CHAPTER 2
IMAGE PROCESSING BASICS

WHAT WILL WE LEARN?

- 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?

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 book, 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 $N-1$) (Figure 2.1):

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

FIGURE 2.1 A monochrome image and the convention used to represent rows (x) and columns (y) adopted in this book.

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

2D image in digital format

format: ***raster*** **(also known as** ***bitmap*****) and** ***vector*****.**

Bitmap representations use one or more two-dimensional arrays of pixels, advantages of bitmap graphics are:

their quality and display speed;

Disadvantages

larger memory storage requirements

size dependence (e.g., enlarging a bitmap image may lead to noticeable artifacts).

vector representations use a series of drawing commands to represent an image.

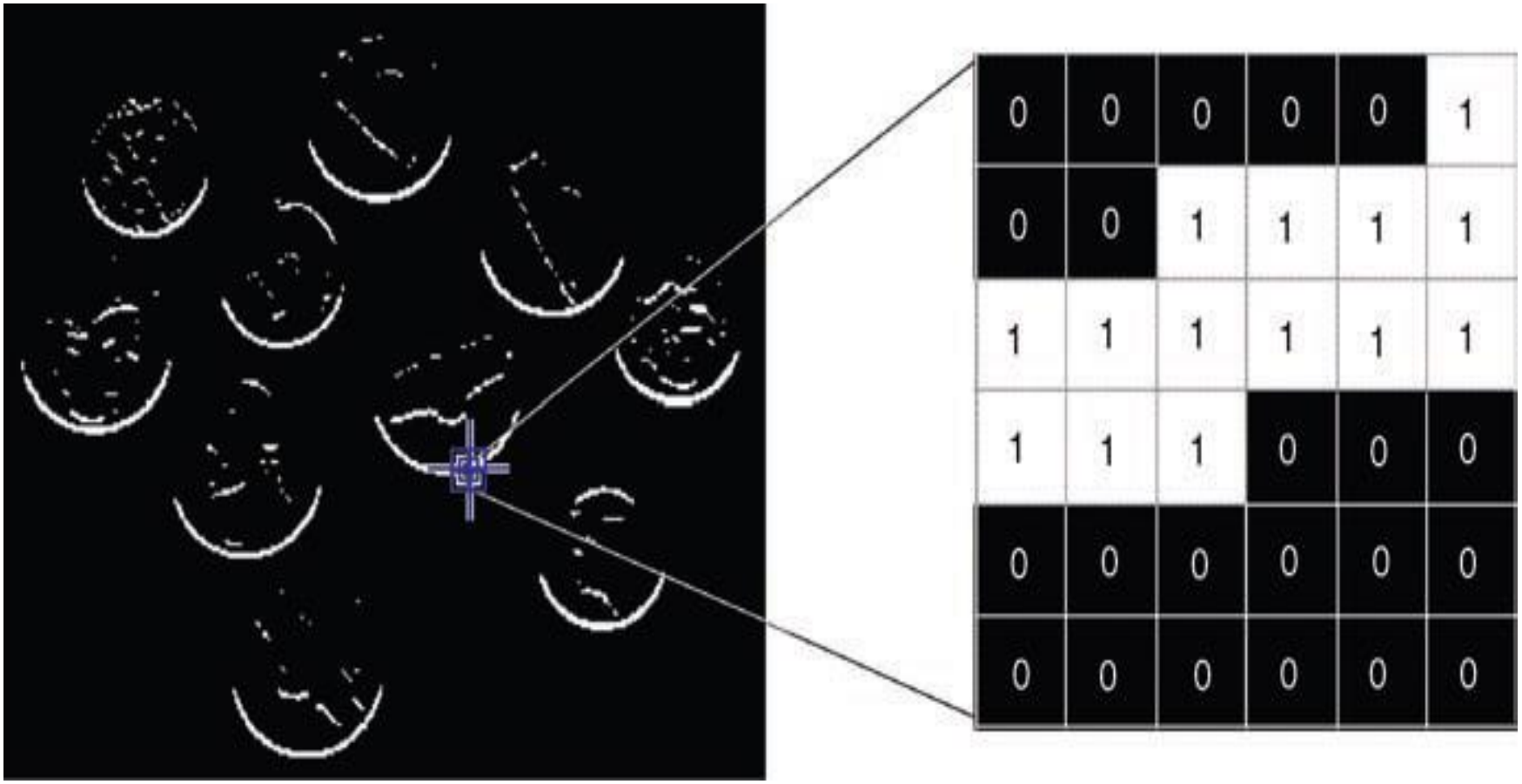
advantages

Vector representations require less memory and allow resizing and geometric manipulations without introducing artifacts,

Disadvantages

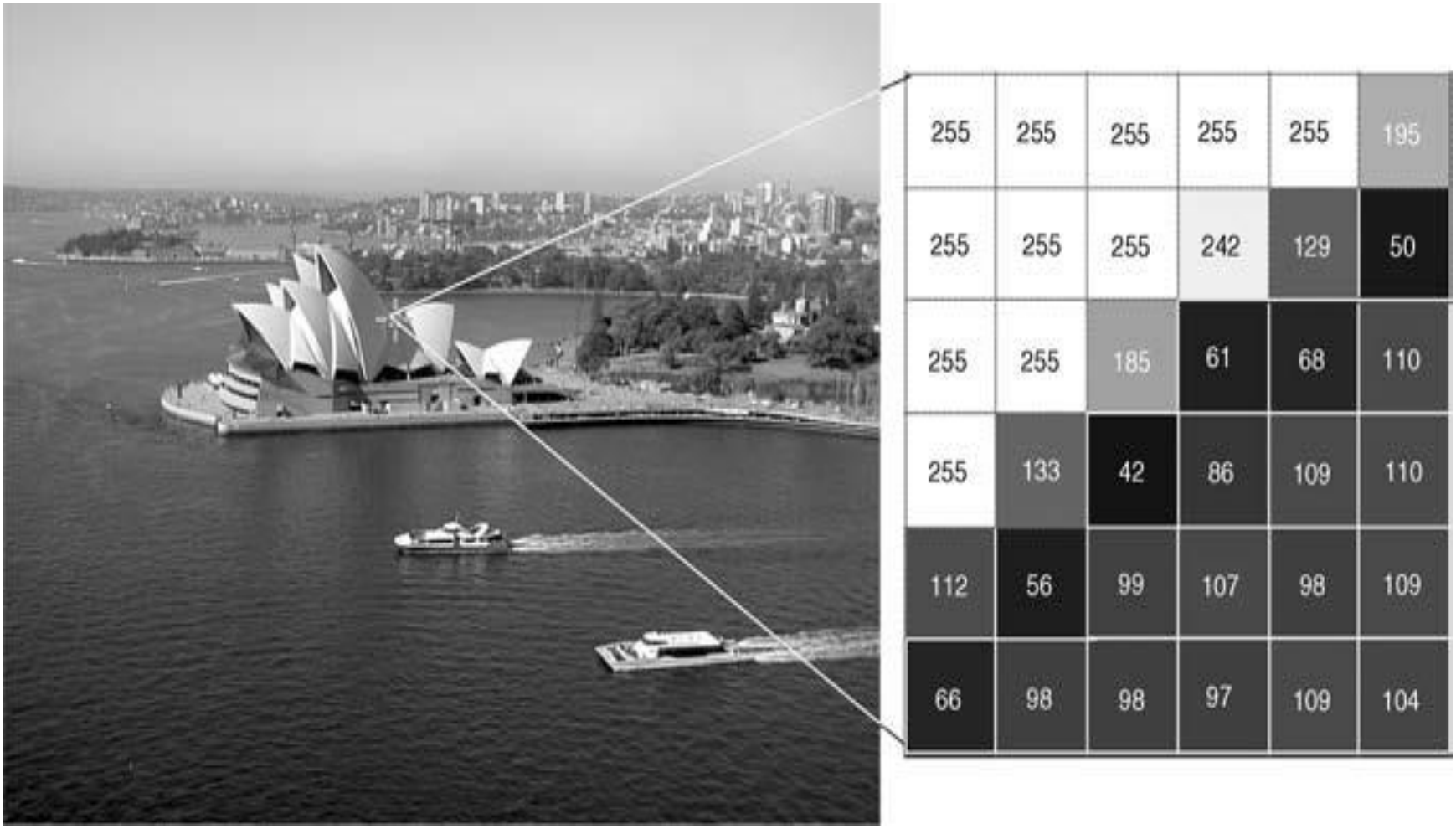
need to be rasterized for most presentation devices.

2.1.1 Binary (1-Bit) Images



2.1.2 Gray-Level (8-Bit) Images

usually with 8 bits per pixel

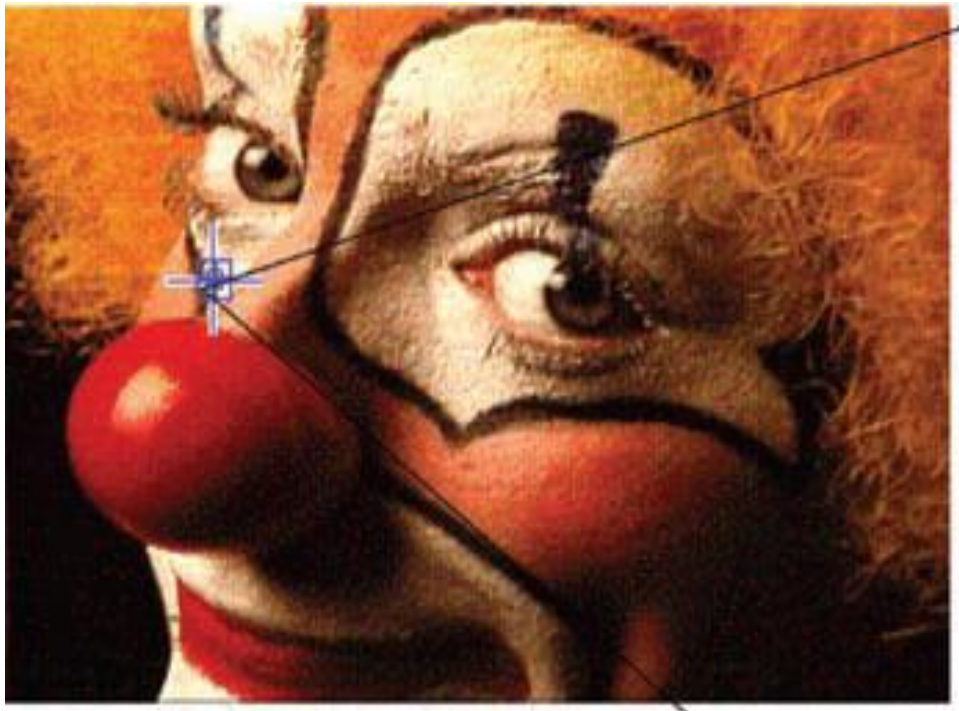


2.1.3 Color Images

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) (Figure 2.4).1 Each array element contains an 8-bit value, indicating the amount of red, green, or blue at that point in a [0, 255] scale.

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

Indexed Color Images



<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

2.1.4 Compression

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)

lossy compression should be used for general-purpose photographic images

lossless compression should be preferred when dealing with line art, drawings, facsimiles, or images in which no loss of detail may be tolerable (most notably, space images and medical images).

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.** In more complex file formats, the header may also contain information about the type of compression used and other parameters that are necessary to decode (i.e., decompress) the image.

- **The BIN format** simply consists of the raw pixel data, **without any header**. Consequently, the user of a BIN file must know the relevant image parameters (such as height and width) beforehand in order to use the image.
- **The PPM format** and its variants (PBM for binary images,

PGM for grayscale images, PPM for color images, and PNM for any of them). The headers for these image formats include **a 2-byte signature, or "magic number," that identifies the file type, the image width and height, the number of bands, and the maximum intensity value** (which determines the number of bpp per band).

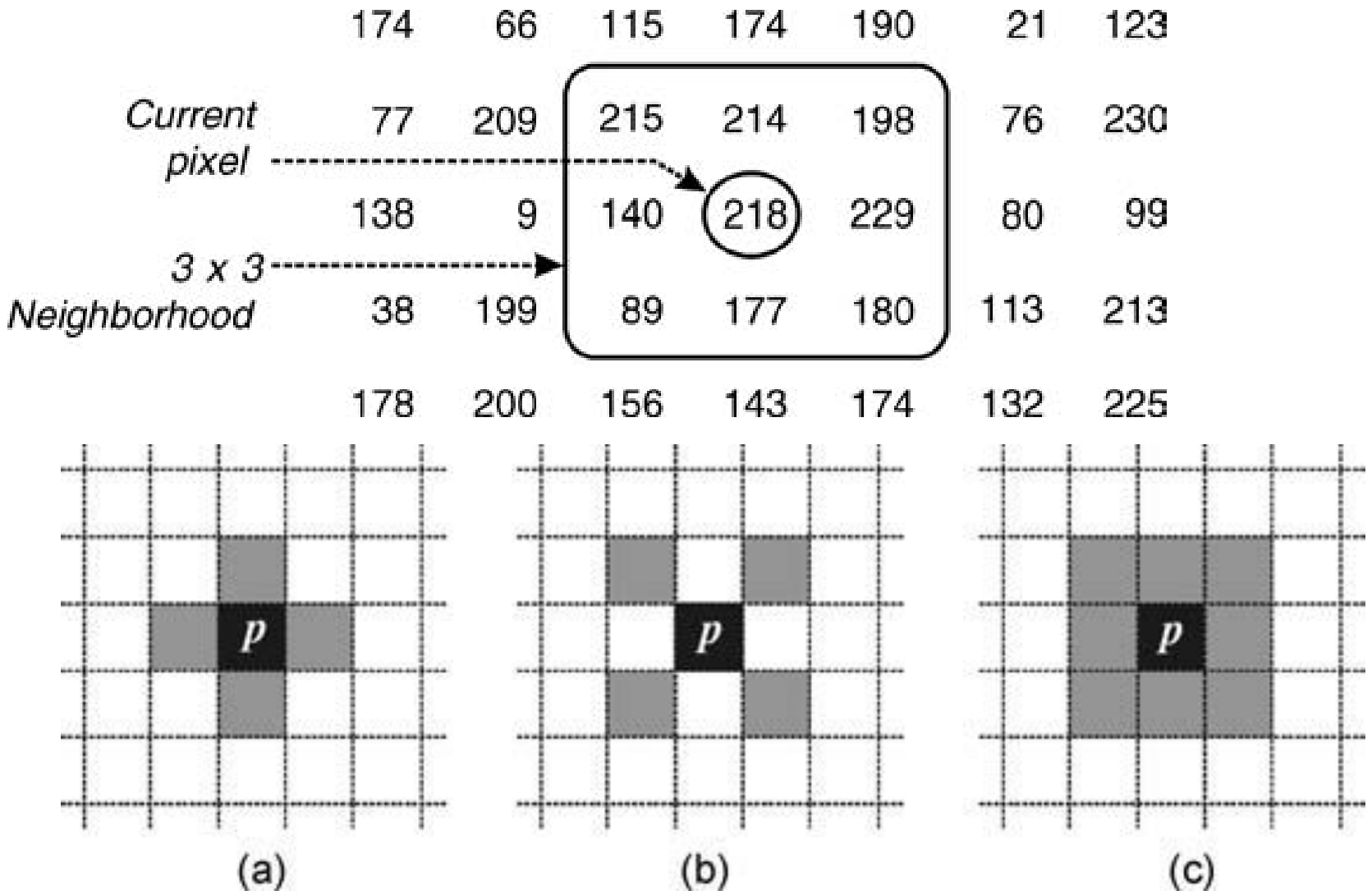
- The Microsoft Windows bitmap (BMP) format is another widely used and fairly simple format, consisting of a header followed by raw pixel data.
- The JPEG format is the most popular file format for photographic quality image representation. It is capable of high degrees of compression with minimal perceptual loss of quality. The technical details of the JPEG compression algorithm (and its presumed successor, the JPEG 2000 standard)

- **GIF (Graphics Interchange Format)** uses an indexed representation for color images (with a palette of a maximum of 256 colors), the LZW (Lempel–Ziv–Welch) compression algorithm, and a 13-byte header.
- **TIFF (Tagged Image File Format)** is a more sophisticated format with many options and capabilities, including the ability to represent truecolor (24 bpp) and support for five different compression schemes.
- **Portable Network Graphics (PNG)** is an increasingly popular file format that supports both indexed and truecolor images.

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 (see Chapter 13)—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, for example, the 3×3 neighborhood



(a) 4-neighborhood; (b) diagonal neighborhood; (c) 8-neighborhood.

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.

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*.

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.

Distances Between Pixels

between two pixels p and q, of coordinates (x_0, y_0) and (x_1, y_1), respectively, are as follows:

- Euclidean distance:

$$D_e(p, q) = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{2.3}$$

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

$$D_4(p, q) = |x_1 - x_0| + |y_1 - y_0| \tag{2.4}$$

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

$$D_8(p, q) = \max(|x_1 - x_0|, |y_1 - y_0|) \tag{2.5}$$

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. *Example:* contrast adjustment

– ***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. *Example:* spatial-domain filters.

– ***Operations Combining Multiple Images*:** Here, 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. *Example:* subtracting one image from another for detecting differences between them.

• *Operations in a Transform Domain*: Here, 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.
Example: frequency-domain filtering techniques

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

$$g(x, y) = T\left[f(x, y)\right] \tag{2.6}$$

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:

$$s = T\,[r] \tag{2.7}$$

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

Figure 2.9 shows an example of a transformation function used to reduce the overall intensity of an image by half: $s = r/2$. Chapter 8 will discuss point operations and transformation functions in more detail.

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.

Each mask coefficient (*W*1, . . . , *W*9) can be interpreted as a *weight*.

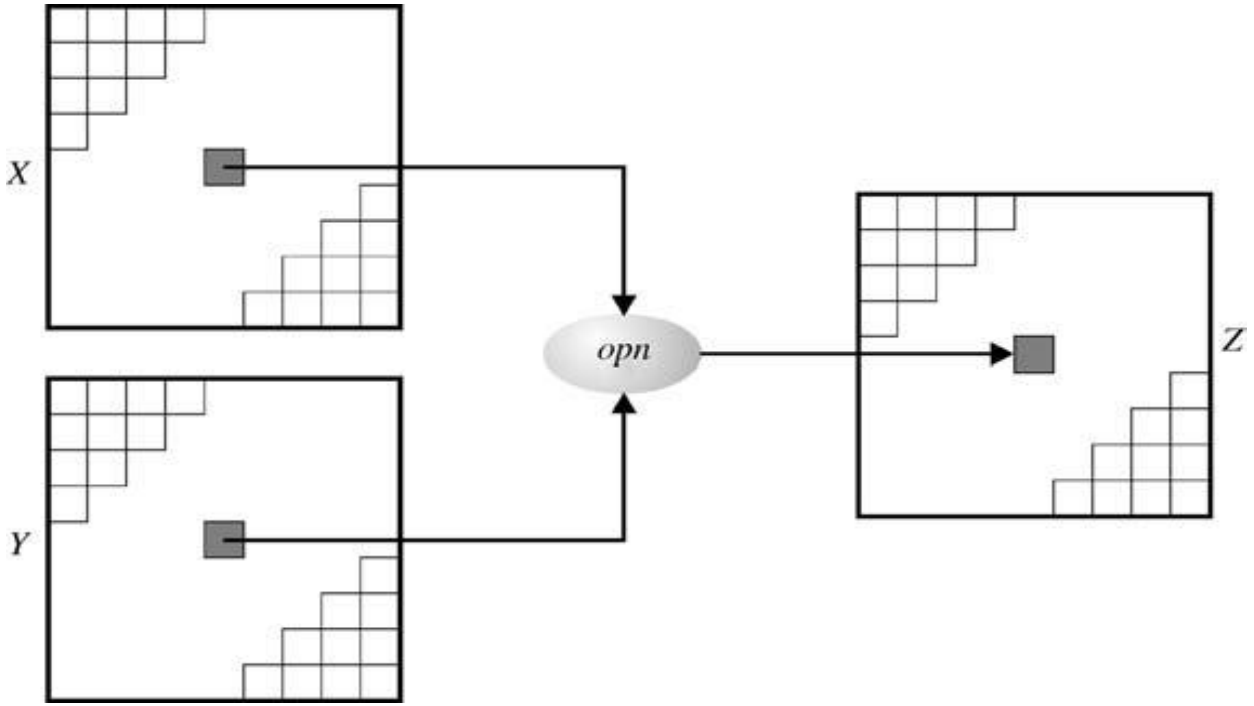
W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

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*:

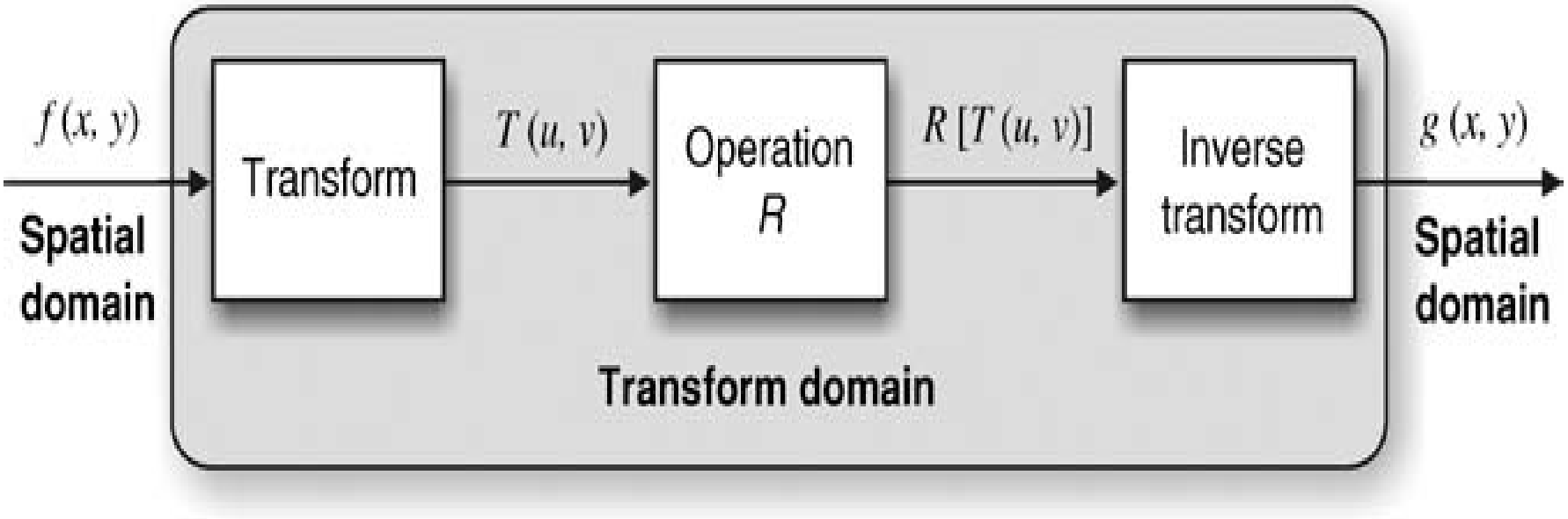
$$X \; opn \; Y = Z \qquad (2.8)$$

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.



2.4.4 Operations in a Transform 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*.**



Operations in a transform domain