Digital Image Processing

Lecture 2: Basic Mathematical Operations and Intensity Transformations

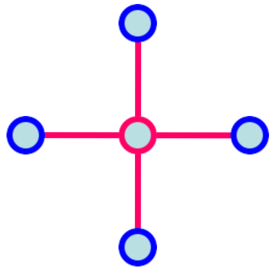
Dr. Behnam Kiani
Department of Computer Science

Spring 2025

Readings: Chapter 2 (sections 2.5 and 2.6), Chapter 3 (sections 3.1 and 3.2)
How to read: Minimum distraction! More Notes!

Table of Content

Neighbors of a Pixel

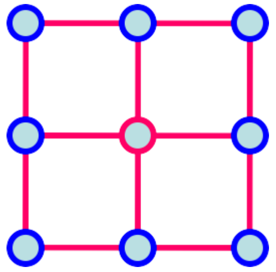


- The 4-neighborhood of pixel **p(x,y)** is the set:

  {(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-adjancent (or connected) if **q** is in $\mathbf{N_4(p)}$ or **p** is in $\mathbf{N_4(q)}$.



- 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-adjancent (or connected) if **q** is in $\mathbf{N_8(p)}$ or **p** is in $\mathbf{N_8(q)}$.

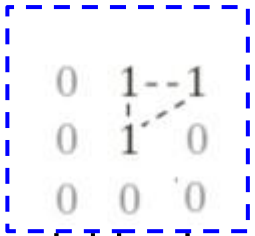
Paths Between Pixels

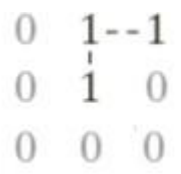
- A digital path from the pixel **p(x,y)** to the pixel **q(s,t)** is a sequence of pixels with coordinates:

$$
\begin{aligned}
&(x_0, y_0), (x_1, y_1) \ldots (x_n, y_n) \\
&\textit{where} \\
&(x, y) = (x_0, y_0),\ \ (s, t) = (x_n, y_n) \\
&\textit{and pixels} \\
&(x_i, y_i)\ \ \textit{and}\ \ (x_{i-1}, y_{i-1})\ \ \textit{are adjacent for}\ 1 \le i \le n
\end{aligned}
$$

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

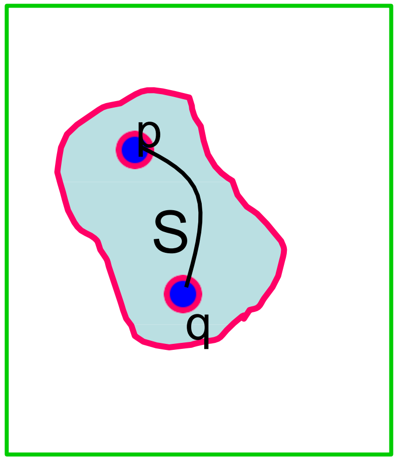
$$
\begin{matrix}
0 & 1 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{matrix}
$$

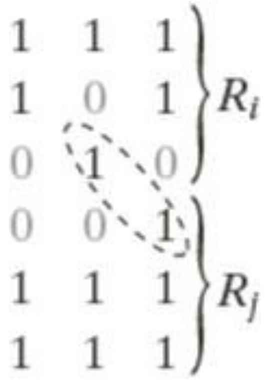

8-neighborhood


4-neighborhood

Paths Between Pixels

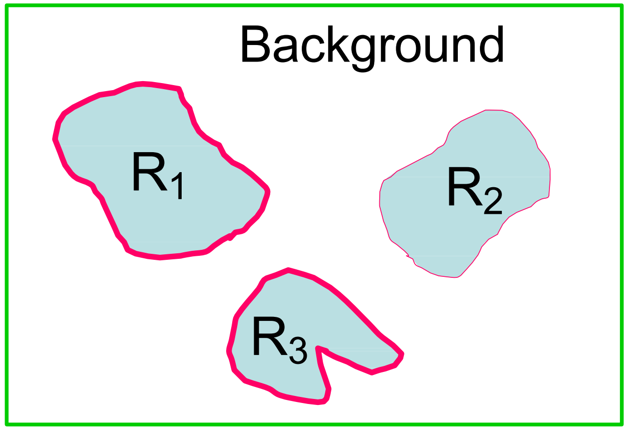
- Assuming **S** a subset of pixels in an image, pixels **p** and **q** are connected in **S** if there exists a path between them consisting entirely of pixels in **S**.
- For any pixel **p** in **S**, the set of pixels that are connected to it in **S** is called a connected component of **S**.
- If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

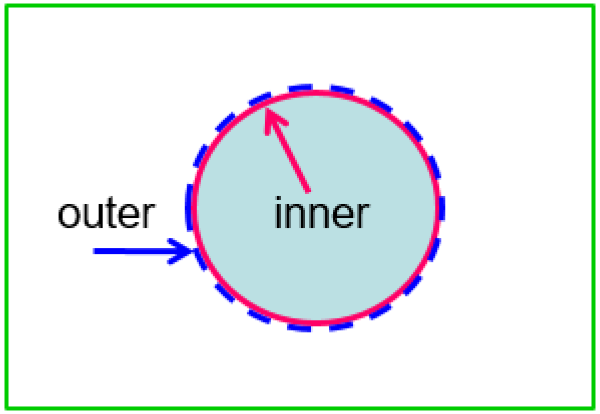




Paths Between Pixels

- Suppose an image contains **K** disjoint regions, **R_k, k=1,2,3,…,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:

$$\begin{aligned}
&\textbf{(a)}\ D(p,q) \geq 0 \quad (D(p,q)=0 \text{ iff } p=q),\\
&\textbf{(b)}\ D(p,q) = D(q,p), \text{ and}\\
&\textbf{(c)}\ D(p,s) \leq D(p,q) + D(q,s).
\end{aligned}$$

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

$$D_e(p,q) = \left[(x-u)^2 + (y-v)^2\right]^{\frac{1}{2}}$$

- The **D$_4$** distance (city-block distance) between **p** and **q** is defined as:

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

$$\begin{array}{ccccc}
 & & 2 & & \\
 & 2 & 1 & 2 & \\
2 & 1 & 0 & 1 & 2\\
 & 2 & 1 & 2 & \\
 & & 2 & &
\end{array}$$

- The **D$_8$** distance (chessboard distance) between **p** and **q** is defined as:

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

$$\begin{array}{ccccc}
2 & 2 & 2 & 2 & 2\\
2 & 1 & 1 & 1 & 2\\
2 & 1 & 0 & 1 & 2\\
2 & 1 & 1 & 1 & 2\\
2 & 2 & 2 & 2 & 2
\end{array}$$

Element-wise vs. Matrix Operations

- An element-wise operation involving one or more images is carried out on a pixel-by-pixel basis.
- The element-wise product of two 2-by-2 images is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

- The matrix product of the same images is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Linear vs. Non-Linear Operations

- Assume a general operator that produces an output image, **g(x,y)**, from a given input image **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:

$$
\begin{aligned}
s(x,y) &= f(x,y) + g(x,y) \\
d(x,y) &= f(x,y) - g(x,y) \\
p(x,y) &= f(x,y) \times g(x,y) \\
v(x,y) &= f(x,y) \div g(x,y)
\end{aligned}
$$

- 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:

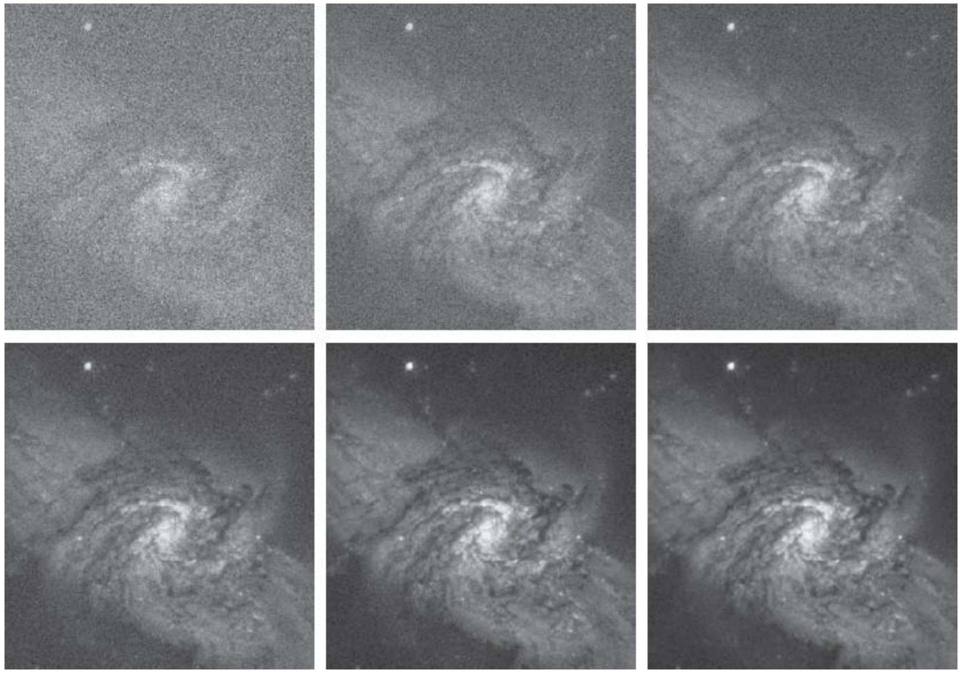
$$\boxed{g(x,y) = f(x,y) + \eta(x,y)}$$

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

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

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

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



Arithmetic Operations: Examples

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

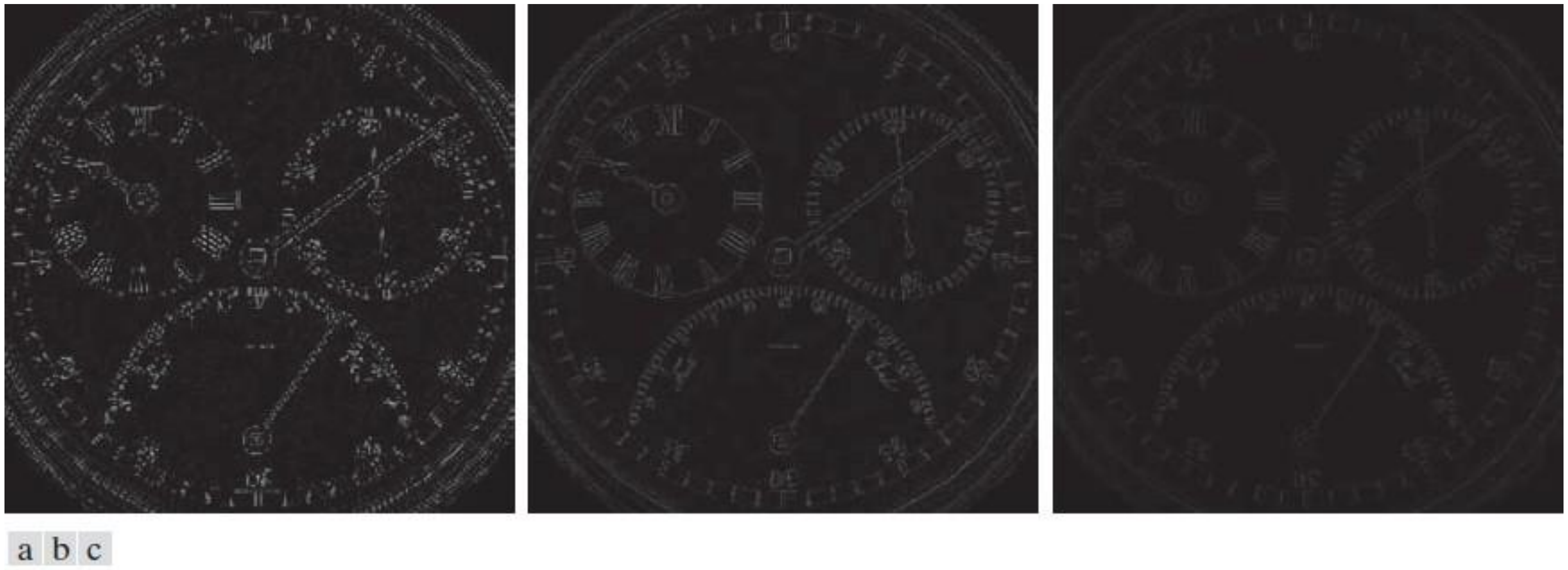


FIGURE 2.31 (a) Difference between the 930 dpi and 72 dpi images in Fig. 2.23. (b) Difference between the 930 dpi and 150 dpi images. (c) Difference between the 930 dpi and 300 dpi images.

Arithmetic Operations: Examples

- Digital Subtraction Angiography (DSA)

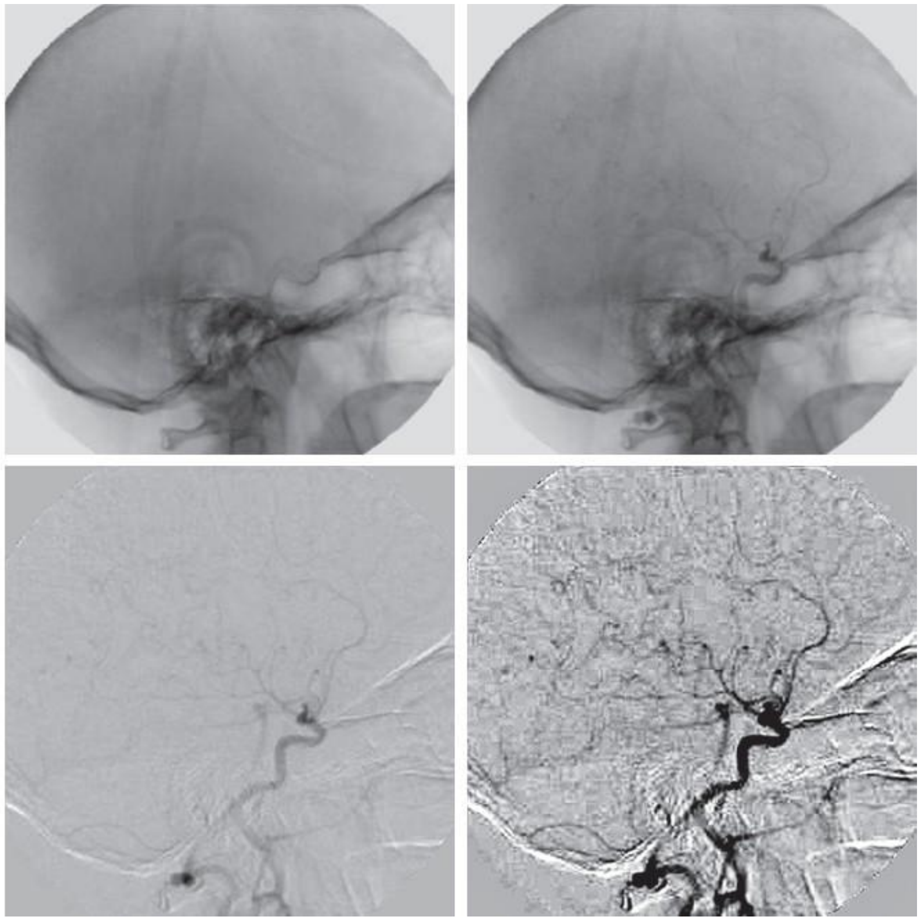


FIGURE 2.32 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of the Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

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)**:

$$\boxed{g(x,y)=f(x,y)h(x,y)}$$

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

$$\boxed{f(x,y)=g(x,y)/h(x,y)}$$

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

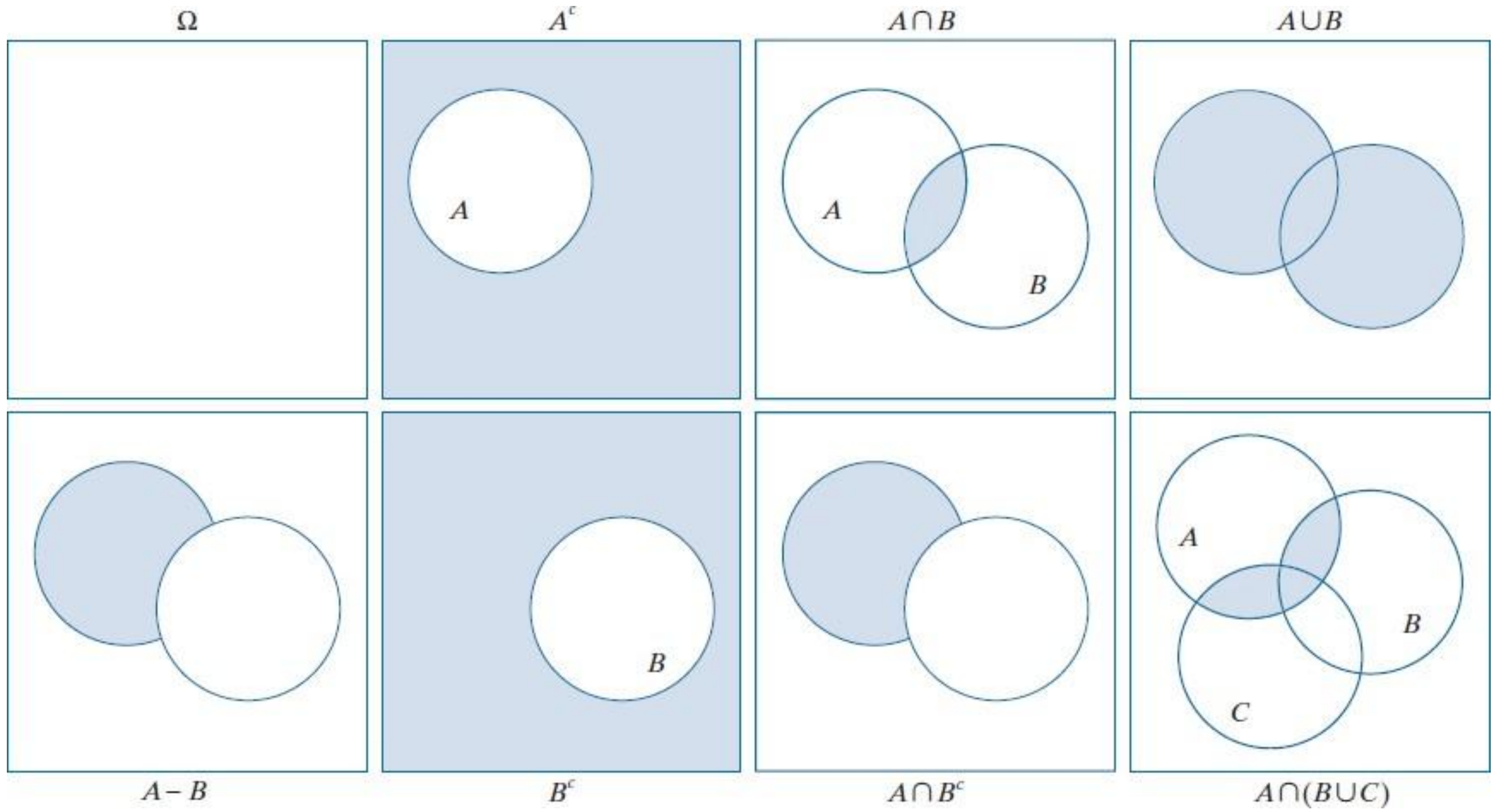
- Since there is a range of values that can be shown using any specific number of bits, it is likely that image processing procedures result in values that are out of this range.
- For example, subtraction of two 8-bit images can range from -255 to +255, which is out of bound for an 8-bit image.
- Clipping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

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

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

Set Operations

- Set: a collection of distinct objects.



Set Operations

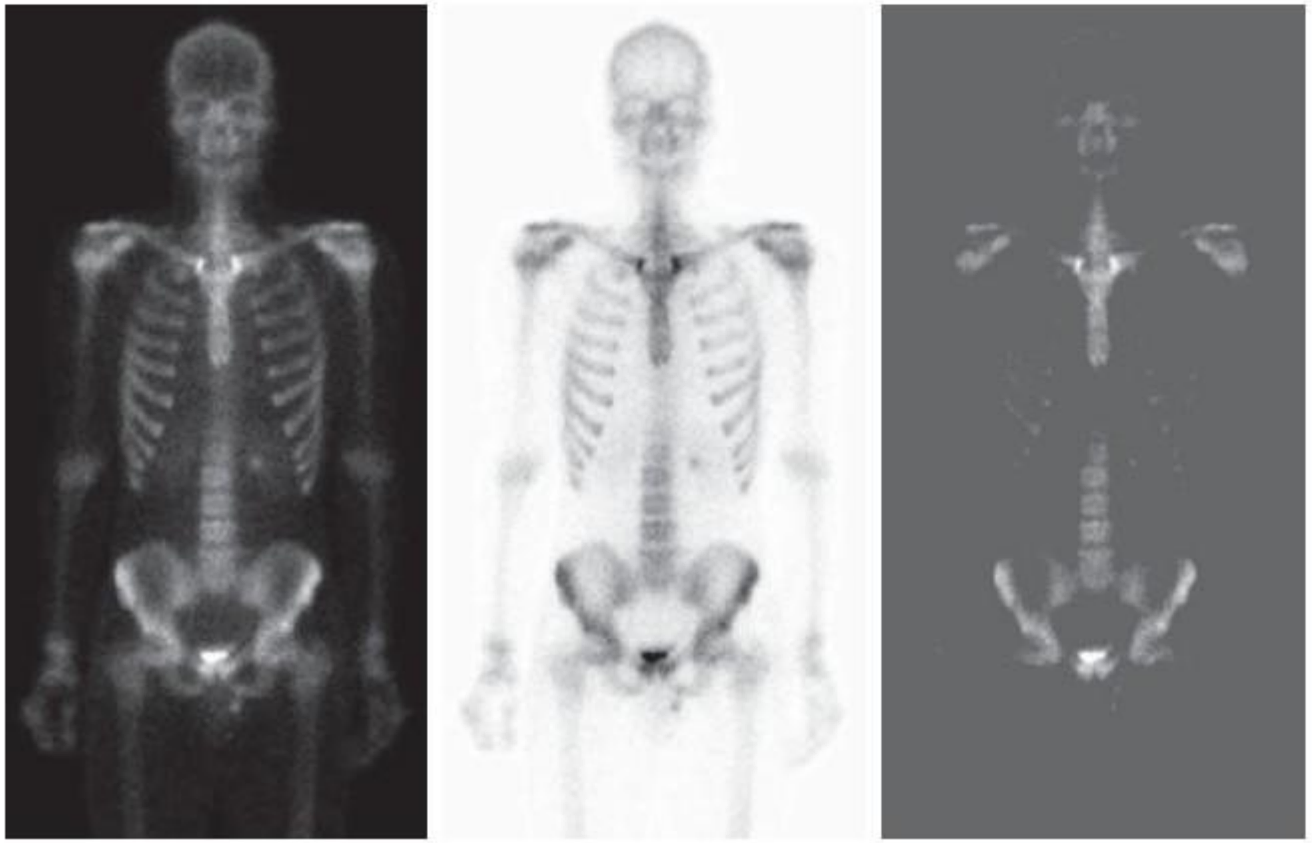
- How set operations can be applied to images? Let’s see an example.
- Complement of an image can be defined using set operations as:

$$A^c = \left\{(x, y, K - z) \middle| (x, y, z) \in A\right\}$$

- **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:

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

Set Operations



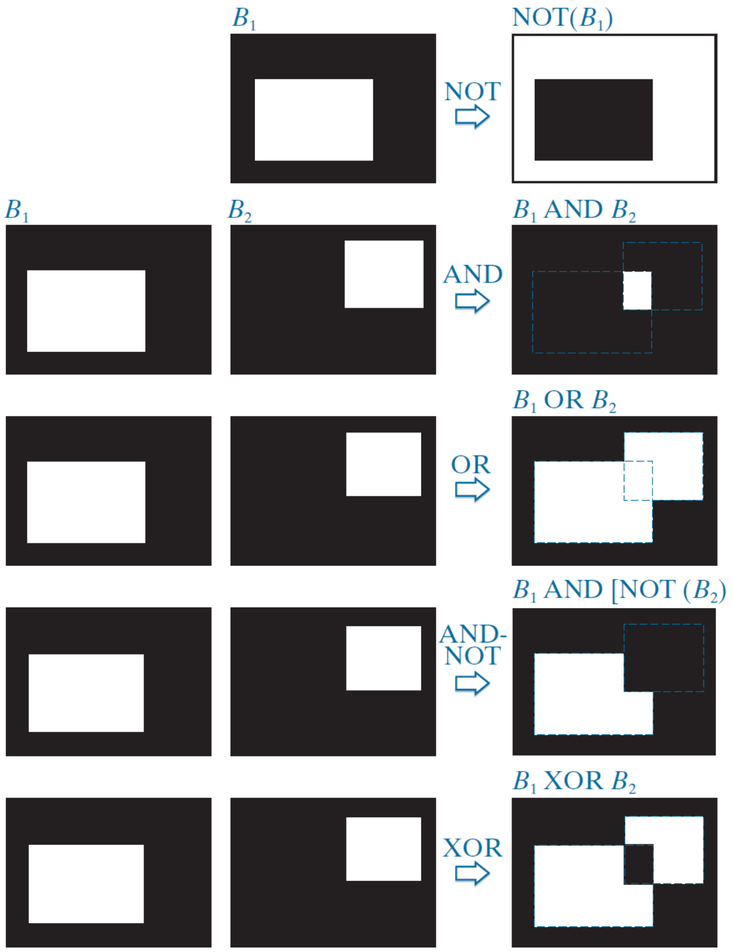
a b c

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 AND b	a OR b	NOT(a)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0



Spatial Operations

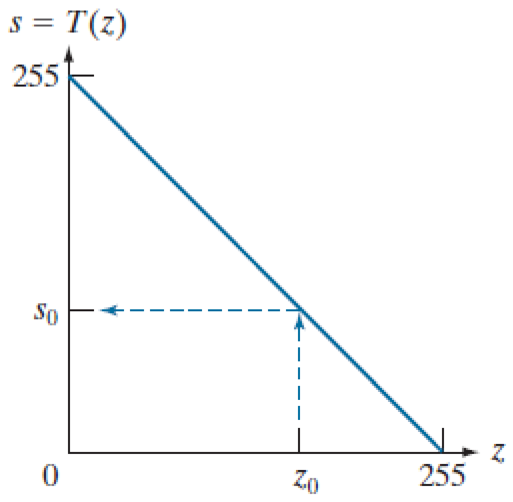
- Spatial operations work directly on the pixels of an image.
- Three types of spatial operations:
  - Single-pixel operations
  - Neighborhood operations
  - Geometric spatial transformations

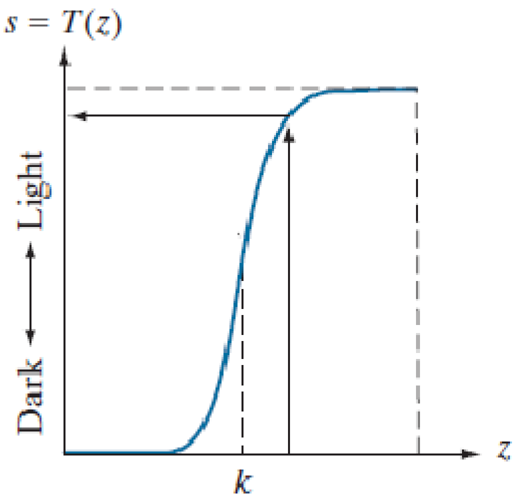
SOs: Single-Pixel Operations

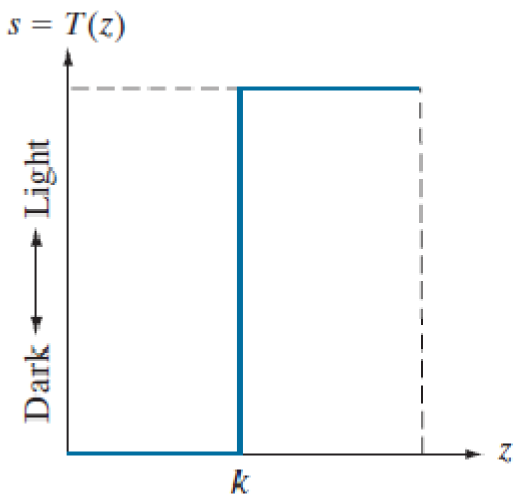
- To alter the intensity of pixels individually using a transformation function **T**, where **z** is the intensity of a pixel in the original image and **s** is the mapped (transformed) intensity of the corresponding pixel in the processed image.

$$\boxed{s = T(z)}$$

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



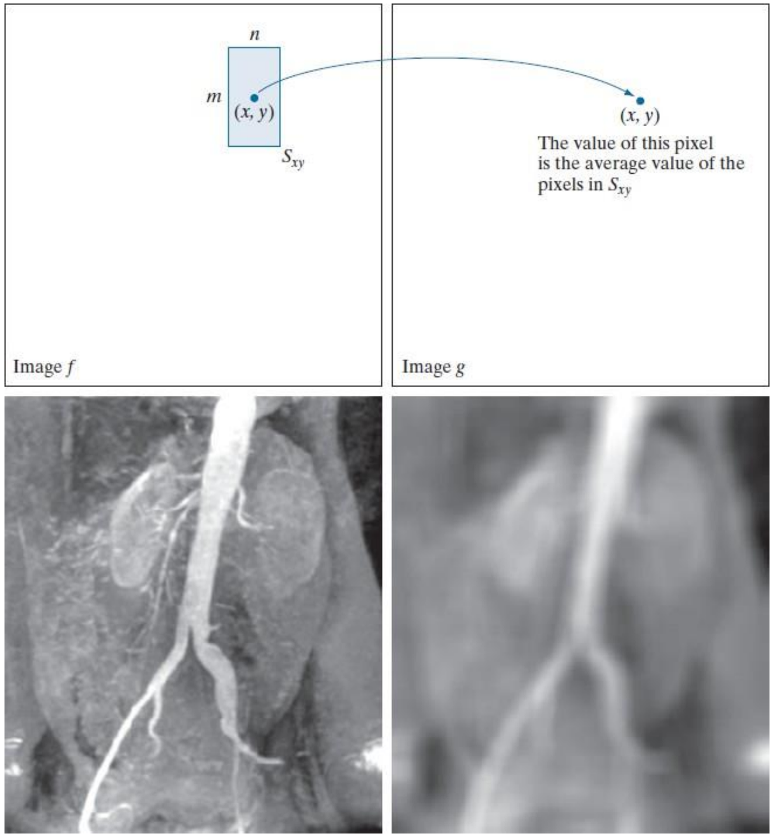




SOs: Neighborhood Operations

- Assume $\mathbf{S}_{xy}$ 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:

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


n
m
(x, y)
S_{xy}
(x, y)
The value of this pixel is the average value of the pixels in S_{xy}
Image f
Image g

SOs: Geometric Spatial Transformations

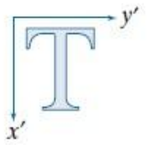
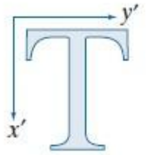
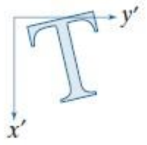
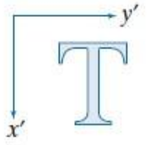
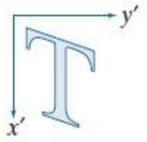
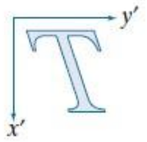
- Geometric operations of digital images consists of two basic operations:
  - Spatial transformation of coordinates;
  - Intensity interpolation for assigning intensity values to spatially transformed pixels.
- ***Affine Transformation*** provides scaling, translation, rotation and shearing.
- Affine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

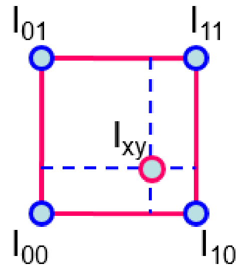
$$A = A_1 * A_2 * A_3 * \ldots * A_n$$

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)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

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)

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

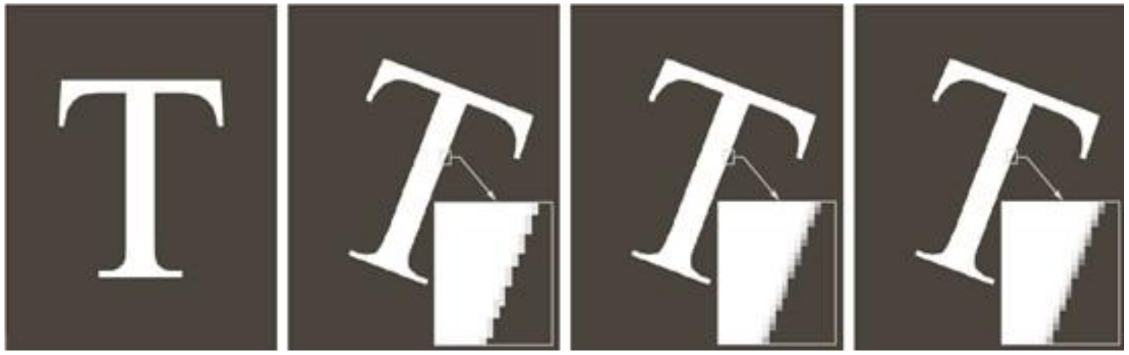


Image Transforms

- Some image processing tasks are best formulated in a ***transform domain***, for example Fourier domain. A general 2D linear transform, **T(u,v)** is formed as:

$$\boxed{T(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\,r(x,y,u,v)}$$

  where **f(x,y)** is the input image, **r(x,y,u,v)** is forward transformation kernel, and the equation is evaluated for **u=0,1,2,…,M-1** and **v=0,1,2,…,N-1**.

- We can recover **f(x,y)** using the inverse transform of **T(u,v)** as:

$$\boxed{f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)\,s(x,y,u,v)}$$

  for **x=0,1,2,…,M-1** and **y=0,1,2,…,N-1**, and **s(x,y,u,v)** is the inverse transformation kernel.

Image Transforms

- The forward transformation kernel is separable if:

$$\boxed{r(x,y,u,v)=r_1(x,u)r_2(y,v)}$$

- The kernel is symmetric if:

$$\boxed{r(x,y,u,v)=r_1(x,y)r_1(y,v)}$$

- The general approach for working in the linear transform domain can be summarized as:

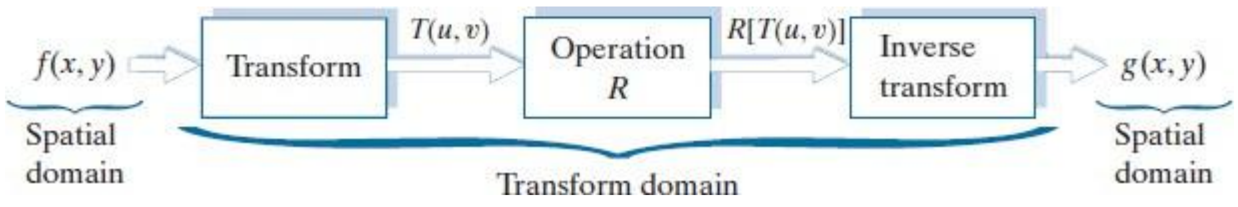


Image Transforms

- The forward and inverse kernels of the Fourier transform are:

$$r(x, y, u, v) = e^{-j2\pi(ux/M + vy/N)}$$

$$s(x, y, u, v) = \frac{1}{MN} e^{j2\pi(ux/M + vy/N)}$$

- The image is corrupted by a sinusoidal interference, which can be seen in the Fourier domain as two bright dots. By applying a mask to the Fourier transform, the interference can be removed.

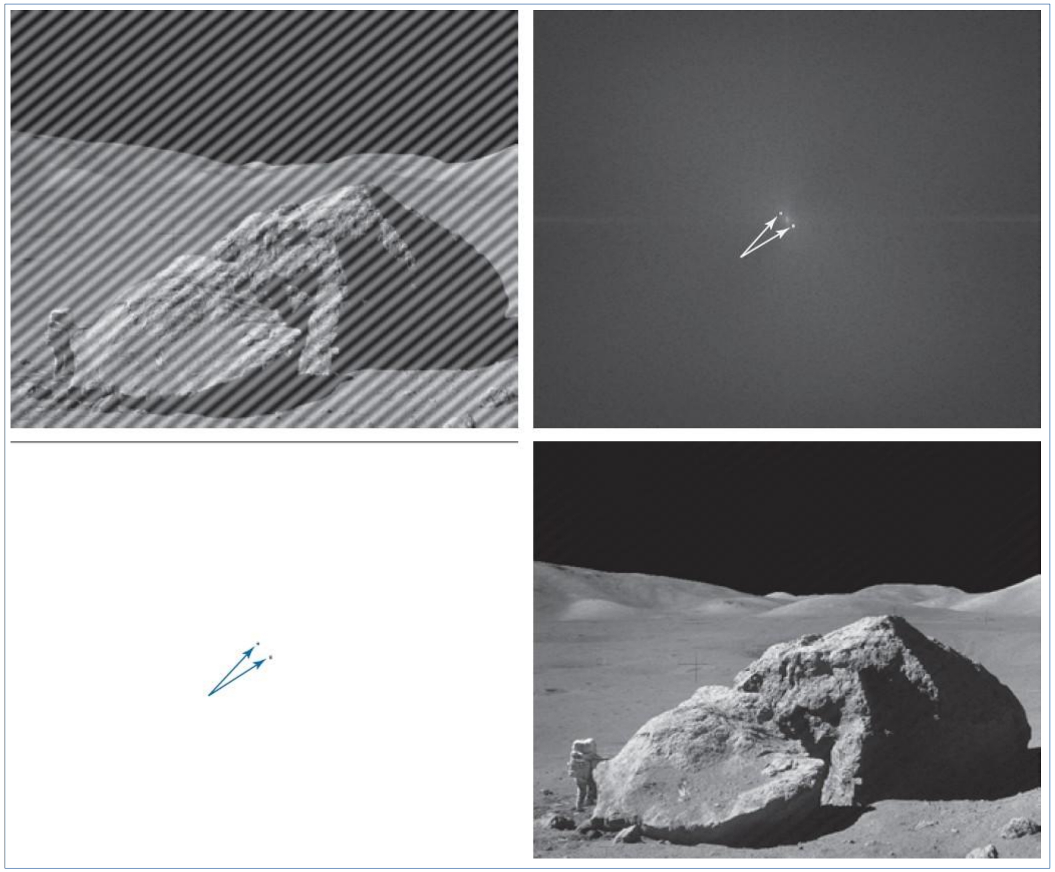


Image Intensities as Random Variables

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

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

where $\mathbf{n_k}$ is the number of times that the intensity level $\mathbf{z_k}$ 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:

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

- The variance of the intensities is:

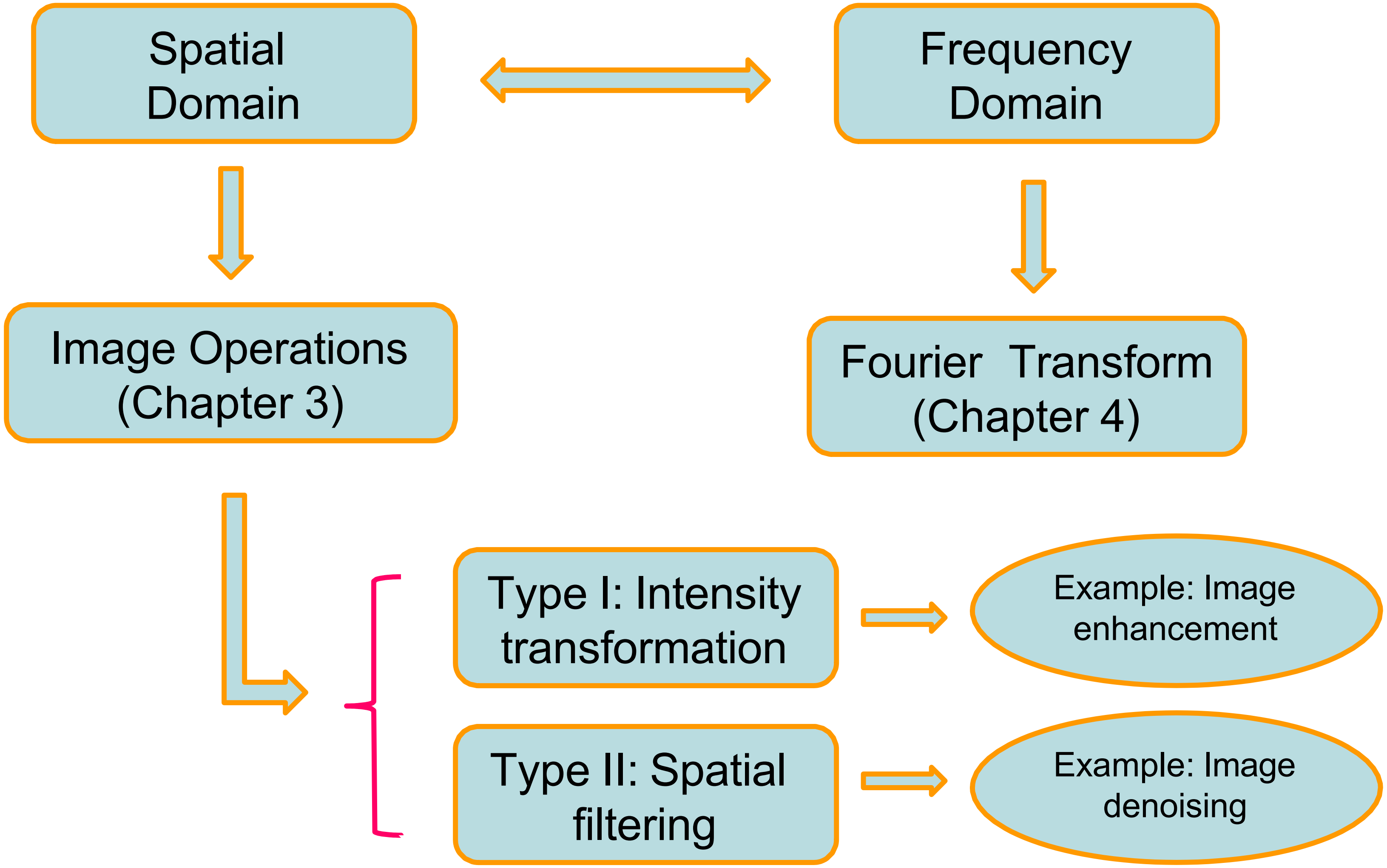
$$\boxed{\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k)}$$

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

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

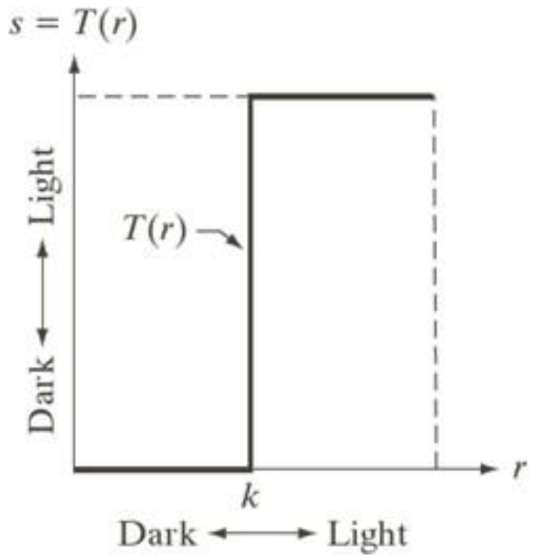
where $\mu_0(z) = 1, \mu_1(z) = 0, \mu_2(z) = \sigma^2$.

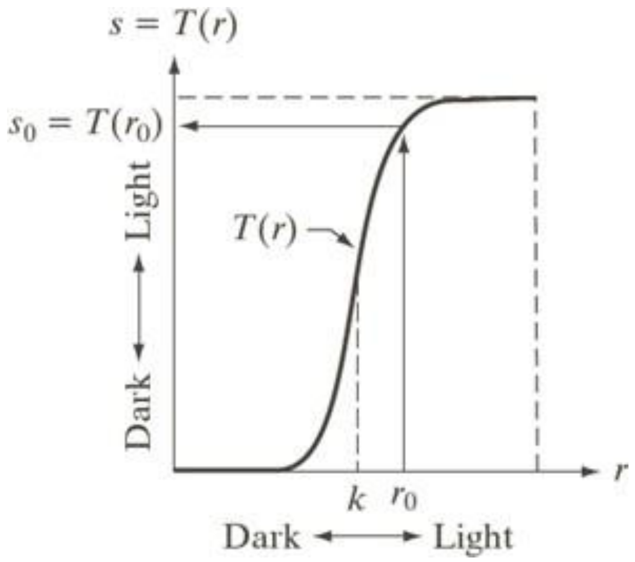
Intensity Transformations and Spatial Filtering


Spatial Domain
Frequency Domain
Image Operations (Chapter 3)
Fourier Transform (Chapter 4)
Type I: Intensity transformation
Type II: Spatial filtering
Example: Image enhancement
Example: Image denoising

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

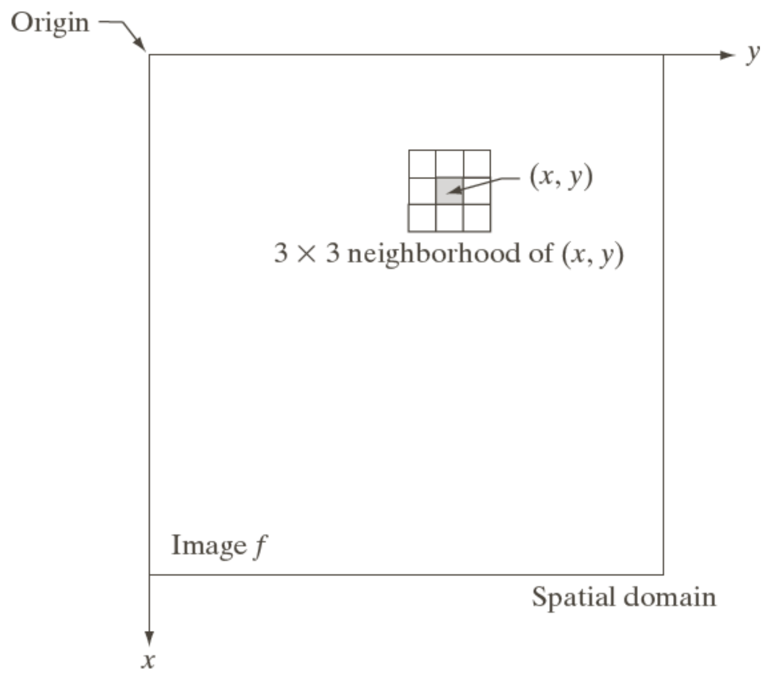




Type II: Intensity Transformations

- 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

$$\boxed{s = T(r)}$$

- In digital images, **r** and **s** are discrete values. We should have a look-up table to define the function **T(*)**.

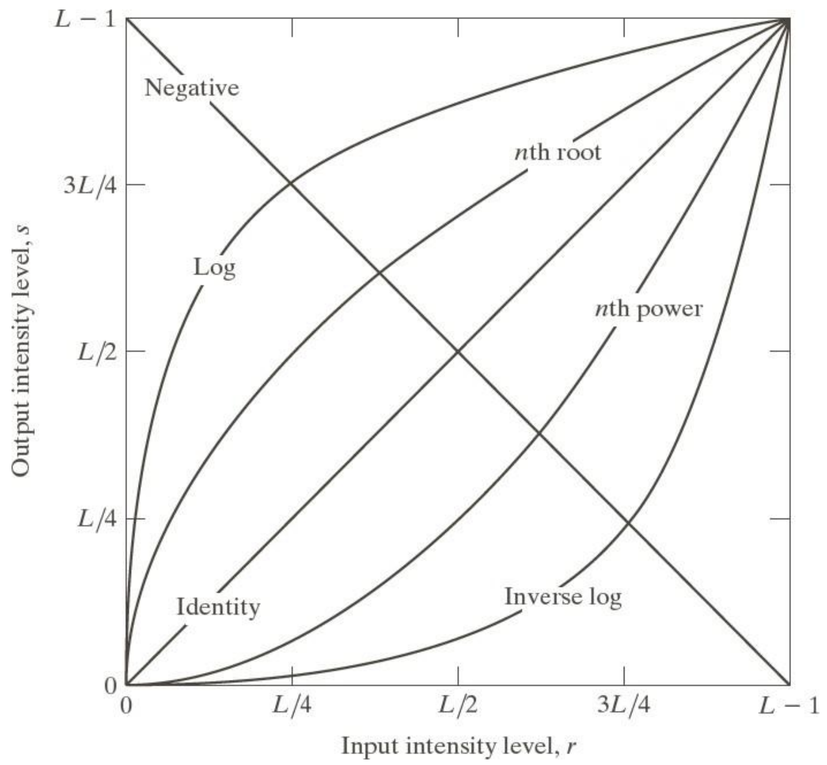
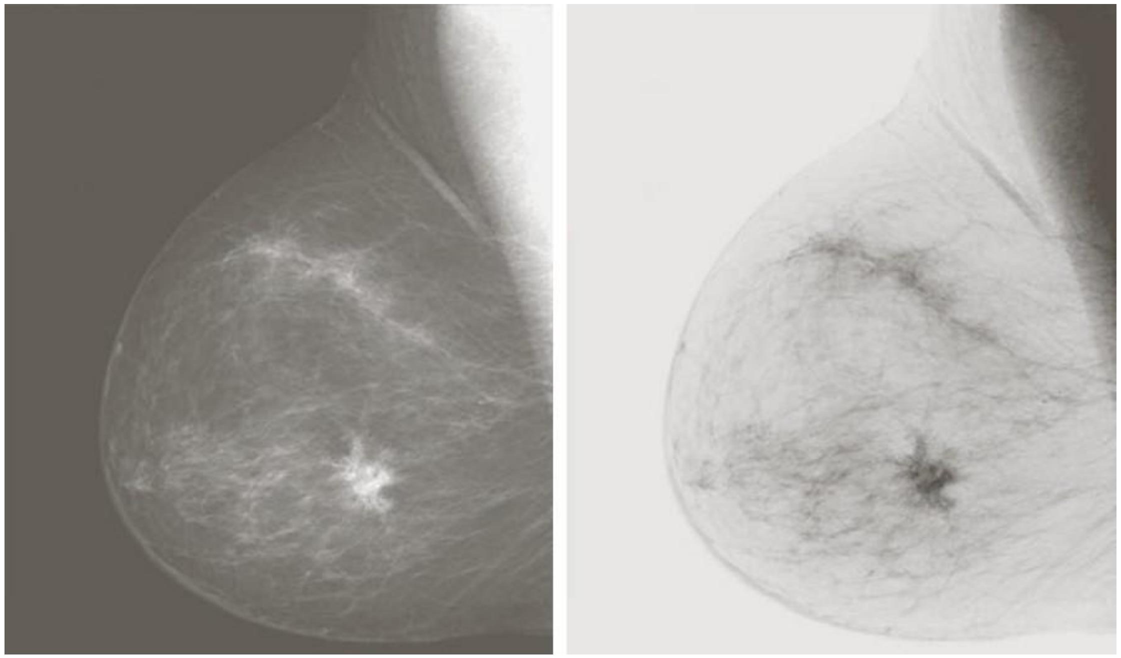


Image Inversion

- Compute the negative of an image:

$$\boxed{s = (L - 1) - r}$$

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

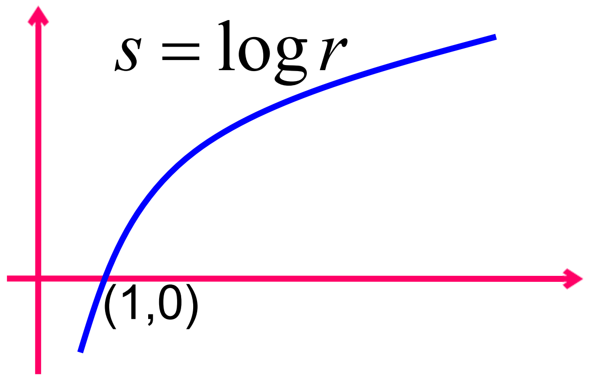

Mammography images

Log Transformation

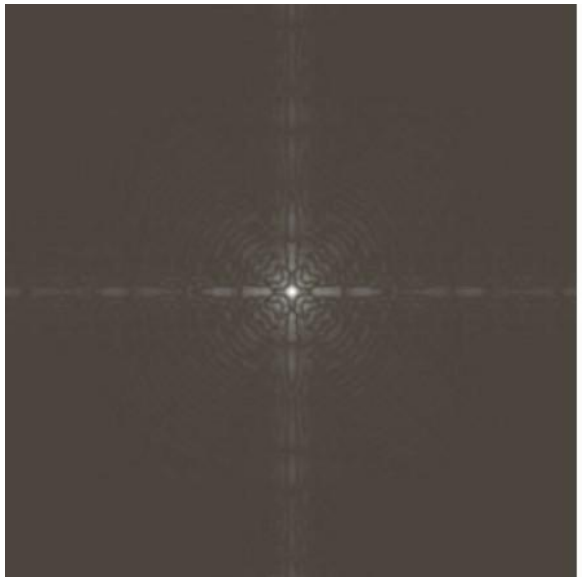
- Transformation function

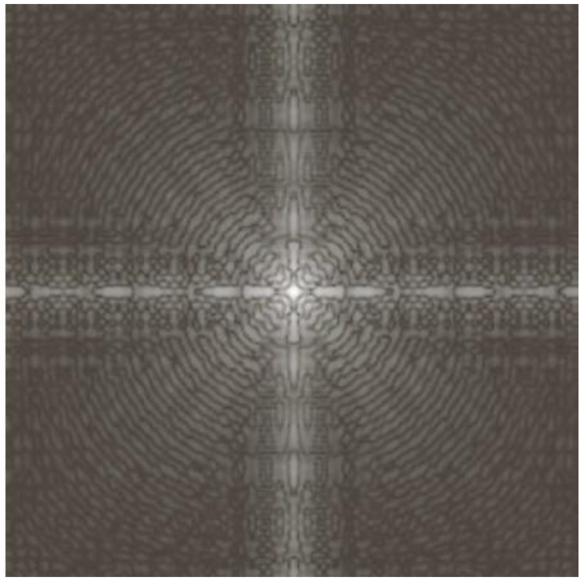
$$\boxed{s = c\,\log(1+r)}$$

  - Enhancing (stretching) dark areas
  - Compressing bright areas



Range:
0 – 1.5X10^6





Range:
0 – 6.2

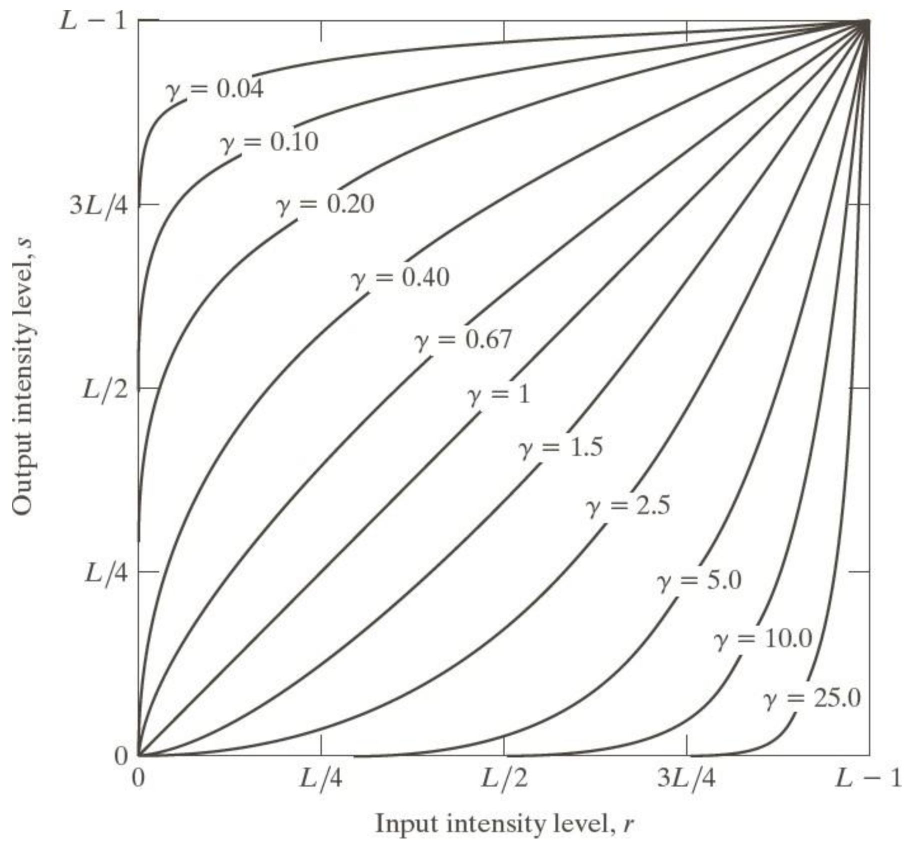
Image in the Fourier domain Enhanced image

Power Law (Gamma) Transformation

- The mathematical form:

$$s = c r^{\gamma} \quad \text{or} \quad s = c\,(r+\epsilon)^{\gamma}$$
for $c > 0$ and $\gamma > 0$
and s and r scaled to $[0,1]$

- If $\gamma \in 1$
  - Enhancing dark areas
- If $\gamma > 1$
  - Enhancing bright areas
- If $\gamma = c = 1$
  - Identity transformation



Application I: Gamma Correction

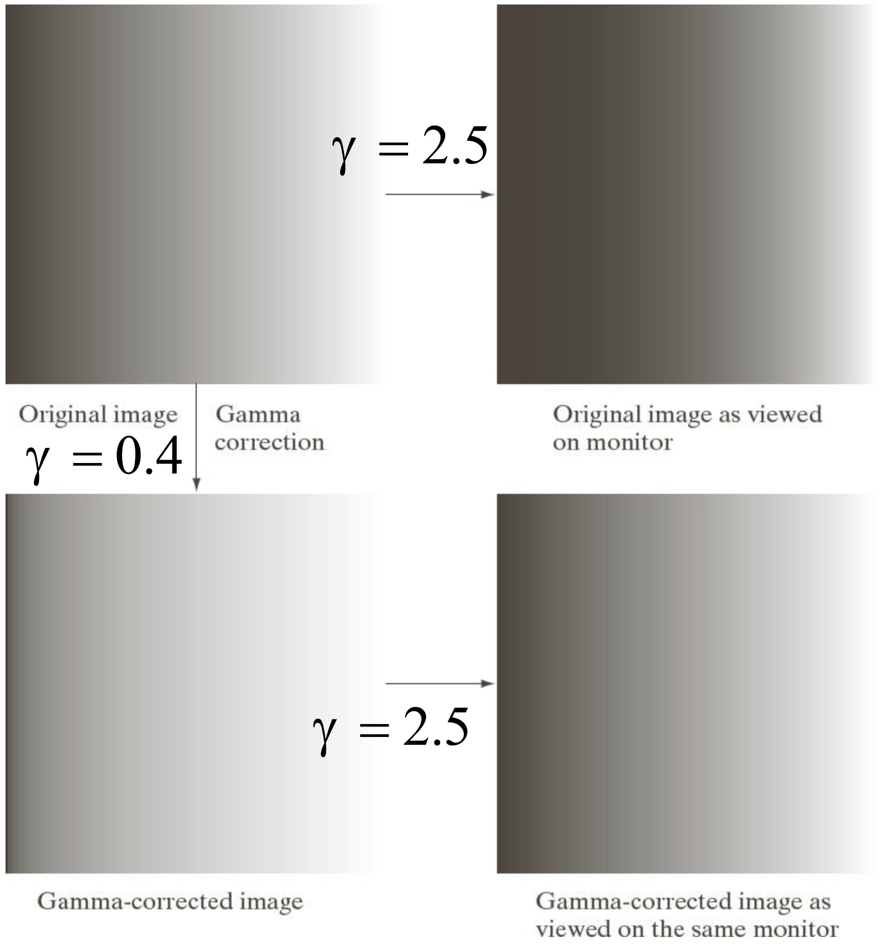
- Many image devices (capturing, printing, and displaying) internally undergo a “gamma” intensity conversion process:

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

- 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

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

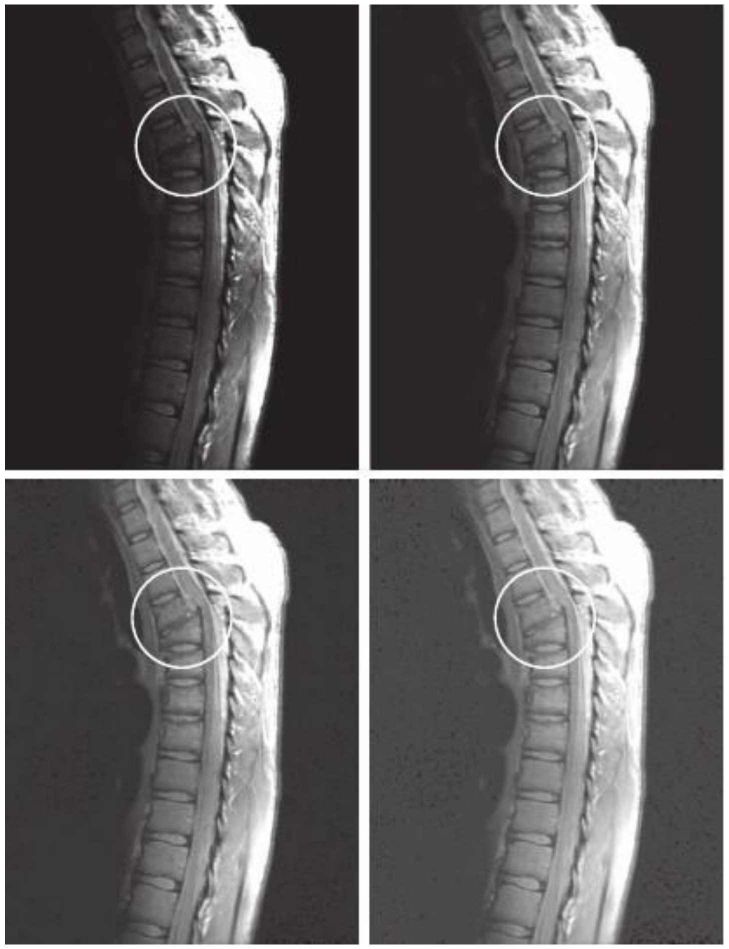


Application II: Contrast Enhancement

- 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 $\gamma < 1$

$$\gamma = 0.6,\quad 0.4,\quad 0.3,\ \text{with}\ c = 1$$



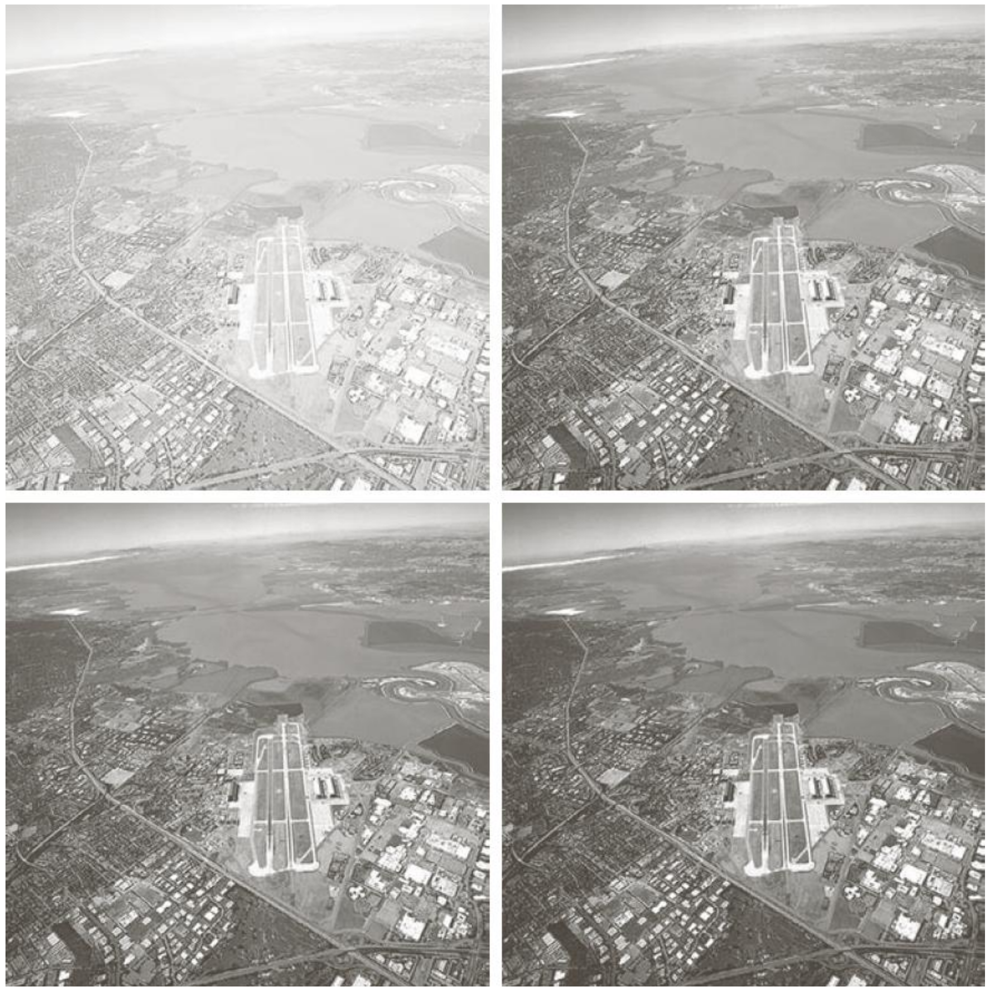
Application II: Contrast Enhancement

- 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 $\gamma > 1$

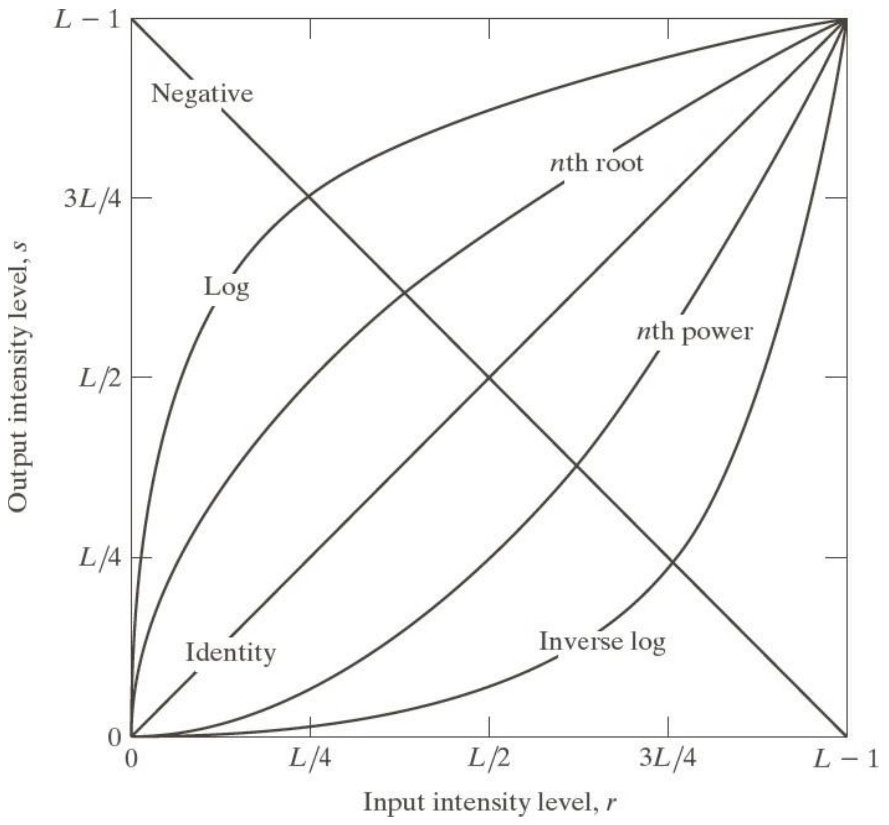
$$\gamma = 3.0, \quad 4.0, \quad 5.0$$

$$c = 1$$



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

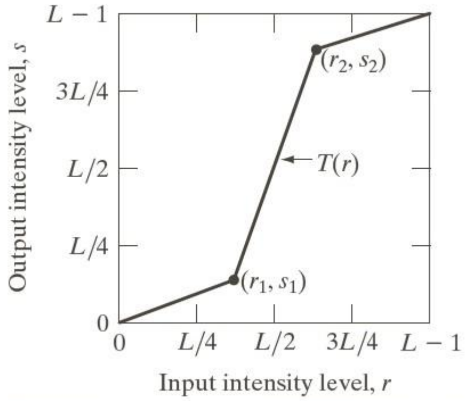
- A process to expand the range of intensity levels so that the contrast between different intensity regions are enhanced
- The function must be increasing to prevent intensity artifacts:

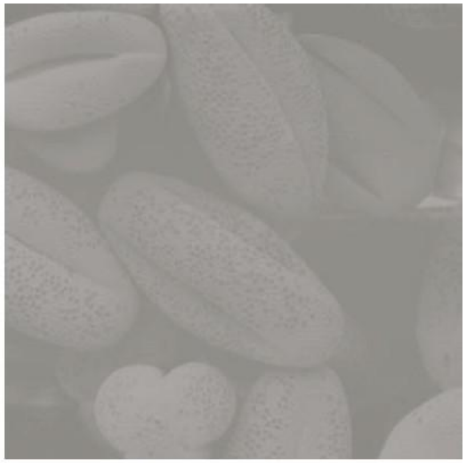
$$r_1 \le r_2 \Rightarrow s_1 \le s_2$$

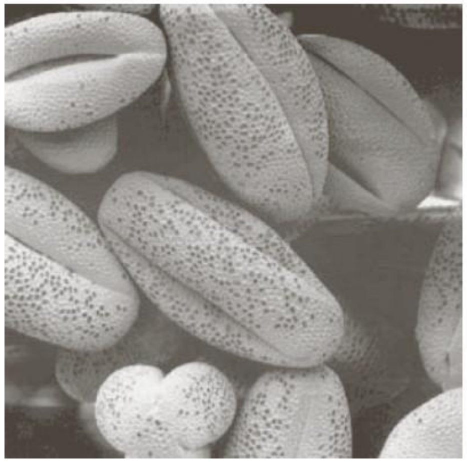
- But may not be strictly increasing:

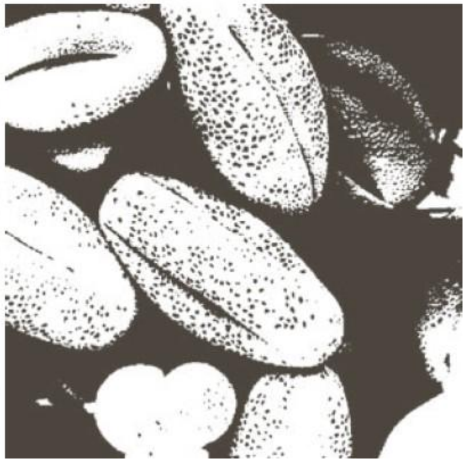
$$r_1 < r_2 \Rightarrow s_1 < s_2$$

Raw image





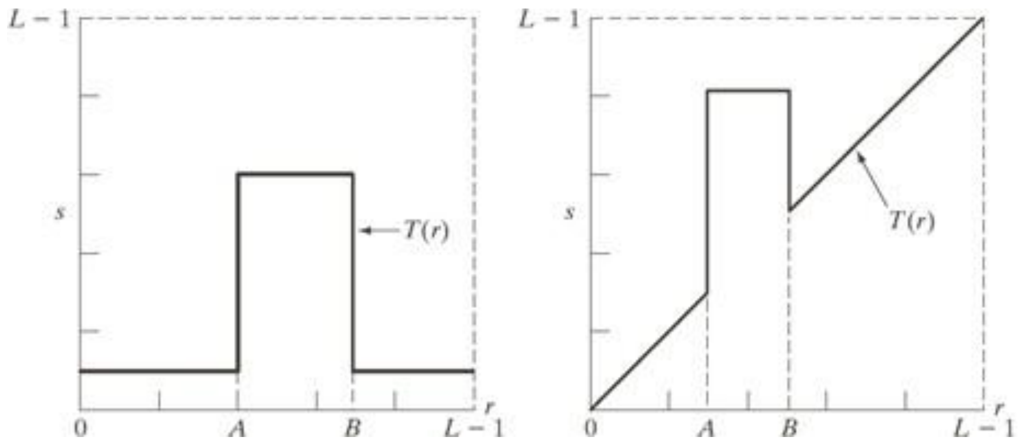




Contrast enhanced thresholding

Intensity-Level Slicing

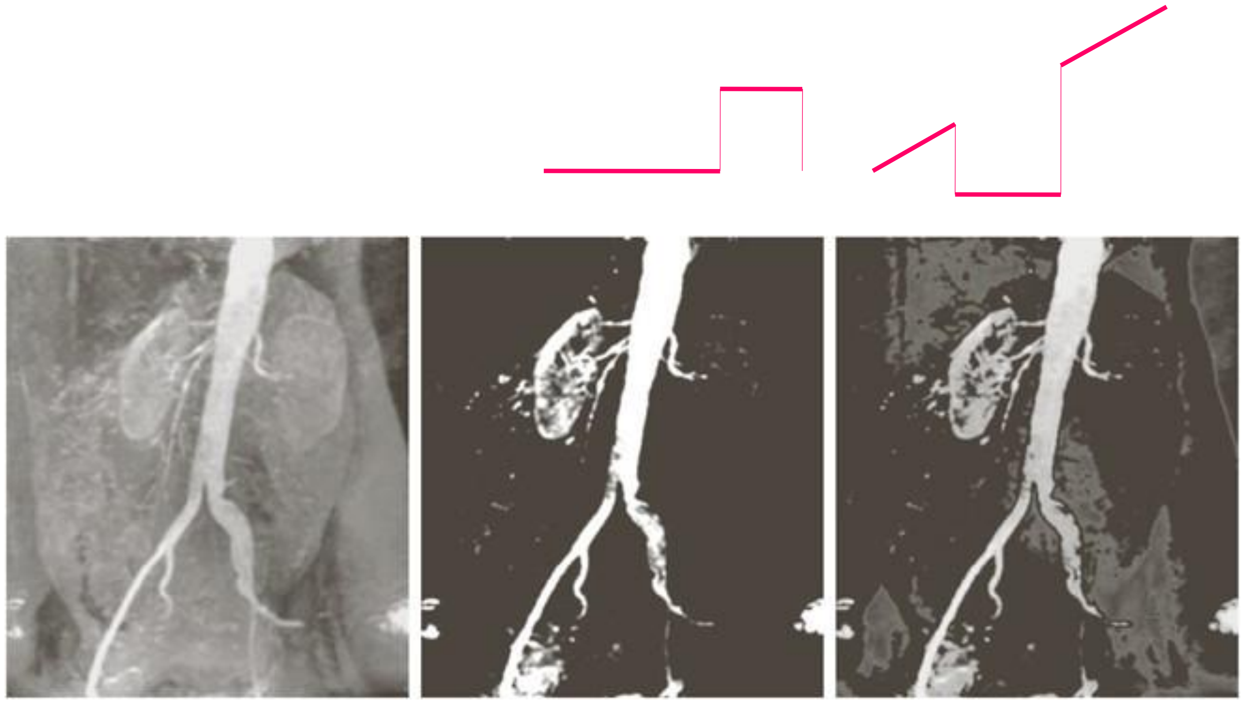
- Features of interest are in a certain intensity range
- We want to highlight the features while keeping the rest of the image dark (left) or unchanged (right)



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

Intensity-Level Slicing: Example

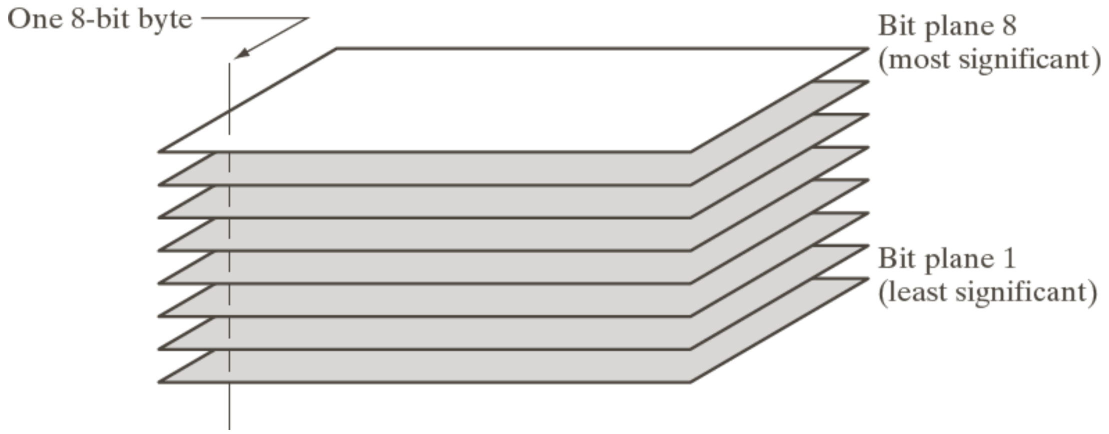
- Left: aortic image near the kidney area
- Middle: the result of highlighting the blood vessels while making the other regions dark.
- Right: the result of keeping dark regions and blood vessels unchanged while darkening the other regions



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

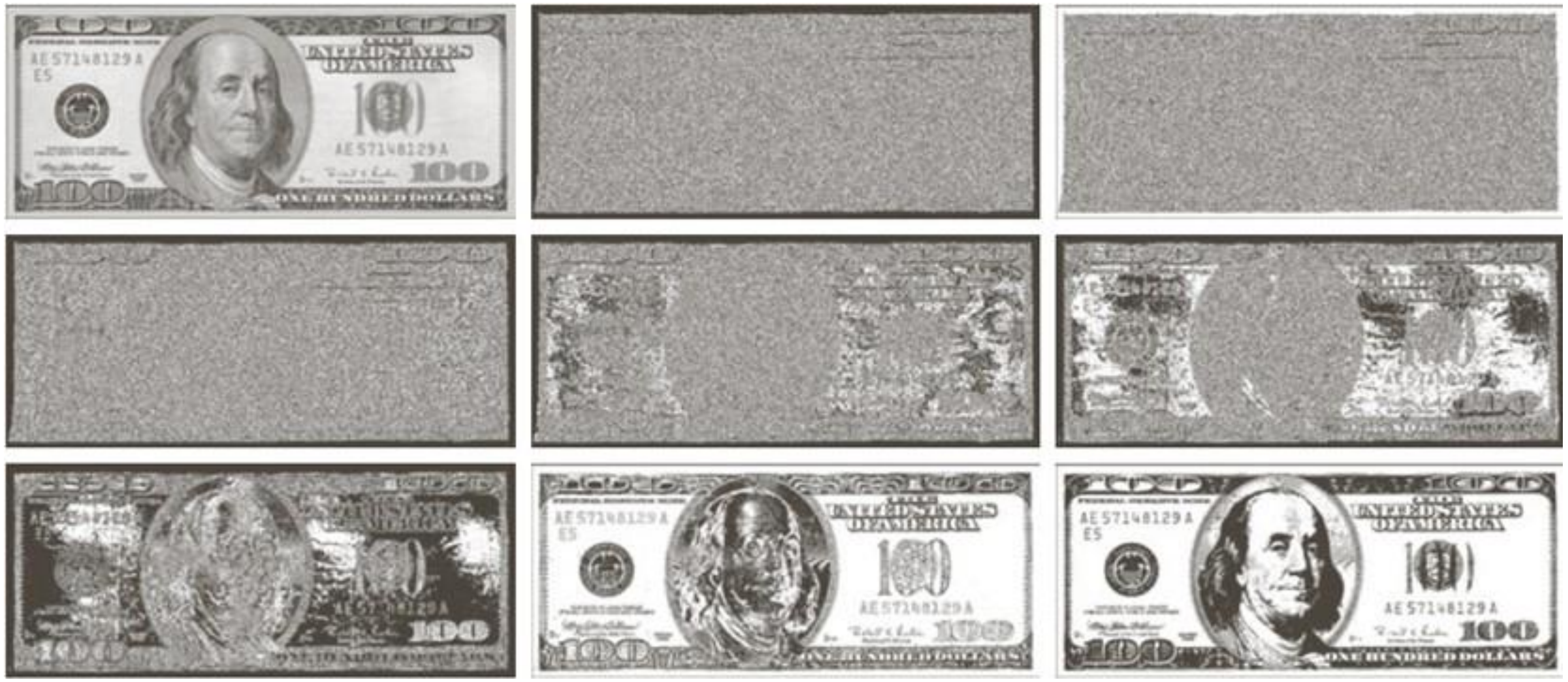
- What we can do is to decompose each image into eight binary images (only one bit for each pixel)



Bit-Plane Slicing

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

Original image Bit plane 1 Bit plane 2

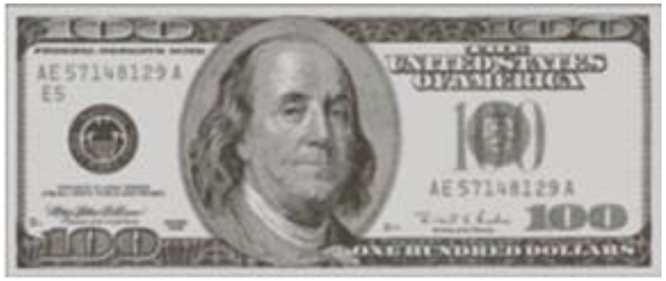


Bit plane 6 Bit plane 7 Bit plane 8

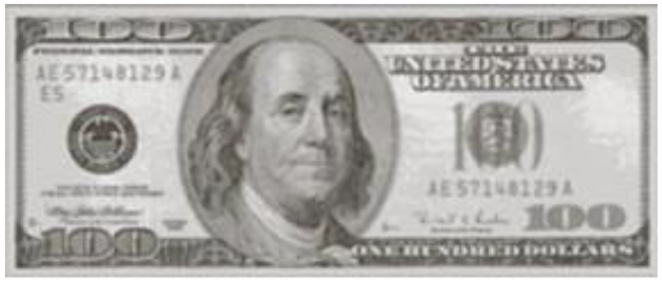
Bit-Plane Slicing

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

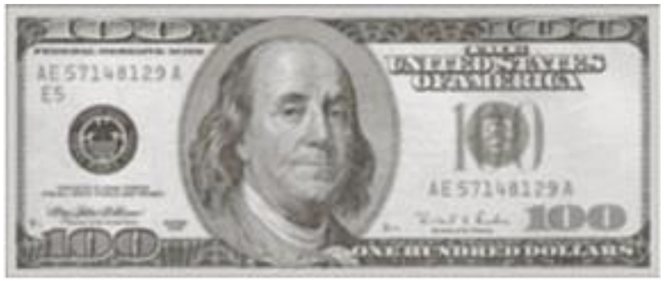
Using bits 8 and 7



Using bits 8, 7, 6



Using bits 8, 7, 6, 5

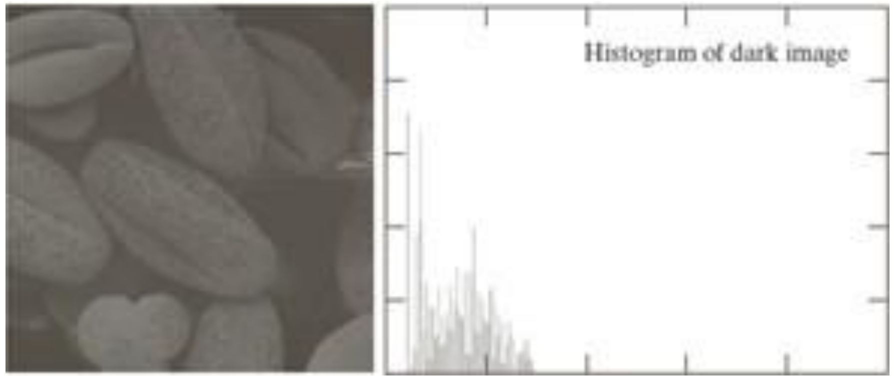


What is Next: Histogram Processing and Spatial Filtering

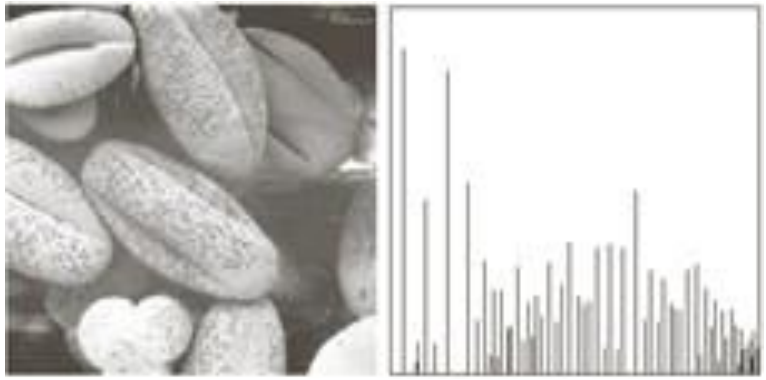
- Image histogram
  - Intensity distribution probability

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

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



Low-contrast image and histogram



High-contrast image and histogram