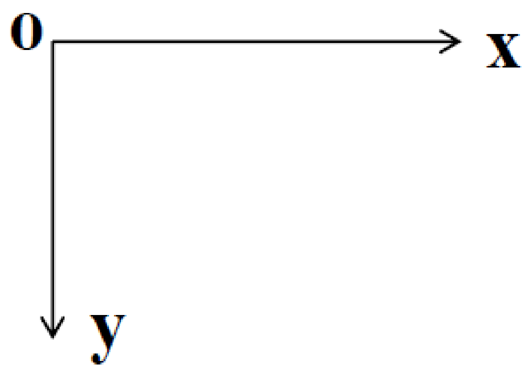
# CSL7320: Digital Image Analysis

Digital Image Processing Fundamentals

# A (2D) Image

**An image = a 2D function *f*(*x*,*y*) where**

- *x* and *y* are spatial coordinates
- *f*(*x*,*y*) is the intensity or gray level




**An digital image:**

- *x*, *y*, and *f*(*x*,*y*) are all finite
- For example $x \in \{1,2,\ldots,M\}$ , $y \in \{1,2,\ldots,N\}$

$$f(x,y) \in \{0,1,2,\ldots,255\}$$

**Digital image processing → processing digital images by means of a digital computer**

**Each element (*x*,*y*) in a digital image is called a pixel (picture element)**

Slide credit: Yan Tong

# A Simple Image Formation Model

$$f(x,y) = i(x,y) \cdot r(x,y)$$

$0 < f(x,y) < \infty$ : **Image (positive and finite)**

**Source:** $0 < i(x,y) < \infty$ : **Illumination component**

**Object:** $0 < r(x,y) < 1$ : **Reflectance/transmission component**

$L_{\min} < f(x,y) < L_{\max}$ **in practice**

**where** $L_{\min} = i_{\min} r_{\min}$ **and** $L_{\max} = i_{\max} r_{\max}$

**i(x,y):** **Sunlight: 10,000 lm/m$^2$ (cloudy), 90,000lm/m$^2$ clear day**
**Office: 1000 lm/m$^2$**

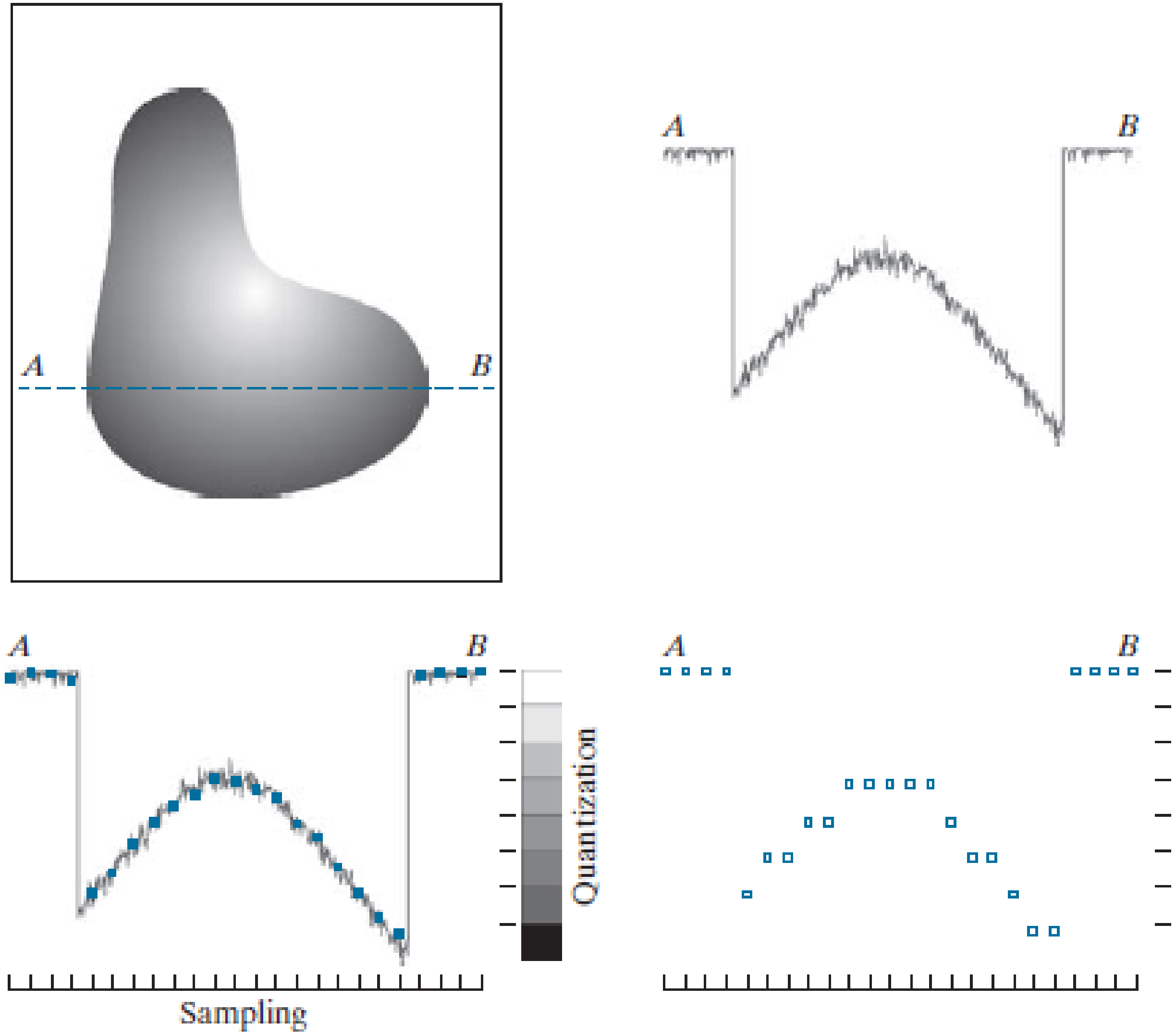
**r(x,y):** **Black velvet 0.01; white pall 0.8; 0.93 snow**

Slide credit: Yan Tong

# Image Sampling and Quantization

a b
c d

**FIGURE 2.16**
(a) Continuous image. (b) A scan line showing intensity variations along line $AB$ in the continuous image. (c) Sampling and quantization. (d) Digital scan line. (The black border in (a) is included for clarity. It is not part of the image).




**Sampling:** Digitizing the coordinate values (usually determined by sensors)

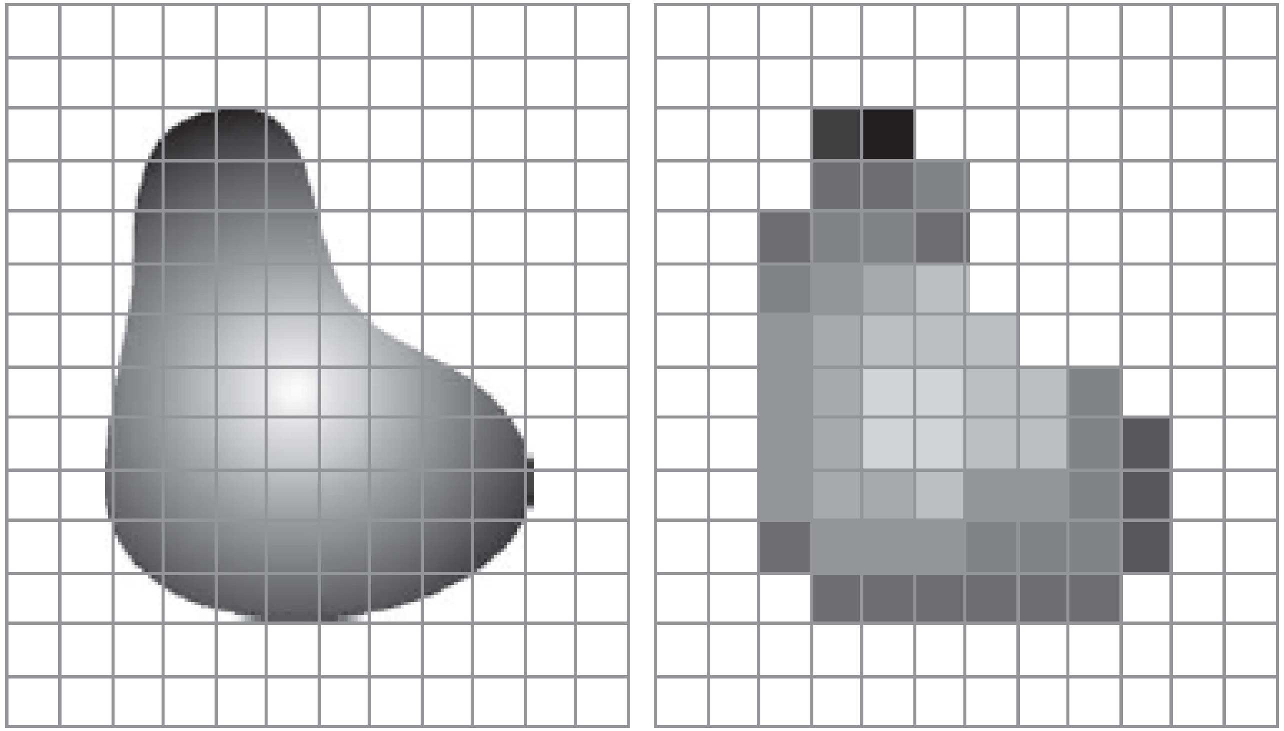
**Quantization:** Digitizing the amplitude values

Slide credit: Yan Tong

# Image Sampling and Quantization

a b

**FIGURE 2.17**
(a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.



continuous image projected onto the plane of a 2-D sensor

after sampling and quantization

# Representing Digital Images

(a)*:* $f(x,y)$, $x = 0, 1, \ldots, M\text{-}1$, $y = 0,1, \ldots, N\text{-}1$

$x$, $y$: spatial coordinates -> spatial domain

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

(b): suitable for visualization

(c): processing and algorithm development

*x: extend downward (rows)*

*y: extend to the right (columns)*

We can also represent a digital image in a traditional matrix form:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

Clearly, $a_{ij} = f(i,j)$

# Representing Digital Images

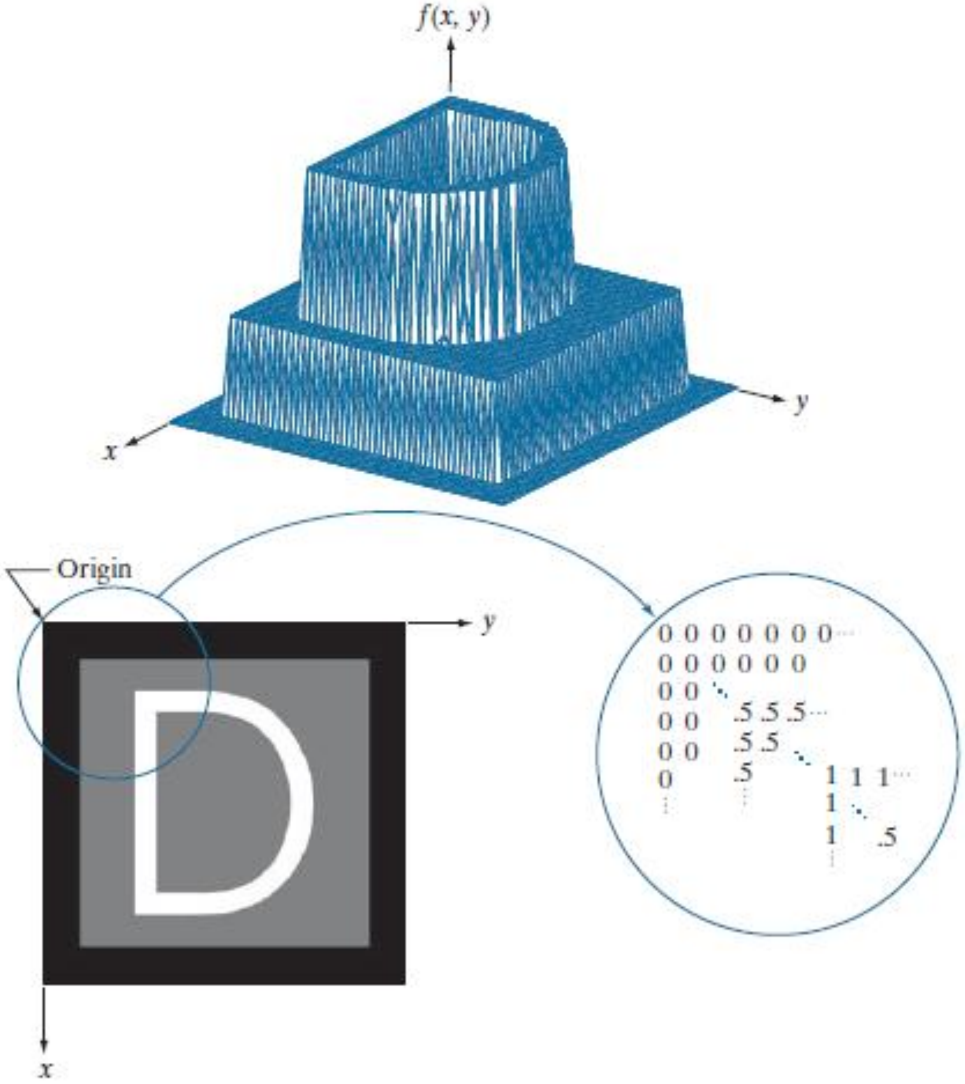
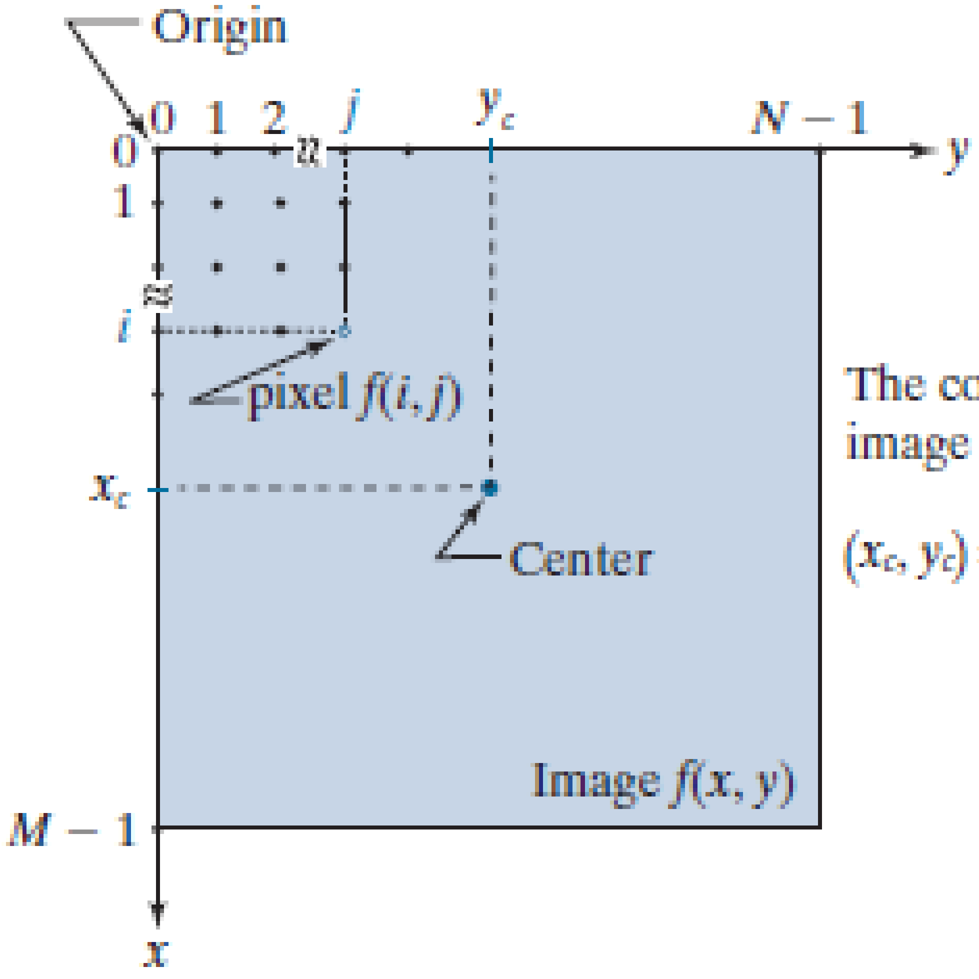



**FIGURE 2.18** (a) Image plotted as a surface. (b) Image displayed as a visual intensity array. (c) Image shown as a 2-D numerical array. (The numbers 0, .5, and 1 represent black, gray, and white, respectively.)

# Representing Digital Images

**FIGURE 2.19**
Coordinate convention used to represent digital images. Because coordinate values are integers, there is a one-to-one correspondence between $x$ and $y$ and the rows ($r$) and columns ($c$) of a matrix.




The coordinates of the image center are

$$(x_c, y_c) = \left(\text{floor}\left(\frac{M}{2}\right), \text{floor}\left(\frac{N}{2}\right)\right)$$

# Representing Digital Images: *Dynamic Range*

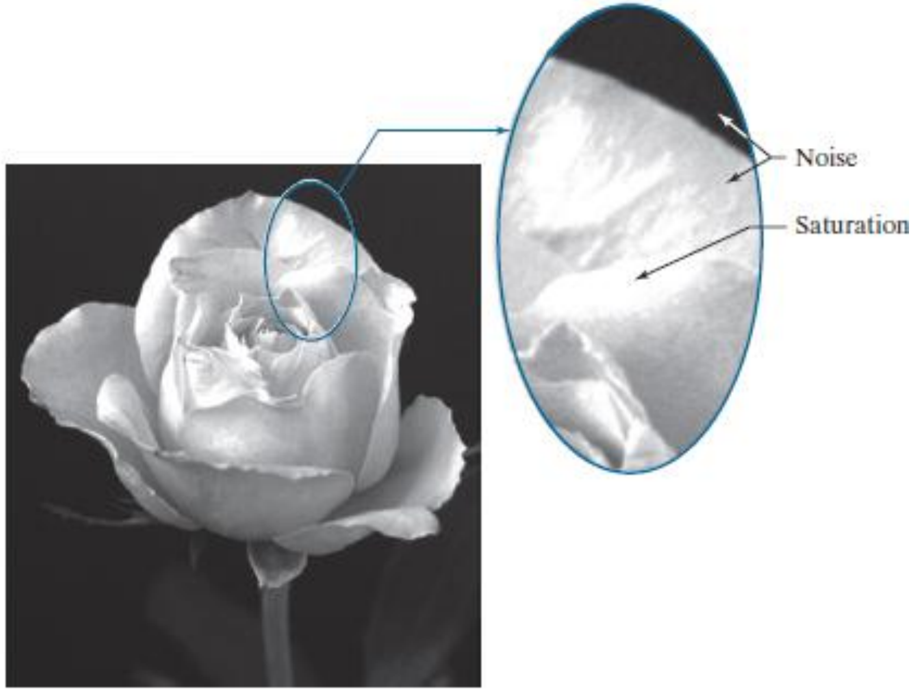
Sometimes, the range of values spanned by the gray scale is referred to as the *dynamic range*, a term used in different ways in different fields.

$$L_{\min} < f(x,y) < L_{\max}$$ **in practice**

**where** $L_{\min} = i_{\min} r_{\min}$ **and** $L_{\max} = i_{\max} r_{\max}$

$$0 \le f(x,y) \le L-1$$ **and** $L = 2^k$

**FIGURE 2.20** An image exhibiting saturation and noise. Saturation is the highest value beyond which all intensity values are clipped (note how the entire saturated area has a high, constant intensity level). Visible noise in this case appears as a grainy texture pattern. The dark background is noisier, but the noise is difficult to see.

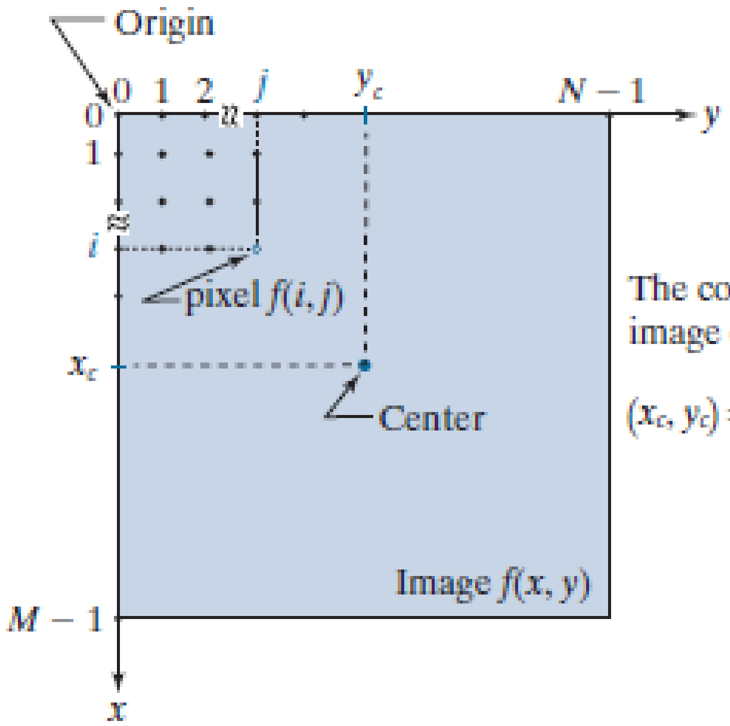



**Dynamic range/contrast ratio:**

the ratio of the maximum detectable intensity level (saturation) to the minimum detectable intensity level (noise)

$$\frac{I_{max}}{I_{min}}$$

# Representing Digital Images




The coordinates of the image center are

$$(x_c, y_c) = \left(\text{floor}\left(\frac{M}{2}\right), \text{floor}\left(\frac{N}{2}\right)\right)$$

$L_{\min} < f(x,y) < L_{\max}$ **in practice**

**where** $L_{\min} = i_{\min} r_{\min}$ **and** $L_{\max} = i_{\max} r_{\max}$

$0 \le f(x,y) \le L-1$ **and** $L = 2^k$

**Number of bits storing the image**

$$\uparrow$$

$$b = M \times N \times k$$

When $M = N$, this equation becomes

$$b = N^2 k$$

# Representing Digital Images

**TABLE 2.1**
Number of storage bits for various values of $N$ and $k$.

| $N/k$ | 1 ($L$ = 2) | 2 ($L$ = 4) | 3 ($L$ = 8) | 4 ($L$ = 16) | 5 ($L$ = 32) | 6 ($L$ = 64) | 7 ($L$ = 128) | 8 ($L$ = 256) |
|---|---|---|---|---|---|---|---|---|
| 32 | 1,024 | 2,048 | 3,072 | 4,096 | 5,120 | 6,144 | 7,168 | 8,192 |
| 64 | 4,096 | 8,192 | 12,288 | 16,384 | 20,480 | 24,576 | 28,672 | 32,768 |
| 128 | 16,384 | 32,768 | 49,152 | 65,536 | 81,920 | 98,304 | 114,688 | 131,072 |
| 256 | 65,536 | 131,072 | 196,608 | 262,144 | 327,680 | 393,216 | 458,752 | 524,288 |
| 512 | 262,144 | 524,288 | 786,432 | 1,048,576 | 1,310,720 | 1,572,864 | 1,835,008 | 2,097,152 |
| 1024 | 1,048,576 | 2,097,152 | 3,145,728 | 4,194,304 | 5,242,880 | 6,291,456 | 7,340,032 | 8,388,608 |
| 2048 | 4,194,304 | 8,388,608 | 12,582,912 | 16,777,216 | 20,971,520 | 25,165,824 | 29,369,128 | 33,554,432 |
| 4096 | 16,777,216 | 33,554,432 | 50,331,648 | 67,108,864 | 83,886,080 | 100,663,296 | 117,440,512 | 134,217,728 |
| 8192 | 67,108,864 | 134,217,728 | 201,326,592 | 268,435,456 | 335,544,320 | 402,653,184 | 469,762,048 | 536,870,912 |

# Spatial Resolution

- **Intuitively**, spatial resolution is a measure of the smallest discernible detail in an image.
- **Quantitatively**, spatial resolution can be stated in several ways
  - # line pairs per unit distance
  - # dots per unit distance
    - Printing and publishing
    - In the U.S., this measure usually is expressed as dots per inch (dpi)

Newspaper → magazines → book

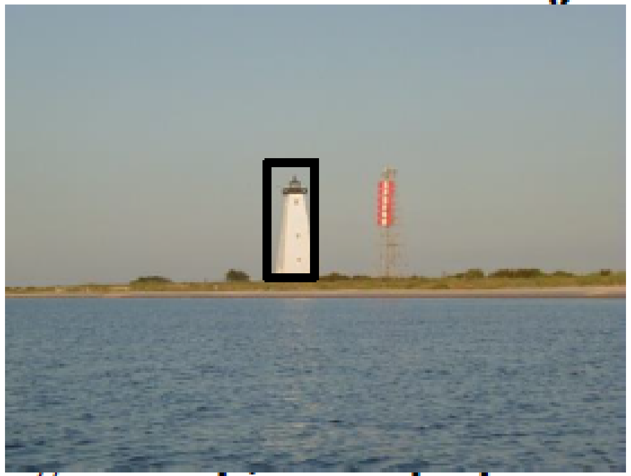
a b
c d

**FIGURE 2.23** Effects of reducing spatial resolution. The images shown are at: (a) 930 dpi, (b) 300 dpi, (c) 150 dpi, and (d) 72 dpi.

# Spatial Resolution

**Large image size itself does not mean high spatial resolution!**

**→ Scene/object size in the image**



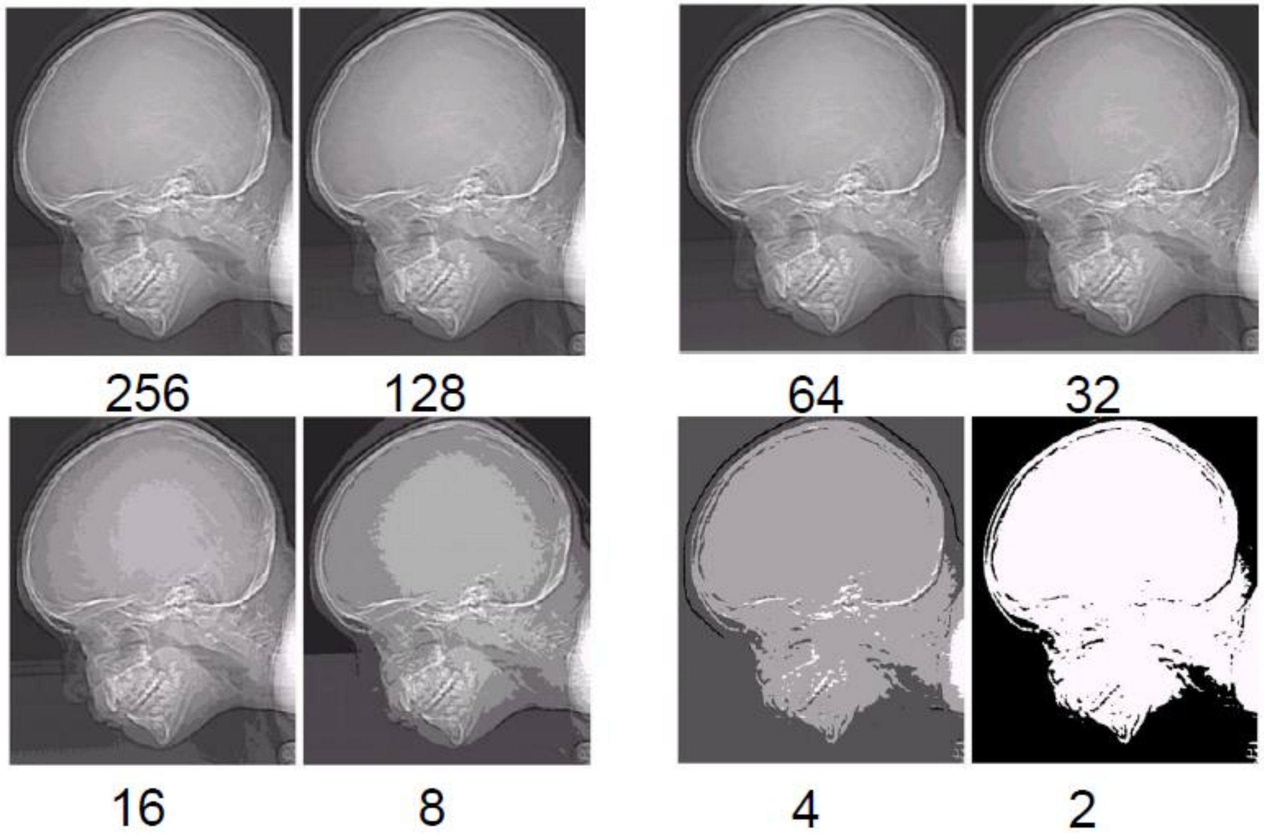
**1280*960**

**http://www.shimanodealer.com/fishing_reports.htm**

Slide credit: Yan Tong

# Intensity Resolution

- Smallest discernible change in intensity levels
- Using the number of levels of intensities
- False contouring (banding) when *k* is small - undersampling




Slide credit: Yan Tong

# Neighbors of A Pixel

$N_4(\mathbf{p})$

| | | |
|---|---|---|
| | (x-1,y) | |
| (x,y-1) | (x,y) | (x,y+1) |
| | (x+1,y) | |

$N_D(\mathbf{p})$

| | | |
|---|---|---|
| (x-1,y-1) | | (x-1,y+1) |
| | (x,y) | |
| (x-1,y+1) | | (x+1,y+1) |

$N_8(\mathbf{p})$

| | | |
|---|---|---|
| (x-1,y-1) | (x-1,y) | (x-1,y+1) |
| (x,y-1) | (x,y) | (x,y+1) |
| (x-1,y+1) | (x+1,y) | (x+1,y+1) |

Slide credit: Yan Tong

# Adjacency

**Adjacency is the relationship between two pixels *p* and *q***

***V* is a set of intensity values used to define adjacency**

- **Binary image: V={1} or V={0}**
- **Gray level image:** $\boldsymbol{V \sqsubseteq \{0, 1, \ldots, 255\}}$

$\boldsymbol{f(p) \in V}$ and $\boldsymbol{f(q) \in V}$ ➡ **Intensity constraints**

**Three types of adjacency:**

**4-adjacency** **8-adjacency** **m-adjacency**

p

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

X

Mixed adjacency

$\boldsymbol{q \in N_4(p)}$ $\boldsymbol{q \in N_8(p)}$ $\boldsymbol{q \in N_D(p)}$ **and** $N_4(q) \cap N_4(p) = \emptyset$ 
$or\ q \in N_4(p)$

Slide credit: Yan Tong

# Connectivity

- Path from p to q: a sequence of distinct and adjacent pixels with coordinates

$$\text{Starting point p} \leftarrow \underbrace{(x_0, y_0), (x_1, y_1)}_{\text{adjacent}}, \dots, (x_n, y_n) \rightarrow \text{ending point q}$$

- *Closed* path: if the starting point is the same as the ending point
- p and q are *connected*: if there is a path from p to q in S
- *Connected component*: all the pixels in S connected to p
- *Connected set*: S has only one connected component

Let S represent a subset of pixels in an image

**Are they connected sets?**

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \qquad \begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \qquad \begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}$$

Slide credit: Yan Tong

# Regions

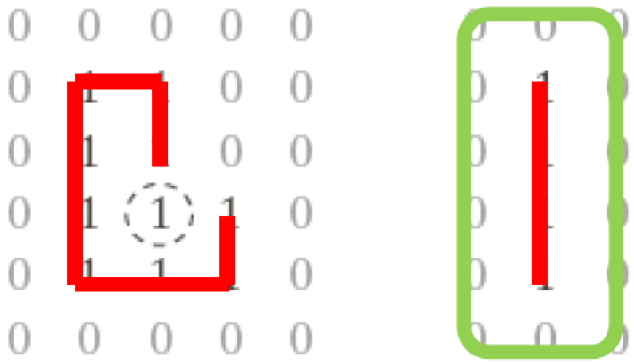
- **R is a region if R is a connected set**
- **$R_i$ and $R_j$ are adjacent if $R_i \cup R_j$ is a connected set**

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

Slide credit: Yan Tong

# Boundaries

- **Inner boundary (boundary) -- the set of pixels each of which has at least one background neighbor**
- **Outer boundary – the boundary pixels in the background**



Slide credit: Yan Tong

# Image Interpolation

- Interpolation is used in tasks such as zooming, shrinking, rotating, and geometrically correcting digital images
- Interpolation is the process of using known data to estimate values at unknown locations
- Approaches
  - **Nearest neighbor interpolation**: it assigns to each new location the intensity of its nearest neighbor in the original image

# Image Interpolation

- Approaches
  - **Bilinear interpolation**: we use the four nearest neighbors to estimate the intensity at a given location.
    - Let (x, y) denote the coordinates of the location to which we want to assign an intensity value (think of it as a point of the grid described previously), and let v(x, y) denote that intensity value, For bilinear interpolation, the assigned value is obtained using the equation

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

Four unknowns that can be written using the *four* nearest neighbors of point (*x*, *y*).

  - **Bicubic interpolation**: which involves the sixteen nearest neighbors of a point. The intensity value assigned to point (x, y) is obtained using the equation

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

The sixteen coefficients are determined from the sixteen equations with sixteen unknowns that can be written using the sixteen nearest neighbors of point (*x*, *y*)

# Distance Measures

**For pixels p, q, and z, with coordinates (x,y), (s,t) and (v,w), D is a distance function or metric if**

$$(a)\ D(p,q) \geq 0 \quad D(p,q) = 0\ iff\ \ p = q$$

$$(b)\ D(p,q) = D(q,p), \quad and$$

$$(c)\ D(p,z) \leq D(p,q) + D(q,z)$$

Slide credit: Yan Tong

# Distance Measures

**Euclidean distance** $D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$

**City-block (D4) distance** $D_4(p,q) = |x-s| + |y-t|$

**Chessboard (D8) distance (Chebyshev distance)**

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

Slide credit: Yan Tong

# Distance Measures

**D4 distance**

6

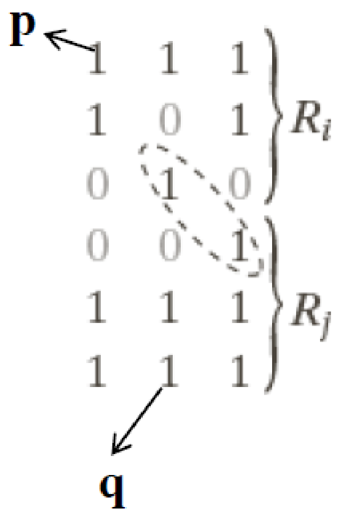
**D8 distance**

5

**Euclidean distance**

$$\sqrt{1+5^2}$$

**Distance vs length of a path?**



# Elementwise Versus Matrix Operations

- **An elementwise operation** involving one or more images is carried out on a pixel-by-pixel basis. Example:

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

The *elementwise product* (often denoted using the symbol $\odot$ or $\otimes$) of these two images is

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

# Elementwise Versus Matrix Operations

- The **matrix product** of the images is formed using the rules of matrix multiplication. For the example matrices given in the previous slide:

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

- The terms elementwise addition and subtraction of two images are redundant because these are elementwise operations by definition.

# Linear Versus Non-Linear Operations

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

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

Given two arbitrary constants, $a$ and $b$, and two arbitrary images $f_1(x,y)$ and $f_2(x,y)$, $\mathcal{H}$ is said to be a *linear operator* if

$$\begin{aligned}\mathcal{H}[af_1(x,y) + bf_2(x,y)] &= a\mathcal{H}[f_1(x,y)] + b\mathcal{H}[f_2(x,y)] \\ &= ag_1(x,y) + bg_2(x,y)\end{aligned} \qquad (1)$$

# Linear Versus Non-Linear Operations

- Linear operations must satisfy the properties:
  - **Additivity**: the output of a linear operation applied to the sum of two inputs is the same as performing the operation individually on the inputs and then summing the results.
  - **Homogeneity**: the output of a linear operation on a constant multiplied by an input is the same as the output of the operation due to the original input multiplied by that constant.
- By definition, an operator that fails to satisfy Eq. 1 in the previous slide, is said to be non-linear

# Test of Linearity

**Suppose that we are working with the max operation, whose function is to find the maximum value of the pixels in an image.**

- Consider the following images: $f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$ and $f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$

and suppose that we let $a = 1$ and $b = -1$. To test for linearity, we again start with the left side of Eq. (1) :

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

$$\max\left\{(1)\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} + (-1)\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = \max\left\{\begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix}\right\}$$
$$= -2$$

Working next with the right side, we obtain

$$(1)\max\left\{\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}\right\} + (-1)\max\left\{\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = 3 + (-1)7 = -4$$

The left and right sides of Eq. (2-23) are not equal in this case, so we have proved that the max operator is nonlinear.

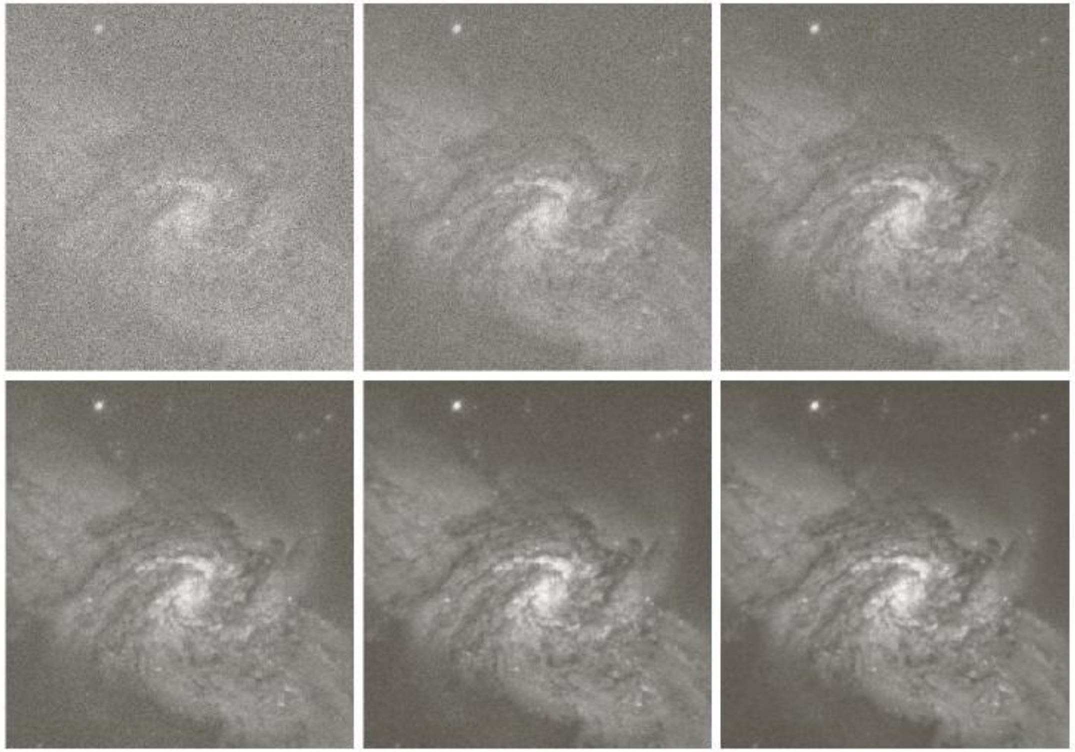
# Arithmetic Operations

Arithmetic operations between two images $f(x, y)$ and $g(x, y)$ are denoted as

$$s(x,y) = f(x,y) + g(x,y)$$
$$d(x,y) = f(x,y) - g(x,y)$$
$$p(x,y) = f(x,y) \times g(x,y)$$
$$v(x,y) = f(x,y) \div g(x,y)$$

# Example: Using image addition (averaging) for noise reduction

It is necessary that the images $g$ must be *registered* (i.e., spatially aligned)



a b c
d e f

**FIGURE 2.26** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

Corrupted image Noiseless image Noise

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

The objective of the following procedure is to reduce the noise content of the output image by adding a set of noisy input images, $\{g_i(x, y)\}$

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

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

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

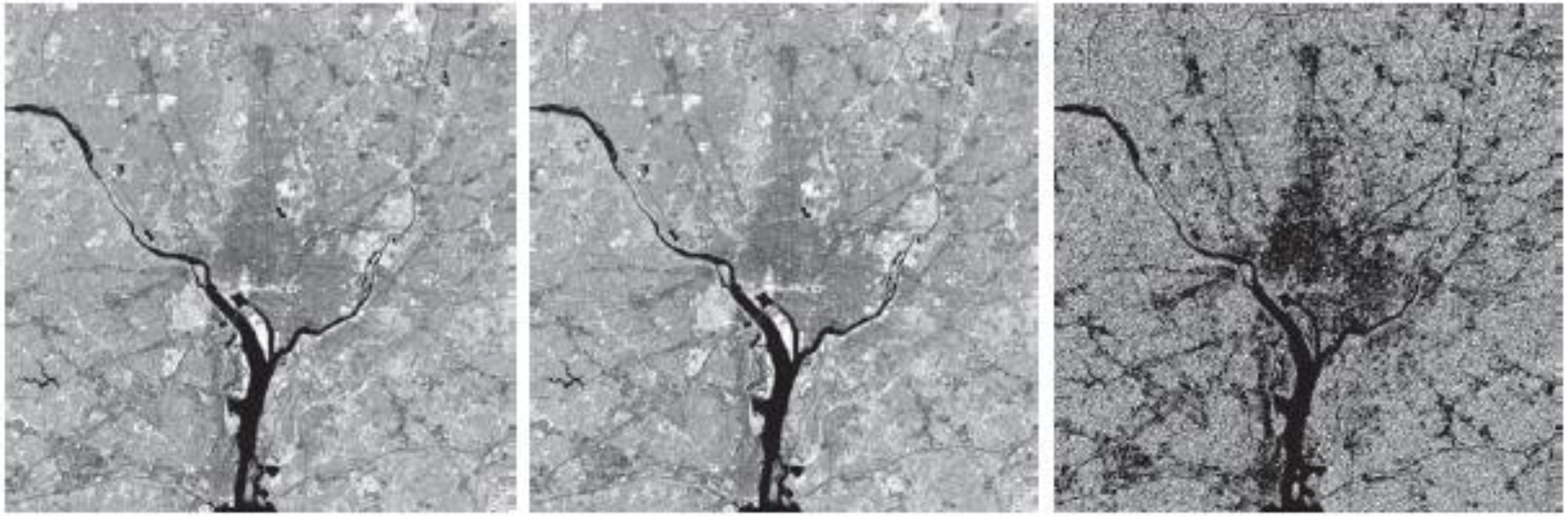
$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}$$

As $K$ increases, these two indicate that the variability (as measured by the variance or the standard deviation) of the pixel values at each location $(x, y)$ decreases

Assumption: the noise is uncorrelated in image and has zero mean

and, noise and image values are uncorrelated (this is a typical assumption for additive noise)

# Example: Comparing images using subtraction



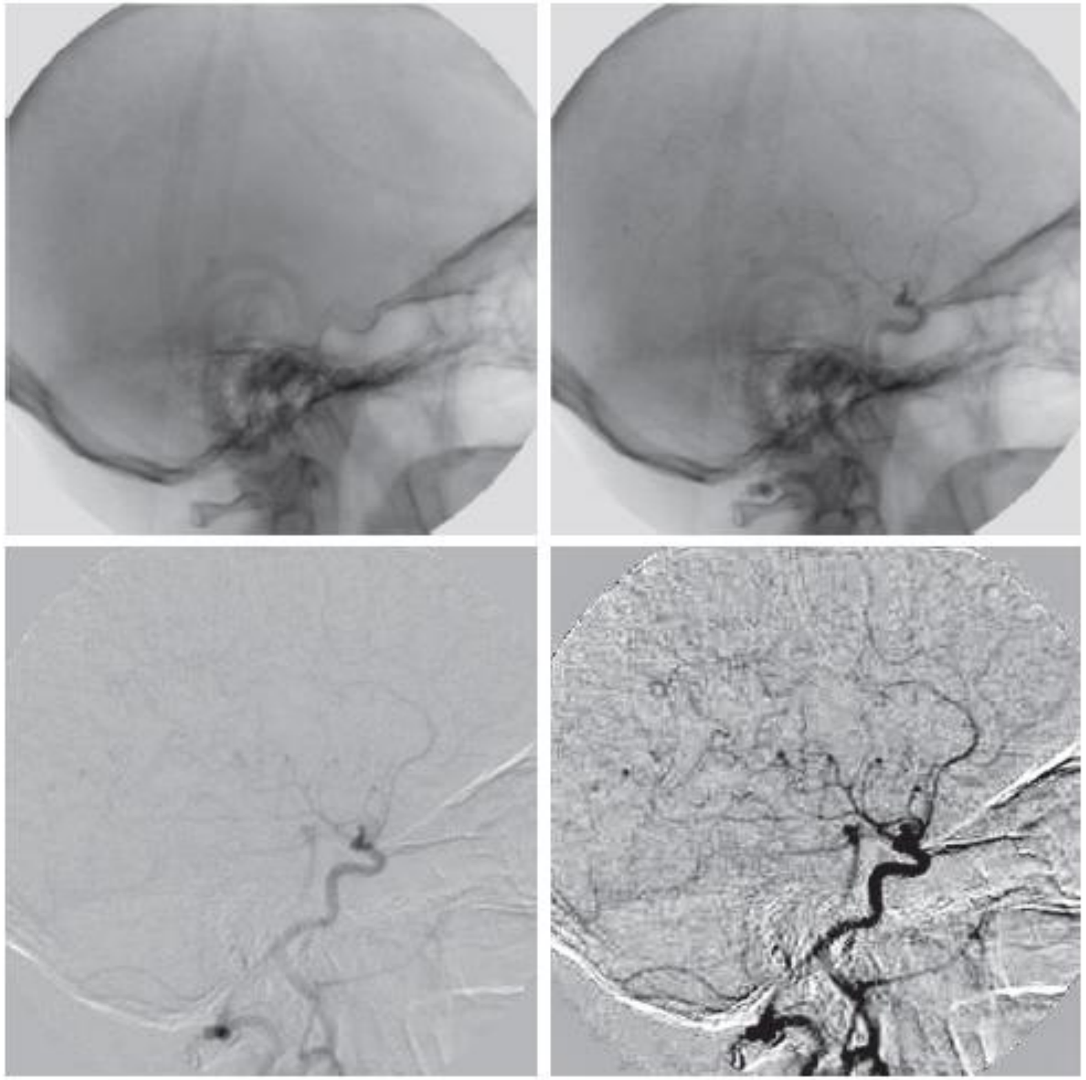
a b c

**FIGURE 2.30** (a) Infrared image of the Washington, D.C. area. (b) Image resulting from setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity. (Original image courtesy of NASA.)

# Example: Comparing images using subtraction

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.)



**The images used in averaging & subtraction must be registered!**

# Notes on Arithmetic Operations

**The images used in averaging, addition & subtraction must be registered!**

Output images should be normalized to the range of [0,255]

$$f_m = f - \min(f)$$ which creates an image whose minimum value is 0

$$f_s = K[f_m / \max(f_m)]$$ which creates a scaled image, $f_s$, whose values are in the range [0, $K$], here K=255 for 8-bit image

# Basic Set Operations

- **A is a set: A={.}        e.g. A={1,…,255} or** $A = \{w|w = 1, \dots, 255\}$

$A = \emptyset$        **for empty set**

- **a is an element of A**$(a \in A)$ **or a isn't an element of A**$(a \notin A)$
- **A is a *subset* of B if every element in A also is in B** $(A \sqsubseteq B)$
- **C is the *union* of two sets A and B** $(C = A \cup B)$
- **C is the *intersection* of A and B** $(C = A \cap B)$
- **Disjoint or mutual exclusive sets** $(A \cap B = \emptyset)$
- ***Set universe* is the set of all elements in an application**
- ***Set difference*** $(A - B = \{w|w \in A, w \notin B\}) = A \cap B^c$

Slide credit: Yan Tong

# Basic Set Operations

**A region in an image is represented by a set of coordinates within the region**

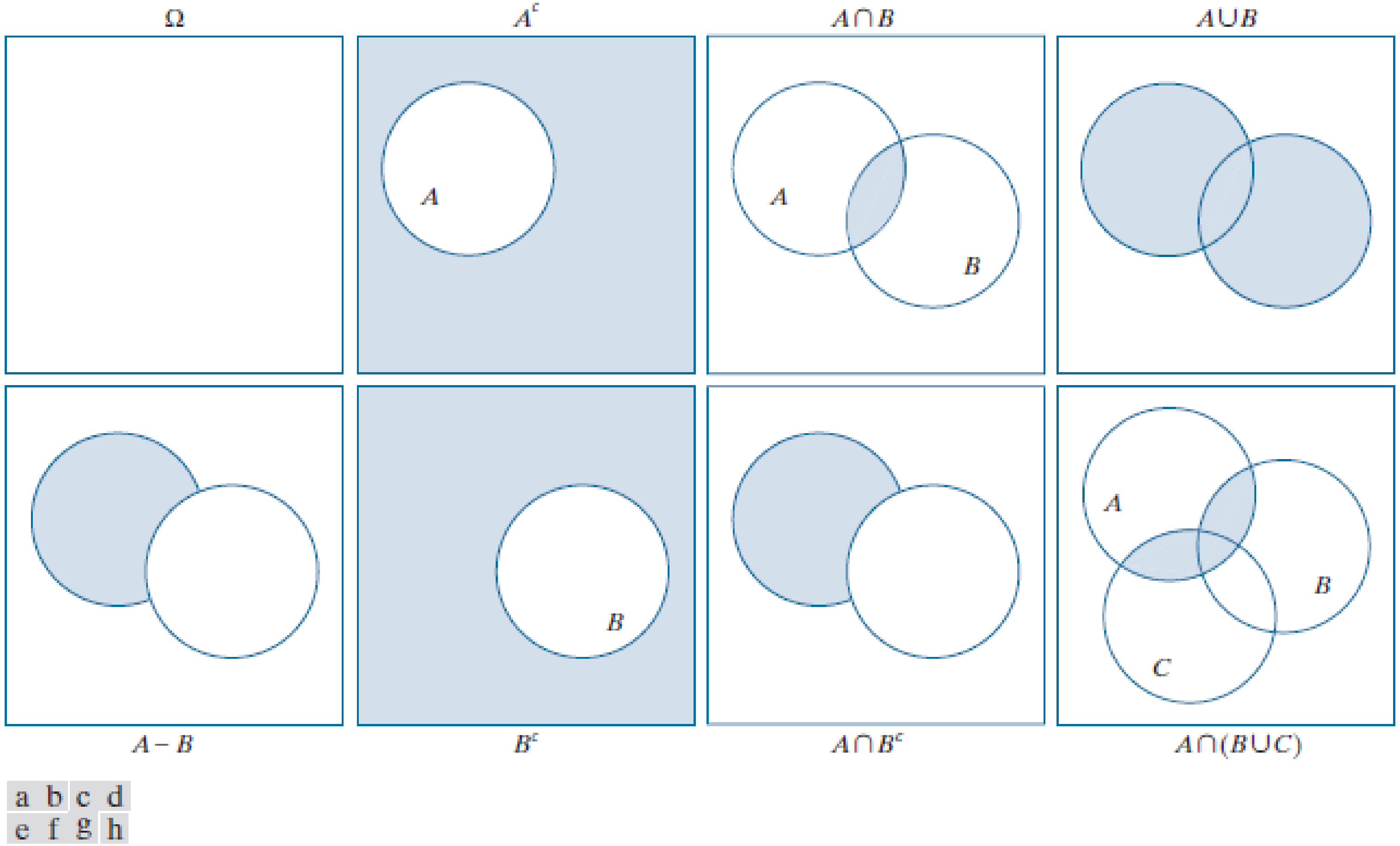



**FIGURE 2.35** Venn diagrams corresponding to some of the set operations in Table 2.1. The results of the operations, such as $A^c$, are shown shaded. Figures (e) and (g) are the same, proving via Venn diagrams that $A - B = A \cap B^c$

# Set operations on images

* ***Complement*** of a grayscale image as the pairwise differences between a constant and the intensity of every pixel in the image

* **Union** and **intersection** operations for grayscale values as the maximum and minimum of corresponding pixel pairs, respectively

Let the elements of a grayscale image be represented by a set *A* whose elements are triplets of the form (*x*, *y*, *z*), where *x* and *y* are spatial coordinates, and *z* denotes intensity values

Complement – negative image

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

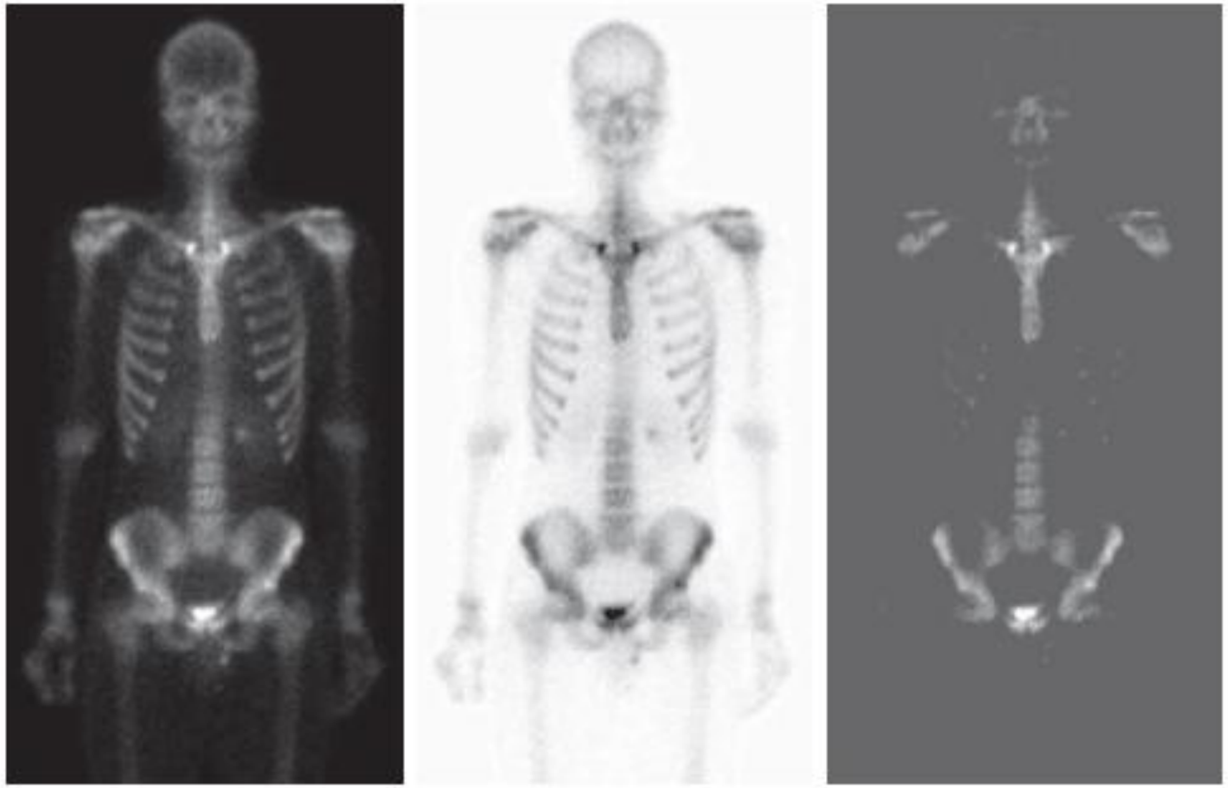
Thresholding

$$\boldsymbol{A} \cup \boldsymbol{B} = \left\{\left(\boldsymbol{x}, \boldsymbol{y}, \max(\boldsymbol{z_a}, \boldsymbol{z_b})\right) \middle| (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z_a}) \in \boldsymbol{A}, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z_b}) \in \boldsymbol{B}\right\}$$

# Illustration of set operations involving grayscale images

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.)



Complement – negative image

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

Thresholding

$$A \cup B = \left\{\left(x, y, \max(z_a, z_b)\right) \middle| (x, y, z_a) \in A, (x, y, z_b) \in B\right\}$$

# Basic Logical Operations

- Logical operations deal with **TRUE** (typically denoted by **1**) and **FALSE** (typically denoted by **0**) variables and expressions
- Binary images composed of foreground (**1-valued**) pixels, and a background composed of **0-valued** pixels
- Logical operators can be defined in terms of truth tables, as shown below for two logical variables *a*, *b*

**TABLE 2.2** Truth table defining the logical operators AND($\wedge$), OR($\vee$), and NOT($\sim$).

| $a$ | $b$ | $a$ 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 |

# Basic Logical Operations

- When applied to two binary images, AND and OR operate on pairs of corresponding pixels between the images.
  - They are elementwise operators

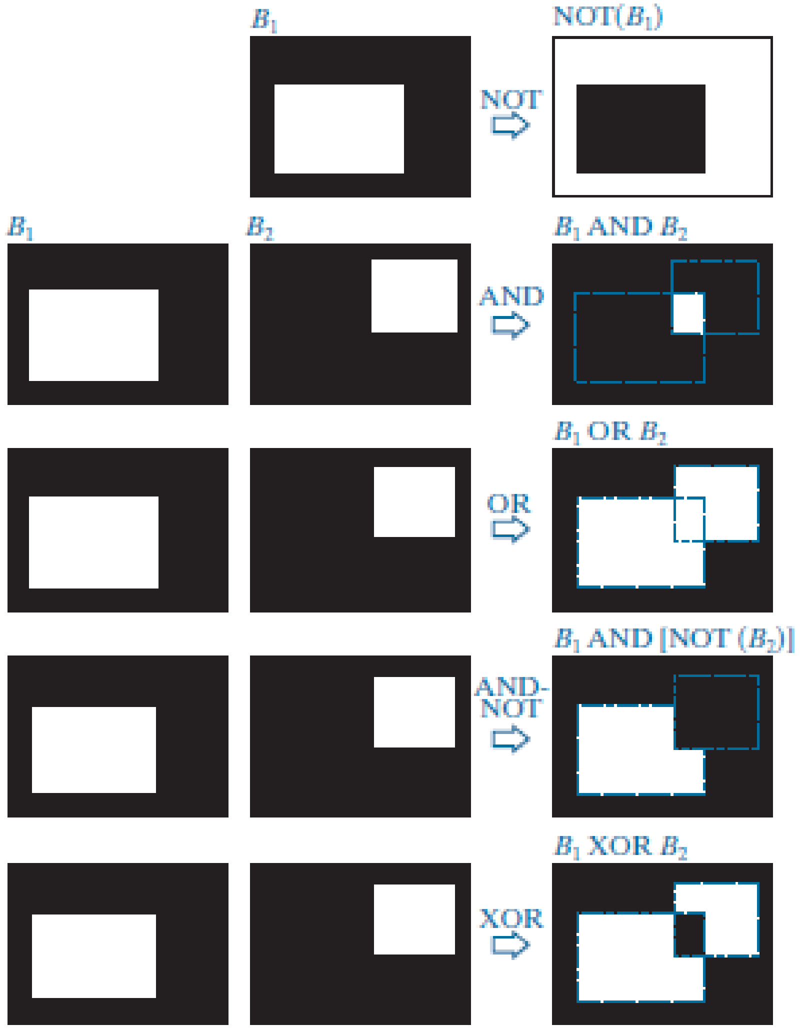



**FIGURE 2.37** Illustration of logical operations involving foreground (white) pixels. Black represents binary 0's and white binary 1's. The dashed lines are shown for reference only. They are not part of the result.

# Spatial Operations

**Perform directly on the pixels of the given image**

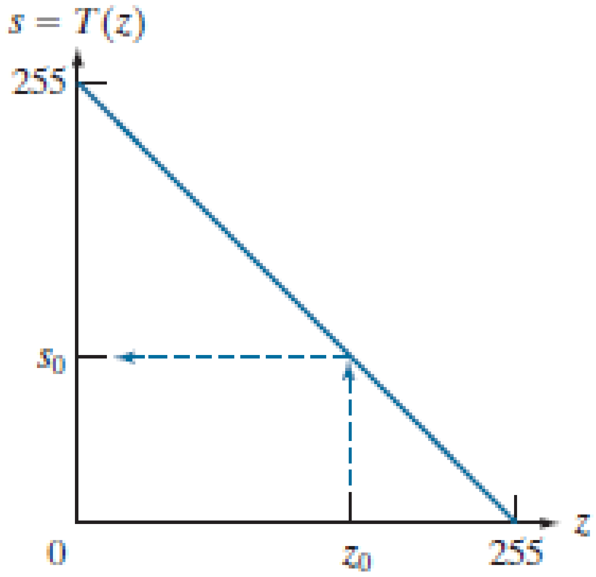
- Intensity transformation – change the intensity
  - Single pixel operations $s = T(z)$
  - Neighborhood operations
- Geometric spatial transformations – change the coordinates

Slide credit: Yan Tong

# Single pixel operations

- Determined by
  - Transformation function T
  - Input intensity value
- Not depend on other pixels and position

**FIGURE 2.38** Intensity transformation function used to obtain the digital equivalent of photographic negative of an 8-bit image..




Slide credit: Yan Tong

# Neighborhood Operations

a b
c d

**FIGURE 2.39** Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood. (c) An aortic angiogram (see Section 1.3). (d) The result of using Eq. (2-43) with $m = n = 41$. The images are of size $790 \times 686$ pixels. (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

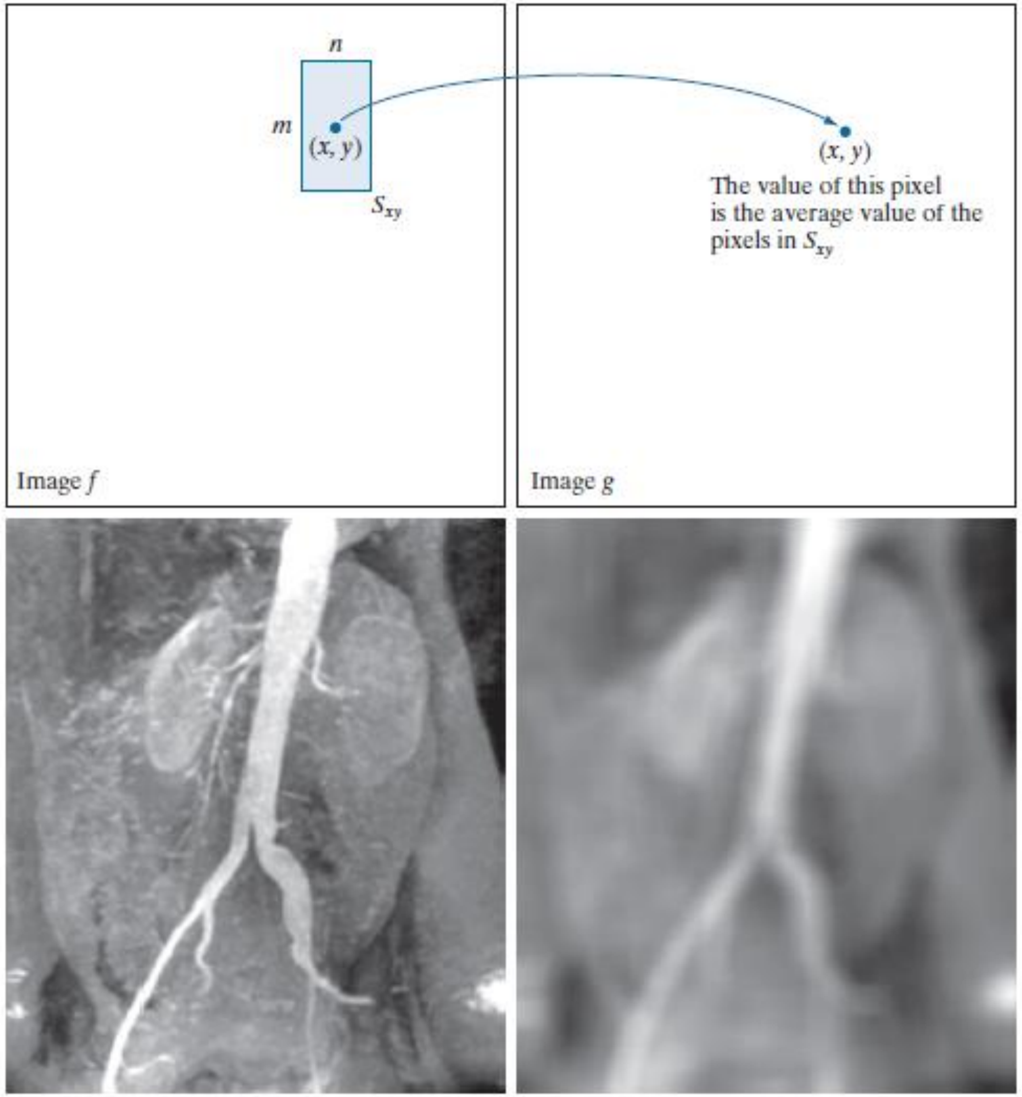



Image smoothing

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

a rectangular neighborhood of size $m \times n$ centered on $(x, y)$

**Other examples:**

- Interpolation
- Image filtering

Slide credit: Yan Tong

# Geometric Transformations

Geometric transformations of digital images consist of two basic operations:

1. Spatial transformation of coordinates.
2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

The transformation of coordinates may be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{2-44}$$

where $(x, y)$ are pixel coordinates in the original image and $(x', y')$ are the corresponding pixel coordinates of the transformed image. For example, the transformation $(x', y') = (x/2, y/2)$ shrinks the original image to half its size in both spatial directions.

# Geometric Transformations - rubber-sheet transformations

Our interest is in so-called *affine transformations*, which include scaling, translation, rotation, and shearing. The key characteristic of an affine transformation in 2-D is that it preserves points, straight lines, and planes.

$$(x, y) = T\{(v, w)\}$$

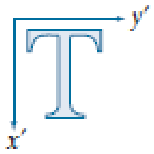
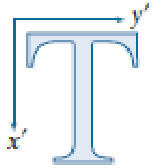
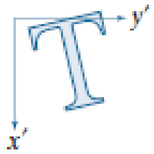
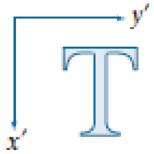
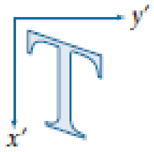
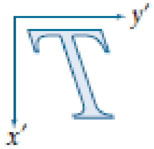
**Affine transform:**

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix}$$

**Inverse mapping**

$$\begin{bmatrix} v \\ w \\ 1 \end{bmatrix} = \mathbf{T}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**TABLE 2.3**
Affine transformations based on Eq. (2-45).

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

# Geometric Transformations – intensity interpolation

Nearest neighbor Bilinear Bicubic

**Note: a neighborhood operation, i.e., interpolation, is required following geometric transformation**

Slide credit: Yan Tong

# Image Registration

**Compensate the geometric change in:**

- view angle
- distance
- orientation
- sensor resolution
- object motion

**Four major steps:**

- Feature detection
- Feature matching
- Transformation model
- Resampling

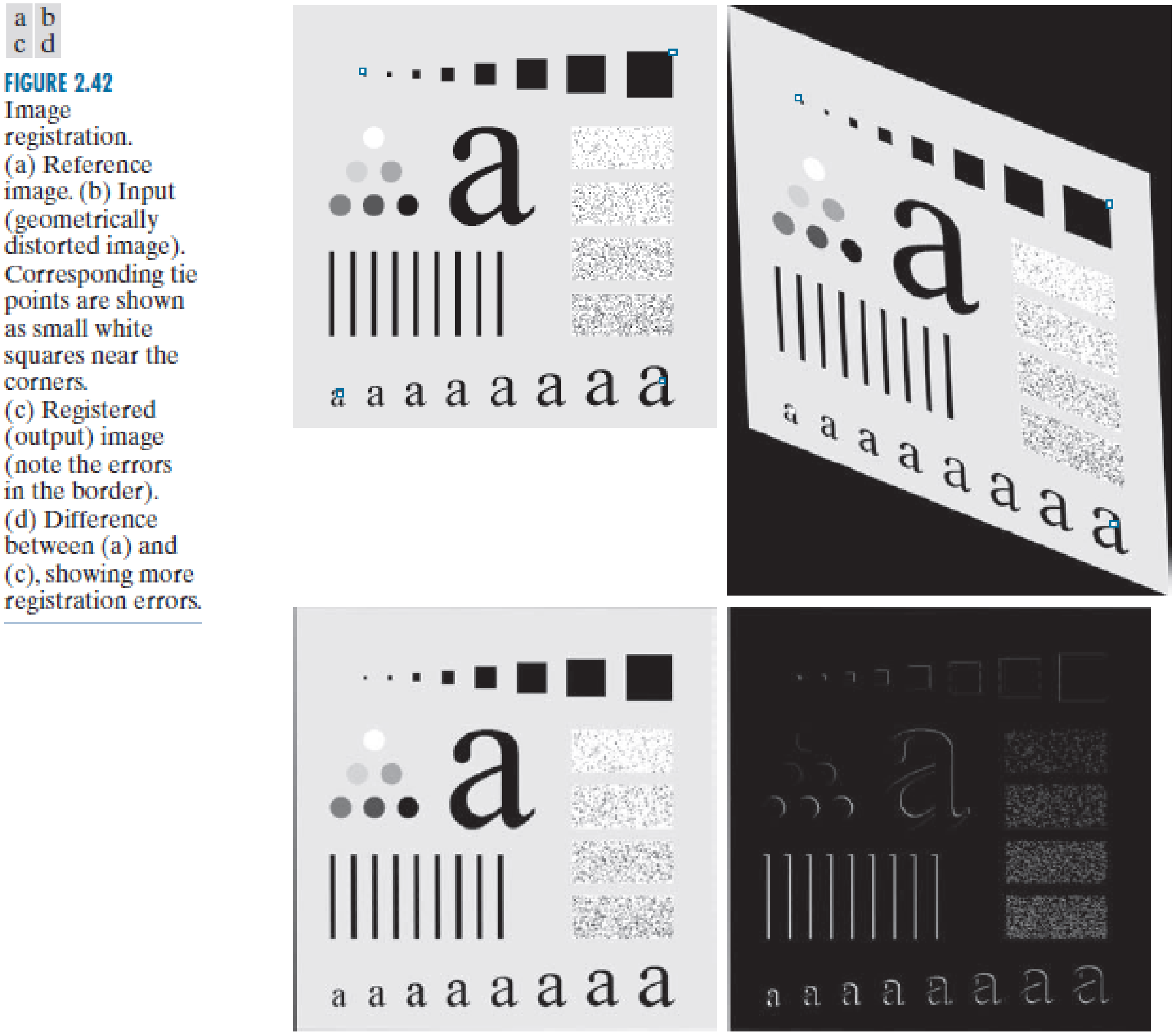


**FIGURE 2.42** Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered (output) image (note the errors in the border). (d) Difference between (a) and (c), showing more registration errors.

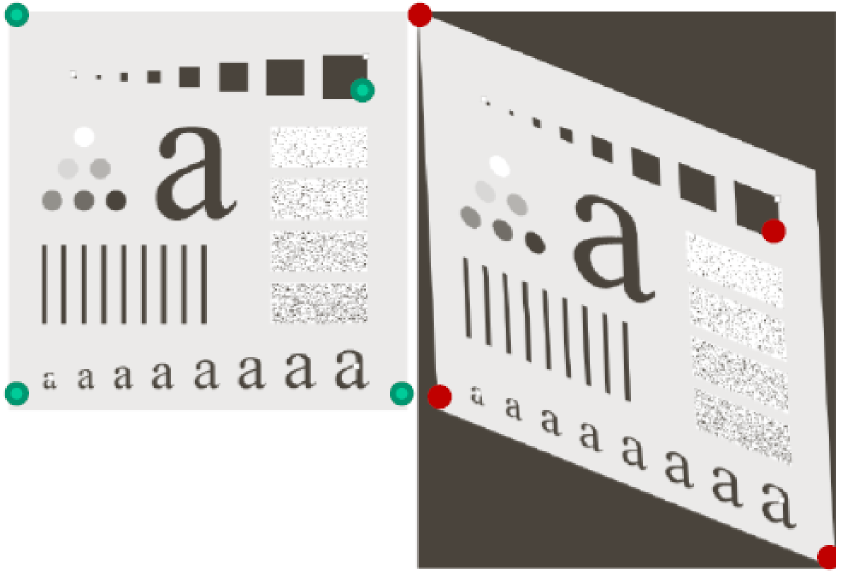
Slide credit: Yan Tong

# Image Registration

**Coordinates in the moving image $(v, w)$**
**Coordinates in the template image $(x, y)$**

$$x = c_1 v + c_2 w + c_3 vw + c_4$$

$$y = c_5 v + c_6 w + c_7 vw + c_8$$



- **Known: coordinates of the points $(x, y)$ and $(v, w)$**
- **Unknown: $c_1$ to $c_8$**

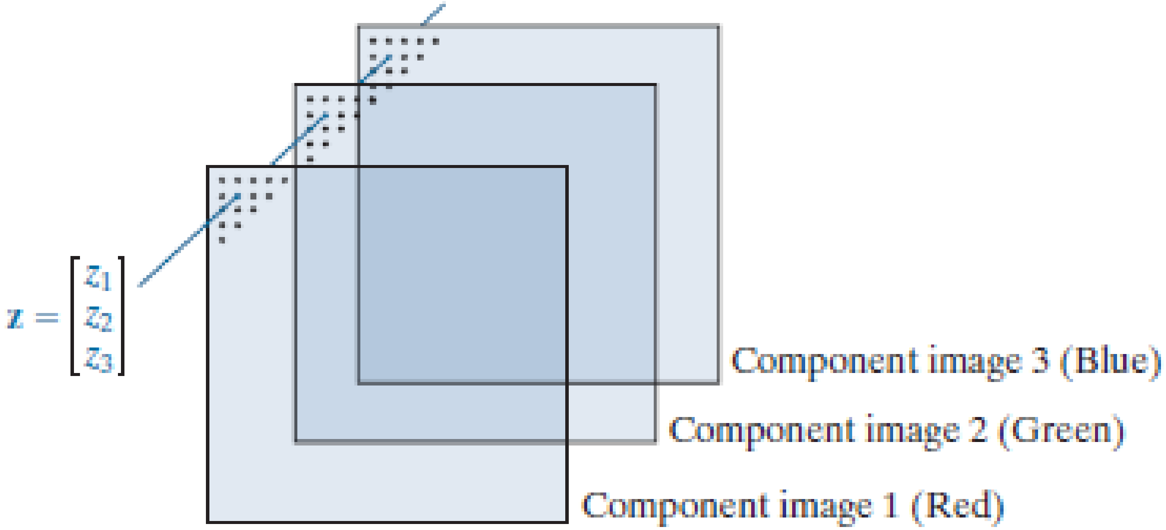
**4 tie points -> 8 equations**

Control Points

Slide credit: Yan Tong

# Vector and Matrix Operations

**FIGURE 2.43** Forming a vector from corresponding pixel values in three RGB component images.




$$\mathbf{z} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}$$

Thus, an RGB color image of size *M X N* can be represented by three component images of this size, or by a total of *MN* vectors of size 3 X 1.

$$D(\mathbf{z}, \mathbf{a}) = \|\mathbf{z} - \mathbf{a}\| = \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}}$$

$$= \left[(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots + (z_n - a_n)^2\right]^{\frac{1}{2}}$$

**Geometric transformations use vector and matrix operations**

Slide credit: Yan Tong

# Spatial-Frequency Domain Transformation

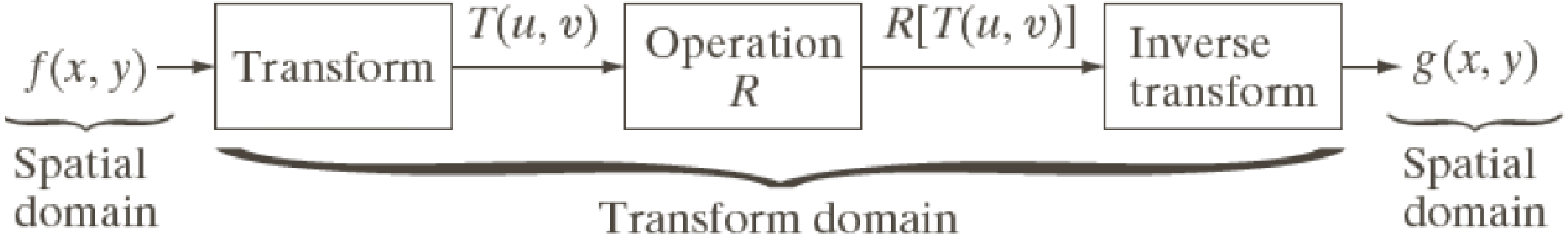



**FIGURE 2.39** General approach for operating in the linear transform domain.

2-D *linear transforms*

Input image

Forward transformation kernel

*transform pair*

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

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

Inverse transformation kernel

$u = 0,1,\ldots, M-1$

$v = 0,1,\ldots, N-1$

$x = 0,1,\ldots, M-1$

$y = 0,1,\ldots, N-1$

$M$ and $N$ are the row and column dimensions of $f$

Slide credit: Yan Tong

# Spatial-Frequency Domain Transformation

The forward transformation kernel is said to be *separable* if

$$r(x, y, u, v) = r_1(x, u) r_2(y, v) \quad (2\text{-}57)$$

In addition, the kernel is said to be *symmetric* if $r_1(x,u)$ is functionally equal to $r_2(y,v)$, so that

$$r(x, y, u, v) = r_1(x, u) r_1(y, v) \quad (2\text{-}58)$$

Identical comments apply to the inverse kernel.

Slide credit: Yan Tong

# Spatial-Frequency Domain Transformation

***Fourier Transform***

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

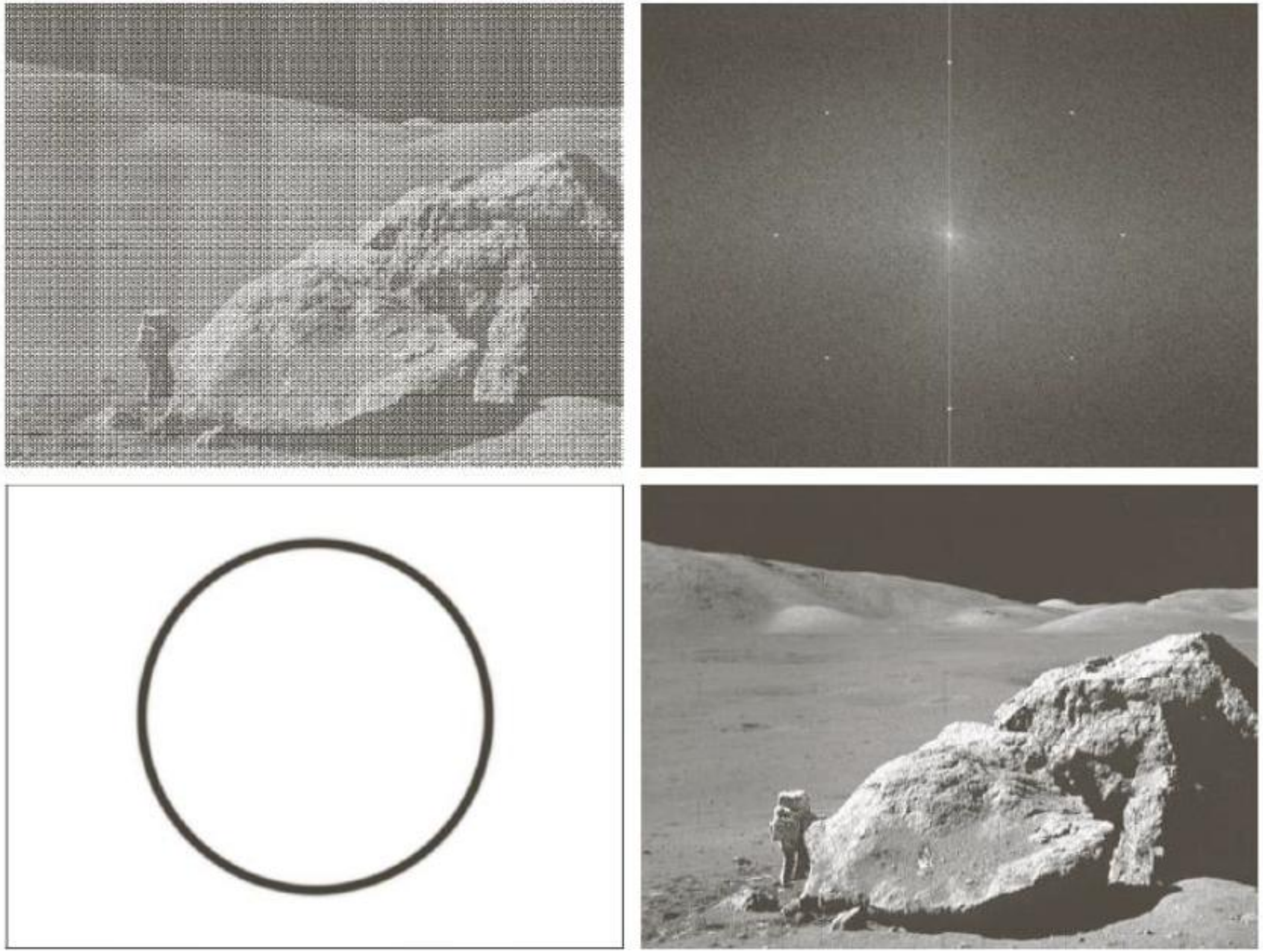
Inverse $$s(x,y,u,v) = \frac{1}{MN} e^{j2\pi(ux/M+vy/N)}$$

***Discrete Fourier Transform***

Forward $$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

Inverse $$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) e^{j2\pi(ux/M + vy/N)}$$

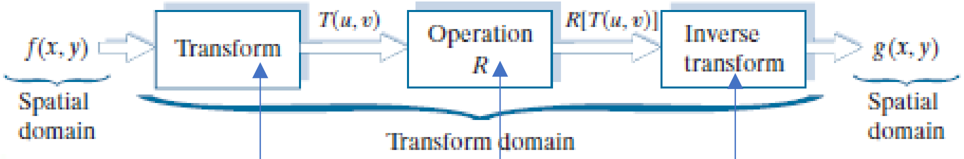
# Spatial-Frequency Domain Transformation



a b
c d

**FIGURE 2.40**
(a) Image corrupted by sinusoidal interference. (b) Magnitude of the Fourier transform showing the bursts of energy responsible for the interference. (c) Mask used to eliminate the energy bursts. (d) Result of computing the inverse of the modified Fourier transform. (Original image courtesy of NASA.)

Sinusoidal interference occurs when two or more sinusoidal waves overlap, resulting in a new wave with a different amplitude.




Figures (b) (b) (c)

Multiplied by filter (c)

Modified transform

# Probability Methods

$z_k$ is the kth intensity value
$n_k$ is the number of pixels having the intensity value $z_k$

Probability of an intensity value

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

Slide credit: Yan Tong

# Probability Methods

Once we have $p(z_k)$, we can determine a number of important image characteristics. For example, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k p(z_k) \tag{2-69}$$

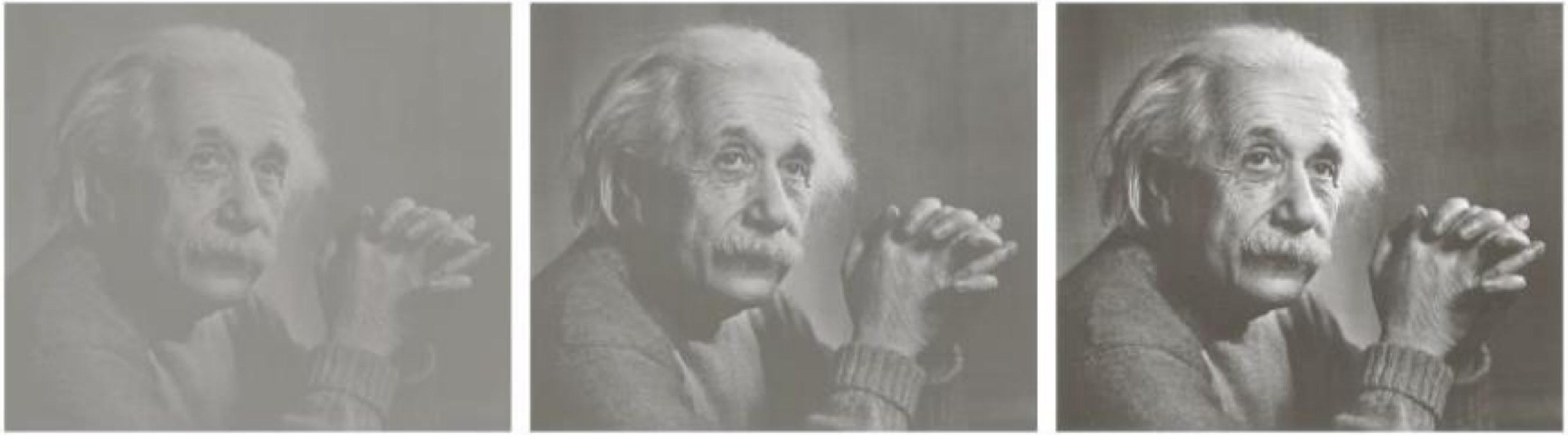
Similarly, the variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \tag{2-70}$$

The variance is a measure of the spread of the values of $z$ about the mean, so it is a useful measure of image contrast. In general, the $n$th central moment of random variable $z$ about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \tag{2-71}$$

# Probability Methods



a b c

**FIGURE 2.41** Images exhibiting (a) low contrast, (b) medium contrast, and (c) high contrast.

Std=14.3    Std=31.6    Std=49.2