Computer Vision and Image Processing (EC 336)

Lecture 3: Operators and Transformations in DIP

by Dr. Rashmi Panda

Dept. of Electronics and communication Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RANCHI

Outline

- Distance Measures
- Mathematical Operations in DIP
  - Arithmetic Operations
  - Logical Operations
- Spatial Operations
- Geometric Spatial Transformations
- Image Interpolation

Distance Measures

- Given pixels *p, q* and *z* with coordinates (x, y), (s, t), (u, v) respectively, the distance function D has following properties:

  a. $D(p, q) \geq 0$ [$D(p, q) = 0$, iff $p = q$]

  b. $D(p, q) = D(q, p)$

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

Distance Measures

The following are the different Distance measures:

a. Euclidean Distance :

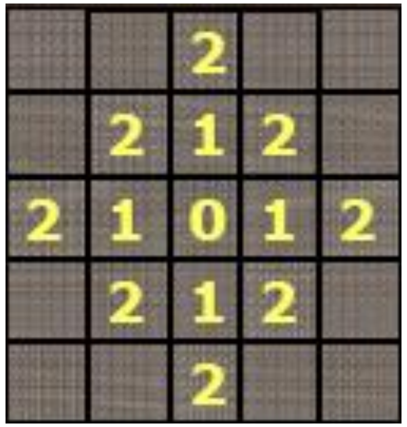
$D_e(p, q) = [(x-s)^2 + (y-t)^2]^{1/2}$

b. City Block Distance:

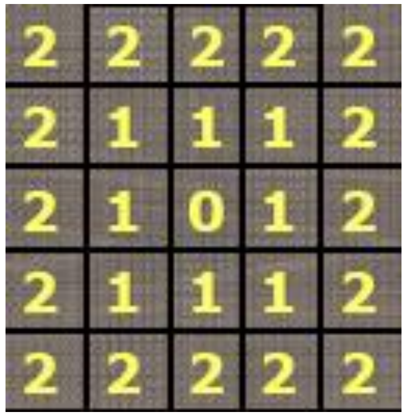
$D_4(p, q) = |x-s| + |y-t|$

c. Chess Board Distance:

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



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

Introduction to Mathematical Operations in DIP

- **Array vs. Matrix Operation**

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Array product operator

$$A \,.{*}\, B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

Array product

Matrix product operator

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

Matrix product

Introduction to Mathematical Operations in DIP

- **Linear vs. Nonlinear Operation**

$$H\left[f(x,y)\right] = g(x,y)$$

$$H\left[a_i f_i(x,y) + a_j f_j(x,y)\right]$$

$$= H\left[a_i f_i(x,y)\right] + H\left[a_j f_j(x,y)\right]$$

Additivity

$$= a_i H\left[f_i(x,y)\right] + a_j H\left[f_j(x,y)\right]$$

Homogeneity

$$= a_i g_i(x,y) + a_j g_j(x,y)$$

H is said to be a **linear operator;**

- *H* is said to be a **nonlinear operator** if it does not meet the above qualification.

Arithmetic Operations

- Arithmetic operations between images are array operations. The four arithmetic operations 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: Addition of Noisy Images for Noise Reduction

Noiseless image: $f(x,y)$

Noise: $n(x,y)$ (at every pair of coordinates (x,y), the noise is uncorrelated and has zero average value)

Corrupted image: $g(x,y)$

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

Reducing the noise by adding a set of noisy images, $\{g_i(x,y)\}$

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

Example: Addition of Noisy Images for Noise Reduction

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

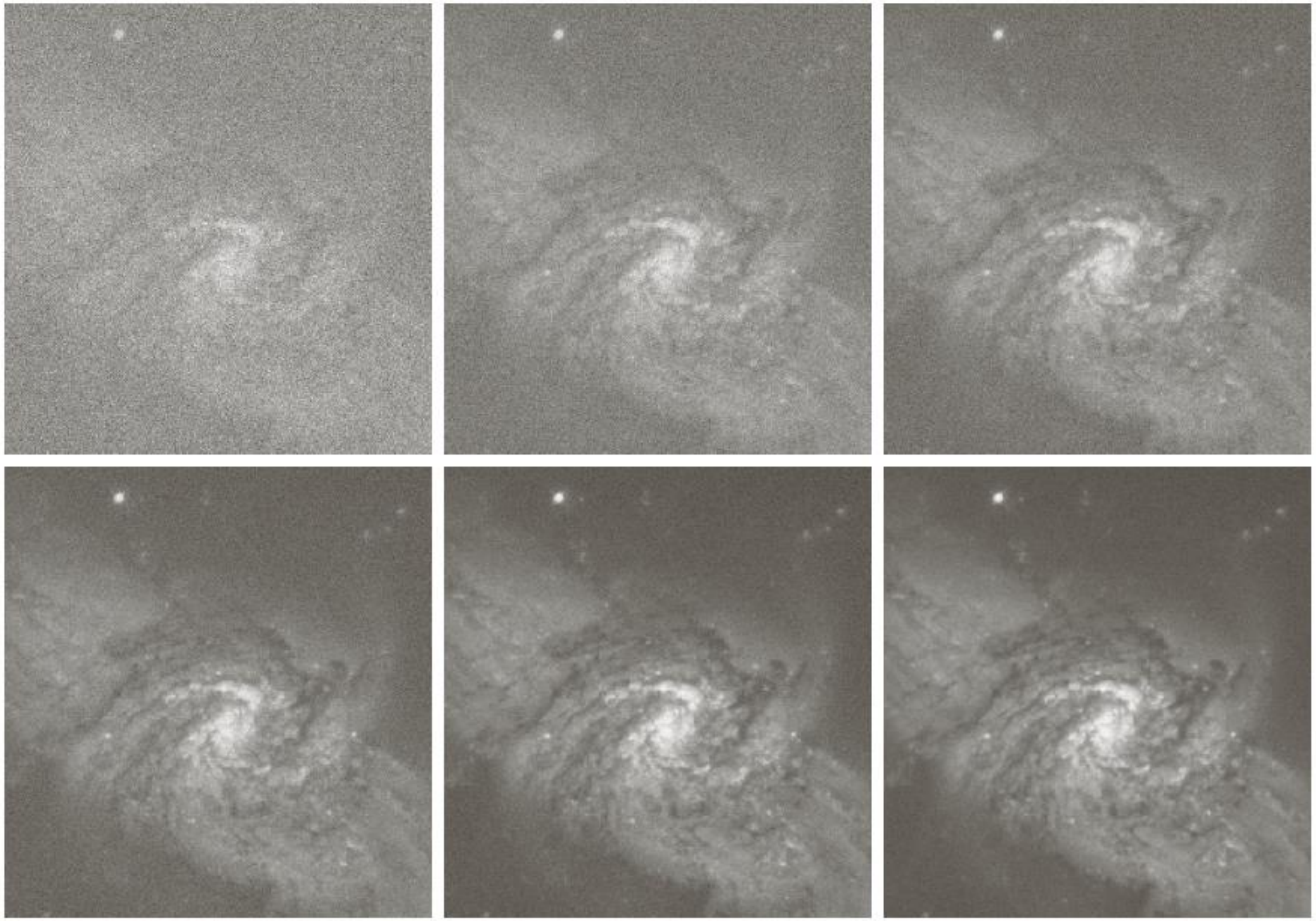
$$E\{\bar{g}(x,y)\} = f(x,y)$$

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

$$= \sigma^2_{\frac{1}{K}\sum_{i=1}^{K} n_i(x,y)} = \frac{1}{K}\sigma^2_{n(x,y)}$$

Example: Addition of Noisy Images for Noise Reduction

- In astronomy, imaging under very low light levels frequently causes sensor noise to render single images virtually useless for analysis.
- In astronomical observations, similar sensors is used for noise reduction by observing the same scene over long periods of time. Image averaging is then used to reduce the noise.



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.)

An Example of Image Subtraction: Mask Mode Radiography

Mask *h(x,y)*: an X-ray image of a region of a patient's body

Live images *f(x,y):* X-ray images captured by an intensified TV camera at TV rates after injection of the contrast medium

Enhanced detail *g(x,y)*

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

The procedure gives a movie showing how the contrast medium propagates through the various arteries in the area being observed.

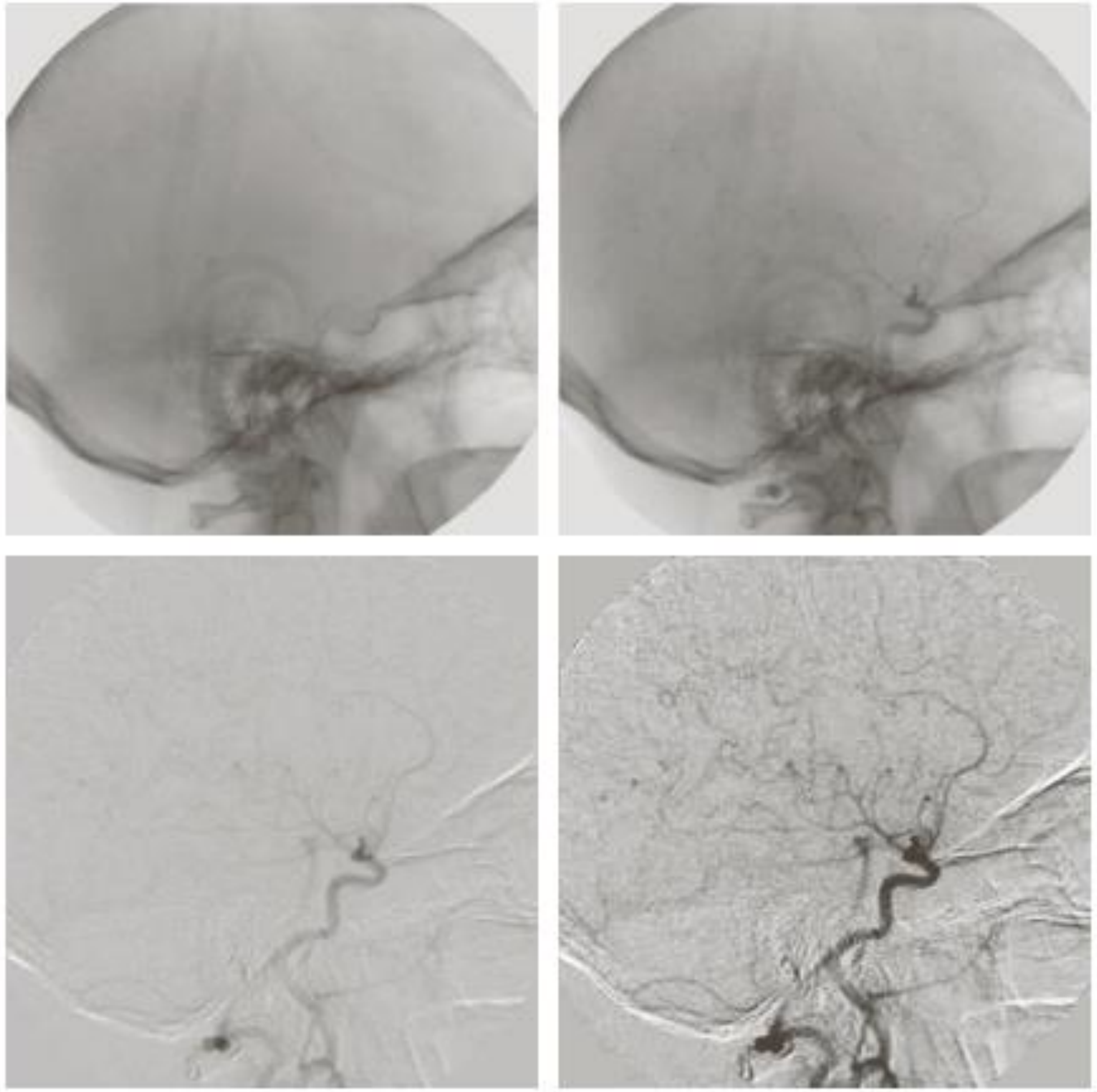


FIGURE 2.28 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

An Example of Image Multiplication

FIGURE 2.29 Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Set and Logical Operations

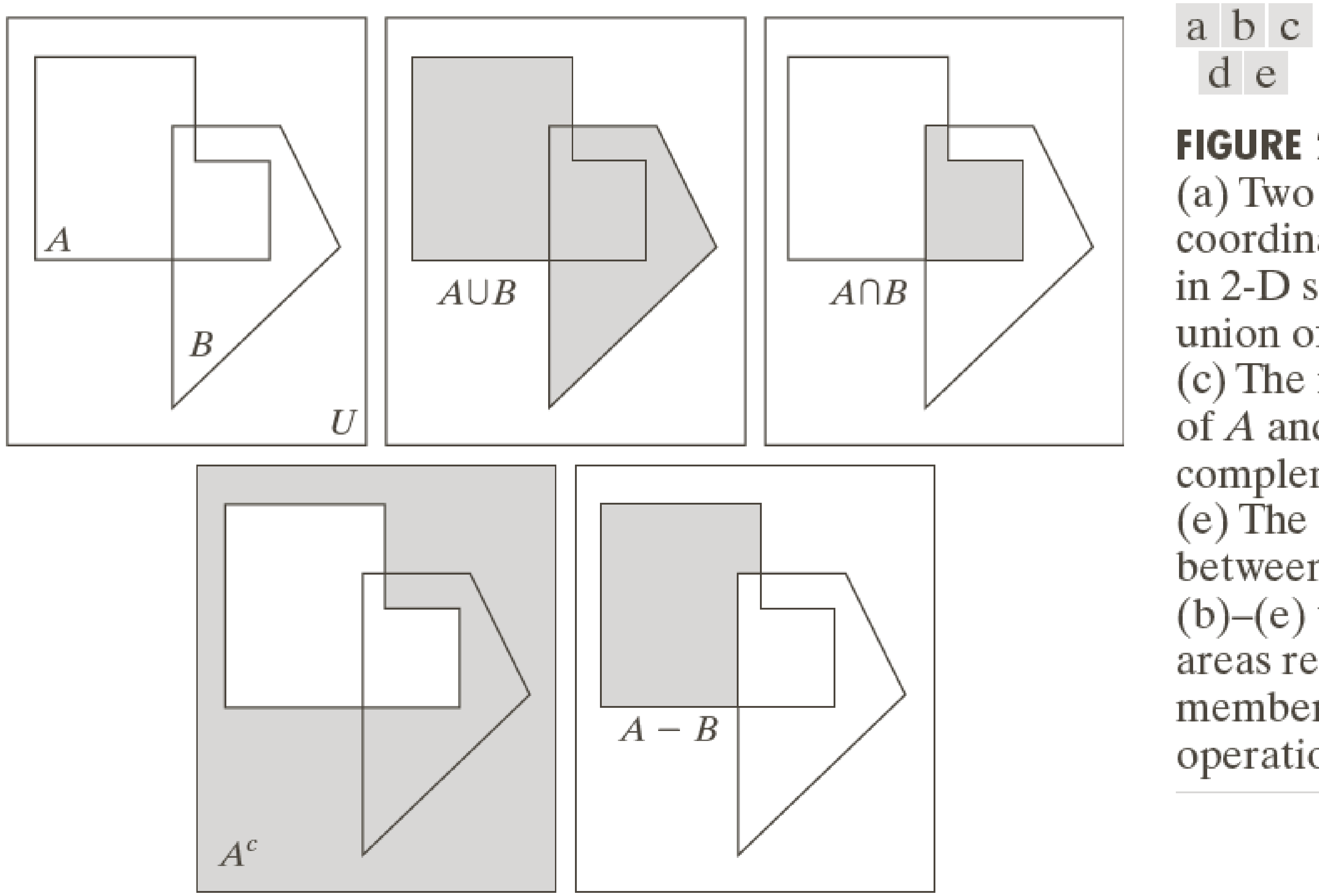


FIGURE 2.31 (a) Two sets of coordinates, *A* and *B*, in 2-D space. (b) The union of *A* and *B*. (c) The intersection of *A* and *B*. (d) The complement of *A*. (e) The difference between *A* and *B*. In (b)–(e) the shaded areas represent the member of the set operation indicated.

Set and Logical Operations

- Let *A* be the elements of a gray-scale image

  The elements of *A* are triplets of the form *(x, y, z),* where x and y are spatial coordinates and *z* denotes the intensity at the point *(x, y).*

$$A = \{(x, y, z) \mid z = f(x, y)\}$$

- The complement of *A* is denoted A^c

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

$$K = 2^k - 1; k \text{ is the number of intensity bits used to represent } z$$

Set and Logical Operations

- The union of two gray-scale images (sets) A and B is defined as the set

$$A \cup B = \{\max_{z}(a,b) \mid a \in A, b \in B\}$$

Set and Logical Operations

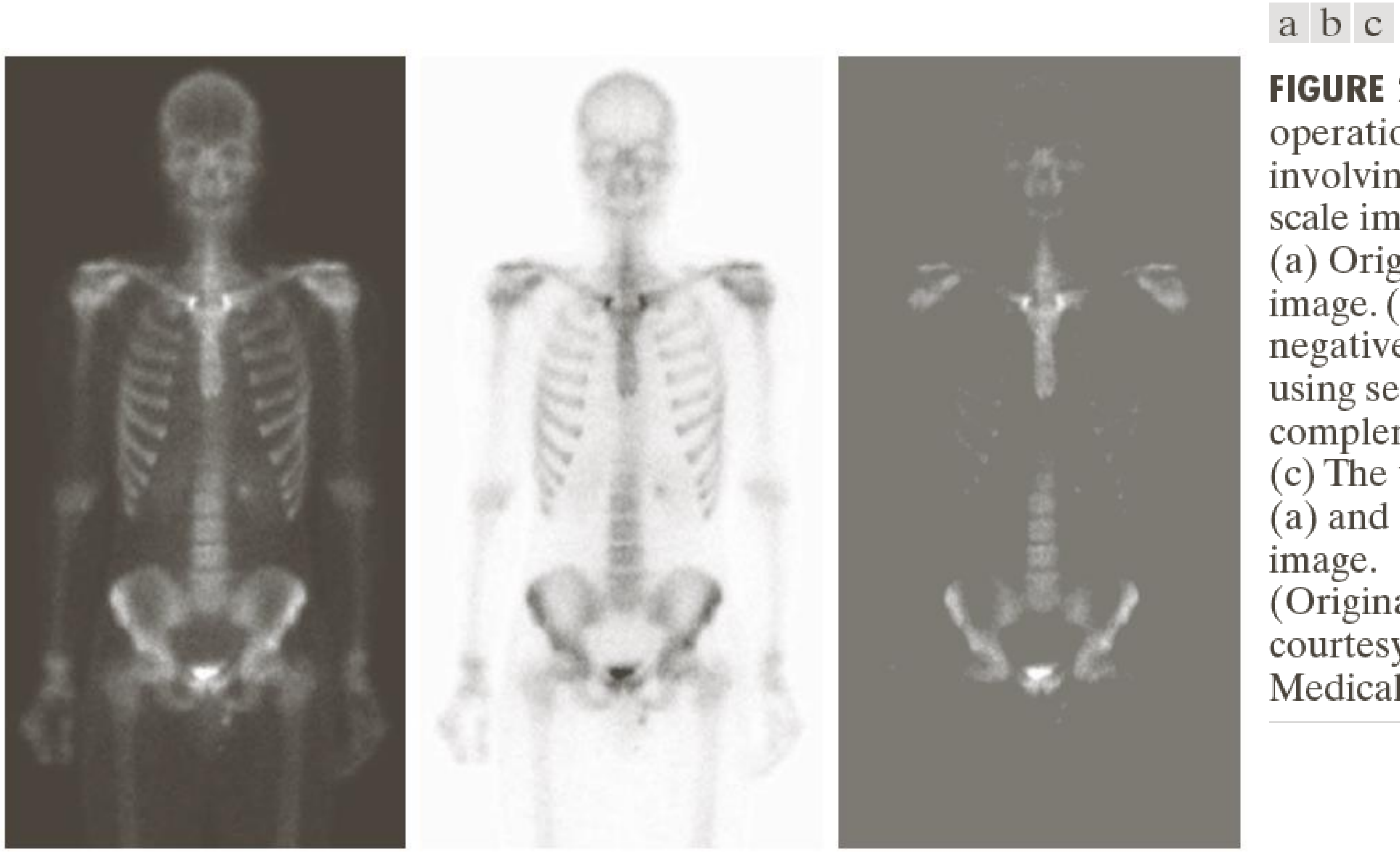


FIGURE 2.32 Set operations involving gray-scale images. (a) Original image. (b) Image negative obtained using set complementation. (c) The union of (a) and a constant image. (Original image courtesy of G.E. Medical Systems.)

Set and Logical Operations

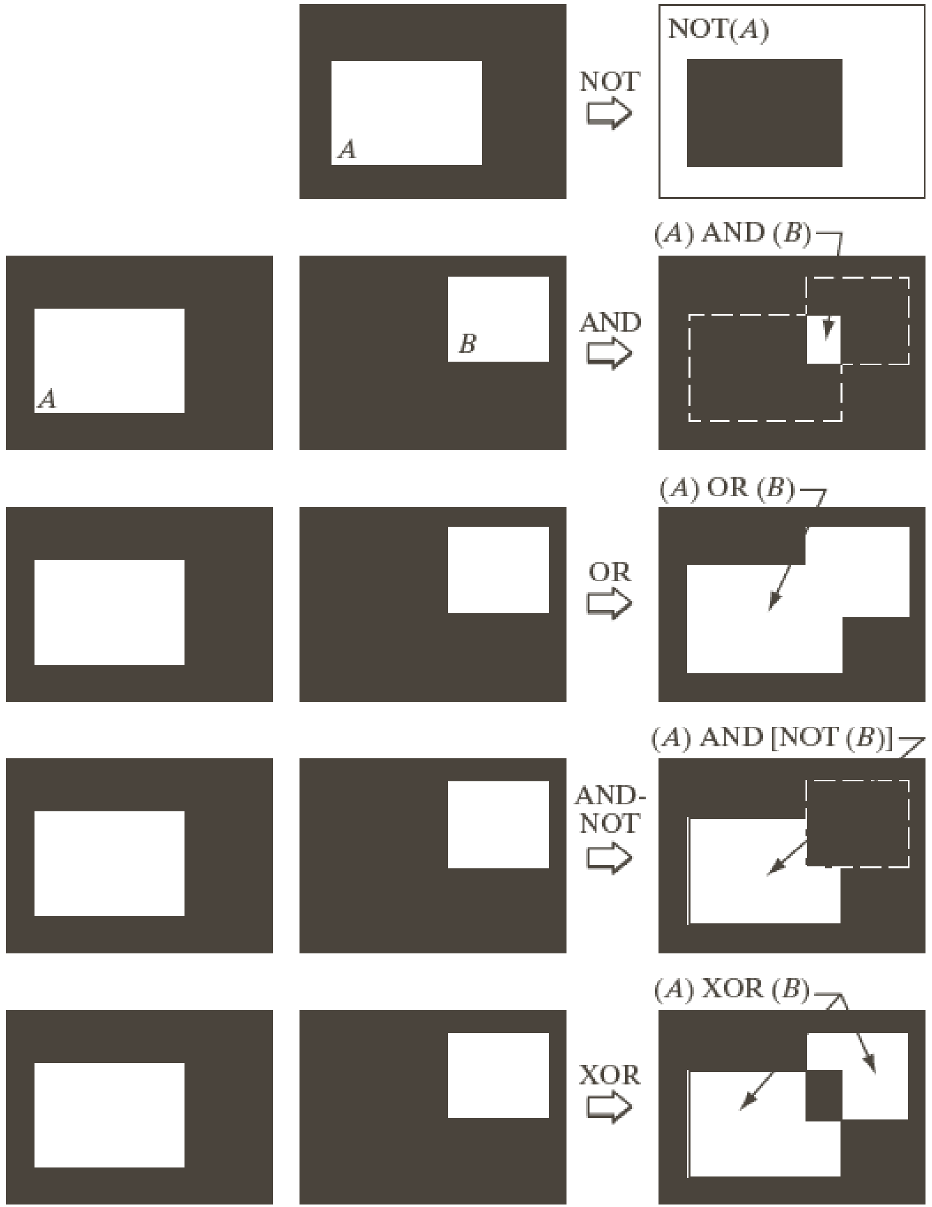


FIGURE 2.33 Illustration of logical operations involving foreground (white) pixels. Black represents binary 0s and white binary 1s. The dashed lines are shown for reference only. They are not part of the result.

Spatial Operations

- Single-pixel operations

Alter the values of an image’s pixels based on the intensity.

$$s = T(z)$$

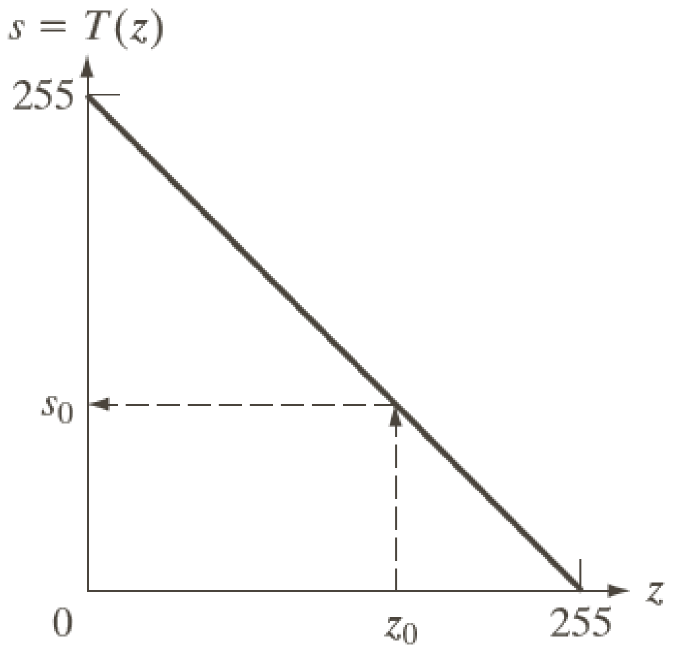
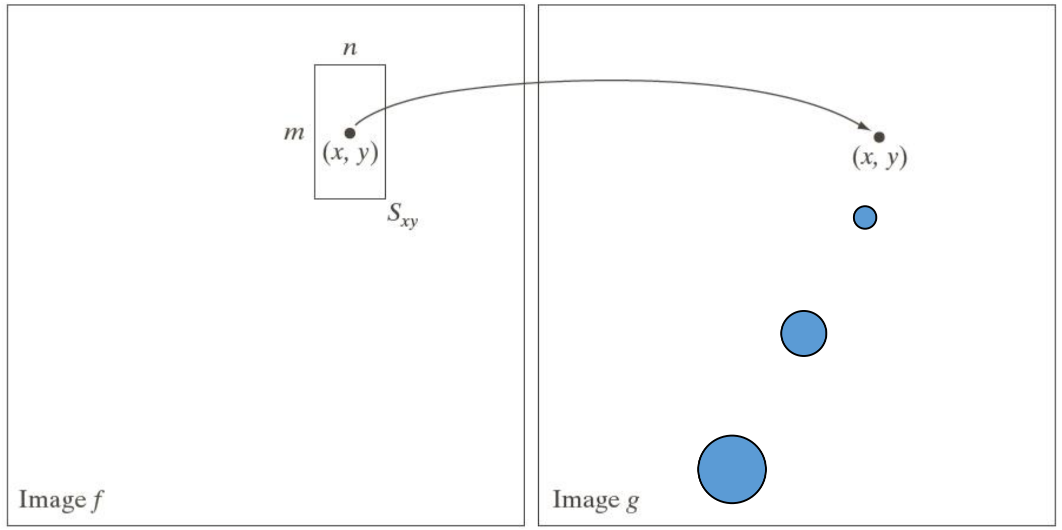


FIGURE 2.34 Intensity transformation function used to obtain the negative of an 8-bit image. The dashed arrows show transformation of an arbitrary input intensity value z_0 into its corresponding output value s_0.

Spatial Operations

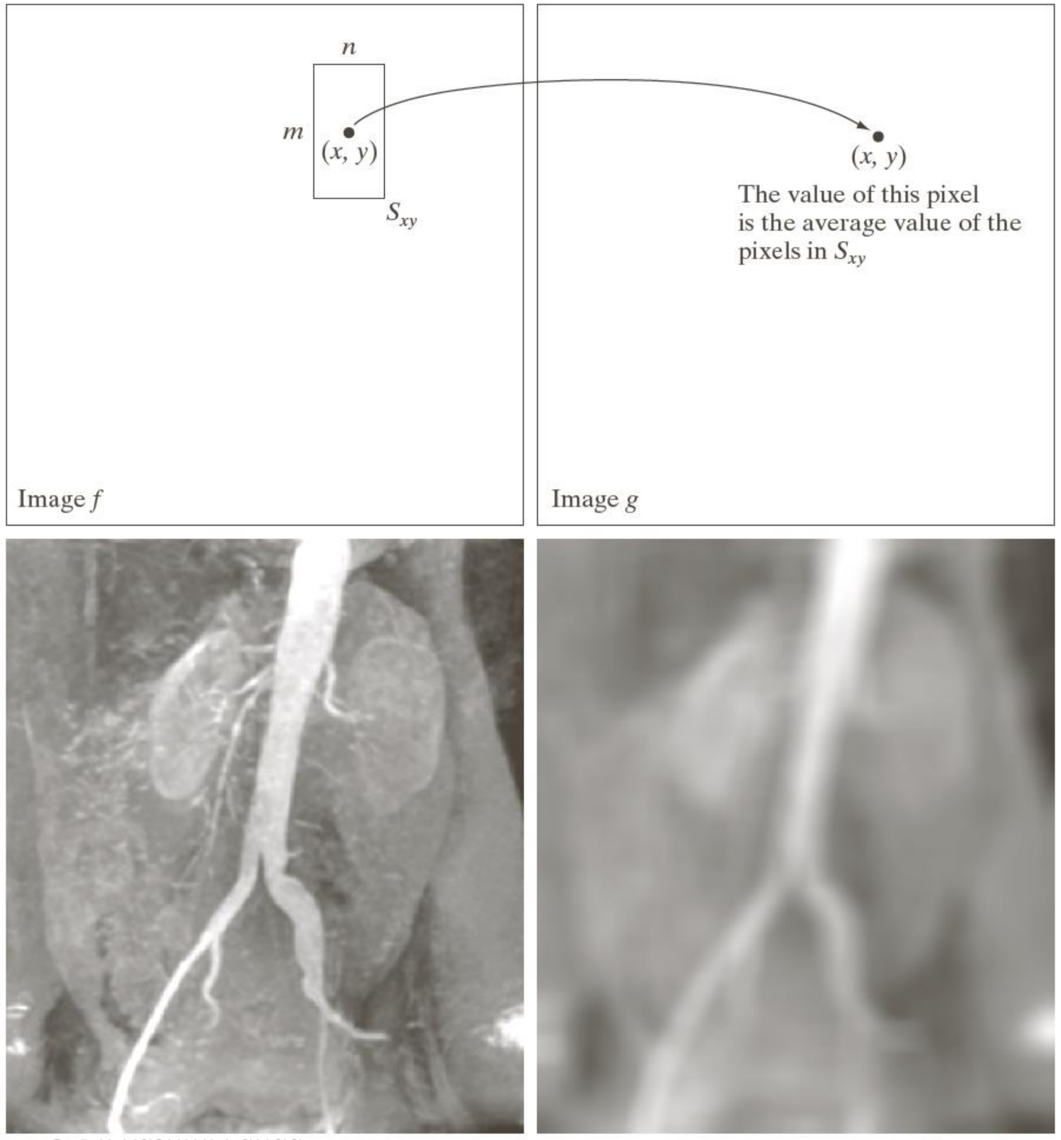
- Neighborhood operations



The value of this pixel is determined by a specified operation involving the pixels in the input image with coordinates in S_{xy}

Spatial Operations

- Neighborhood operations


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

Geometric Spatial Transformations

- Geometric transformation (rubber-sheet transformation)

  — A spatial transformation of coordinates

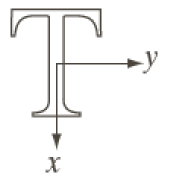
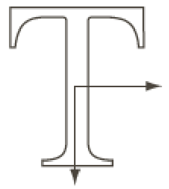
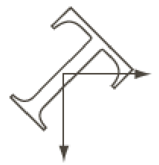
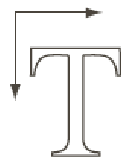
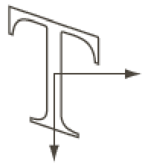
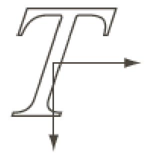
$$(x, y) = T\{(v, w)\}$$

  — intensity interpolation that assigns intensity values to the spatially transformed pixels.

- Affine transform

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

TABLE 2.2
Affine transformations based on Eq. (2.6.–23).

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Intensity Assignment

- **Forward Mapping**

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

It’s possible that two or more pixels can be transformed to the same location in the output image.

- **Inverse Mapping**

$$(v, w) = T^{-1}\{(x, y)\}$$

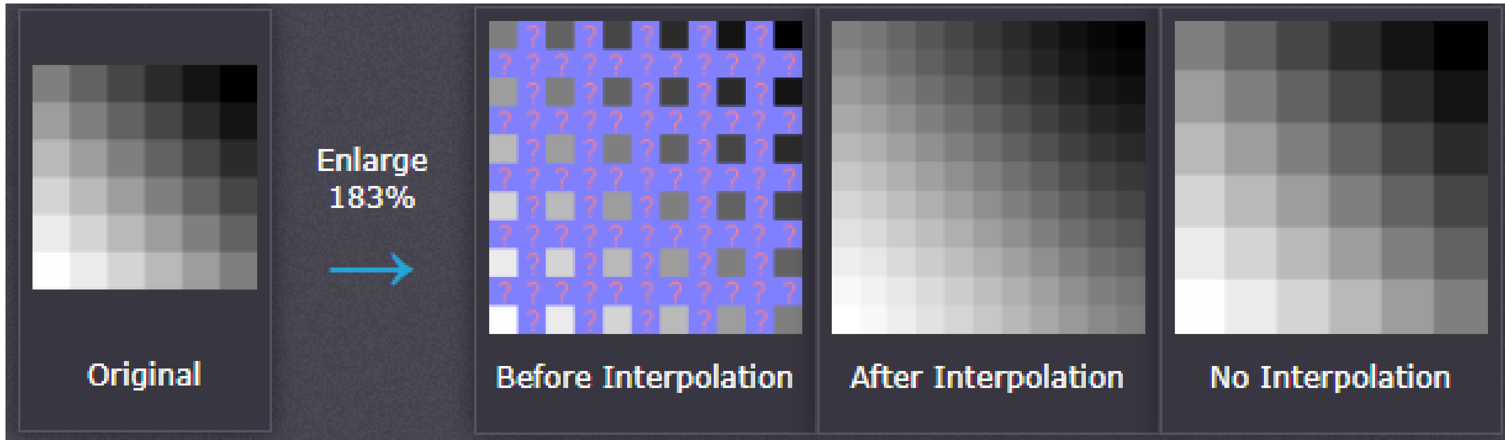
Image Interpolation

- **Interpolation** — Process of using known data to estimate unknown values

  e.g., zooming, shrinking, rotating, and geometric correction

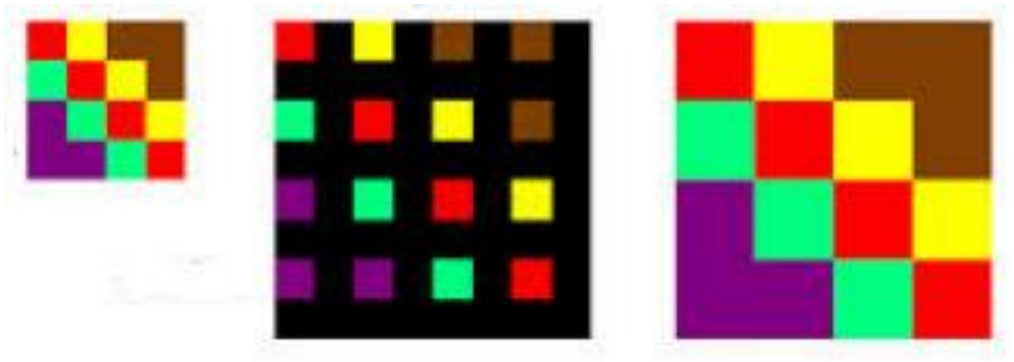
- **Interpolation** (sometimes called ***resampling***) — an imaging method to increase (or decrease) the number of pixels in a digital image.

  Some digital cameras use interpolation to produce a larger image than the sensor captured or to create digital zoom

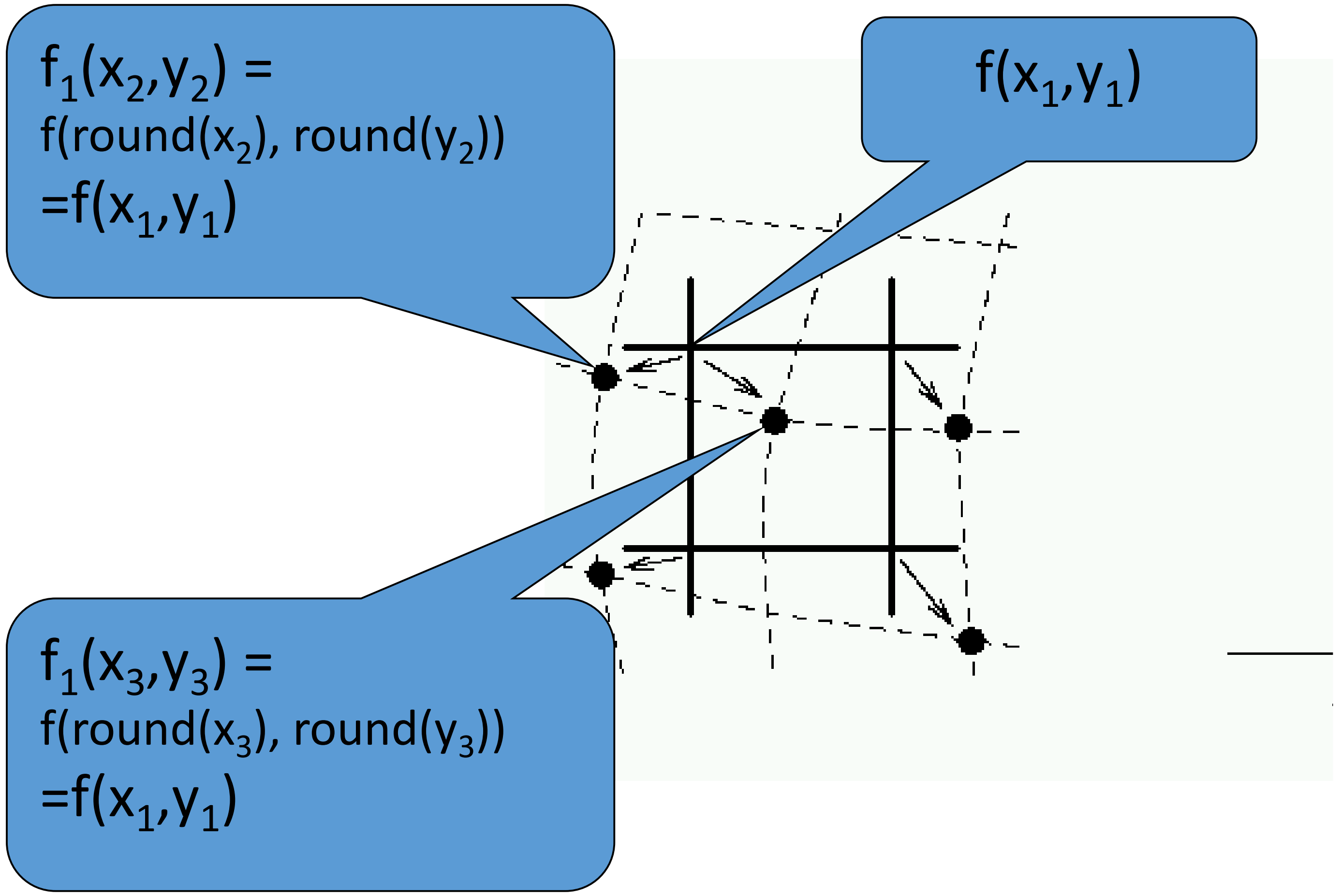


Nearest Neighbor Interpolation

- Most basic method Requires the least processing time
- Only considers one pixel: the closest one to the interpolated point
- Has the effect of simply making each pixel bigger

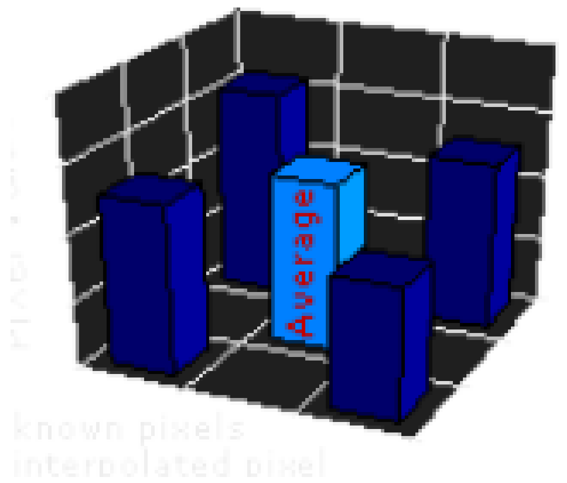


Nearest Neighbor Interpolation

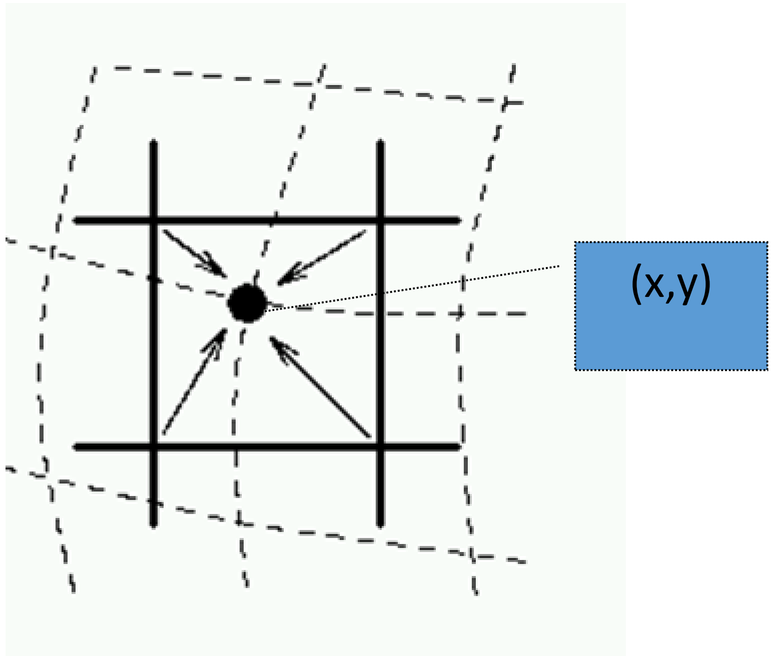

$f_1(x_2,y_2) =$
$f(\text{round}(x_2), \text{round}(y_2))$
$=f(x_1,y_1)$
$f(x_1,y_1)$
$f_1(x_3,y_3) =$
$f(\text{round}(x_3), \text{round}(y_3))$
$=f(x_1,y_1)$

Bilinear Interpolation

- Considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixels
- Takes a weighted average of these 4 pixels to arrive at the final interpolated values
- Results in smoother looking images than nearest neighborhood
- Needs of more processing time

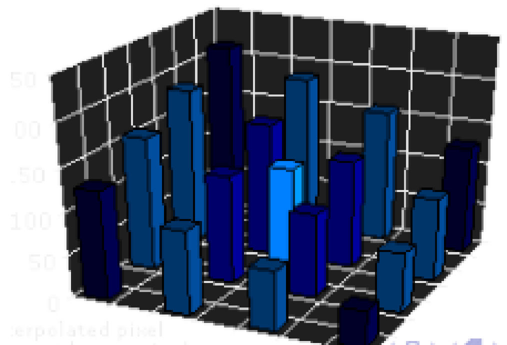


Bilinear Interpolation



Bicubic Interpolation

- One step beyond bilinear by considering the closest 4x4 neighborhood of known pixels, for a total of 16 pixels
- Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation
- Produces sharper images than the previous two methods.
- Good compromise between processing time and output quality
- Standard in many image editing programs, printer drivers and in-camera interpolation



Bicubic Interpolation

- The intensity value assigned to point (x,y) is obtained by the following equation

$$f_3(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j$$

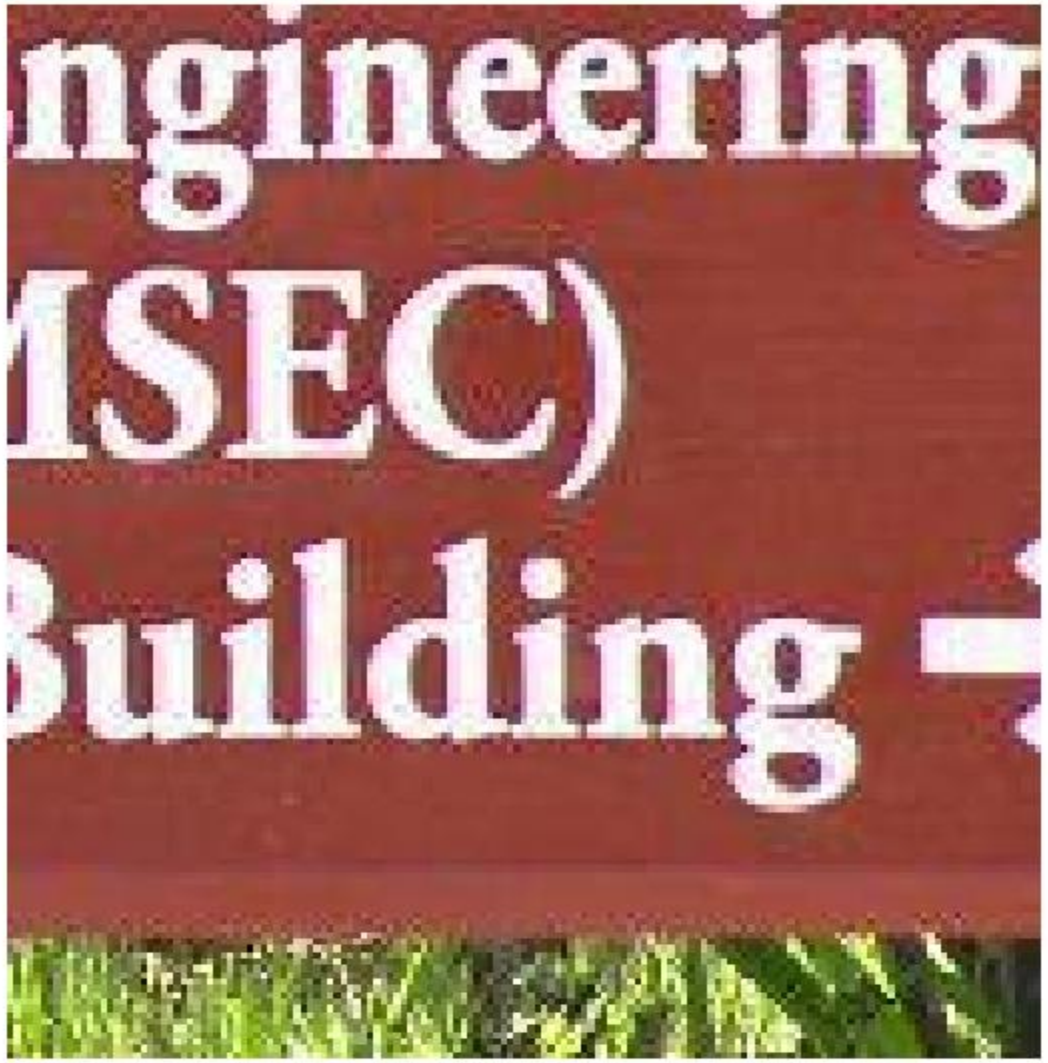
- The sixteen coefficients are determined by using the sixteen nearest neighbors.

Examples: Interpolation

Original Image

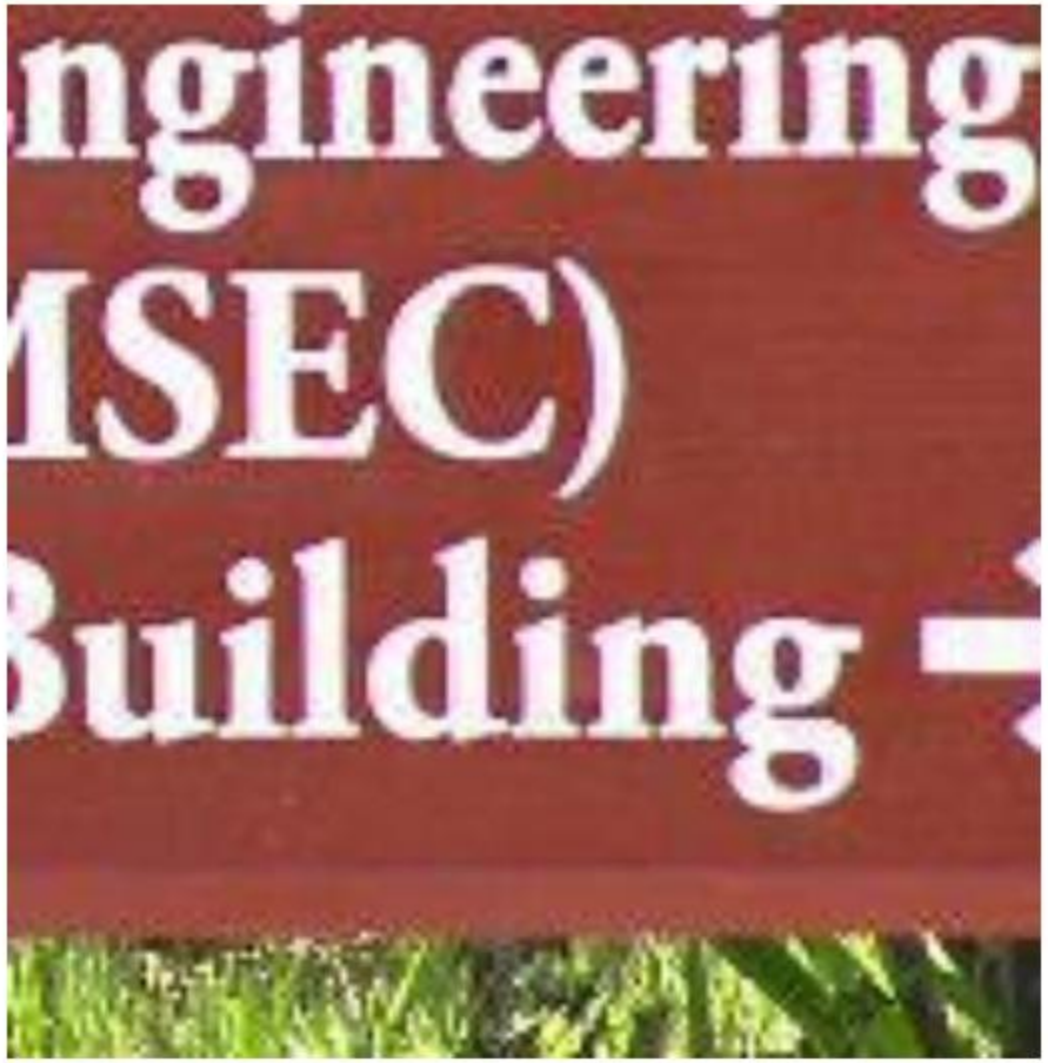
Examples: Interpolation

Nearest Neighbor Interpolation



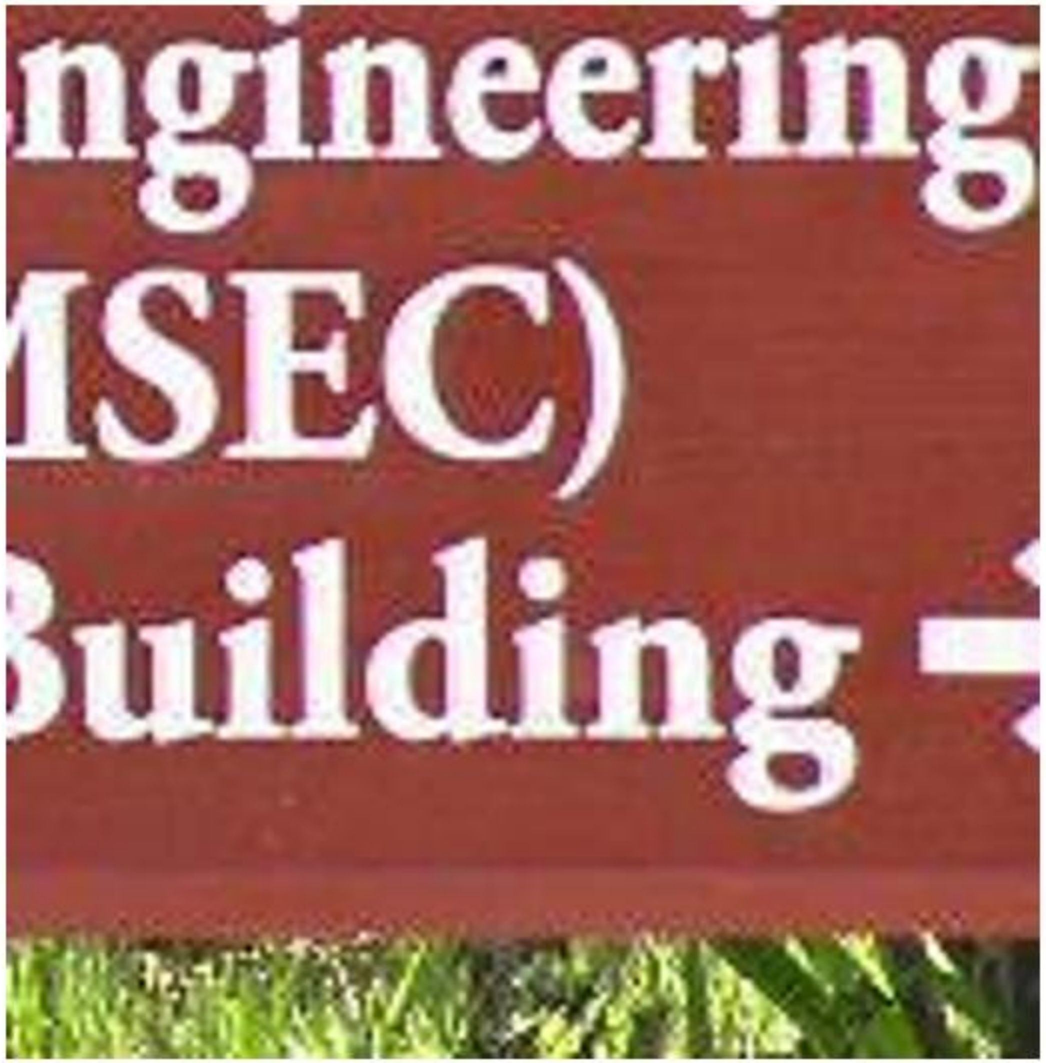
Examples: Interpolation

Bilinear Interpolation



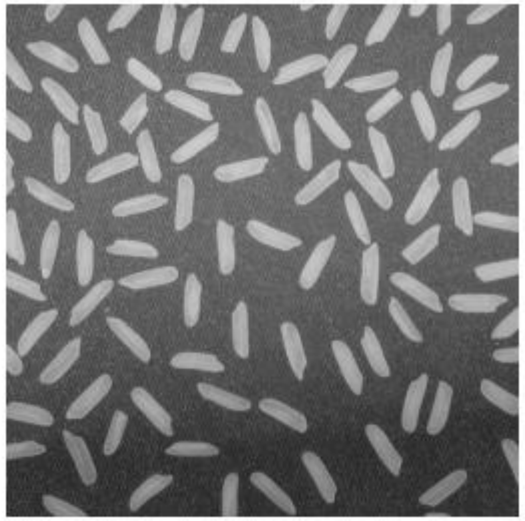
Examples: Interpolation

Bicubic Interpolation

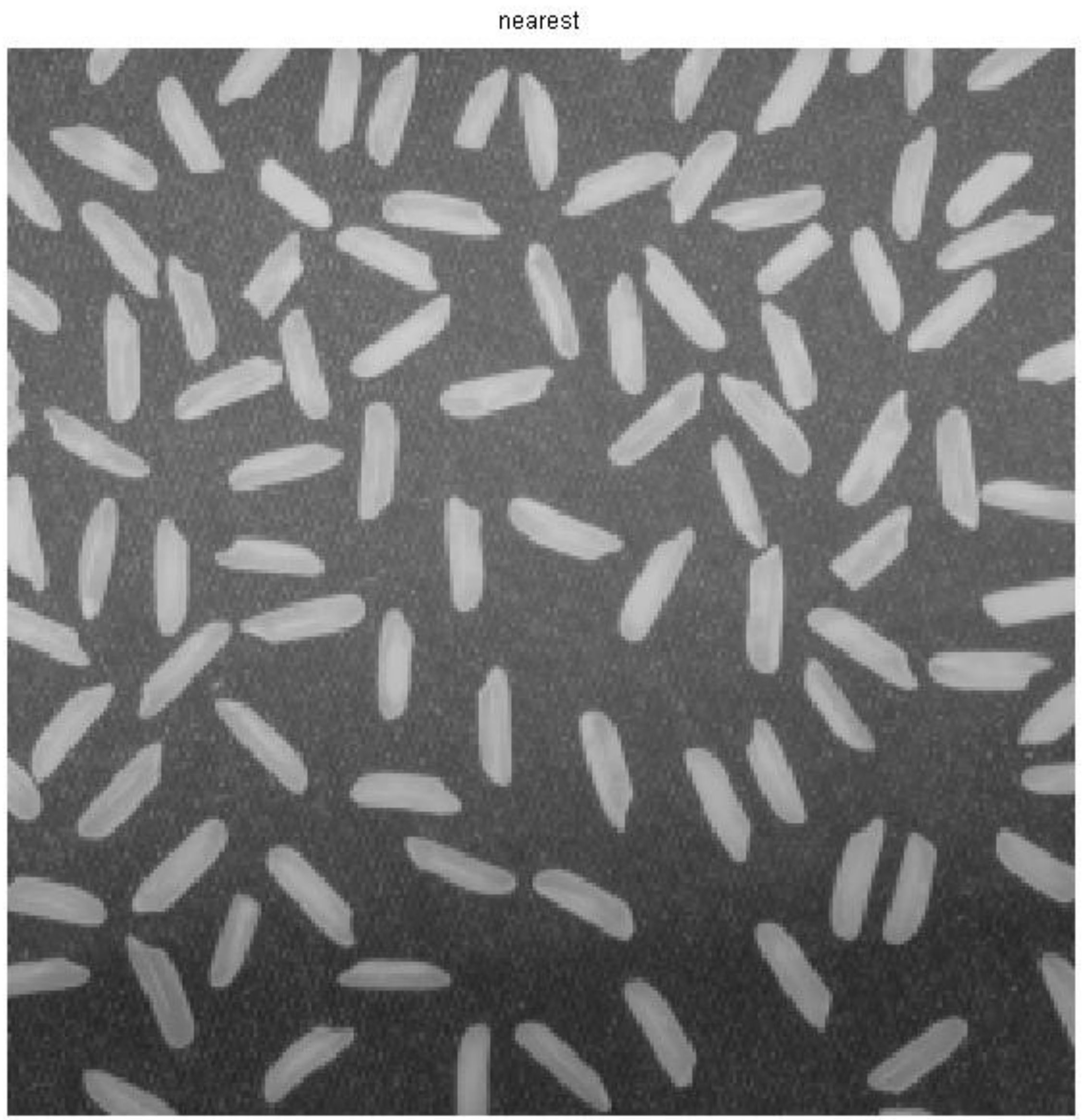


Examples: Interpolation

original image

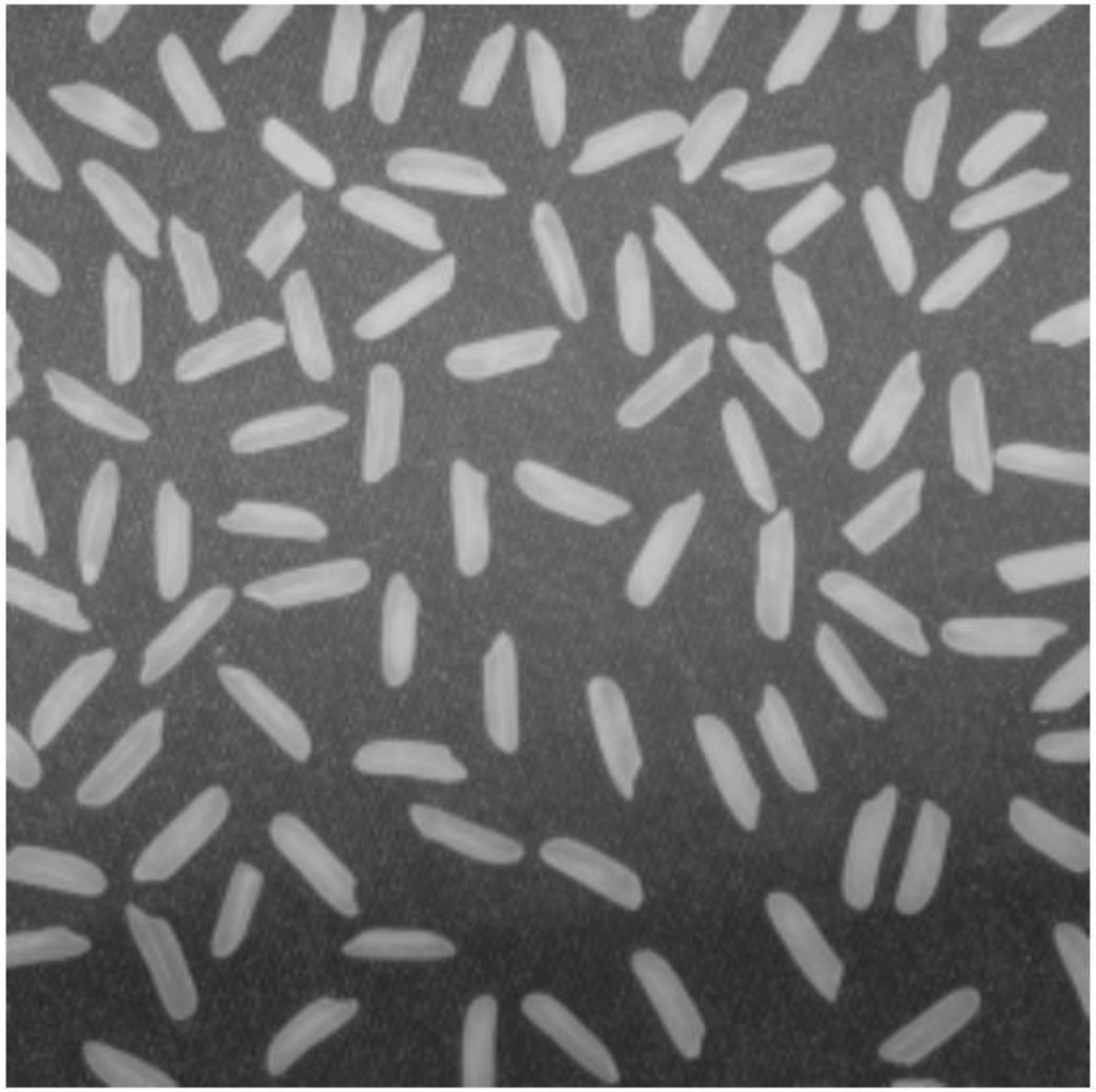


Examples: Interpolation



Examples: Interpolation

bilinear



Examples: Interpolation

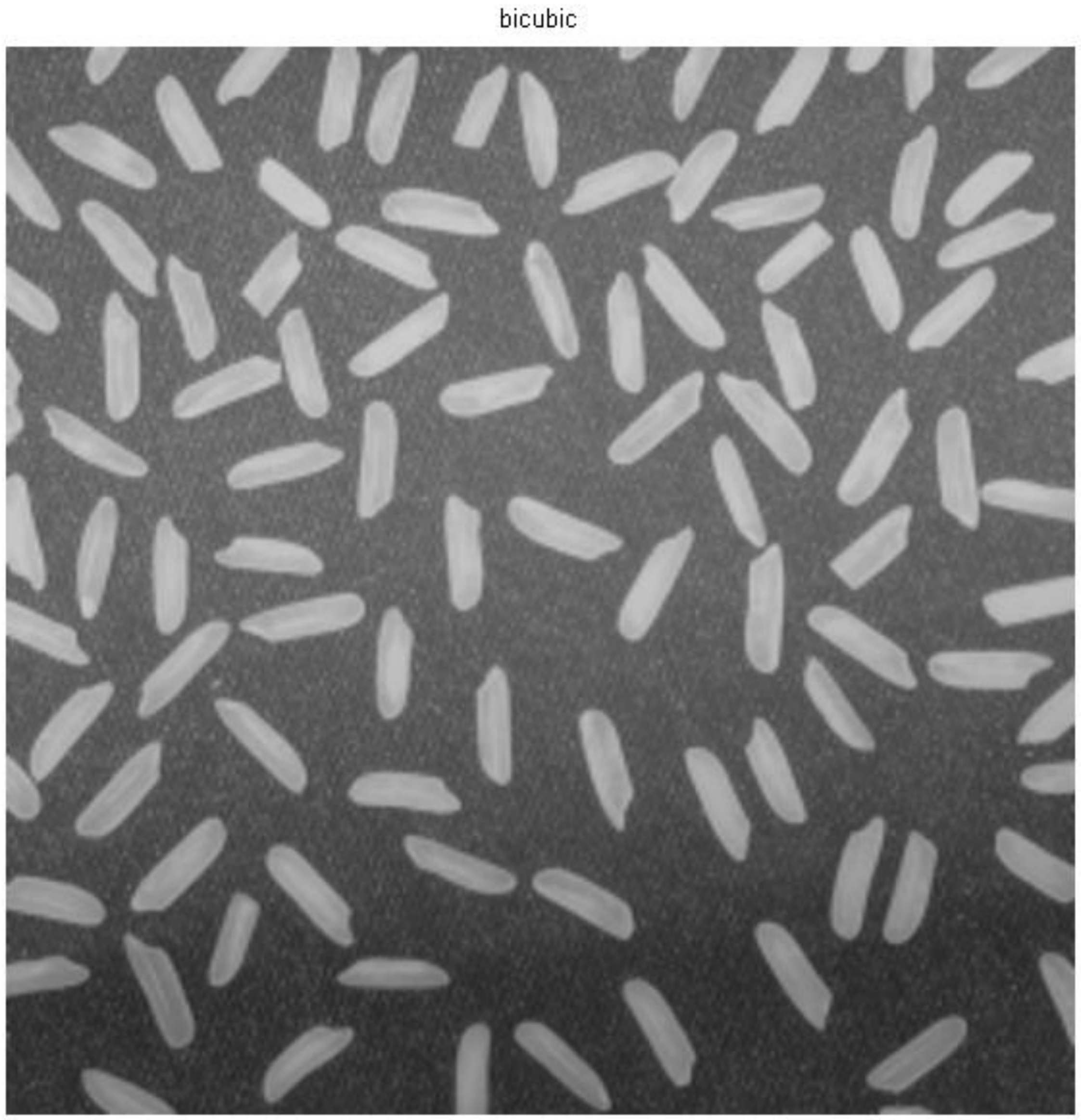


Image Registration

- Input and output images are available but the transformation function is unknown.

  Goal: estimate the transformation function and use it to register the two images.

- One of the principal approaches for image registration is to use ***tie points*** (also called ***control points***)

➢ The corresponding points are known precisely in the input and output (**reference**) images.

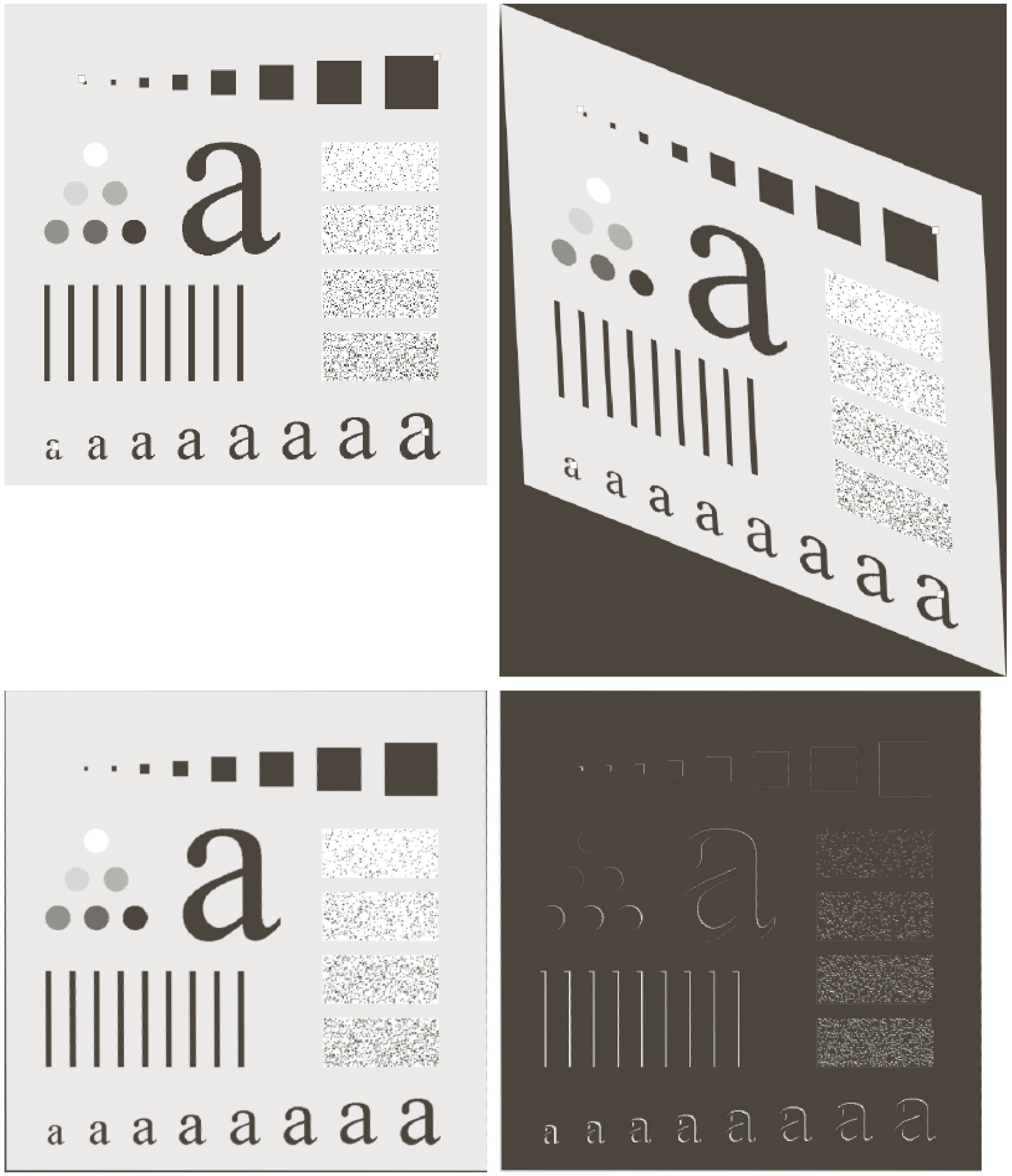


FIGURE 2.37 Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered image (note the errors in the borders). (d) Difference between (a) and (c), showing more registration errors.

Thank you !

© Dr. Rashmi Panda