input image.
- For example, if we want to perform the neighborhood average:

$$g(x, y) = \frac{1}{mn} \sum_{(r, c) \in S_{xy}} f(r, c)$$



SOs: Geometric Spatial Transformations

- Geometric operations of digital images consists of two basic operations:
 - Spatial transformation of coordinates;
 - Intensity interpolation for assigning intensity values to spatially transformed pixels.
- **Affine Transformation** provides scaling, translation, rotation and shearing.
- Affine transformations preserve points, straight lines, and planes.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

SOs: Geometric Spatial Transformations

- Different transformations can be concatenated by calculating the product of their transformation matrices.

$$A = A_1 * A_2 * A_3 * \dots * A_n$$

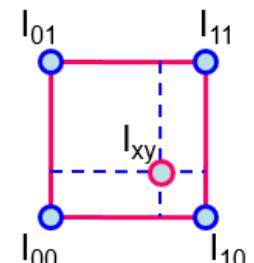
Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x \cos \theta - y \sin \theta$ $y' = x \sin \theta + y \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

Interpolation is a method used in mathematics and computer science to estimate unknown values that fall between known values. It involves constructing new data points within the range of a discrete set of known data points

SOs: Geometric Spatial Transformations

- To complete the process, we assign intensity values to the spatially transformed locations by means of intensity interpolation.
- Basic interpolation methods are:
 - Nearest neighbor
 - Bilinear interpolation (4 nearest neighbor pixels are needed)

$$v(x, y) = ax + by + cxy + d$$



- Bicubic interpolation (16 nearest neighbor pixels are needed)

$$v(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$$

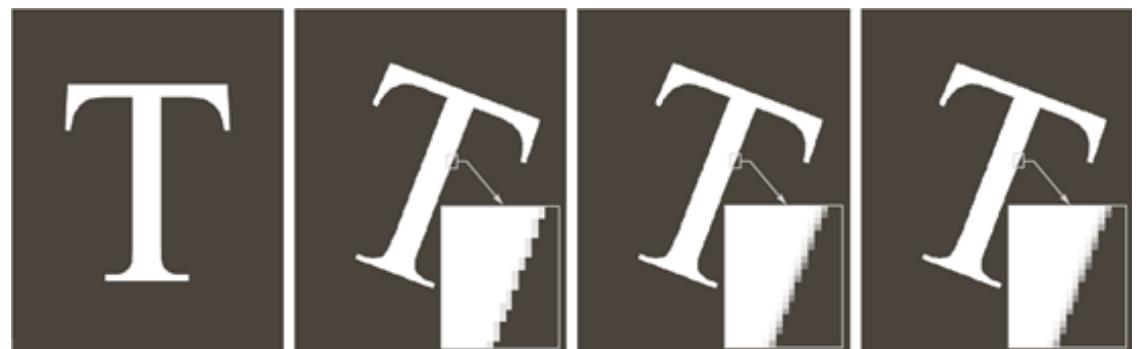


Image Intensities as Random Variables

- What is the probability of an intensity level in an image?
- Let's have z_i , for $i=0,1,2,\dots,L-1$ as the values of all possible intensity levels in an image.
- The probability $p(z_i)$ of intensity level z_k occurring in the image is:

$$p(z_k) = \frac{n_k}{MN}$$

where n_k is the number of times that the intensity level z_k occurs in the image, and MN is the total number of pixels in the image.

- We have:

$$\sum_{k=0}^{L-1} p(z_k) = \frac{1}{MN} \sum_{k=0}^{L-1} n_k = \frac{MN}{MN} = 1$$

Image Intensities as Random Variables

- Like any other random variable, we can determine important characteristics by using the probabilities.
- The mean (average) intensity is:

$$m = \sum_{k=0}^{L-1} z_k p(z_k)$$

- The variance of the intensities is:

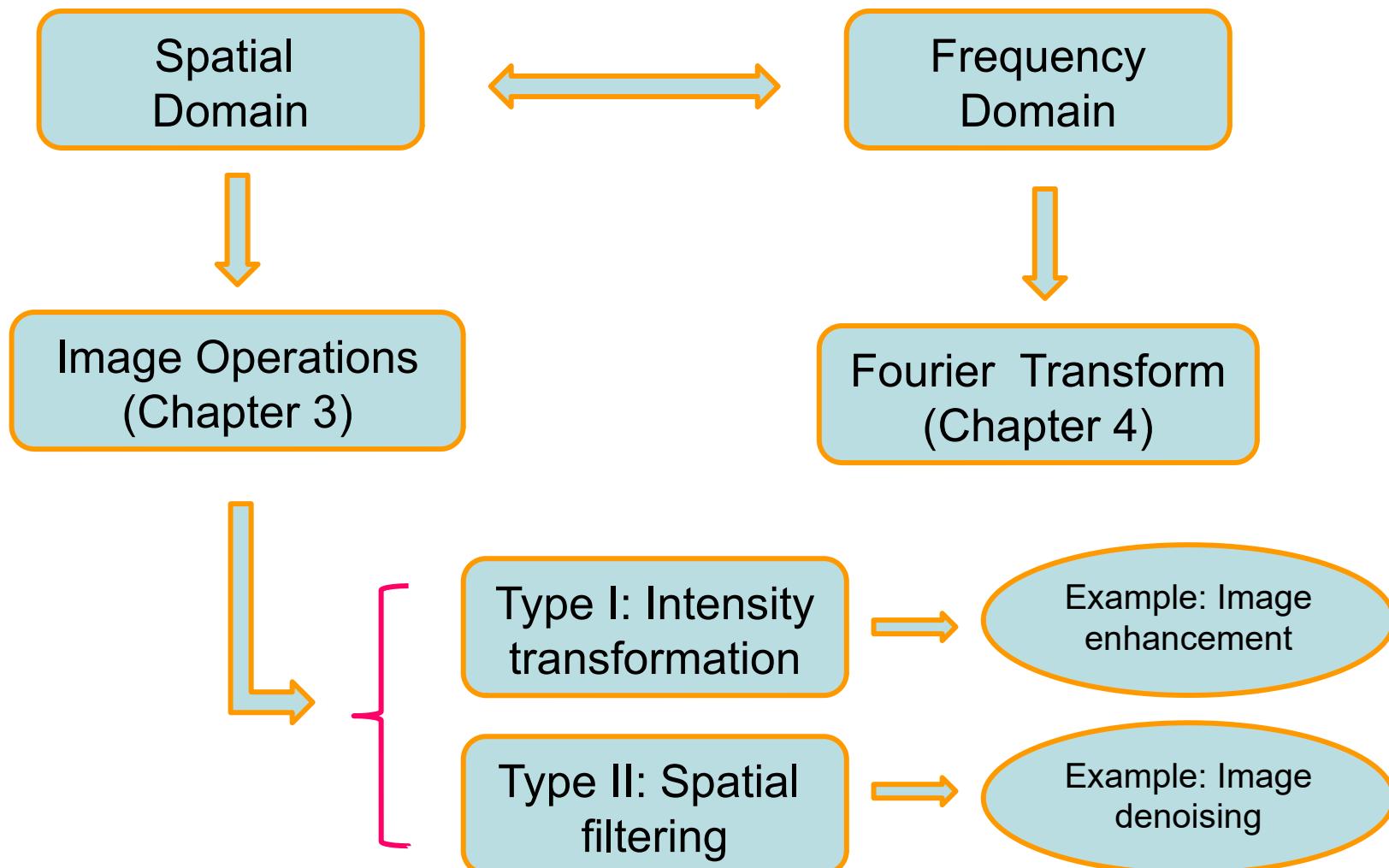
$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k)$$

- Generally, the n'th central moment of random variable z about the mean is:

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k)$$

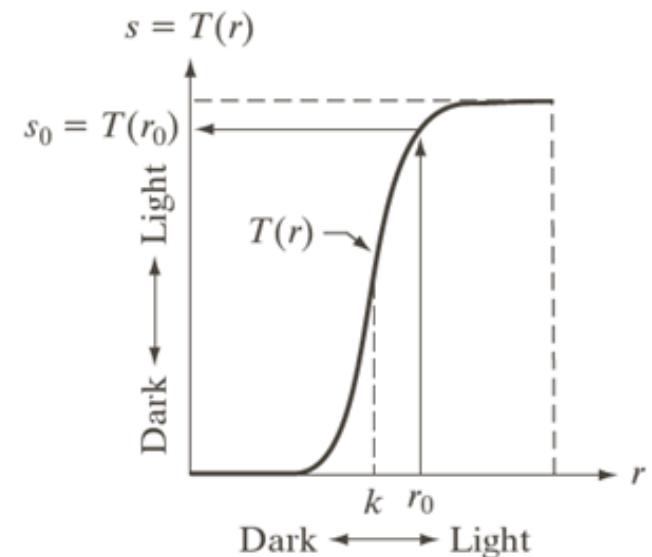
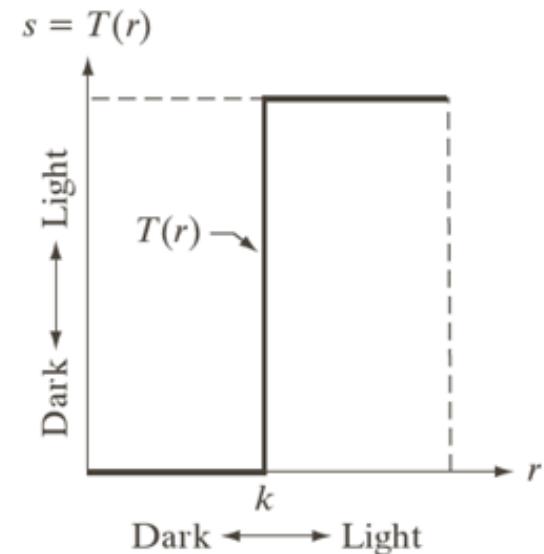
where $\mu_0(z) = 1, \mu_1(z) = 0, \mu_2(z) = \sigma^2$.

Intensity Transformations and Spatial Filtering



Type I: Intensity Transformations

- Intensity transformation: $s = T(r)$
 - Operating on single pixels
 - Example: thresholding & contrast stretching
 - The transformation is independent of locations, determined only by the intensity values.
- Procedure
 - Check the pixels in a raster scanning method
 - Calculate the new value for each pixel
 - Update the intensity
 - Repeat for every pixel



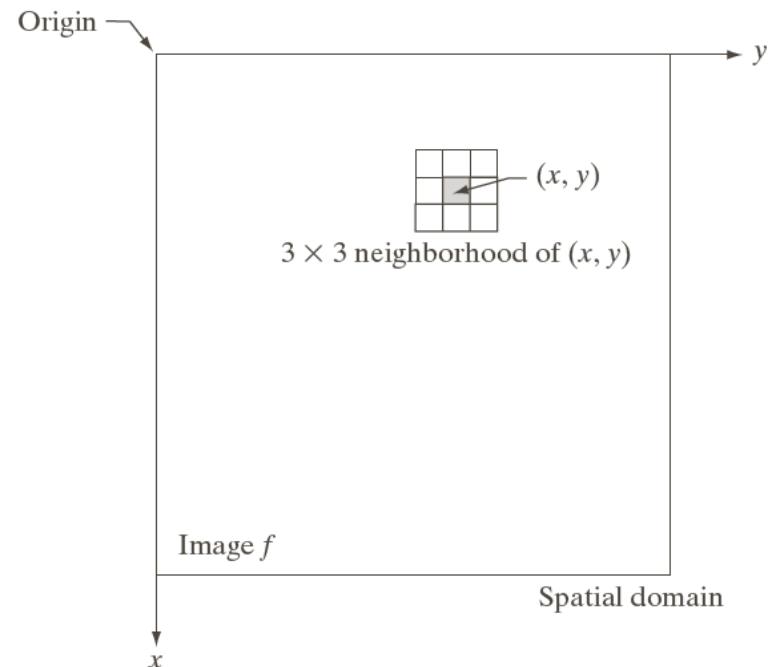
Type II: Intensity Transformations

- Spatial filtering:
$$g(x, y) = T[f(x, y)]$$

- In a neighborhood

- Typically a square shape
 - Example: intensity averaging

- Procedure
 - Check the pixels in a raster scanning method
 - Calculate the new value for each pixel
 - Update the intensity
 - Repeat for every pixel



Basic Intensity Transformation Functions

- Two types of low-quality images we deal with here
 - Too dark
 - Too bright
- Transformation function
$$s = T(r)$$
- In digital images, r and s are discrete values. We should have a look-up table to define the function $T(*)$.

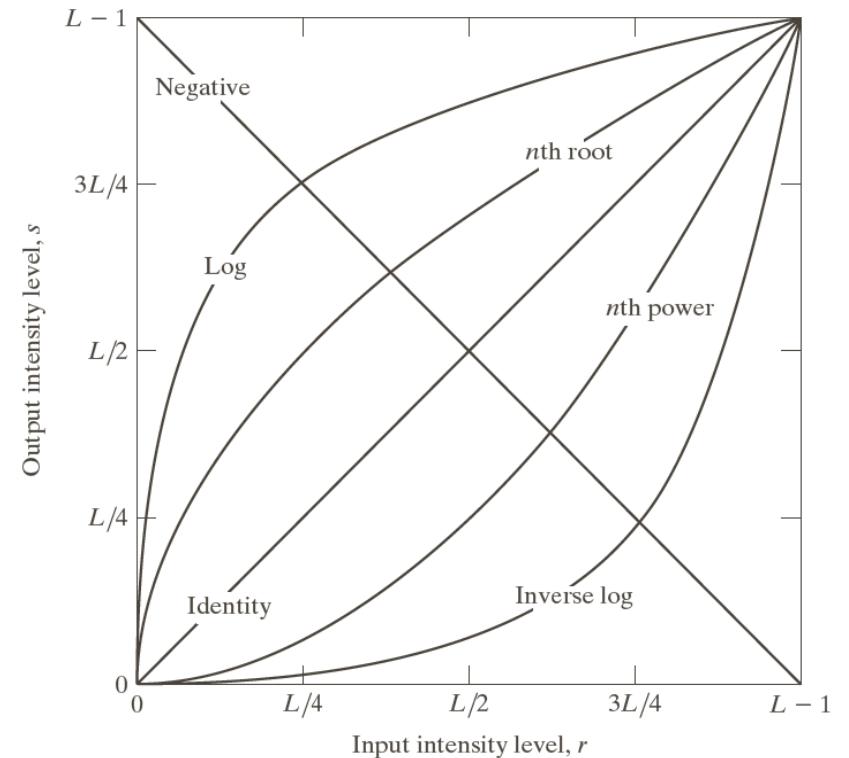
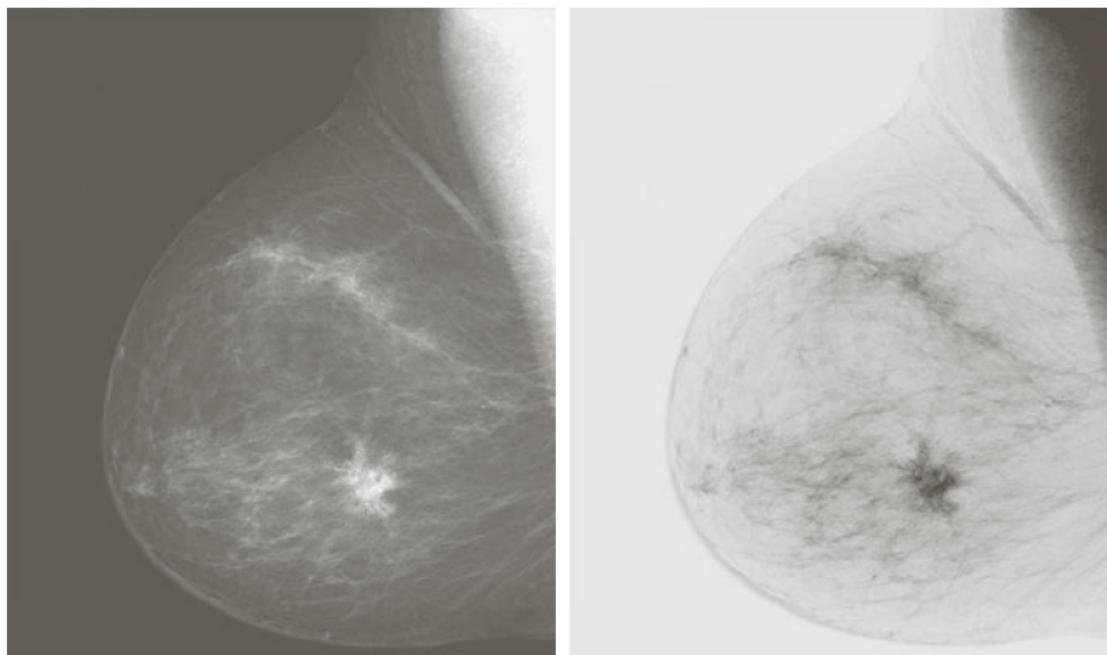


Image Inversion

- Compute the negative of an image:

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

- Good to visualize the features when black areas are dominant in size.



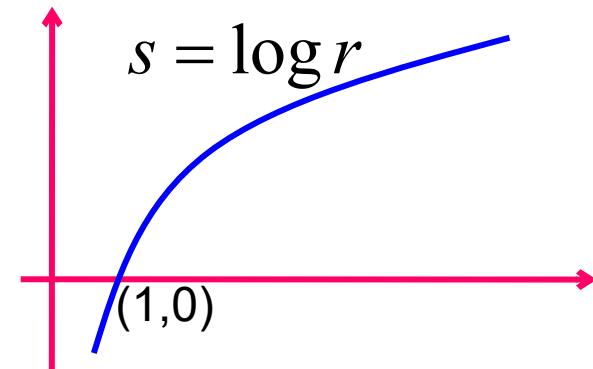
Mammography images

Log Transformation

- Transformation function

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

- Enhancing (stretching) dark areas
- Compressing bright areas



Range:
0 – 1.5×10^6

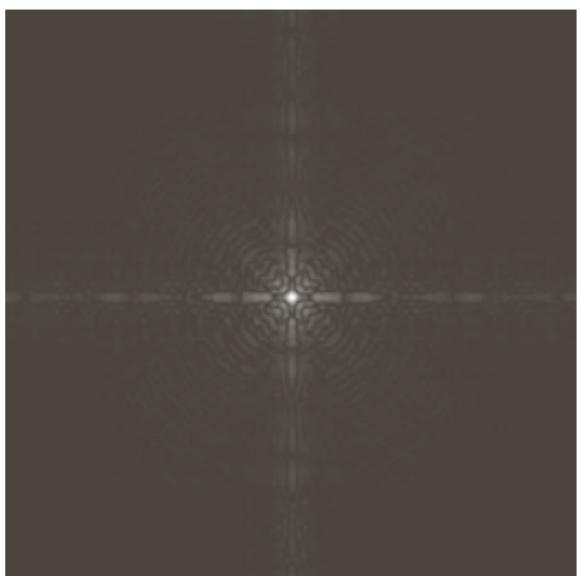
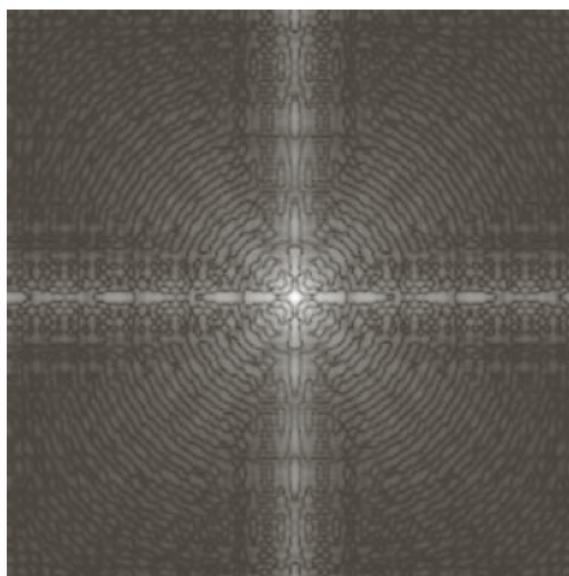


Image in the Fourier domain



Enhanced image

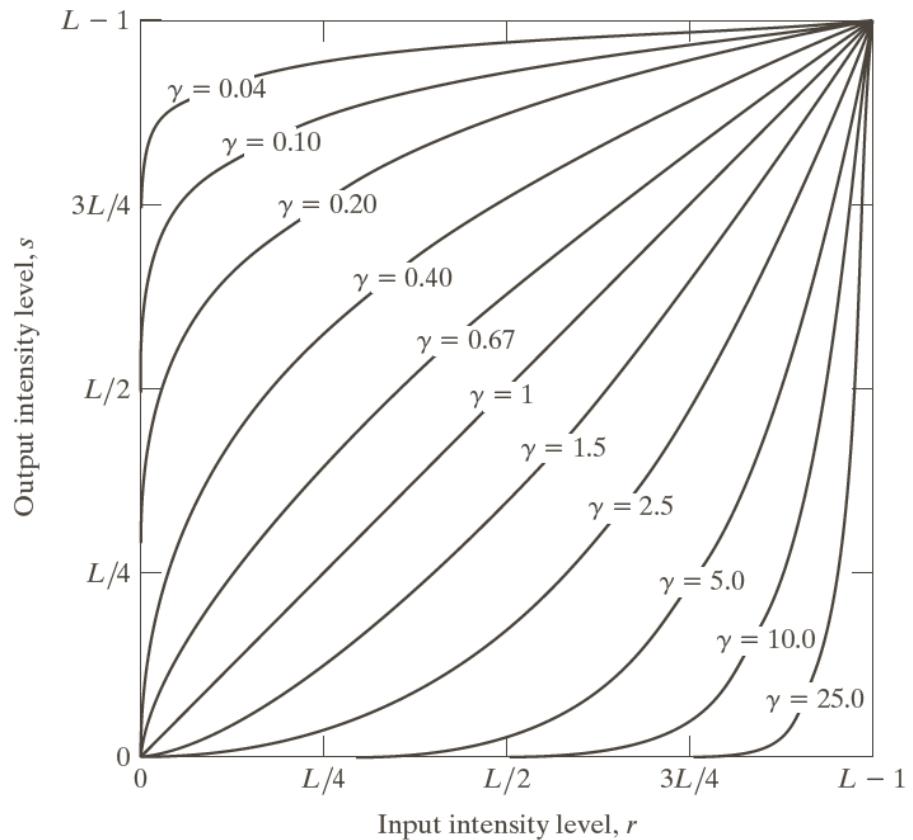
Range:
0 – 6.2

Power Law (Gamma) Transformation

- The mathematical form:
- If $\gamma < 1$
 - Enhancing dark areas
- If $\gamma > 1$
 - Enhancing bright areas
- If $\gamma = c = 1$
 - Identity transformation

$$s = cr^\gamma \text{ or } s = c(r + \epsilon)^\gamma$$

for $c > 0$ and $\gamma > 0$
and s and r scaled to $[0, 1]$



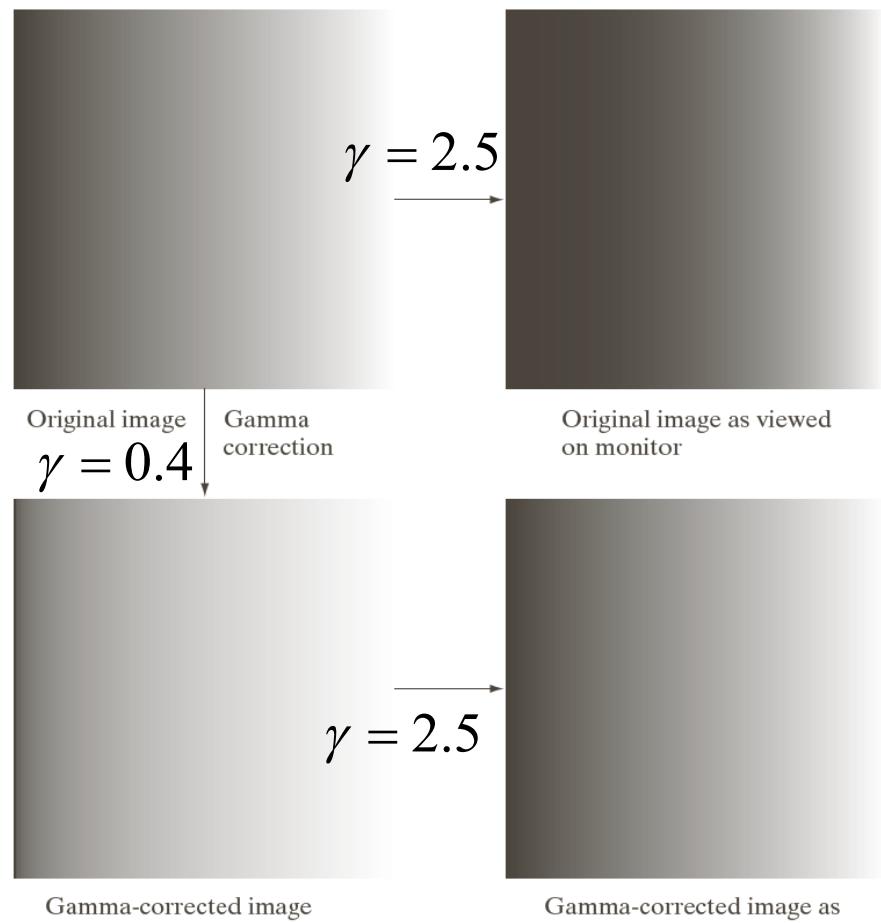
Application I: Gamma Correction

- Many image devices (capturing, printing, and displaying) internally undergo a “gamma” intensity conversion process:

$$\hat{s} = r^\gamma$$

- For instance, CRT devices have a gamma value of 1.8 – 2.5, which tend to darken the output images
- Correction:
 - Estimate gamma
 - Make the correction

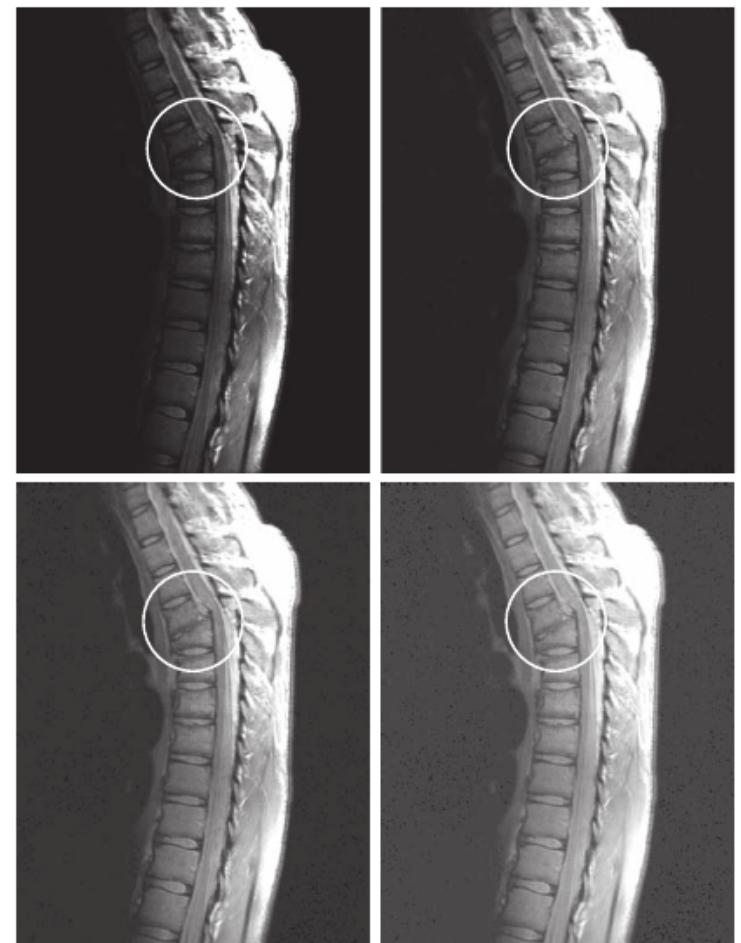
$$s = \hat{s}^{1/\gamma} = (r^\gamma)^{1/\gamma} = r$$



Application II: Contrast Enhancement

- Can be used to enhance dark or bright regions
 - Dark regions $\gamma < 1$
 - Bright regions $\gamma > 1$
- Example on the right
 - Original image (upper-left) is too dark
 - Use $\gamma < 1$

$\gamma = 0.6, 0.4, 0.3$, with $c=1$



Application II: Contrast Enhancement

- When the input image is too bright, it appears washed-out and detail in the bright regions is unclear.
- Example on the right
 - Original image (upper-left) is too bright
 - Use $\gamma > 1$

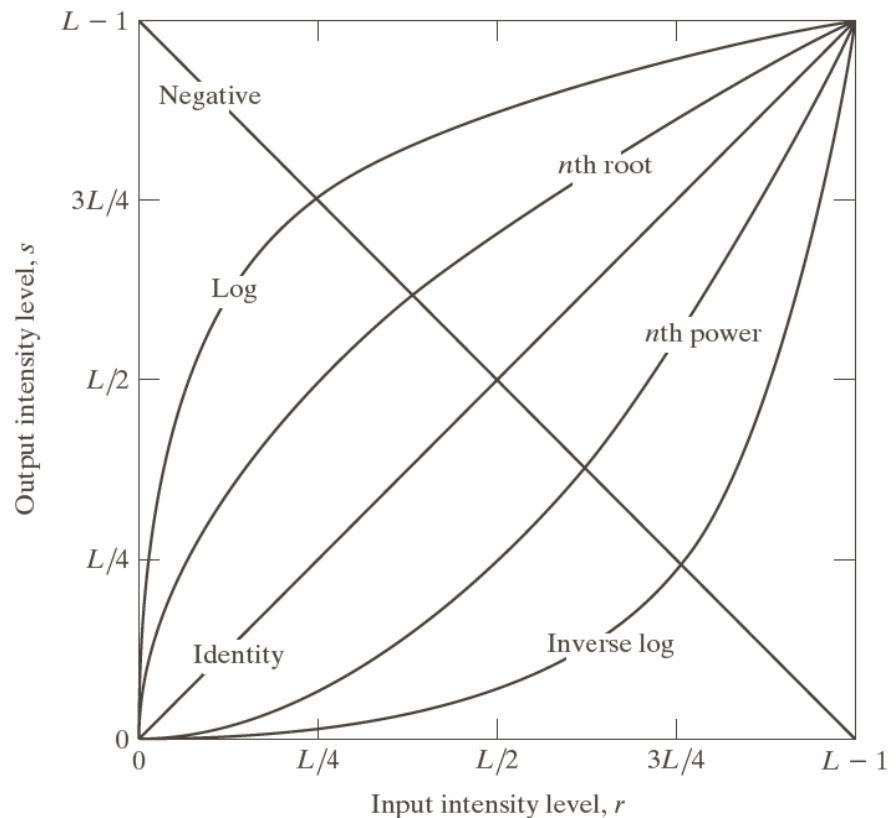
$$\gamma = 3.0, 4.0, 5.0$$

$$c = 1$$



Summary

- Identity (not useful)
- Negative
- Log and root functions
 - For dark images (convex curves)
- Inverse log and power functions
 - For bright images (concave curves)



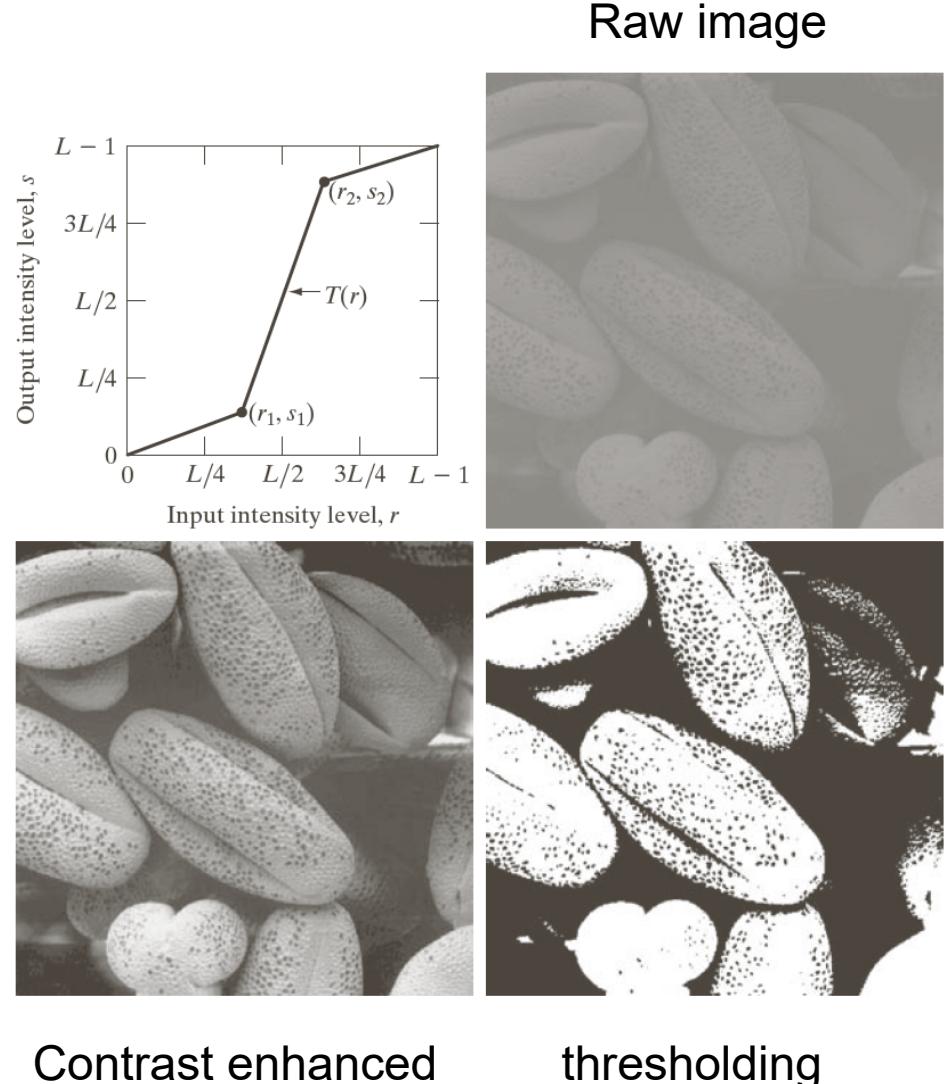
Contrast Stretching

- A process to expand the range of intensity levels so that the contrast between different intensity regions are enhanced
- The function must be increasing to prevent intensity artifacts:

$$r_1 \leq r_2 \Rightarrow s_1 \leq s_2$$

- But may not be strictly increasing:

$$r_1 < r_2 \Rightarrow s_1 < s_2$$

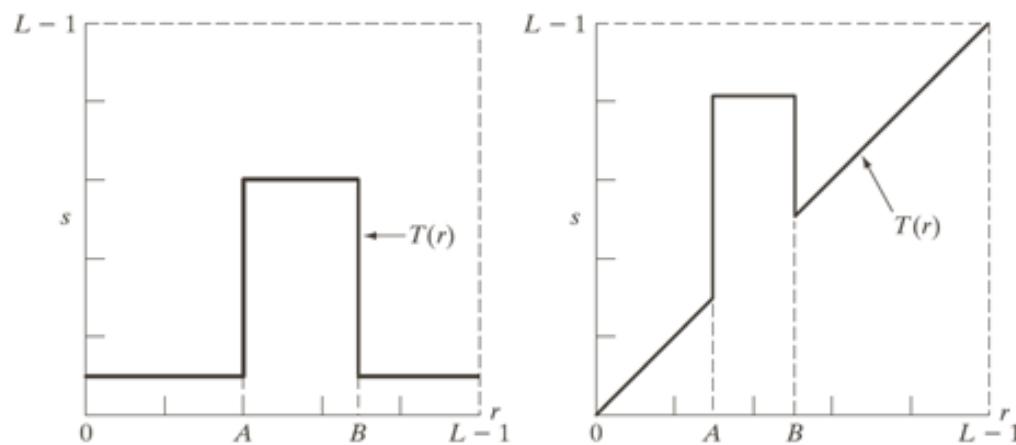


Contrast enhanced

thresholding

Intensity-Level Slicing

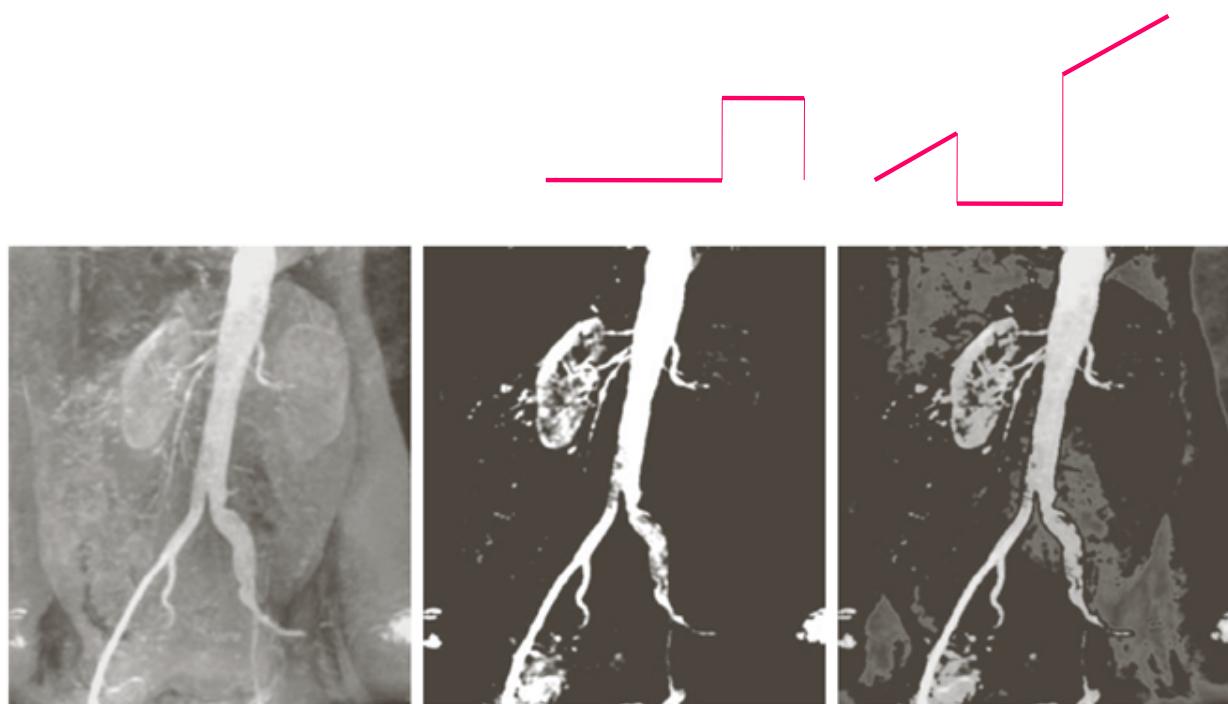
- Features of interest are in a certain intensity range
- We want to highlight the features while keeping the rest of the image dark (left) or unchanged (right)



- What if you want to keep the feature's intensities while darkening the non-features?

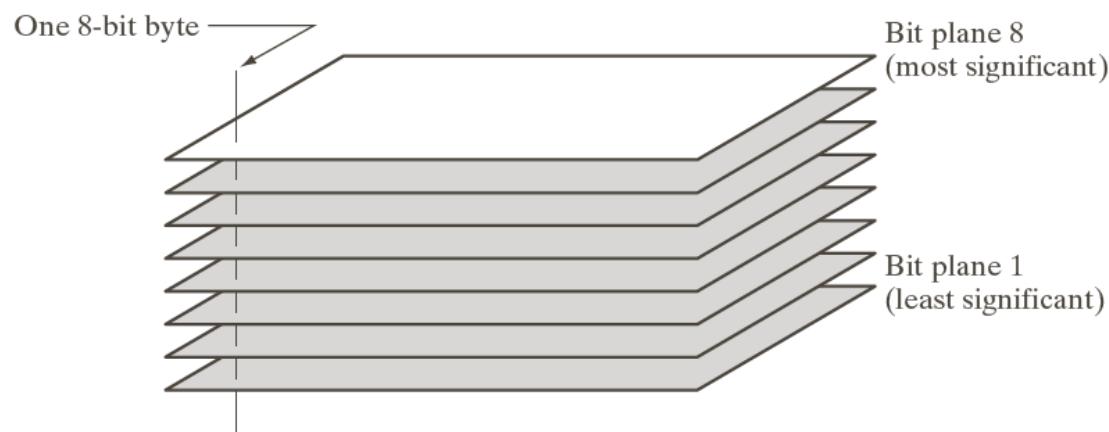
Intensity-Level Slicing: Example

- Left: aortic image near the kidney area
- Middle: the result of highlighting the blood vessels while making the other regions dark.
- Right: the result of keeping dark regions and blood vessels unchanged while darkening the other regions



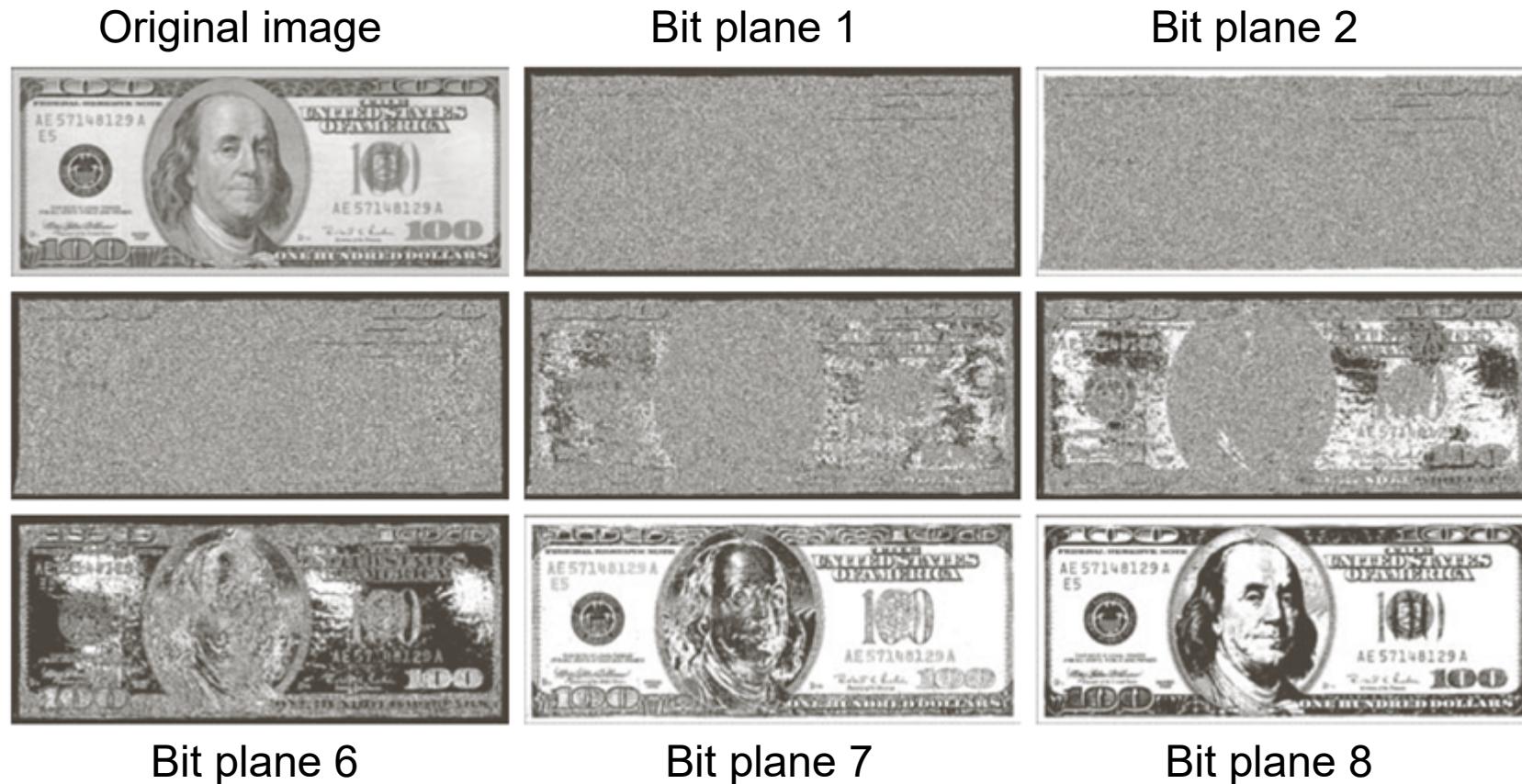
Bit-Plane Slicing

- Consider 8-bit for each pixel
 - The first (rightmost) bit of the intensity value at each pixel is the least significant
 - The last (leftmost) bit of the intensity value at each pixel is the most significant
- What we can do is to decompose each image into eight binary images (only one bit for each pixel)



Bit-Plane Slicing

- Example: the eight (binary) images from the least significant to the most significant



Bit-Plane Slicing

- Can be used for image compression
- Idea: use only the most significant bits to “recover” the original image
- Example:

Using bits 8 and 7



Using bits 8, 7, 6



Using bits 8, 7, 6, 5

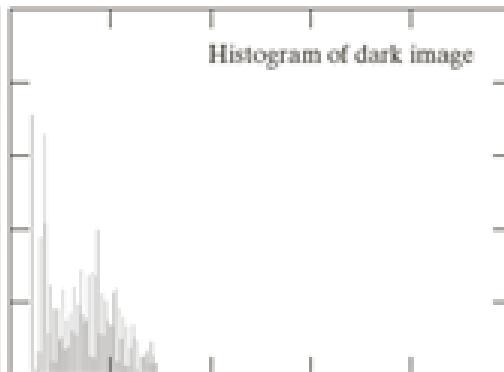


What is Next: Histogram Processing and Spatial Filtering

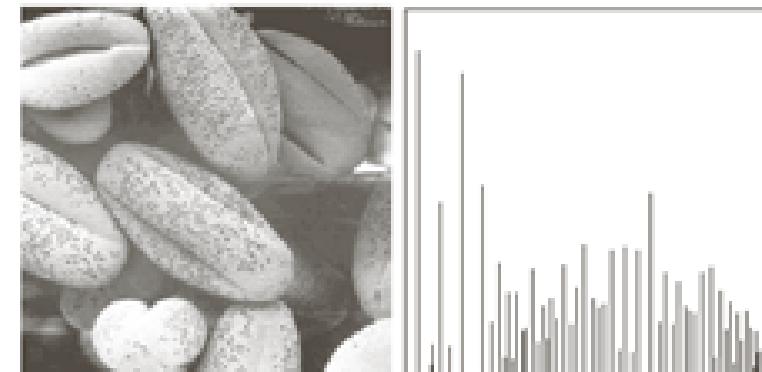
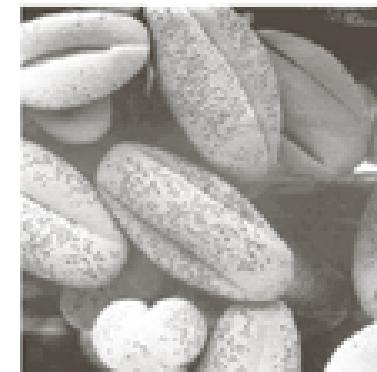
- Image histogram
 - Intensity distribution probability

$$p(z_k) = \frac{n_k}{MN}$$

- Histogram processing, very useful in contrast enhancement and other applications.



Low-contrast image and histogram



High-contrast image and histogram