FPT Education

FPT UNIVERSITY

PART I
IMAGE PROCESSING

INSTRUCTOR: DR. NGUYEN NGOC TRUONG MINH

SCHOOL OF ELECTRICAL ENGINEERING, INTERNATIONAL UNIVERSITY (VNU-HCMC)

Ho Chi Minh City, June 2023

LECTURE II – IMAGE PROCESSING BASICS

INSTRUCTOR: DR. NGUYEN NGOC TRUONG MINH

SCHOOL OF ELECTRICAL ENGINEERING, INTERNATIONAL UNIVERSITY (VNU-HCMC)

Ho Chi Minh City, June 2023

LECTURE CONTENT

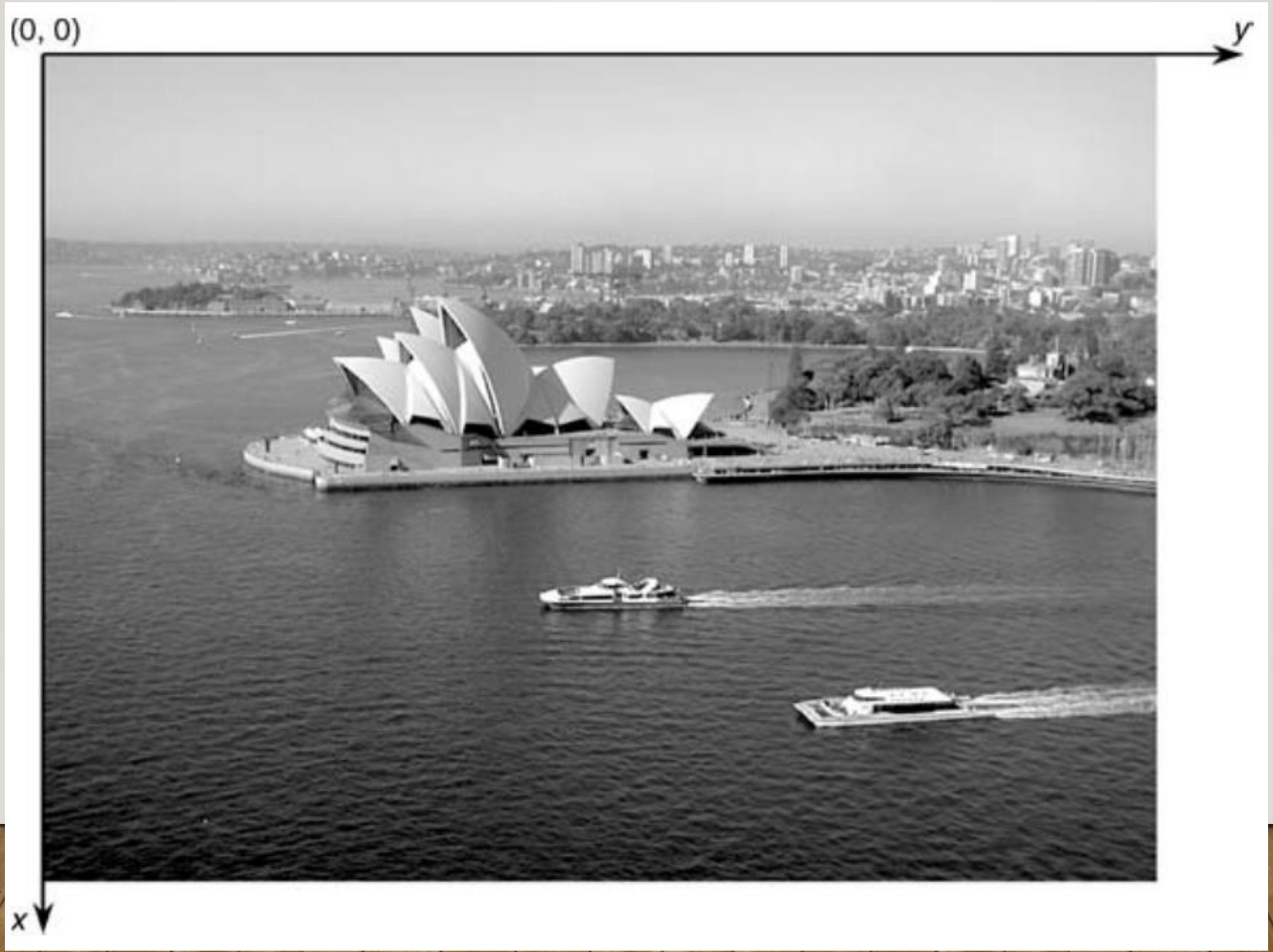
- How is a digital image represented and stored in memory?
- What are the main types of digital image representation?
- What are the most popular image file formats?
- What are the most common types of image processing operations and how do they affect pixel values?
- Lecture Summary – What have we learned?
- Problems

2.1 DIGITAL IMAGE REPRESENTATION

- A digital image—whether it was obtained as a result of *sampling* and *quantization* of an analog image or created already in digital form—can be represented as a two-dimensional (2D) matrix of real numbers.
- In this lecture, we adopt the convention $f(x, y)$ to refer to monochrome images of size $M \times N$, where x denotes the row number (from 0 to $M - 1$) and y represents the column number (between 0 and $N - 1$):

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

2.1 DIGITAL IMAGE REPRESENTATION

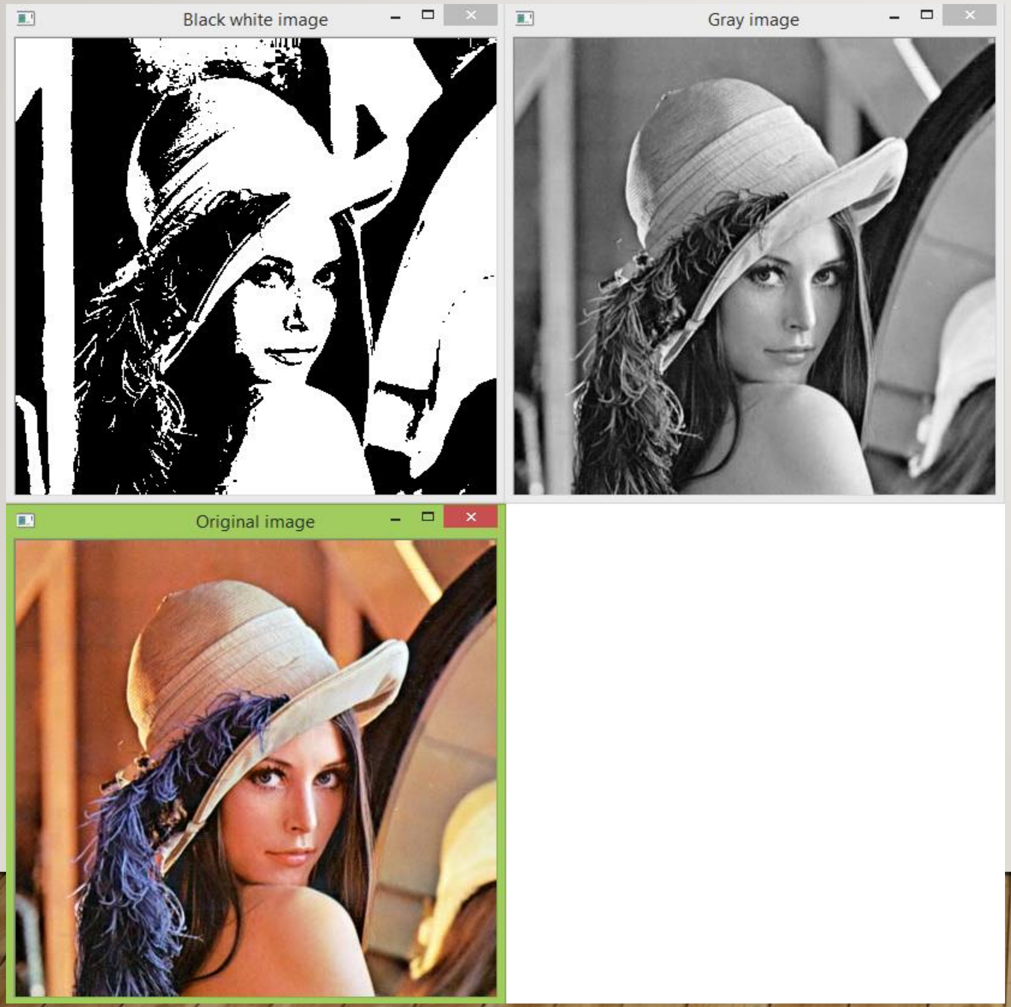


2.1 DIGITAL IMAGE REPRESENTATION

- The value of the two-dimensional function $f(x, y)$ at any given pixel of coordinates (x_0, y_0), denoted by $f(x_0, y_0)$, is called the intensity or gray level of the image at that pixel. The maximum and minimum values that a pixel intensity can assume will vary depending on the data type and convention used. Common ranges are as follows: *0.0 (black) to 1.0 (white)* for *double* data type and *0 (black) to 255 (white)* for *uint8* (unsigned integer, 8 bits) representation.
- MATLAB and its Image Processing Toolbox (IPT), which use 1-based array notation:

$$f(p,q) = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

2.1 DIGITAL IMAGE REPRESENTATION



Black and White (*line 1 – left*)
Monochrome Grayscale (*line 1 – right*)
Color (*line 2 – left*)

2.1 DIGITAL IMAGE REPRESENTATION

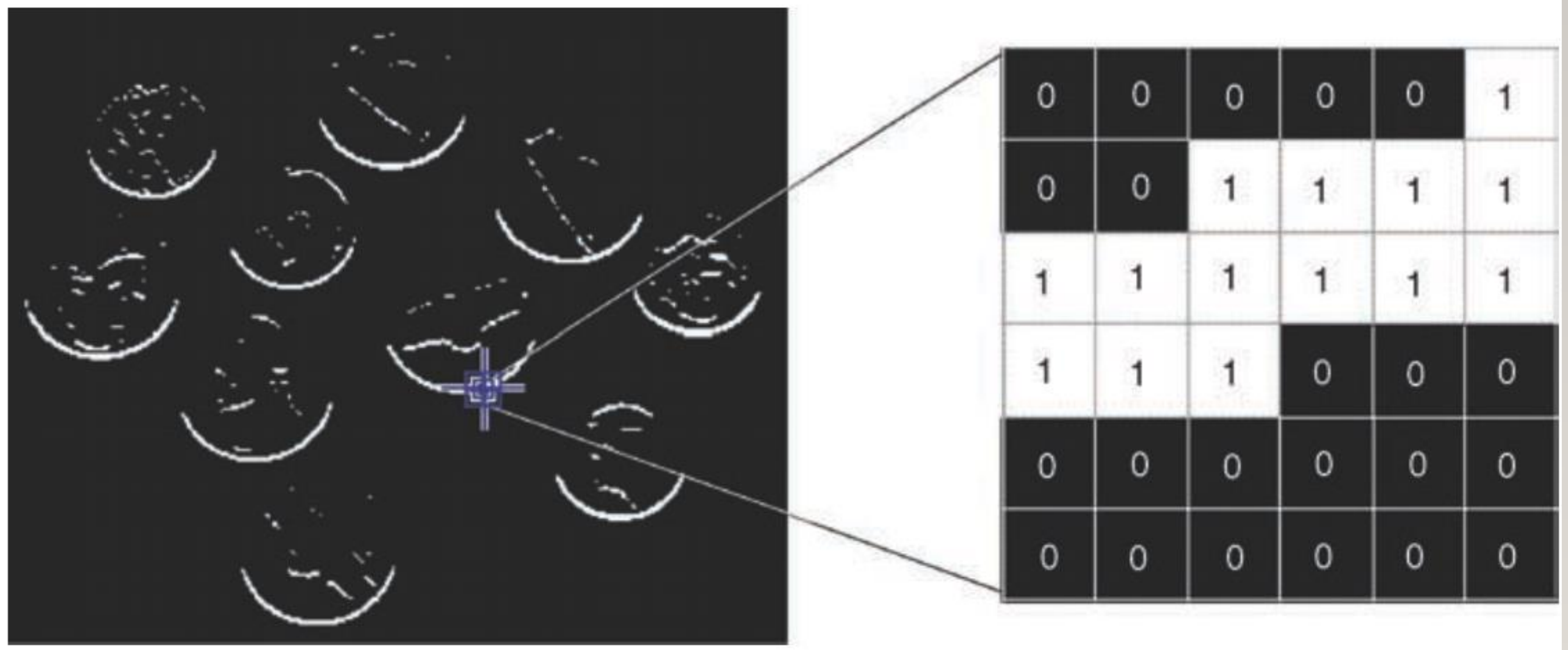
- Monochrome images are essentially *2D matrices* (or *arrays*). Since MATLAB treats matrices as a built-in data type, it is easier to manipulate them without resorting to common programming language constructs.
- Images are represented in digital format in a variety of ways. At the most basic level, there are two different ways of encoding the contents of a 2D image in digital format: *raster* (also known as bitmap) and *vector*.

2.1 DIGITAL IMAGE REPRESENTATION

- Bitmap representations use *one or more two-dimensional arrays of pixels*, whereas vector representations use a series of drawing commands to represent an image.
  - Each encoding method has its pros and cons: the greatest advantages of bitmap graphics are *their quality and display speed*; their main disadvantages include *larger memory storage requirements* and *size dependence* (e.g., enlarging a bitmap image may lead to noticeable artifacts). Vector representations require *less memory and allow resizing and geometric manipulations without introducing artifacts* but *need to be rasterized for most presentation devices*.

2.1.1 BINARY (1-BIT) IMAGES

- Binary images are encoded as a 2D array, typically using 1 bit per pixel, where *a 0 usually means "black" and a 1 means "white"* (although there is no universal agreement on that).
- The main advantage of this representation—usually suitable for images containing simple graphics, text, or line art—is *its small size*.

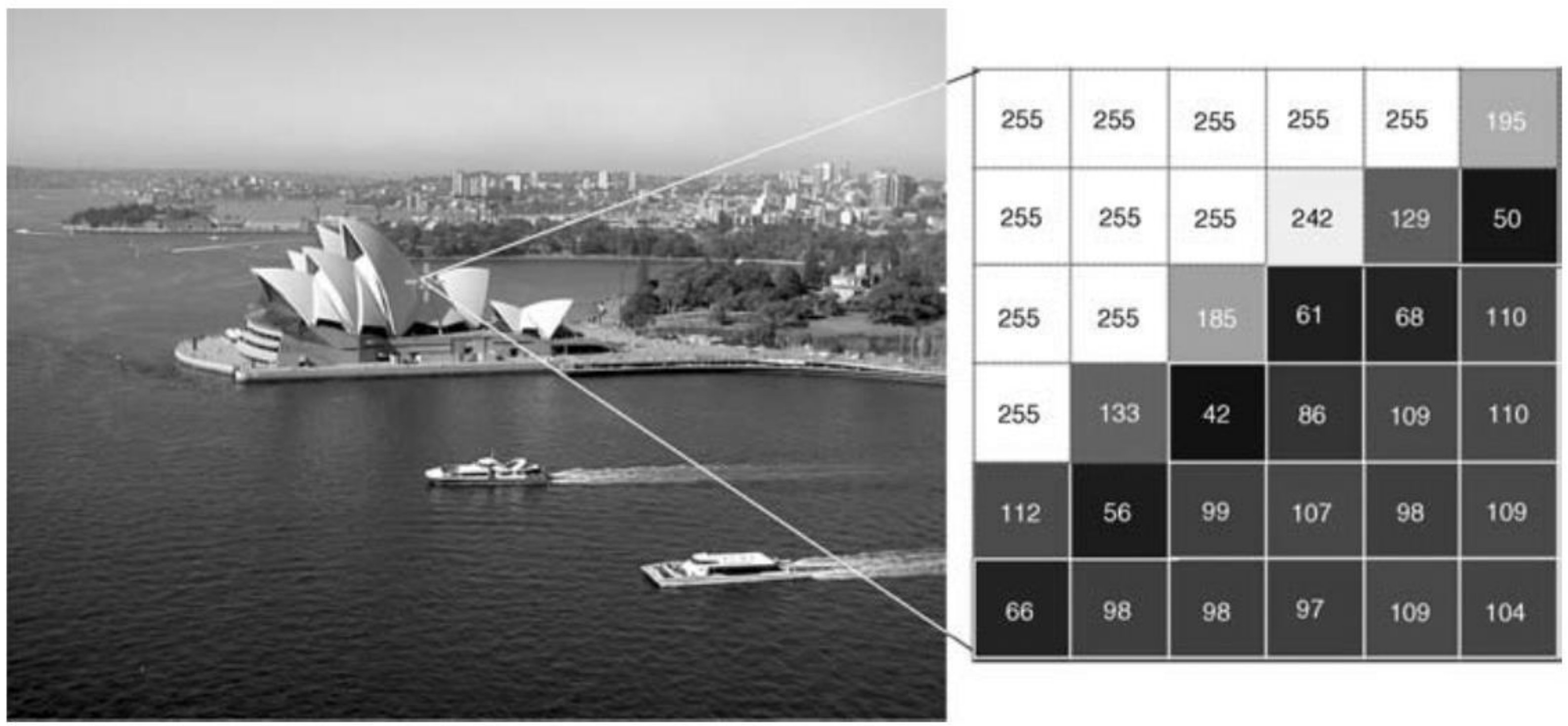


0	0	0	0	0	1
0	0	1	1	1	1
1	1	1	1	1	1
1	1	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

FIGURE 2.2 A binary image and the pixel values in a 6 × 6 neighborhood. Original image: courtesy of MathWorks.

2.1.2 GRAY-LEVEL (8-BIT) IMAGES

- Gray-level (also referred to as monochrome) images are also encoded as a 2D array of pixels, usually with 8 bits per pixel, where *a pixel value of 0 corresponds to "black," a pixel value of 255 means "white,"* and intermediate values indicate varying shades of gray. *The total number of gray levels is larger than the human visual system requirements.*



255	255	255	255	255	195
255	255	255	242	129	50
255	255	185	61	68	110
255	133	42	86	109	110
112	56	99	107	98	109
66	98	98	97	109	104

FIGURE 2.3 A grayscale image and the pixel values in a 6 × 6 neighborhood.

2.1.3 COLOR IMAGES

- Representation of color images is more complex and varied. The two most common ways of storing color image contents are *RGB representation*—in which *each pixel is usually represented by a 24-bit number containing the amount of its red (R), green (G), and blue (B) components*—and *Indexed representation*—where a 2D array contains indices to *a color palette (or lookup table - (LUT))*.

✓ ***24-Bit (RGB) Color Images:*** *Color images can be represented using three 2D arrays of same size, one for each color channel: red (R), green (G), and blue (B).*

✓ *Each array element contains an 8-bit value, indicating the amount of red, green, or blue at that point in a [0, 255] scale. The combination of the three 8-bit values into a 24-bit number allows 2^{24} (16,777,216, usually referred to as 16 million or 16M) color combinations.*

2.1.3 COLOR IMAGES

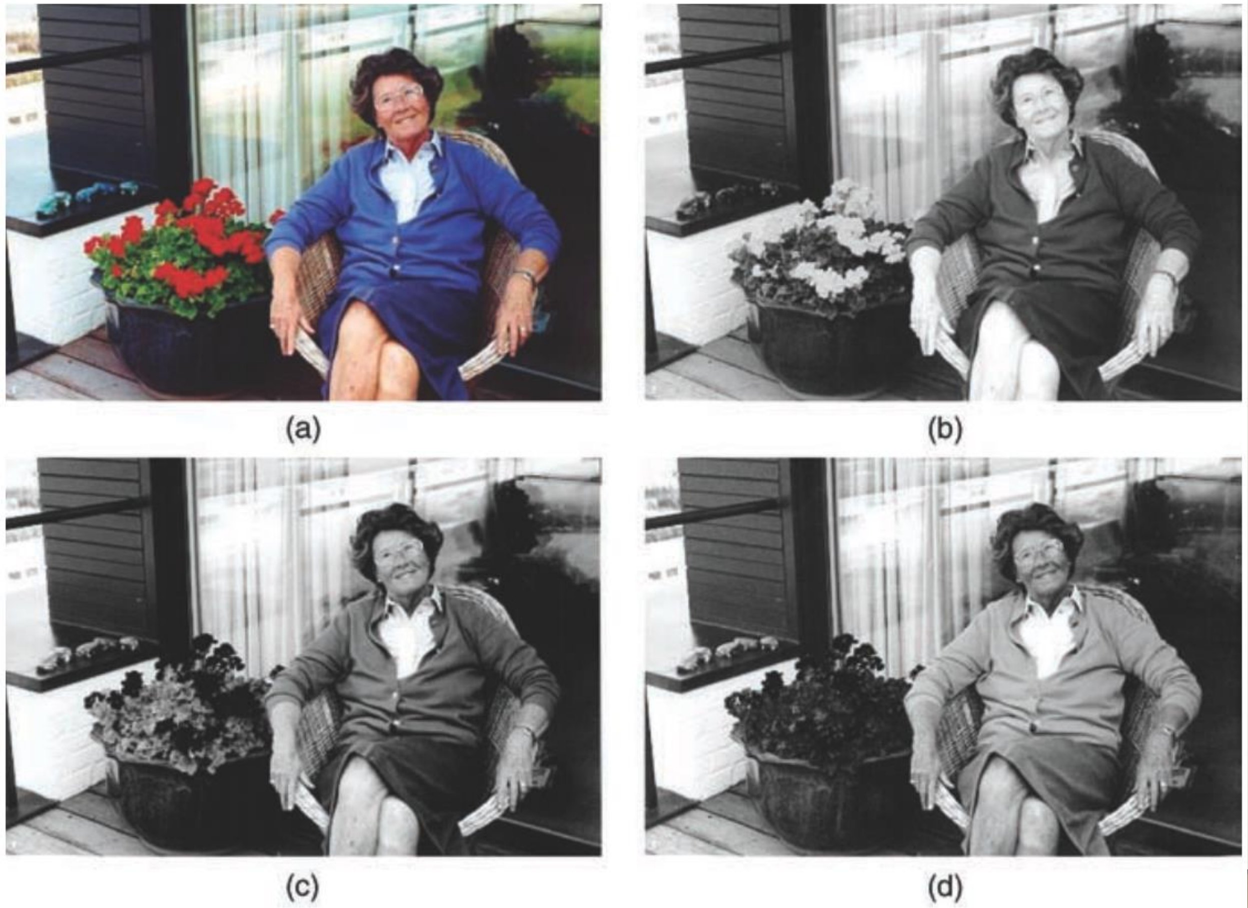


FIGURE 2.4 Color image (a) and its R (b), G (c), and B (d) components.

2.1.3 COLOR IMAGES

- Indexed Color Images: A problem with 24-bit color representations is backward compatibility with older hardware that may not be able to display the 16 million colors simultaneously. A solution—*devised before 24-bit color displays and video cards were widely available*—consisted of an indexed representation, in which a 2D array of the same size as the image contains indices (*pointers*) to a color palette (or *color map*) of fixed maximum size (usually 256 colors). The color map is simply a list of colors used in that image.

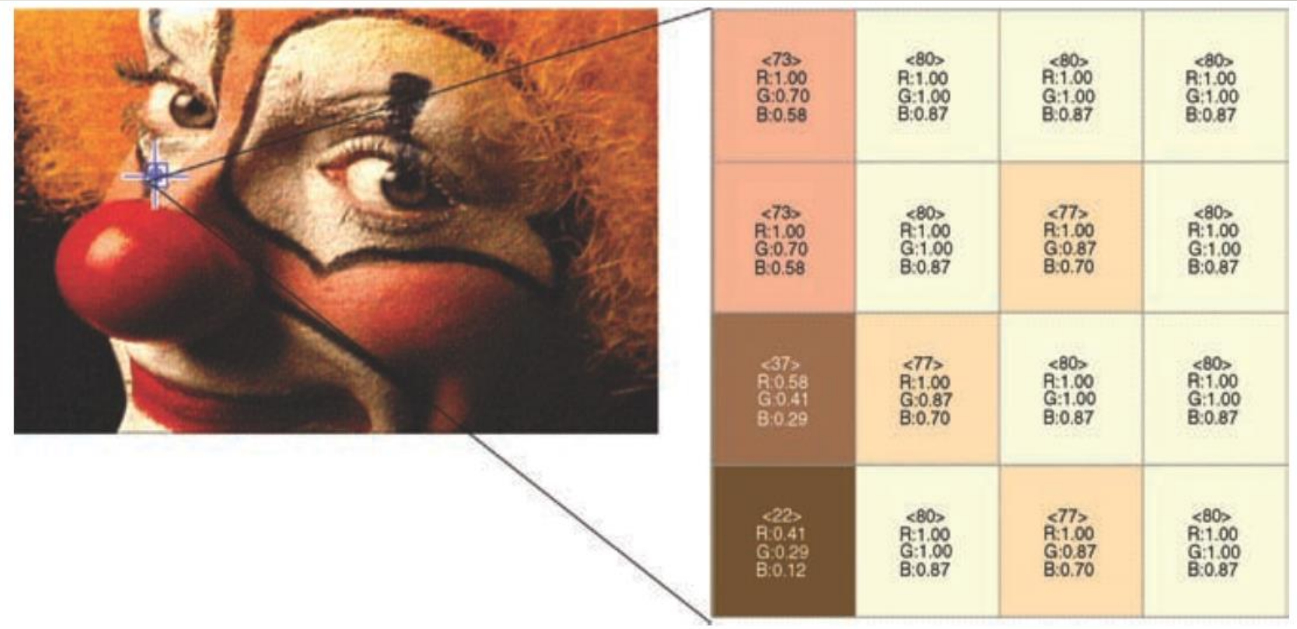

<73> R:1.00 G:0.70 B:0.58
<80> R:1.00 G:1.00 B:0.87
<80> R:1.00 G:1.00 B:0.87
<80> R:1.00 G:1.00 B:0.87
<73> R:1.00 G:0.70 B:0.58
<80> R:1.00 G:1.00 B:0.87
<77> R:1.00 G:0.87 B:0.70
<80> R:1.00 G:1.00 B:0.87
<37> R:0.58 G:0.41 B:0.29
<77> R:1.00 G:0.87 B:0.70
<80> R:1.00 G:1.00 B:0.87
<80> R:1.00 G:1.00 B:0.87
<22> R:0.41 G:0.29 B:0.12
<80> R:1.00 G:1.00 B:0.87
<77> R:1.00 G:0.87 B:0.70
<80> R:1.00 G:1.00 B:0.87

FIGURE 2.5 An indexed color image and the indices in a 4 × 4 neighborhood. Original image: courtesy of MathWorks.

2.1.4 COMPRESSIONS

- Since raw image representations usually require *a large amount of storage space* (and proportionally long transmission times in the case of file uploads/downloads), most image file formats employ some type of compression.
- Compression methods can be *lossy*—when a tolerable degree of deterioration in the visual quality of the resulting image is acceptable—or *lossless*—when the image is encoded in its full quality.
- The overall results of the compression process in terms of both storage savings—usually expressed in terms of compression ratio or *bits per pixel* (*bpp*)—and resulting quality loss (for the case of lossy techniques) may vary depending on the technique, format, options (such as the quality setting for JPEG), and actual image contents.

2.2 IMAGE FILE FORMATS

- Most of the image file formats used to represent bitmap images consist of a file header followed by (often compressed) *pixel data*. The image file header stores information about the image, such as *image height and width*, *number of bands*, *number of bits per pixel*, and *some signature bytes* indicating the file type.
- The simplest file formats are the BIN and PPM formats. The BIN format simply consists of *the raw pixel data*, without any header. The PPM format and its variants (PBM for binary images, PGM for grayscale images, PPM for color images, and PNM for any of them) are widely used in image processing research and many free tools for *format conversion include them*.

2.3 BASIC TERMINOLOGY

- *Image Topology* It involves the investigation of *fundamental image properties*—usually done on binary images and with *the help of morphological operators*—such as number of occurrences of a particular object, number of separate (not connected) regions, and number of holes in an object, to mention but a few.
- *Neighborhood* The pixels surrounding a given pixel constitute *its neighborhood*, which can be interpreted as *a smaller matrix* containing (and usually centered around) *the reference pixel.* Most neighborhoods used in image processing algorithms are *small square arrays with an odd number of pixels*.

2.3 BASIC TERMINOLOGY

- In the context of image topology, neighborhood takes *a slightly different meaning*. It is common to refer to *the 4-neighborhood of a pixel* as the set of pixels situated above, below, to the right, and to the left of *the reference pixel* (*p*), whereas the set of all of *p*'s immediate neighbors is referred to as *its 8-neighborhood*. The pixels that belong to the 8-neighborhood, but not to the 4-neighborhood, make up *the diagonal neighborhood of p*.

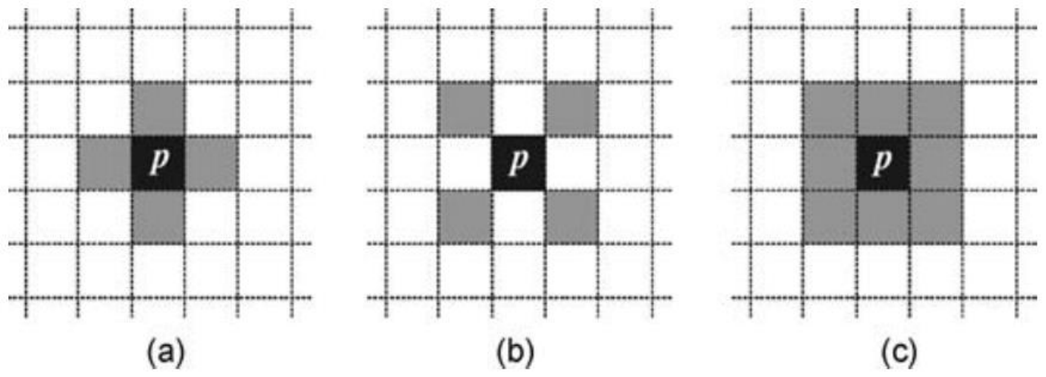


FIGURE 2.7 Concept of neighborhood of pixel *p* (from an image topology perspective): (a) 4-neighborhood; (b) diagonal neighborhood; (c) 8-neighborhood.

2.3 BASIC TERMINOLOGY

- ***Adjacency*** In the context of image topology, two pixels *p* and *q* are *4-adjacent* if they are *4-neighbors of each other* and *8-adjacent* if they are *8-neighbors of one another*. A third type of adjacency—known as *mixed adjacency* (or simply *m-adjacency*)—is sometimes used to eliminate ambiguities (i.e., redundant paths) that may arise when 8-adjacency is used.
- ***Paths*** In the context of image topology, a 4-path between two pixels *p* and *q* is *a sequence of pixels starting with p and ending with q* such that each pixel in the sequence is 4-adjacent to its predecessor in the sequence. Similarly, an 8-path indicates that each pixel in the sequence is 8-adjacent to its predecessor.

2.3 BASIC TERMINOLOGY

- *Connectivity* If there is a 4-path between pixels *p* and *q*, they are said to be *4-connected*. Similarly, the existence of an 8-path between them means that they are *8-connected* (link: Neighbors of Pixels)
- *Components* A set of pixels that are connected to each other is called *a component*. If the pixels are 4-connected, *the expression 4-component is used*; if the pixels are 8-connected, *the set is called an 8-component*. Components are often *labeled* (and optionally *pseudo-colored*) in a unique way, resulting in a labeled image, $L(x, y)$, whose pixel values are symbols of a chosen alphabet. The symbol value of a pixel typically denotes *the outcome of a decision made for that pixel*—in this case, the unique number of the component to which it belongs.

2.3 BASIC TERMINOLOGY

- Figure 2.8 shows an example of using *bwlabel* and *label2rgb* and highlights the fact that the number of connected components will vary from 2 (when 8-connectivity is used, Figure 2.8b) to 3 (when 4-connectivity is used, Figure 2.8c).

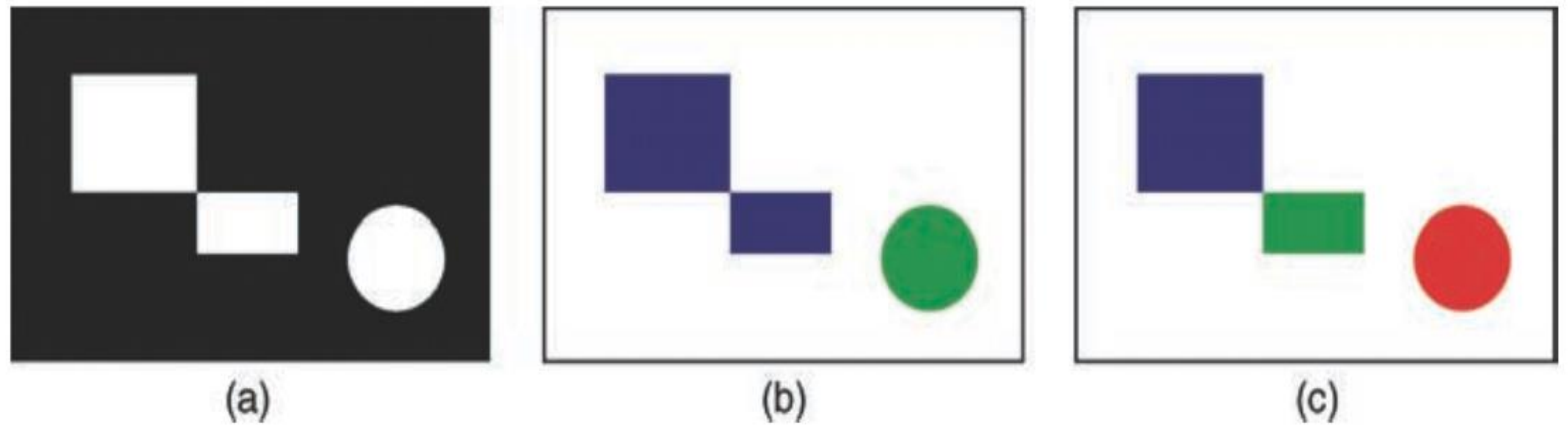


FIGURE 2.8 Connected components: (a) original (binary) image; (b) results for 8-connectivity; (c) results for 4-connectivity.

2.3 BASIC TERMINOLOGY

- *Distances Between Pixels* There are many image processing applications that require measuring distances between pixels. The most common distance measures between two pixels *p* and *q*, of coordinates (x_0, y_0) and (x_1, y_1), respectively, are as follows:
- Euclidean distance:

$$D_e(p,q) = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

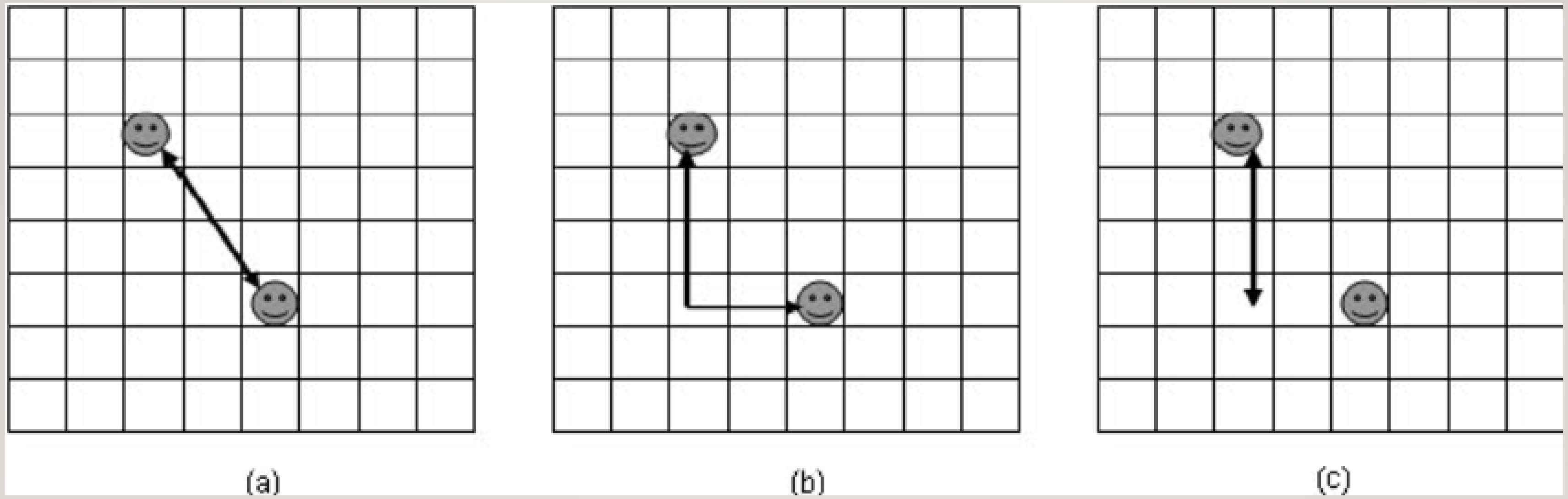
- D_4 (also known as *Manhattan* or *city block*) distance:

$$D_4(p,q) = |x_1 - x_0| + |y_1 - y_0|$$

- D_8 (also known as *chessboard*) distance:

$$D_8(p,q) = \max(|x_1 - x_0|, |y_1 - y_0|)$$

2.3 BASIC TERMINOLOGY



(a) Euclidean; (b) City-Block; and (c) Chess-Board distance

2.4 OVERVIEW OF IMAGE PROCESSING OPERATIONS

- *Operations in the Spatial Domain*: Here, arithmetic calculations and/or logical operations are performed on *the original pixel values*. They can be further divided into three types:
  - *Global Operations:* Also known as point operations, in which the entire image is treated in a uniform manner and the resulting value for a processed pixel is a function of its original value, regardless of its location within the image.
  - *Neighborhood-Oriented Operations:* Also known as local or area operations, in which the input image is treated on a pixel-by-pixel basis and the resulting value for a processed pixel is a function of its original value and the values of its neighbors.
  - *Operations Combining Multiple Images:* Two or more images are used as an input and the result is obtained by applying a (series of) arithmetic or logical operator(s) to them.

2.4 OVERVIEW OF IMAGE PROCESSING OPERATIONS

- Operations in a Transform Domain: The image undergoes a *mathematical transformation*—such as ***Fourier Transform*** (FT) or ***Discrete Cosine Transform*** (DCT)—and the image processing algorithm works in the transform domain.

2.4.1 GLOBAL (POINT) OPERATIONS

- Point operations apply the same mathematical function, often called *transformation function*, to all pixels, regardless of their location in the image or the values of their neighbors. Transformation functions in the spatial domain can be expressed as

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

where $g(x,y)$ is the processed image, $f(x,y)$ is the original image, and T is an operator on $f(x,y)$.

- Since the actual coordinates do not play any role in the way the transformation function processes the original image, a shorthand notation can be used:

$$s = T[r]$$

where r is the original gray level and s is the resulting gray level after processing.

2.4.1 GLOBAL (POINT) OPERATIONS

- Figure 2.9 shows an example of a transformation function used to reduce the overall intensity of an image by half: $s = r/2$.

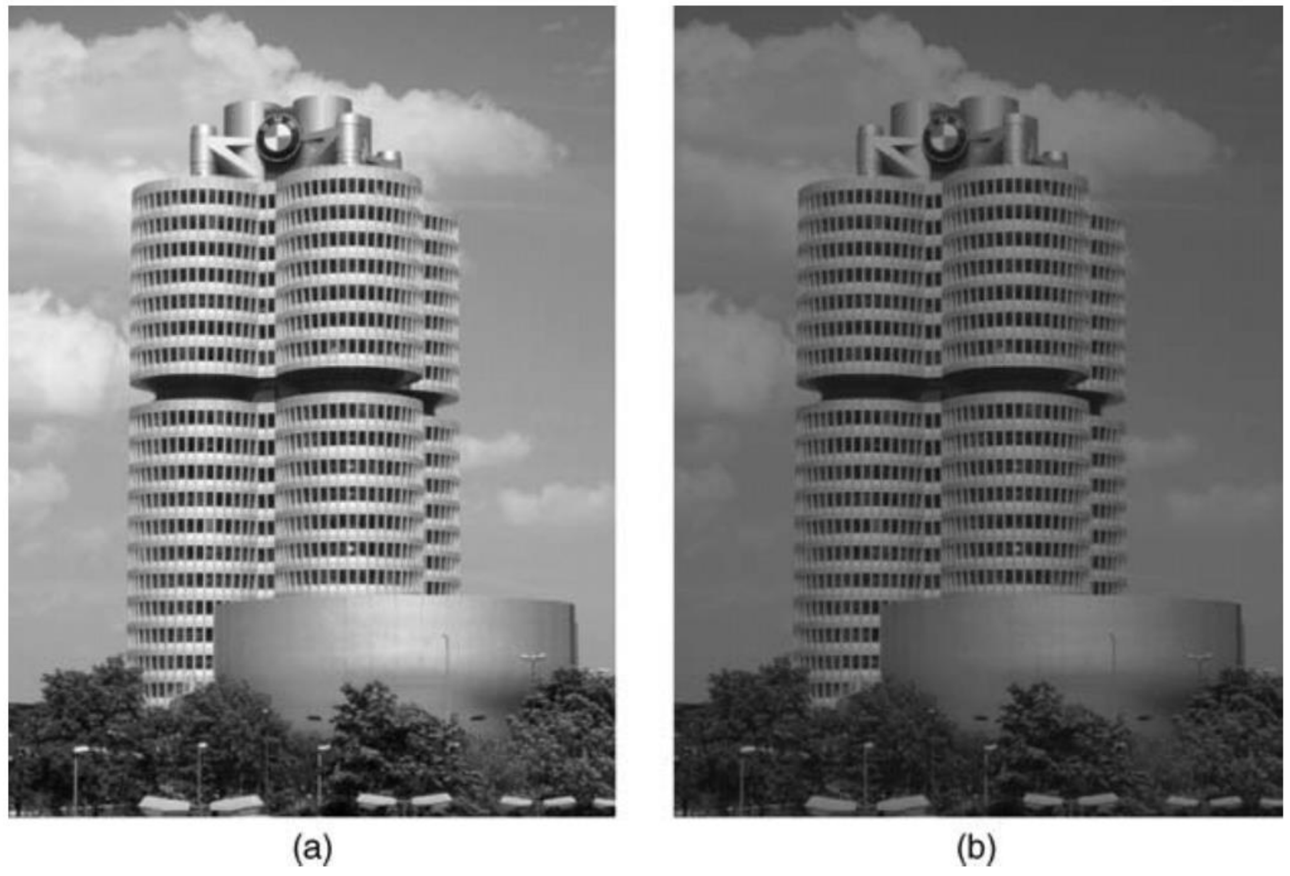


FIGURE 2.9 Example of intensity reduction using a transformation function: (a) original image; (b) output image.

2.4.2 NEIGHBORHOOD-ORIENTED OPERATIONS

- *Neighborhood-oriented* (also known as *local or area*) operations consist of determining the resulting pixel value at coordinates (x, y) as a function of its original value and the value of (some of) its neighbors, typically using a convolution operation.
- The convolution of a source image with a small 2D array (known as *window*, *template*, *mask*, or *kernel*) produces a destination image in which *each pixel value depends on its original value and the value of (some of) its neighbors*. The convolution mask determines which neighbors are used as well as the relative weight of their original values.

2.4.2 NEIGHBORHOOD-ORIENTED OPERATIONS

- Masks are normally 3x3, such as the one shown in Figure 2.10. Each mask coefficient (W_1,...,W_9) can be interpreted as *a weight*. The mask can be thought of as *a small window* that is *overlaid on the image to perform the calculation on one pixel at a time*.
- As each pixel is processed, the window moves to the next pixel in the source image and the process is repeated until the last pixel has been processed.

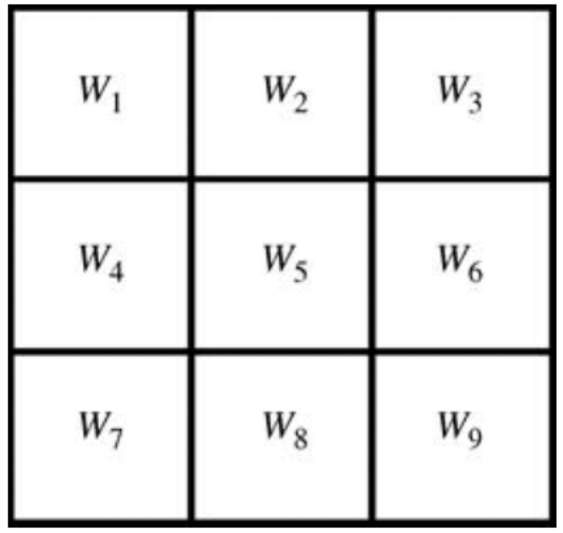


FIGURE 2.10 A 3 × 3 convolution mask, whose generic weights are W_1, ..., W_9.

2.4.3 OPERATIONS COMBINING MULTIPLE IMAGES

- There are many image processing applications that combine two images, *pixel by pixel*, using an arithmetic or logical operator, resulting in a third image, Z:

$$X \quad opn \quad Y \quad = \quad Z$$

where X and Y may be images (arrays) or scalars, Z is necessarily an array, and opn is a *binary mathematical* (+, −, ×, /) or *logical* (AND, OR, XOR) operator.

2.4.3 OPERATIONS COMBINING MULTIPLE IMAGES

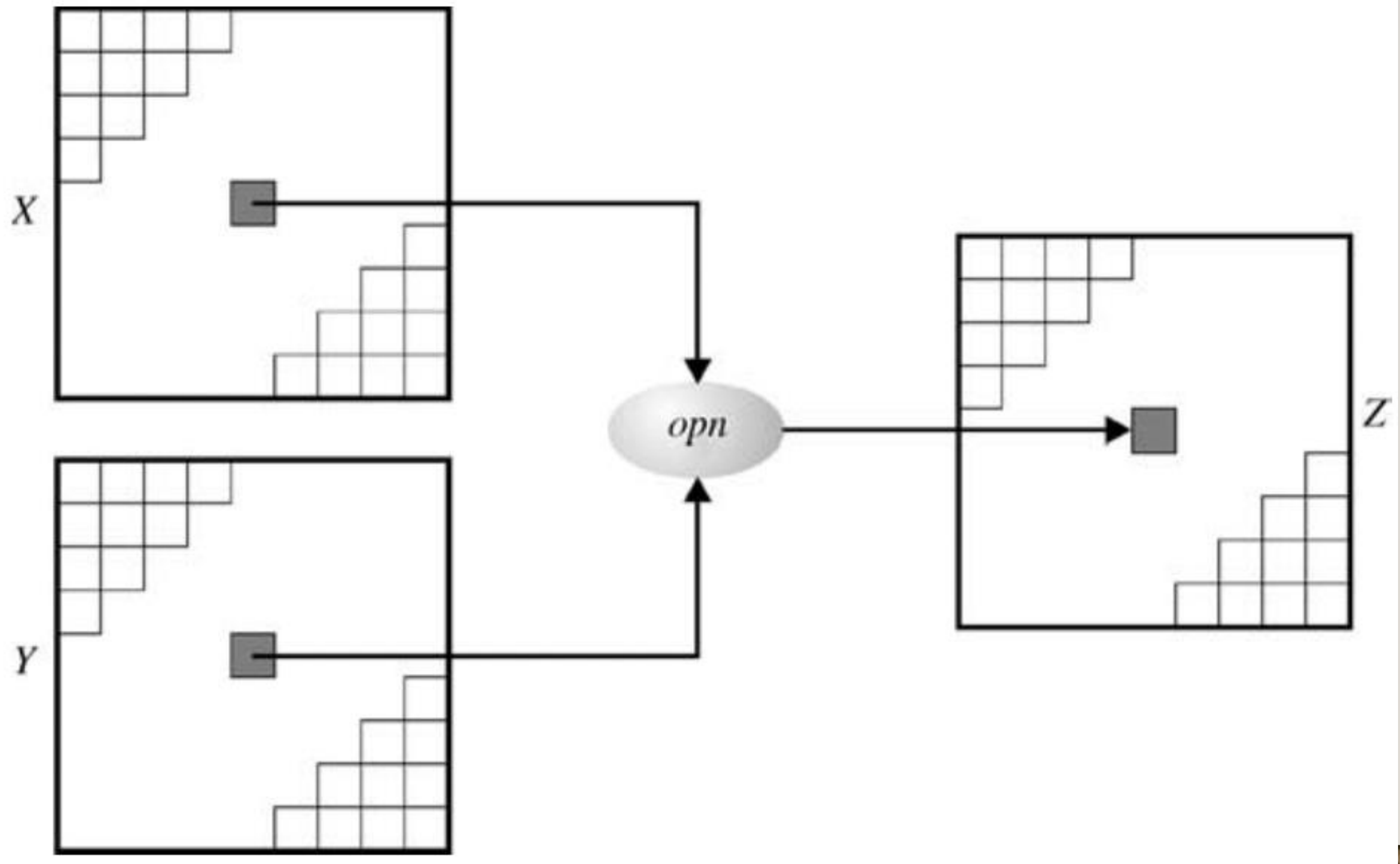
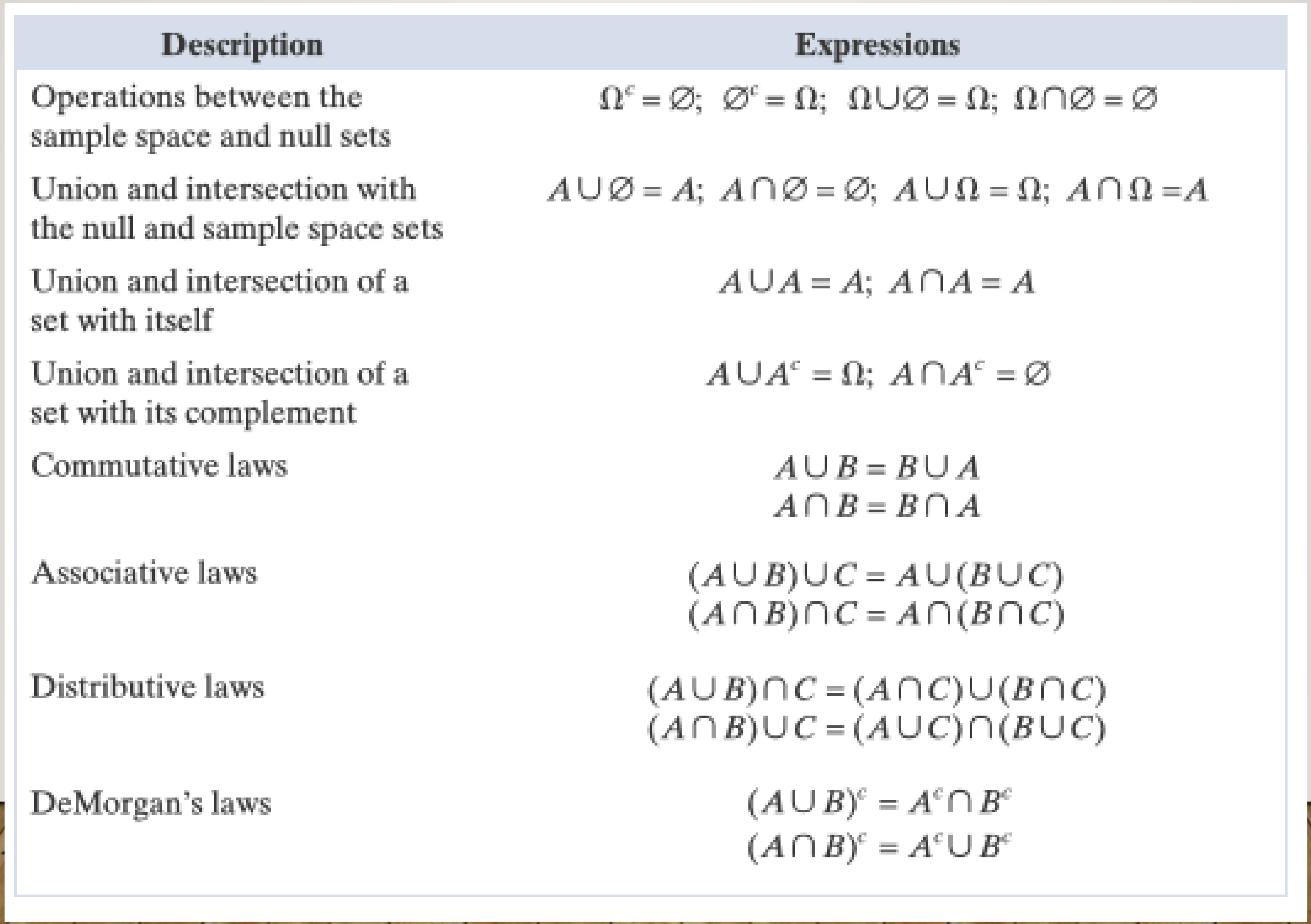


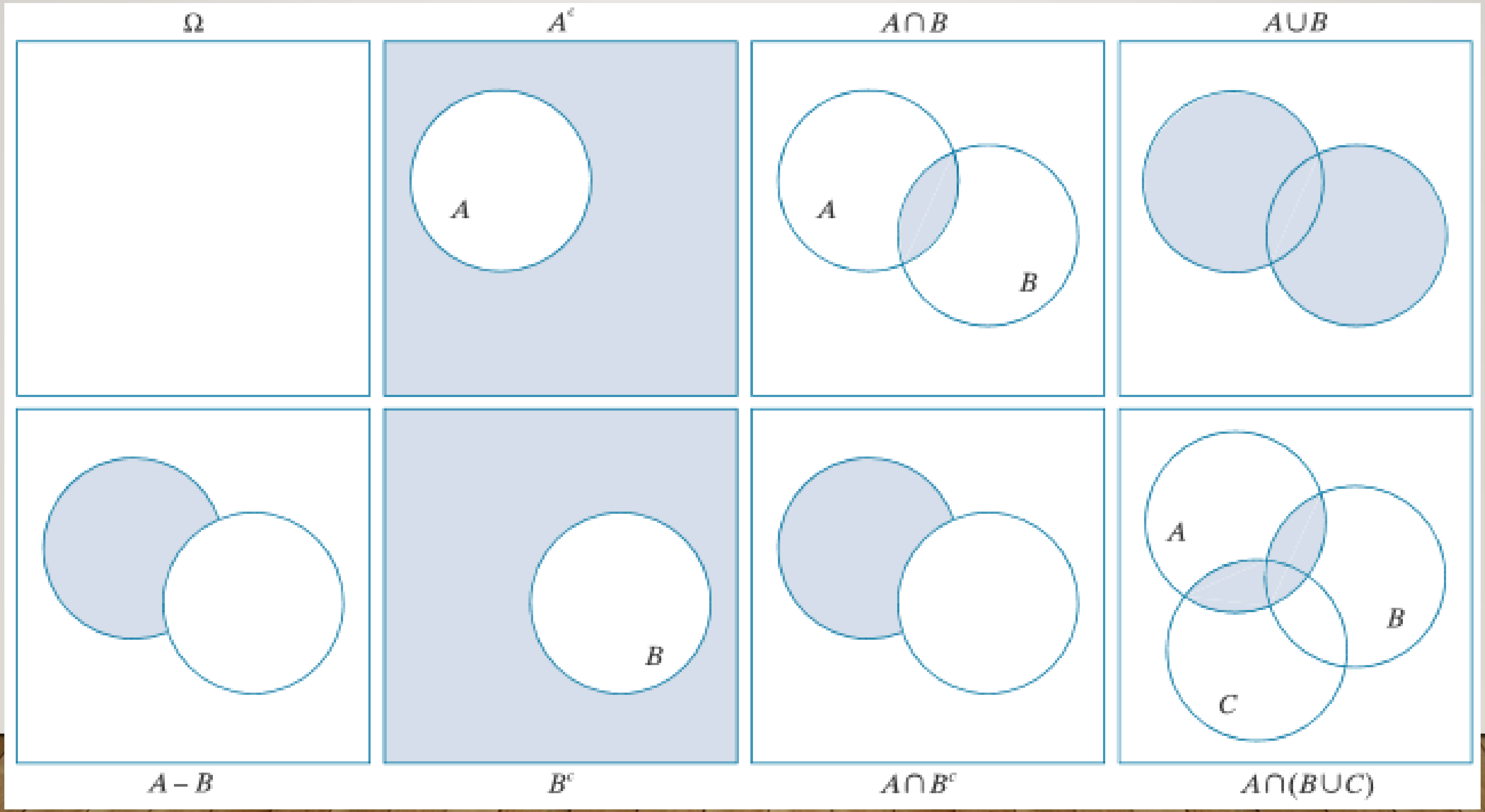
FIGURE 2.11 Pixel-by-pixel arithmetic and logic operations.

2.4.3 OPERATIONS COMBINING MULTIPLE IMAGES

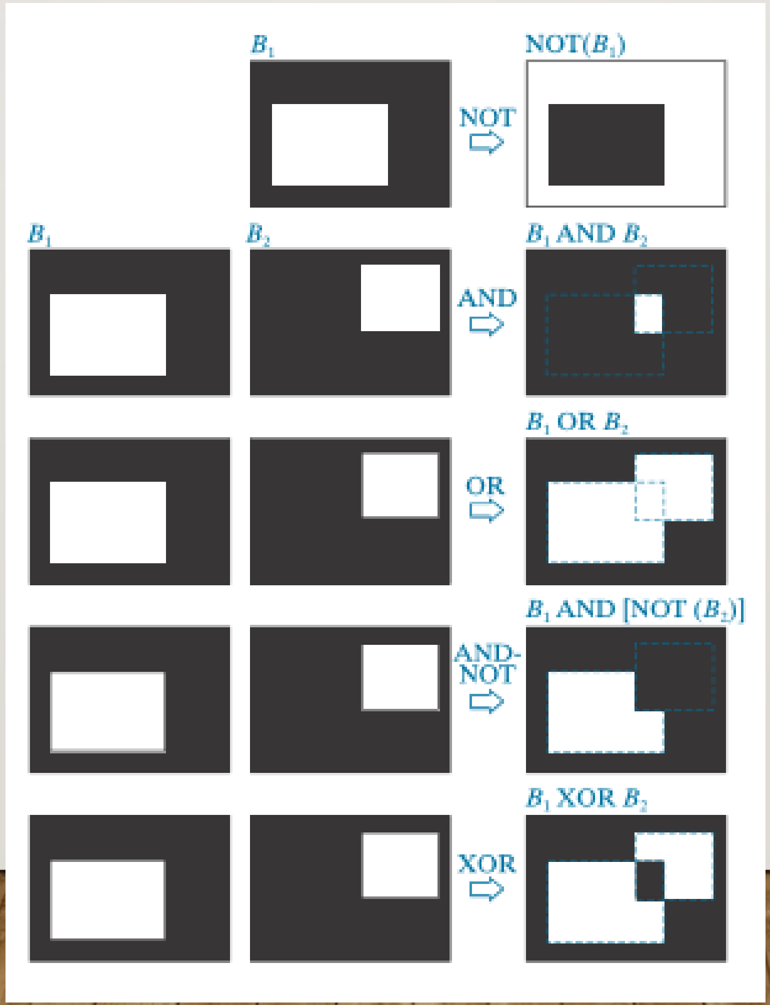


Description	Expressions
Operations between the sample space and null sets	$\Omega^c = \varnothing;\ \varnothing^c = \Omega;\ \Omega \cup \varnothing = \Omega;\ \Omega \cap \varnothing = \varnothing$
Union and intersection with the null and sample space sets	$A \cup \varnothing = A;\ A \cap \varnothing = \varnothing;\ A \cup \Omega = \Omega;\ A \cap \Omega = A$
Union and intersection of a set with itself	$A \cup A = A;\ A \cap A = A$
Union and intersection of a set with its complement	$A \cup A^c = \Omega;\ A \cap A^c = \varnothing$
Commutative laws	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associative laws	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributive laws	$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$
DeMorgan's laws	$(A \cup B)^c = A^c \cap B^c$ $(A \cap B)^c = A^c \cup B^c$

2.4.3 OPERATIONS COMBINING MULTIPLE IMAGES



2.4.3 OPERATIONS COMBINING MULTIPLE IMAGES



2.4.4 OPERATIONS IN A TRANSFORMATION DOMAIN

- A transform is *a mathematical tool* that allows the conversion of a set of values to another set of values, creating, therefore, a new way of representing the same information.
- In the field of image processing, *the original domain is referred to as spatial domain*, whereas the results are said to lie *in the transform domain*.
- The motivation for using mathematical transforms in image processing stems from the fact that *some tasks are best performed by transforming the input images*, applying selected algorithms in the transform domain, and eventually applying the inverse transformation to the result.

2.4.4 OPERATIONS IN A TRANSFORMATION DOMAIN

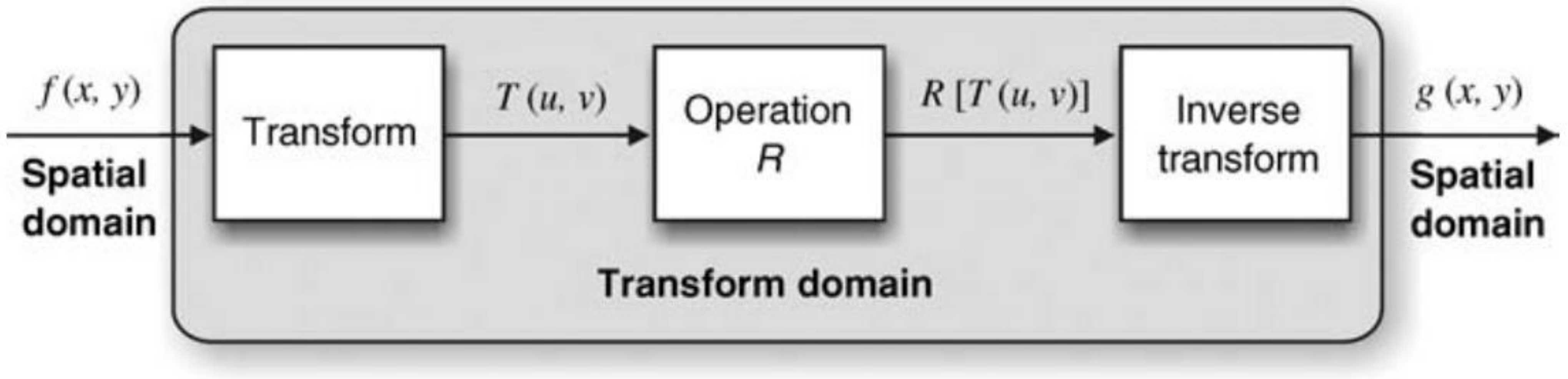
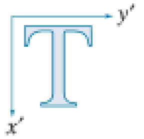
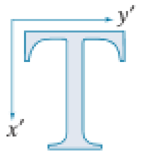
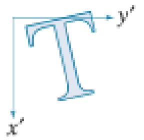
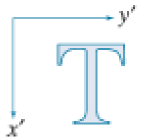
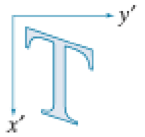
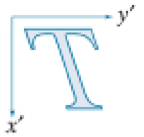
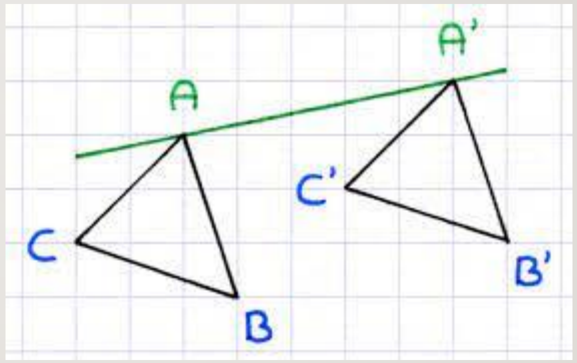
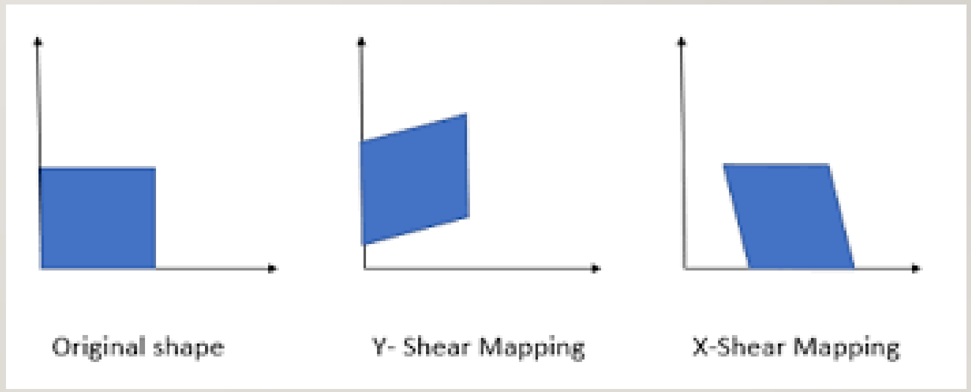


FIGURE 2.12 Operations in a transform domain.

2.4.4 OPERATIONS IN A TRANSFORMATION DOMAIN

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





2.4.4 OPERATIONS IN A TRANSFORMATION DOMAIN

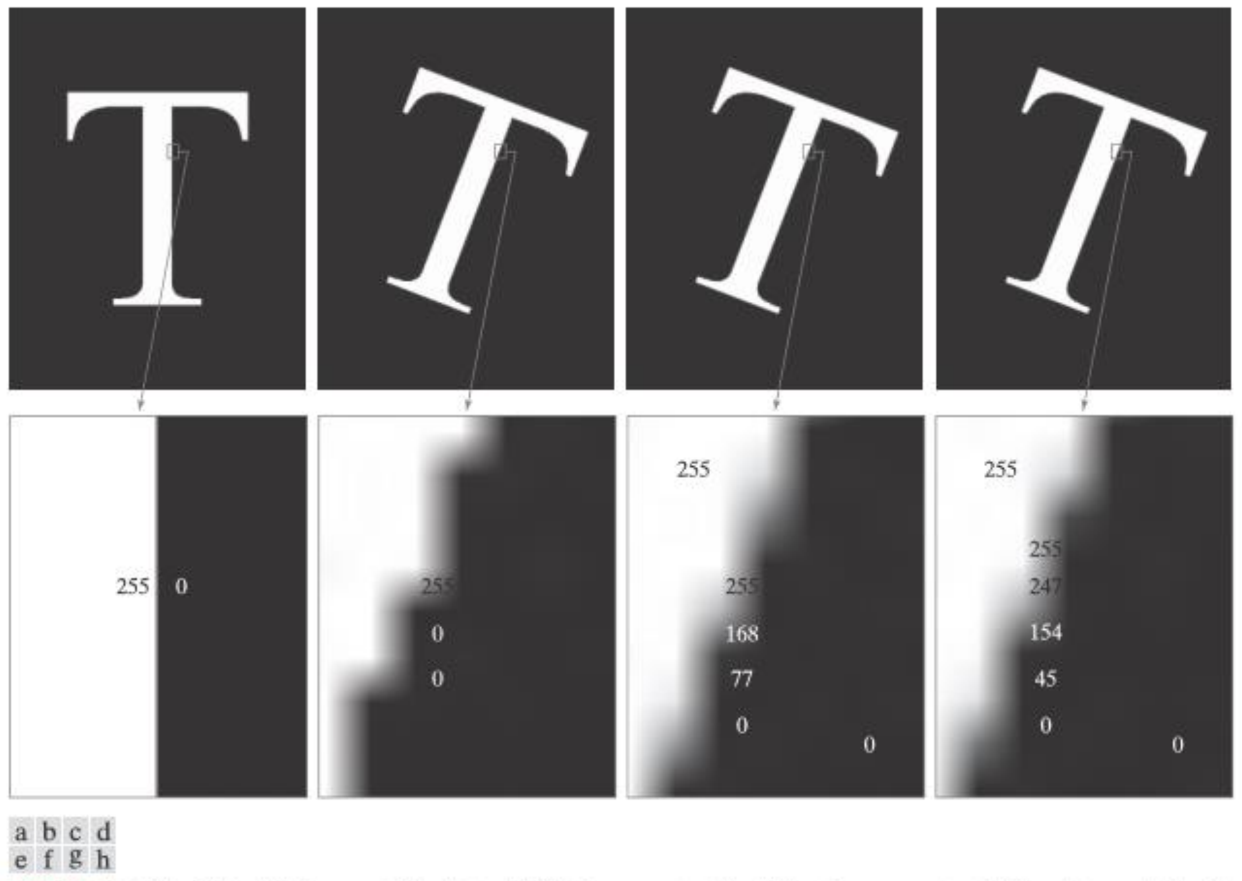


FIGURE 2.40 (a) A 541 × 421 image of the letter T. (b) Image rotated −21° using nearest-neighbor interpolation for intensity assignments. (c) Image rotated −21° using bilinear interpolation. (d) Image rotated −21° using bicubic interpolation. (e)-(h) Zoomed sections (each square is one pixel, and the numbers shown are intensity values).

2.4.4 OPERATIONS IN A TRANSFORMATION DOMAIN

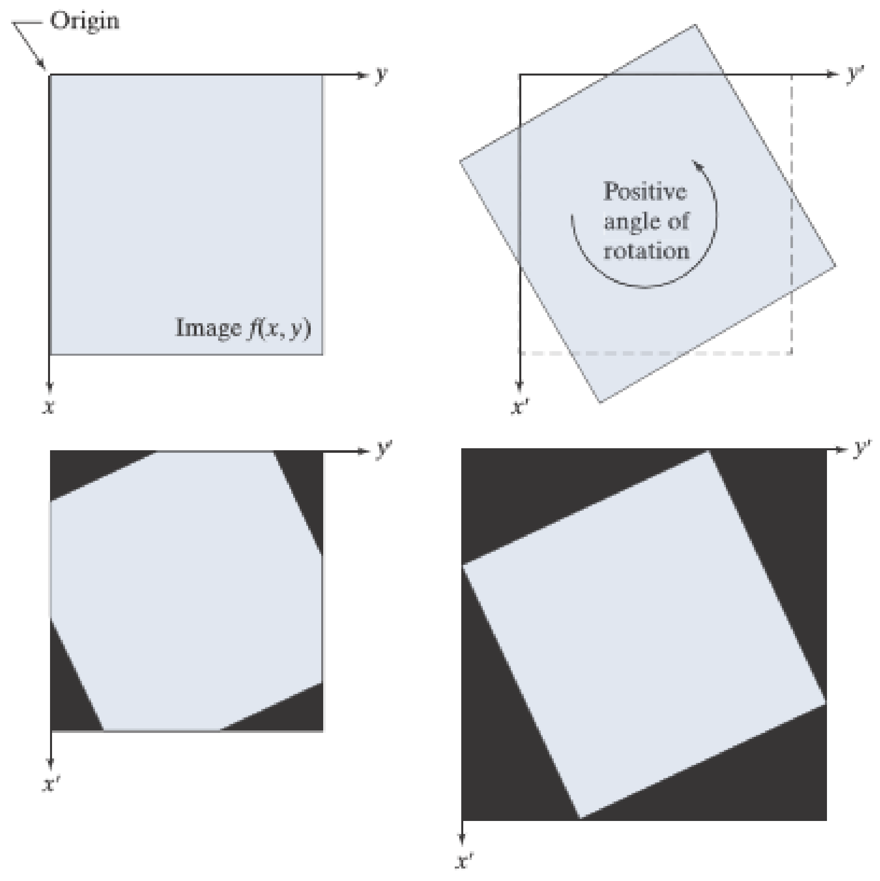


FIGURE 2.41
(a) A digital image.
(b) Rotated image (note the counterclockwise direction for a positive angle of rotation).
(c) Rotated image cropped to fit the same area as the original image.
(d) Image enlarged to accommodate the entire rotated image.

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

2.4.5 IMAGE REGISTRATION

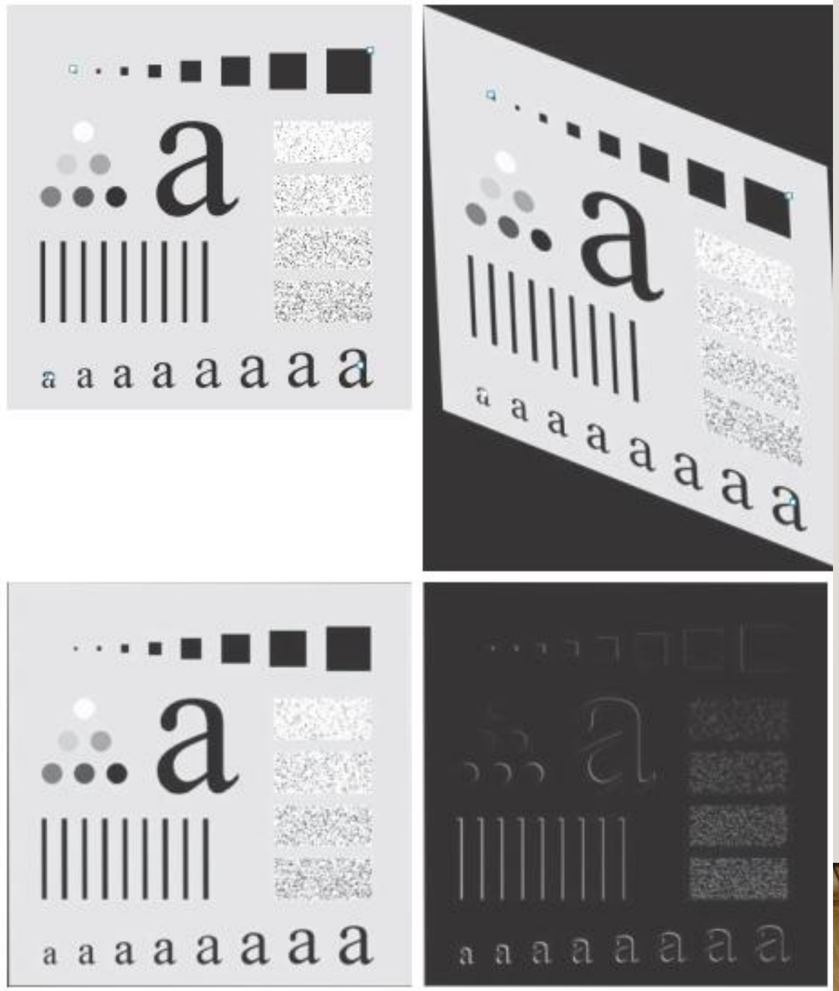


FIGURE 2.42 Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered (output) image (note the errors in the border). (d) Difference between (a) and (c), showing more registration errors.

WHAT HAVE WE LEARNED?

- Images are represented in *digital format* in a variety of ways. Bitmap (also known as *raster*) representations use one or more two-dimensional arrays of pixels (picture elements), whereas vector representations use a series of drawing commands to represent an image.
- *Binary images* are encoded as a 2D array, using 1 bit per pixel, where usually—but not always—a 0 means "black" and a 1 means "white."
- *Gray-level* (*monochrome*) images are encoded as a 2D array of pixels, using 8 bits per pixel, where a pixel value of 0 usually means "black" and a pixel value of 255 means "white," with intermediate values corresponding to varying shades of gray.

WHAT HAVE WE LEARNED?

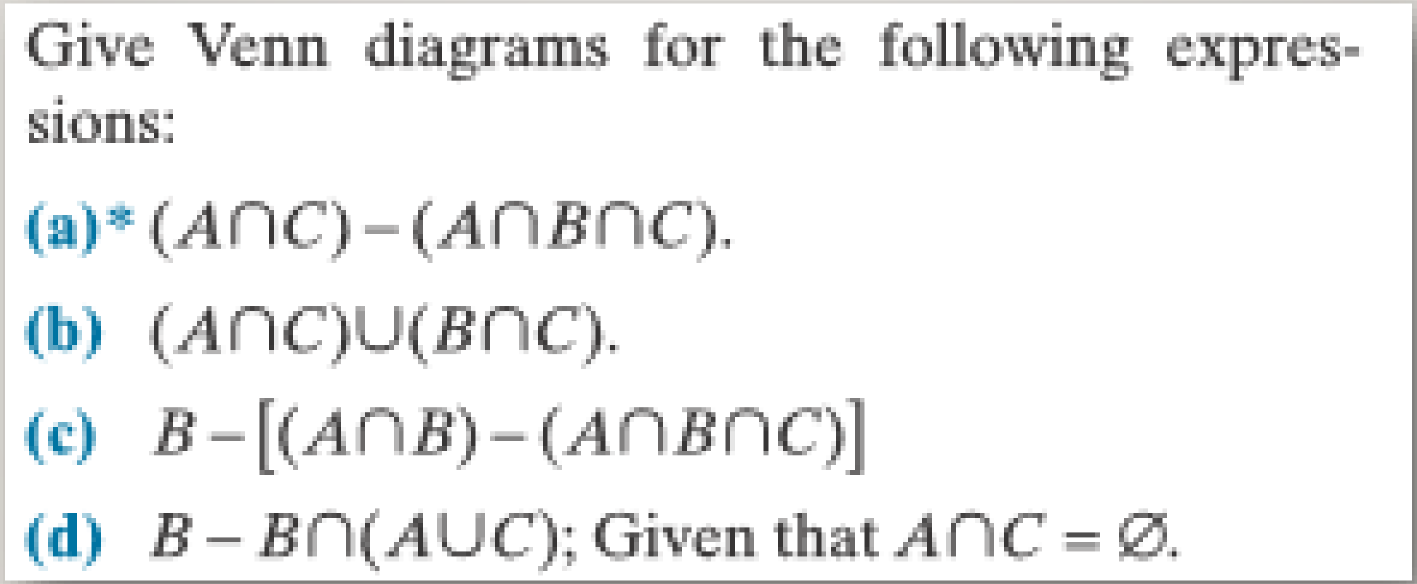
- The two most common ways of storing color image contents are *RGB representation*—in which each pixel is usually represented by a 24-bit number containing the amount of its red (R), green (G), and blue (B) components—and *indexed representation*—where a 2D array contains indices to a color palette (or look up table).
- Some of the most popular image file formats in use today are BMP, GIF, JPEG, TIFF, and PNG.
- MATLAB's built-in functions for reading images from files and writing images to files (*imread* and *imwrite*, respectively) support most file formats and their variants and options.

WHAT HAVE WE LEARNED?

- *Image topology* is the field of image processing concerned with investigation of fundamental image properties (e.g., number of connected components and number of holes in an object) using concepts such as *adjacency* and *connectivity*.

- Image processing operations can be divided into two big groups: *spatial domain* and *transform domain*. Spatial-domain techniques can be further divided into *pixel-by-pixel* (*point*) or *neighborhood-oriented* (*area*) operations.

PROBLEMS


Give Venn diagrams for the following expressions:

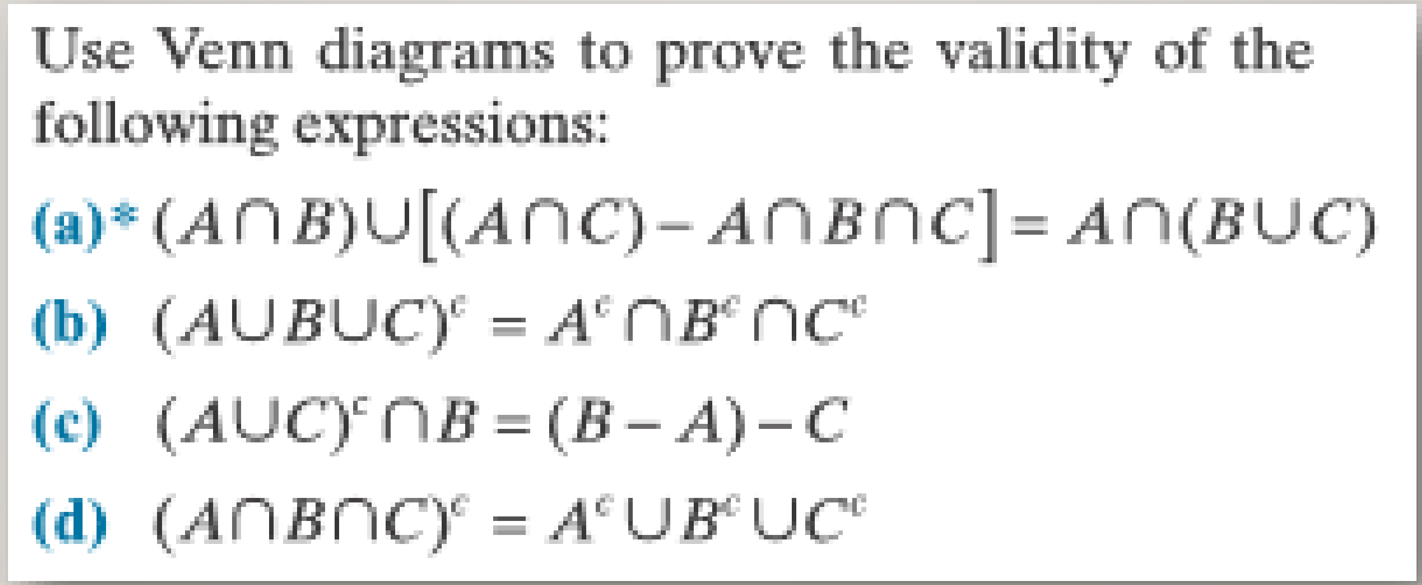
(a)* $(A \cap C) - (A \cap B \cap C)$.

(b) $(A \cap C) \cup (B \cap C)$.

(c) $B - [(A \cap B) - (A \cap B \cap C)]$

(d) $B - B \cap (A \cup C)$; Given that $A \cap C = \varnothing$.

PROBLEMS



Use Venn diagrams to prove the validity of the following expressions:

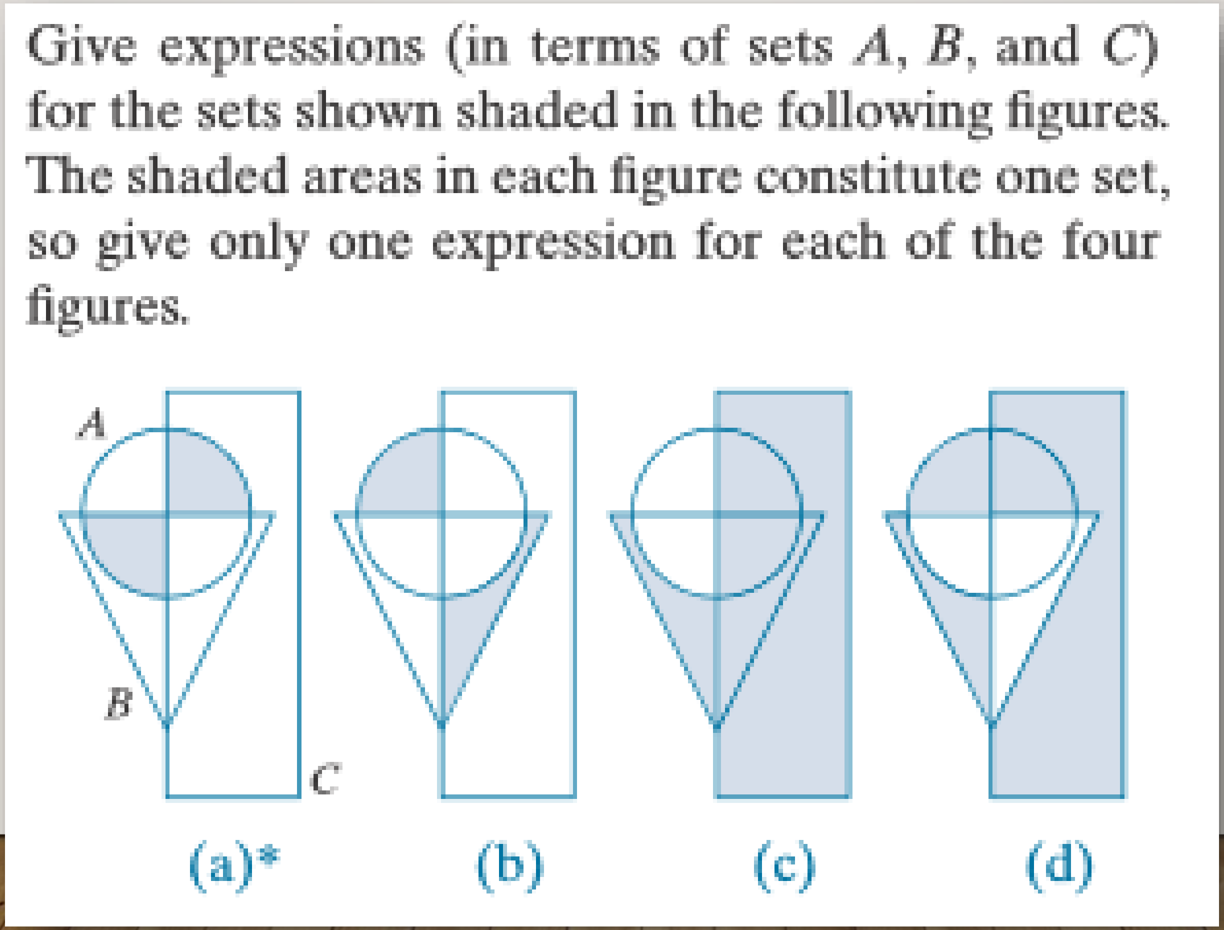
(a)* $(A \cap B) \cup [(A \cap C) - A \cap B \cap C] = A \cap (B \cup C)$

(b) $(A \cup B \cup C)^c = A^c \cap B^c \cap C^c$

(c) $(A \cup C)^c \cap B = (B - A) - C$

(d) $(A \cap B \cap C)^c = A^c \cup B^c \cup C^c$

PROBLEMS


Give expressions (in terms of sets A, B, and C) for the sets shown shaded in the following figures. The shaded areas in each figure constitute one set, so give only one expression for each of the four figures.

END OF LECTURE 2

LECTURE 3 – MATLAB BASICS AND THE IMAGE PROCESSING TOOLBOX AT A GLANCE