CHAPTER 3

Digital Image Processing Operations

Photography is the simplest thing in the world, but it is incredibly complicated to make it really work.

–Martin Parr

LEARNING OBJECTIVES

This chapter provides an overview of image processing operations. The concepts of image topology such as neighbourhood and connectivity are introduced in this chapter. The important arithmetic, logical, statistical, spatial, geometrical, and set operations are discussed. After studying this chapter, the reader will become familiar with the following:

- Concepts of image geometry and topology
- Image processing operations
- Logical operations
- Geometrical operations
- Interpolation techniques
- Set operations
- Spatial operations

.1 BASIC RELATIONSHIPS AND DISTANCE METRICS

This chapter discusses the image processing operations. To carry out any image processing operation, the image must be stored in the memory in a suitable form. Chapter 2 discusses how the images are sampled and quantized with rectangular grids. During this sampling process, the digital image is a set of discrete pixels arranged in a rectangular grid. The pixel itself assumes the shape of a small rectangle. If the width and height of the pixel are the same, the shape of the pixel is a square. Even though grids of other shapes such as hexagonal and triangular are available, rectangular grids are widely used as they are intuitive, simple, and are inherently used in the discretization of images by CCD (charge coupled device) cameras and scanners.

Image Coordinate System

Images can be easily represented as a two-dimensional array or matrix. The popularity of the matrix form is due to the fact that most of the programming languages support 2D array data structure and can easily implement matrix-level computation.

Pixels can be visualized logically and physically. Logical pixels specify the points of a continuous 2D function. These are logical in the sense that they specify a location but occupy no physical area. Normally, this is represented in the Cartesian first coordinate system. Physical pixels, on the other hand, occupy a small amount of space when displayed on the output device. Digitized images indicating physical pixels are represented in the Cartesian fourth coordinate system. For example, an analog image of size 3 × 3 is represented in the first quadrant of the Cartesian coordinate system as shown in Fig. 3.1.

$$\begin{array}{c|ccc} 2 & f(0,2) & f(1,2) & f(2,2) \\ 1 & f(0,1) & f(1,1) & f(2,1) \\ y=0 & f(0,0) & f(1,0) & f(2,0) \\ \hline & x=0 & 1 & 2 \end{array}$$

Fig. 3.1 Analog image *f(x, y)* in the first quadrant of Cartesian coordinate system

Figure 3.1 illustrates an image $f(x, y)$ of dimension 3 × 3, where $f(0, 0)$ is the bottom left corner. Since it starts from the coordinate position (0, 0), it ends with $f(2, 2)$, that is, $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. x and y define the dimensions of the image.

However, in digital image processing, the discrete form of the image is often used. Discrete images are usually represented in the fourth quadrant of the Cartesian coordinate system. A discrete image $f(x, y)$, of dimension 3 × 3, is shown in Fig. 3.2(a).

Many programming environments including MATLAB start with an index of (1, 1). The equivalent representation of the given matrix is shown in Fig. 3.2(b).

The coordinates used for discrete image is, by default, the fourth quadrant of the Cartesian system.

$$\begin{array}{c|ccc} & y=0 & y=1 & y=2 \\ \hline x=0 & f(0,0) & f(0,1) & f(0,2) \\ x=1 & f(1,0) & f(1,1) & f(1,2) \\ x=2 & f(2,0) & f(2,1) & f(2,2) \end{array} \qquad \begin{array}{c|ccc} & y=1 & y=2 & y=3 \\ \hline x=1 & f(1,1) & f(1,2) & f(1,3) \\ x=2 & f(2,1) & f(2,2) & f(2,3) \\ x=3 & f(3,1) & f(3,2) & f(3,3) \end{array}$$

(a) (b)

Fig. 3.2 Discrete image (a) Image in the fourth quadrant of Cartesian coordinate system (b) Image coordinates as handled by software environments such as MATLAB

3.1.2 Image Topology

Image topology is a branch of image processing that deals with the fundamental properties of the image such as image neighbourhood, paths among pixels, boundary, and connected components. It characterizes the image with topological properties such as neighbourhood, adjacency, and connectivity. *Neighbourhood* is fundamental to understanding image topology. In the simplest case, the neighbours of a given reference pixel are those pixels with which the given reference pixel shares its edges and corners.

In $N_4(p)$, the reference pixel $p(x, y)$ at the coordinate position (x, y) has two horizontal and two vertical pixels as neighbours. This is shown graphically in Fig. 3.3.

$$\begin{pmatrix} 0 & X & 0 \\ X & p(x,y) & X \\ 0 & X & 0 \end{pmatrix}$$

Fig. 3.3 4-Neighbourhood $N_4(p)$

The set of pixels $\{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$, called the 4-neighbours of p, is denoted as $N_4(p)$. Thus 4-neighbourhood includes the four direct neighbours of the pixel $p(x, y)$. By duality, these are pixels that share a common edge with the given reference pixel $p(x,y)$. Similarly, the pixel may have four diagonal neighbours. They are $(x-1, y-1)$, $(x+1, y+1)$, $(x-1, y+1)$, and $(x+1, y-1)$. The diagonal pixels for the reference pixel $p(x, y)$ are shown graphically in Fig. 3.4.

$$\begin{pmatrix} X & 0 & X \\ 0 & p(x, y) & 0 \\ X & 0 & X \end{pmatrix}$$

Fig. 3.4 Diagonal elements $N_D(p)$

The diagonal neighbours of pixel $p(x, y)$ are represented as $N_D(p)$. The 4-neighbourhood and N_D are collectively called the 8-neighbourhood. This refers to all the neighbours and pixels that share a common corner with the reference pixel $p(x, y)$. These pixels are called indirect neighbours. This is represented as $N_8(p)$ and is shown graphically in Fig. 3.5. The set of pixels $N_8(x) = N_D(x) \cup N_4(x)$.

$$\begin{pmatrix} X & X & X \\ X & p(x, y) & X \\ X & X & X \end{pmatrix}$$

Fig. 3.5 8-Neighbourhood $N_8(p)$

3.1.3 Connectivity

The relationship between two or more pixels is defined by pixel connectivity. Connectivity information is used to establish the boundaries of the objects. The pixels p and q are said to be connected if certain conditions on pixel brightness specified by the set V and spatial adjacency are satisfied. For a binary image, this set V will be $\{0, 1\}$ and for grey scale images, V might be any range of grey levels.

4-Connectivity The pixels p and q are said to be in 4-connectivity when both have the same values as specified by the set V and if q is said to be in the set $N_4(p)$. This implies any path from p to q on which every other pixel is 4-connected to the next pixel.

8-Connectivity It is assumed that the pixels p and q share a common grey scale value. The pixels p and q are said to be in 8-connectivity if q is in the set $N_8(p)$.

Mixed connectivity Mixed connectivity is also known as m-connectivity. Two pixels p and q are said to be in m-connectivity when

1. q is in $N_4(p)$ or
2. q is in $N_D(p)$ and the intersection of $N_4(p)$ and $N_4(q)$ is empty.

For example, Fig. 3.6 shows 8-connectivity when $V = \{0, 1\}$.

8-Connectivity is shown as lines. Here, a multiple path or loop is present. In m-connectivity, there are no such multiple paths. The m-connectivity for the image in Fig. 3.6 is as shown in Fig. 3.7.

It can be observed that the multiple paths have been removed.

3.1.4 Relations

A binary relation between two pixels a and b, denoted as aRb, specifies a pair of elements of an image. For example, consider the image pattern given in Fig. 3.8.

The set is given as $A = \{x_1, x_2, x_3\}$. The set based on the 4-connectivity relation is given as $A = \{x_1, x_2\}$. It can be observed that $x3$ is ignored as it is not connected to any other element of the image by 4-connectivity.

The following are the properties of the binary relations:

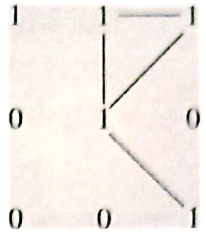


Fig. 3.6 8-Connectivity represented as lines

Reflexive For any element a in the set A, if the relation aRa holds, this is known as a reflexive relation.

Symmetric If aRb implies that bRa also exists, this is known as a symmetric relation.

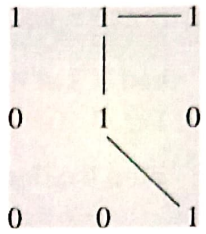


Fig. 3.7 *m*-Connectivity

Transitive If the relations aRb and bRc exist, it implies that the relationship aRc also exists. This is called the transitivity property.

If all these three properties hold, the relationship is called an *equivalence relation*. The relationships can be expressed as a matrix. Assume that the set A is divided into K disjoint sets, where K is an integer ranging from 1 to ∞. If a relation aRb exists, and if a and b belong to the same set, the relationship is said to be an equivalence relation. Transitive closure implies that if aRb and bRc exist, aRc also exists. The set containing all the implied relations is called the transitive closure of R, and is denoted as $R+$.

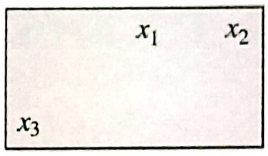


Fig. 3.8 Image pattern

3.1.5 Distance Measures

The distance between the pixels p and q in an image can be given by distance measures such as Euclidian distance, D_4 distance, and D_8 distance. Consider three pixels p, q, and z. If the coordinates of the pixels are $P(x, y)$, $Q(s, t)$, and $Z(u, w)$ as shown in Fig. 3.9, the distances between the pixels can be calculated.

The distance function can be called metric if the following properties are satisfied:

1. $D(p, q)$ is well-defined and finite for all p and q.
2. $D(p, q) \geq 0$ if $p = q$, then $D(p, q) = 0$.
3. The distance $D(p, q) = D(q, p)$.
4. $D(p, q) + D(q, z) \geq D(p, z)$. This is called the property of triangular inequality.

0	1	1	1(z)
1	0	0	1
1	1	1	1(q)
1	1	1	1
(p)			

Fig. 3.9 Sample image

The Euclidean distance between the pixels p and q, with coordinates (x, y) and (s, t), respectively, can be defined as

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

The advantage of the Euclidean distance is its simplicity. However, since its calculation involves a square root operation, it is computationally very costly.

The D_4 distance or city block distance can be simply calculated as

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

The D_8 distance or chessboard distance can be calculated as

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

Example 3.1 Let $V = \{0, 1\}$. Compute the D_e, D_4, D_8, and D_m distances between two pixels p and q. Let the pixel coordinates of p and q be (3, 0) and (2, 3), respectively, for the image shown in Fig. 3.10.

Find the distance measures.

	0	1	2	3
0	0	1	1	1
1	1	0	0	1
2	1	1	1	1(q)
3	1	1	1	1
	(p)			

Fig. 3.10 Sample image

Solution The Euclidean distance is

$$D_e = \sqrt{(x-s)^2 + (y-t)^2} = \sqrt{(3-2)^2 + (0-3)^2}$$

$$= \sqrt{1+9} = \sqrt{10}$$

$$D_4 = |x-s| + |y-t| = |3-2| + |0-3|$$

$$= 1 + 3 = 4$$

$$D_8 = \max\left(|x-s|,\ |y-t|\right) = \max\left(|3-2|,\ |0-3|\right)$$

$$= \max(1, 3) = 3$$

The distances can be checked with Figs 3.11(a)–3.11(d). The transition from one element to another is a hop. It can be verified that the calculations match with the hops in Fig. 3.11. The distance D_m depends on the values of the set V. The implication of the set V is that the path should be constructed only using the elements of V. So if the values of the set are changed, the path also changes.

The simplest D_m distance can be calculated along the diagonal path. The distance is 3. Suppose the set $V = \{1\}$, the path of distance D_m also changes. It is shown in Fig. 3.11.

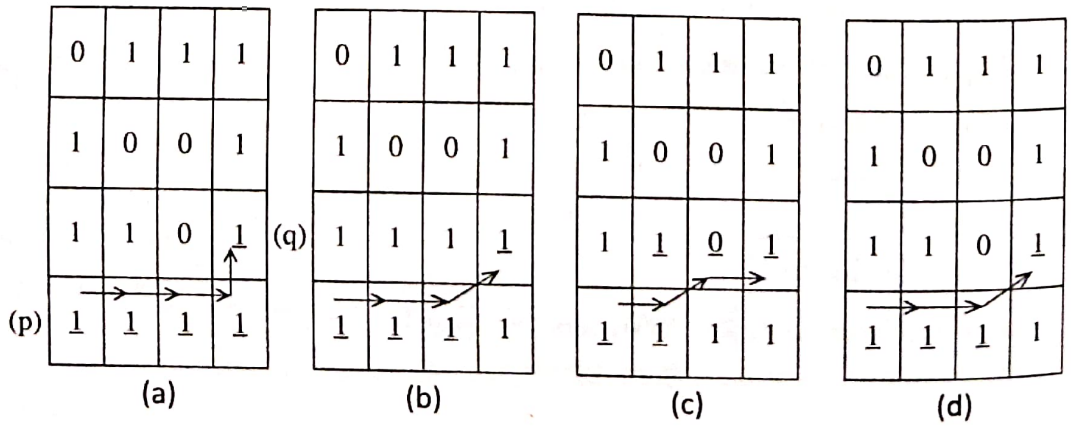


Fig. 3.11 Distance measures (a) Distance D_1 (b) Distance D_0 when $V = \{0, 1\}$ (c) Distance D_m when $V = \{0, 1\}$ (d) Distance D_m when $V = \{1\}$

Example 3.2 Consider the following image with marked pixels p and q.

3	1	2	1	(q)
2	2	0	2	
1	2	1	1	
1	0	1	2	
(p)				

If $V = \{1, 2\}$, find the shortest m-path between pixels p and q.

Solution The shortest path for set V {1,2} is shown here:

3	1	2	1	(q)
2	2	0	2	
1	2	1	1	
1	0	1	2	
(p)				

The shortest m-path is 1 – 2 – 1 – 2 –1. Therefore, D_m distance is 4.

3.1.6 Important Image Characteristics

Some important characteristics of images are as follows:

1. The set of pixels that has connectivity in a binary image is said to be characterized by the connected set.
2. A digital path or curve from pixel p to another pixel q is a set of points $p_1, p_2, \ldots, p_n$. If the coordinates of those points are $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, then $p = (x_0, y_0)$ and $q = (x_n, y_n)$. The number of pixels is called the length. If $x_0 = x_n$ and $y_0 = y_n$, then the path is called a closed path.
3. R is called a region if it is a connected component.
4. If a path between any two pixels p and q lies within the connected set S, it is called a connected component of S. If the set has only one connected component, then the set S is called a connected set. A connected set is called a region.
5. Two regions R_1 and R_2 are called adjacent if the union of these sets also forms a connected component. If the regions are not adjacent, it is called disjoint set. In Fig. 3.12, two regions R_1 and R_2 are shown. These regions are 8-connected because the pixels (underlined pixel '1') have 8-connectivity. If the regions are not adjacent, they are called disjoint.
6. The border of the image is called contour or boundary. A boundary is a set of pixels covering a region that has one or more neighbours outside the region. Typically, in a binary image, there is a foreground object and a background object. The border of the foreground object may have at least one neighbour in the background. If the border pixels are within the region itself, it is called inner boundary. This need not be closed.

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

Region 1 Region 2

Fig. 3.12 Neighbouring regions

7. Edges are present wherever there is an abrupt intensity change among the pixels. Edges are similar to boundaries, but may or may not be connected. If edges are disjoint, they have to be linked together by edge linking algorithms. However, boundaries are global and have a closed path. Figure 3.13 illustrates two regions and an edge. It can be observed that edges provide an outline of the object. The pixels that are covered by the edges lead to regions.

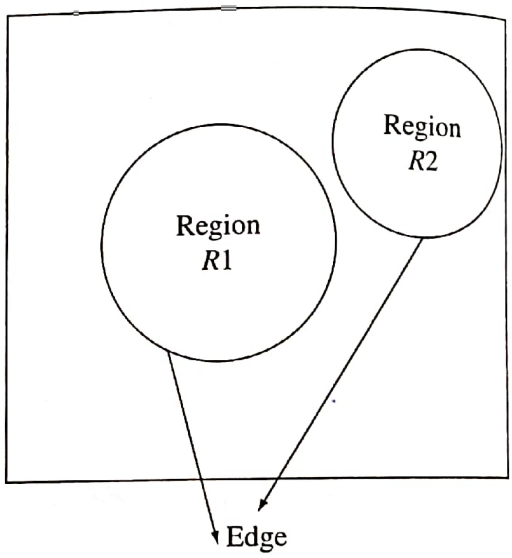


Fig. 3.13 Edge and regions

3.2 CLASSIFICATION OF IMAGE PROCESSING OPERATIONS

There are various ways to classify image operations. The reason for categorizing the operations is to gain an insight into the nature of the operations, the expected results, and the kind of computational burden that is associated with them.

One way of categorizing the operations based on neighbourhood is as follows:

1. Point operations 2. Local operations 3. Global operations

Point operations are those whose output value at a specific coordinate is dependent only on the input value. A local operation is one whose output value at a specific coordinate is dependent on the input values in the neighbourhood of that pixel. Global operations are those whose output value at a specific coordinate is dependent on all the values in the input image.

Another way of categorizing operations is as follows:

1. Linear operations 2. Non-linear operations

An operator is called a linear operator if it obeys the following rules of additivity and homogeneity.

1. Property of additivity

$$\begin{aligned} H(a_1 f_1(x, y) + a_2 f_2(x, y)) &= H(a_1 f_1(x, y)) + H(a_2 f_2(x, y)) \\ &= a_1 H(f_1(x, y)) + a_2 H(f_2(x, y)) \\ &= a_1 \times g_1(x, y) + a_2 \times g_2(x, y) \end{aligned}$$

2. Property of homogeneity

$$H(kf_1(x,y)) = kH(f_1(x,y)) = kg_1(x,y)$$

A non-linear operator, as the name suggests, does not follow these rules.

Image operations are array operations. These operations are done on a pixel-by-pixel basis. Array operations are different from matrix operations. For example, consider two images

$$F_1 = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \text{ and } F_2 = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

The multiplication of F_1 and F_2 is element-wise, as follows:

$$F_1 \times F_2 = \begin{pmatrix} AE & BF \\ CG & HD \end{pmatrix}$$

In addition, one can observe that $F_1 \times F_2 = F_2 \times F_1$, whereas matrix multiplication is clearly different, since in matrices, $A \times B \neq B \times A$. By default, image operations are array operations only.

3.2.1 Arithmetic Operations

Arithmetic operations include image addition, subtraction, multiplication, division, and blending. The following sections discuss the usage of these operations.

3.2.1.1 *Image addition*

Two images can be added in a direct manner, as given by

$$g(x,y) = f_1(x,y) + f_2(x,y)$$

The pixels of the input images $f_1(x,y)$ and $f_2(x,y)$ are added to obtain the resultant image $g(x,y)$. Figure 3.14 shows the effect of adding a noise pattern to an image. However, during the image addition process, care should be taken to ensure that the sum does not cross the allowed range. For example, in a grey scale image, the allowed range is 0–255, using eight bits. If the sum is above the allowed range, the pixel value is set to the maximum allowed value. The range values of select MATLAB data types are listed in Table 3.1.

Table 3.1 Data type and allowed ranges

S. no.	Data type	Data range
1	Uint 8	0–255
2	Uint 16	0–65,535
3	Uint 32	0–4,29,49,67,295
4	Uint 64	0–1,84,46,74,40,73,70,95,51,615

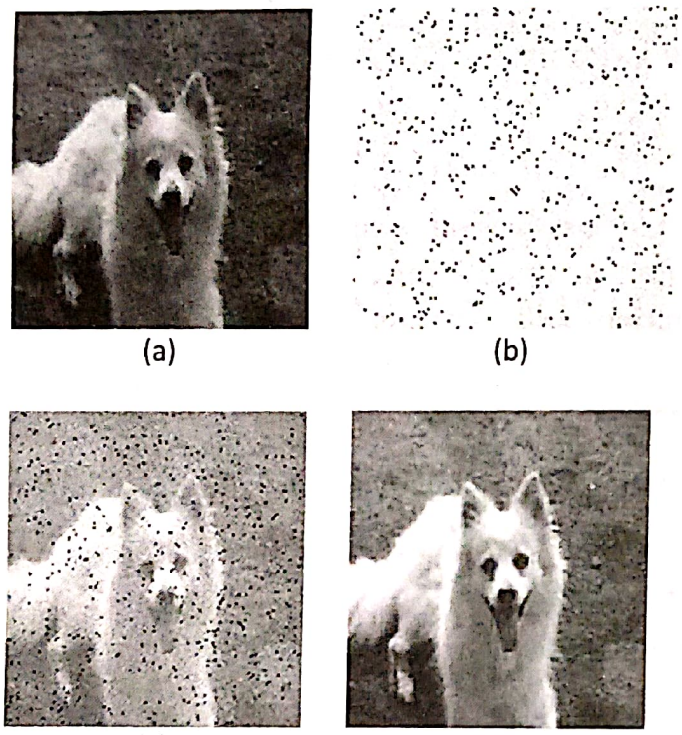
Similarly, it is possible to add a constant value to a single image, as follows:

$$g(x,y) = f_1(x,y) + k$$

If the value of k is larger than 0, the overall brightness is increased. Figure 3.14(d) illustrates that the addition of the constant 50 increases the brightness of the image. Why?

The brightness of an image is the average pixel intensity of an image. If a positive or negative constant is added to all the pixels of an image, the average pixel intensity of the image increases or decreases, respectively. This is covered in Section 3.2.1.2. The practical applications of image addition are as follows:

1. To create double exposure. Double exposure is the technique of superimposing an image on another image to produce the resultant. This gives a scenario equivalent to exposing a film to two pictures. This is illustrated in Figs 3.14(a)–3.14(c).
2. To increase the brightness of an image.



(a) (b) (c) (d)

Fig. 3.14 Results of the image addition operation (a) Image 1 (b) Image 2 (c) Addition of images 1 and 2 (d) Addition of image 1 and constant 50

3.2.1.2 Image subtraction

The subtraction of two images can be done as follows. Consider

$$g(x, y) = f_1(x, y) - f_2(x, y)$$

where $f_1(x, y)$ and $f_2(x, y)$ are two input images and $g(x, y)$ is the output image. To avoid negative values, it is desirable to find the modulus of the difference as

$$g(x, y) = |f_1(x, y) - f_2(x, y)|$$

It is also possible to subtract a constant value k from the image, i.e., $g(x, y) = |f_1(x, y) - k|$, as k is constant. As discussed earlier, the decrease in the average intensity reduces the brightness of the image. Some of the practical applications of image subtraction are as follows:

1. Background elimination
2. Brightness reduction
3. Change detection

If there is no difference between the frames, the subtraction process yields zero, and if there is any difference, it indicates the change. Figures 3.15(a)–3.15(d) show the difference between the images. In addition, it illustrates that the subtraction of a constant results in a decrease of the brightness.

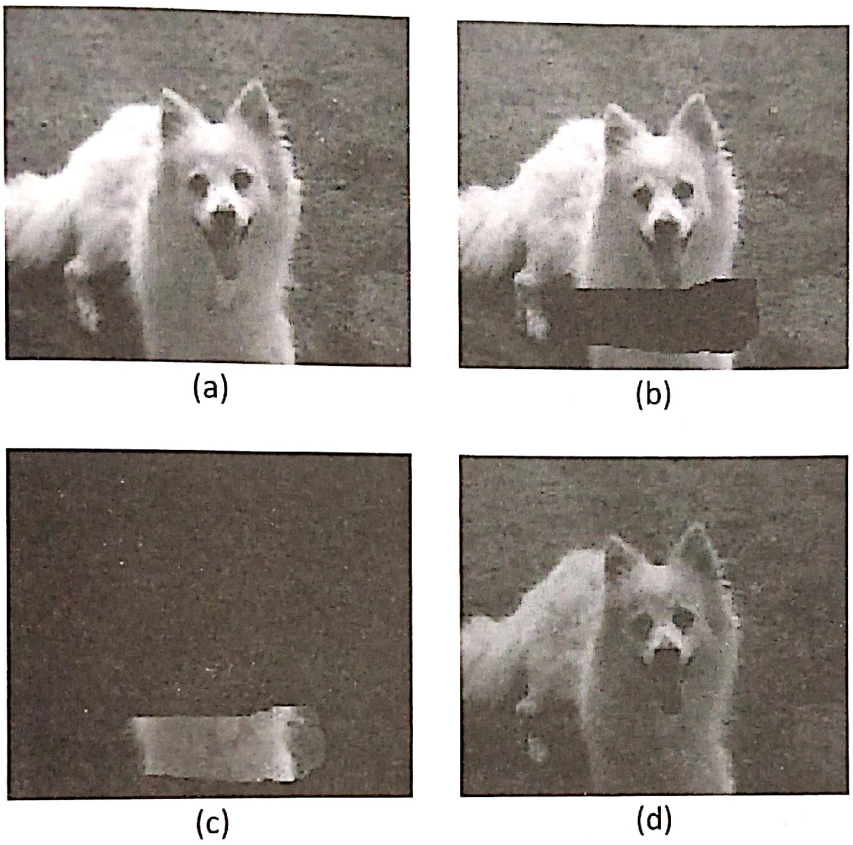


Fig. 3.15 Results of the image subtraction operation (a) Image 1 (b) Image 2 (c) Subtraction of images 1 and 2 (d) Subtraction of constant 50 from image 1

3.2.1.3 Image multiplication

Image multiplication can be done in the following manner:
Consider

$$g(x, y) = f_1(x, y) \times f_2(x, y)$$

Here $f_1(x, y)$ and $f_2(x, y)$ are two input images and $g(x, y)$ is the output image. If the multiplied value crosses the maximum value of the data type of the images, the value of the pixel is reset to the maximum allowed value. Similarly, scaling by a constant can be performed as

$$g(x, y) = f(x, y) \times k$$

where k is a constant.

If k is greater than 1, the overall contrast increases. If k is less than 1, the contrast decreases. The brightness and contrast can be manipulated together as

$$g(x, y) = af(x, y) + k$$

Here, the parameters a and k are used to manipulate the brightness and contrast of the input image. $g(x, y)$ is the output image. Some of the practical applications of image multiplication are as follows:

1. It increases contrast. If a fraction less than 1 is multiplied with the image, it results in decrease of contrast. Figure 3.16 shows that by multiplying a factor of 1.25 with the original image, the contrast of the image increases.
2. It is useful for designing filter masks.
3. It is useful for creating a mask to highlight the area of interest.

Fig. 3.16 Result of multiplication operation (image × 1.25) resulting in good contrast

3.2.1.4 Image division

Similar to the other operations, division can be performed as

$$g(x, y) = \frac{f_1(x, y)}{f_2(x, y)}$$

where $f_1(x, y)$ and $f_2(x, y)$ are two input images and $g(x, y)$ is the output image.

The division process may result in floating-point numbers. Hence, the float data type should be used in programming. Improper data type specification of the image may result in loss of information. Division using a constant can also be performed as

$$g(x, y) = \frac{f(x, y)}{k}, \text{ where } k \text{ is a constant.}$$

Some of the practical applications of image division are as follows:

1. Change detection
2. Separation of luminance and reflectance components
3. Contrast reduction. Figure 3.17(a) shows such an effect when the original image is divided by 1.25.

Figures 3.17(b)–3.17(e) show the multiplication and division operations used to create a mask. It can be observed that image 2 is used as a mask. The multiplication of image 1 with image 2 results in highlighting certain portions of image 1 while suppressing the other portions. It can be observed that division yields back the original image.

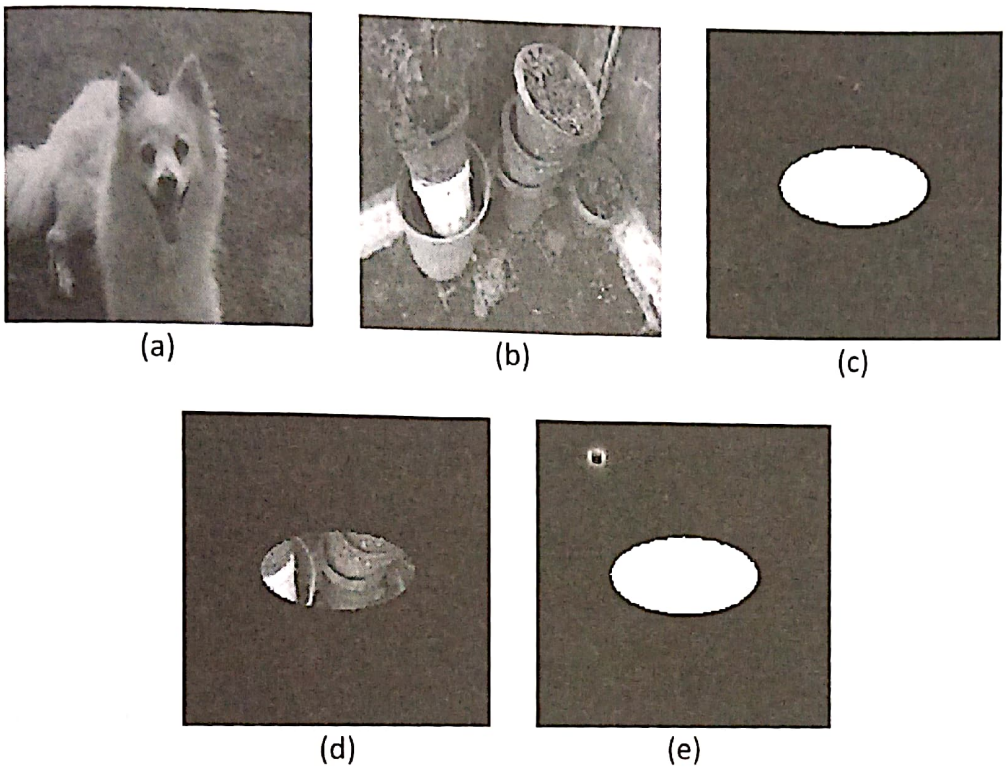


Fig. 3.17 Image division operation (a) Result of the image division operation (image/1.25) (b) Image 1 (c) Image 2 used as a mask (d) Image 3 = image 1 × image 2 (e) Image 4 = image 3/image 1

Example 3.3 Consider the following two images.

$$f_1 = \begin{pmatrix} 1 & 3 & 7 \\ 5 & 15 & 75 \\ 200 & 50 & 150 \end{pmatrix} \quad f_2 = \begin{pmatrix} 50 & 150 & 125 \\ 45 & 55 & 155 \\ 200 & 50 & 75 \end{pmatrix}$$

Perform addition, subtraction, multiplication, division, and image blending operations. Assume both the images are of the 8-bit integer type (uint8 of MATLAB type).

Solution

Addition:

$$g = f_1 + f_2 = \begin{pmatrix} 1+50 & 3+150 & 7+125 \\ 5+45 & 15+55 & 75+155 \\ 200+200 & 50+50 & 150+75 \end{pmatrix} = \begin{pmatrix} 51 & 153 & 132 \\ 50 & 70 & 230 \\ 400 & 100 & 225 \end{pmatrix}$$

If the data type unit8 is assumed, the minimum and maximum allowed values are 0 and 255, respectively. So if the value is larger than 255, it is reset to 255. Similarly, if the value is less than 0, it is reset to 0 (Table 3.1).

So the result of the image addition is

$$g = \begin{pmatrix} 51 & 153 & 132 \\ 50 & 70 & 230 \\ 255 & 100 & 225 \end{pmatrix}$$

Subtraction:

$$g = f_1 - f_2 = \begin{pmatrix} 1-50 & 3-150 & 7-125 \\ 5-45 & 15-55 & 75-155 \\ 200-200 & 50-50 & 150-75 \end{pmatrix} = \begin{pmatrix} -49 & -147 & -118 \\ -40 & -40 & -80 \\ 0 & 0 & 75 \end{pmatrix}$$

Since the data type is unit8, values less than 0 are reset to 0.

$$g = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 75 \end{pmatrix}$$

It can be observed that the modulus of the difference results in a different image.

Multiplication:

$$g = f_1 \times f_2 = \begin{pmatrix} 1\times 50 & 3\times 150 & 7\times 125 \\ 5\times 45 & 15\times 55 & 75\times 155 \\ 200\times 200 & 50\times 50 & 150\times 75 \end{pmatrix} = \begin{pmatrix} 50 & 450 & 875 \\ 225 & 825 & 11625 \\ 40000 & 2500 & 11250 \end{pmatrix}$$

Since the data type is unit8, values greater than 255 are reset to 255.

$$g = \begin{pmatrix} 50 & 255 & 255 \\ 225 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}$$

Division:

$$g = f_1 / f_2 = \begin{pmatrix} 1/50 & 3/150 & 7/125 \\ 5/45 & 15/55 & 75/155 \\ 200/200 & 50/50 & 150/75 \end{pmatrix} = \begin{pmatrix} 0.02 & 0.02 & 0.056 \\ 0.11 & 0.272 & 0.484 \\ 1 & 1 & 2 \end{pmatrix}$$

$$g = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 2 \end{pmatrix}$$

This resultant image is due to truncation.

Image or alpha blending: This operation blends two images of the same size to yield a resultant image. This operation is useful for transparency and compositing. The resultant image is a linear combination of two input images. This can be mathematically stated as

$$g(x, y) = \alpha \times f_1(x, y) + (1 - \alpha) \times f_2(x, y)$$

where $f_1(x, y)$ and $f_2(x, y)$ are two input images and $g(x, y)$ is the output image, and the parameter α is called blending ratio, which determines the influence of each image on the resultant image.

Example 3.4 Consider the following 4×4, 8 level images A and B. Find $A + B$, $A - B$, $A \times B$, and A/B.

$A =$

1	2	3	4
5	5	6	6
6	7	6	6
6	7	2	3

$B =$

1	3	5	7
8	7	0	1
3	5	6	7
1	3	5	7

Solution The given images are 4×4 and 8 grey level (0 – 7) images. As the grey levels are 0 – 7, any value above 7 is reduced to 7.

$A + B =$

1 + 1	2 + 3	3 + 5	4 + 7
5 + 8	5 + 7	6 + 0	6 + 1
6 + 3	7 + 5	6 + 6	6 + 7
6 + 1	7 + 3	2 + 5	3 + 7

=

2	5	8	11
13	12	6	7
9	12	12	13
7	10	7	10

=

2	5	7	7
7	7	6	7
7	7	7	7
7	7	7	7

$A - B =$

1 – 1	2 – 3	3 – 5	4 – 7
5 – 8	5 – 7	6 – 0	6 – 1
6 – 3	7 – 5	6 – 6	6 – 7
6 – 1	7 – 3	2 – 5	3 – 7

=

0	–1	–2	–3
–3	–2	6	5
3	2	0	–1
5	4	–3	–4

=

0	0	0	0
0	0	6	5
3	2	0	0
5	4	0	0

$A \times B =$

1 × 1	2 × 3	3 × 5	4 × 7
5 × 8	5 × 7	6 × 0	6 × 1
6 × 3	7 × 5	6 × 6	6 × 7
6 × 1	7 × 3	2 × 5	3 × 7

=

1	6	15	28
40	35	0	6
18	35	36	42
6	21	10	21

=

1	6	7	7
7	7	0	6
7	7	7	7
6	7	7	7

$A/B =$

1/1	2/3	3/5	4/7
5/8	5/7	6/0	6/1
6/3	7/5	6/6	6/7
6/1	7/3	2/5	3/7

=

1	0	0	0
0	0	0	6
2	1	1	0
6	2	0	0

3.2.1.5 Applications of arithmetic operations

Arithmetic operations can be combined and put to effective use. For example, the image averaging process can be used to remove noise. Noise is a random fluctuation of pixel values, which affects the quality of the image. A noisy image can be considered as an image plus noise:

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

where $f(x, y)$ is the input image and $g(x, y)$ is the output image. Several instances of noisy images can be taken and averaged as

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

where M is the number of noisy images. As M increases, the averaging process reduces the intensity of the noise and it becomes so low that it can automatically be removed. As M becomes large, the expectation $E\{\bar{g}(x, y)\} = f(x, y)$.

3.2.2 Logical Operations

Bitwise operations can be applied to image pixels. The resultant pixel is defined by the rules of the particular operation. Some of the logical operations that are widely used in image processing are as follows:

1. AND/NAND
2. OR/NOR
3. EXOR/EXNOR
4. Invert/Logical NOT

3.2.2.1 AND/NAND

The truth table of the AND and NAND operators is given in Table 3.2.

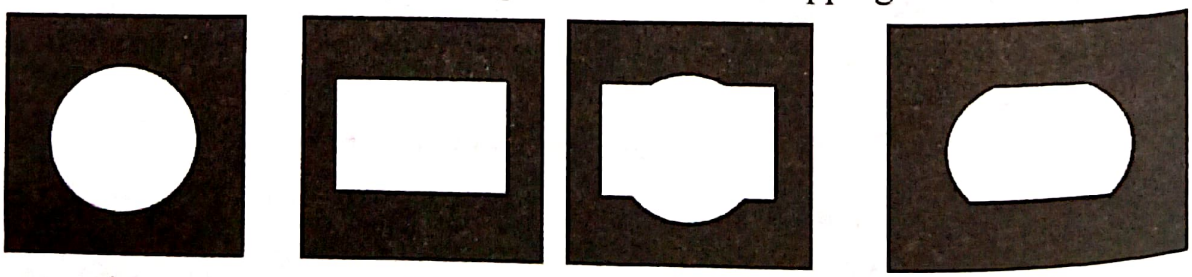
Table 3.2 Truth table of the AND and NAND operators

A	B	C (AND)	C (NAND)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

The operators AND and NAND take two images as input and produce one output image. The output image pixels are the output of the logical AND/NAND of the individual pixels. Some of the practical applications of the AND and NAND operators are as follows:

1. Computation of the intersection of images
2. Design of filter masks
3. Slicing of grey scale images; for example, the pixel value of the grey scale image may be 1100 0000. The first bits of the pixels of an image constitute one slice. To extract the first slice, a mask of value 1000 0000 can be designed. The AND operation of the image pixel and the mask can extract the first bit and first slice of the image.

Figures 3.18(a)–3.18(d) shows the effects of the AND and OR logical operators. It illustrates that the AND operator shows overlapping regions of the two input images and the OR operator shows all the input images with their overlapping.



(a) (b) (c) (d)

Fig. 3.18 Results of the AND and OR logical operators (a) Image 1 (b) Image 2 (c) Result of image 1 OR image 2 (d) Result of image 1 AND image 2

3.2.2.2 OR/NOR

The truth table of the OR and NOR operators is given in Table 3.3.

Table 3.3 Truth table of the OR and NOR operators

A	B	C (OR)	C (NOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

The practical applications of the OR and NOR operators are as follows:

1. OR is used as the union operator of two images.
2. OR can be used as a merging operator.

3.2.2.3 XOR/XNOR

The truth table of the XOR and XNOR operators is given in Table 3.4.

Table 3.4 Truth table of the XOR and XNOR operators

A	B	C (XOR)	C (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The practical applications of the XOR and XNOR operators are as follows:

1. Change detection
2. Use as a subcomponent of a complex imaging operation. XOR for identical inputs is zero. Hence it can be observed that the common region of image 1 and image 2 in Figs 3.18(a) and 3.18(b), respectively, is zero and hence dark. This is illustrated in Fig. 3.19.

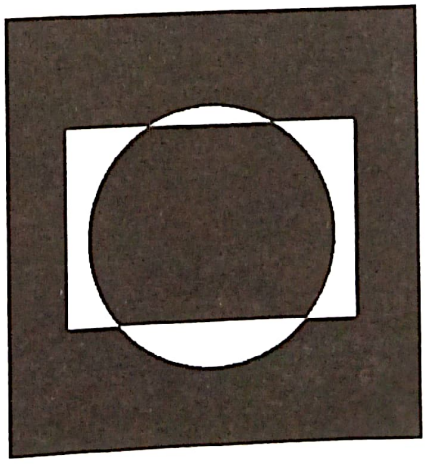


Fig. 3.19 Result of the XOR operation

3.2.2.4 Invert/Logical NOT

The truth table of the NOT operator is given in Table 3.5.

For grey scale values, the inversion operation is described as

$$g(x, y) = 255 - f(x, y)$$

The practical applications of the inversion operator are as follows:

1. Obtaining the negative of an image. Figure 3.20 shows the negative of the original image shown in Fig. 3.18(a).
2. Making features clear to the observer
3. Morphological processing

Table 3.5 Truth table of the NOT operator

A	C (NOT)
0	1
1	0

Similarly, two images can be compared using operators such as

$=$ Equal to

$>$ Greater than

$\geq$ Greater than or equal to

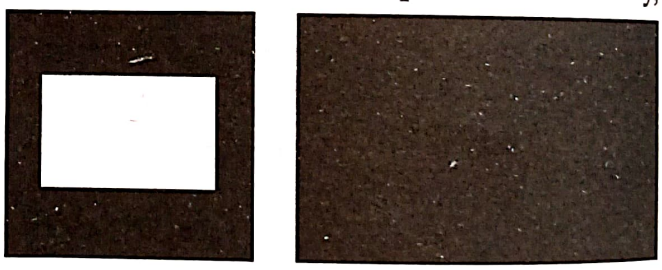
$<$ Less than

$\leq$ Less than or equal to

$\neq$ Not equal to

The resultant image pixel represents the truth or falsehood of the comparisons. Similarly, shifting operations are also very useful. Shifting of I bits of the image pixel to the right results in division by 2^I. Similarly, shifting of I bits of the image pixel to the left results in multiplication by 2^I.

Shifting operators are helpful in dividing and multiplying an image by a power of two. In addition, this operation is computationally less expensive.



(a) (b)

Fig. 3.20 Effect of the NOT operator (a) Original Image (b) NOT of original Image

Example 3.5 Consider the following two images.

$$f_1 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad f_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Perform the logical AND, OR, NOT, and difference operations.

Solution

AND

$$f_1 \text{ AND } f_2 = \begin{pmatrix} 1 \wedge 1 & 0 \wedge 1 & 0 \wedge 1 \\ 1 \wedge 1 & 1 \wedge 1 & 1 \wedge 1 \\ 0 \wedge 1 & 0 \wedge 1 & 1 \wedge 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

OR

$$f_1 \text{ OR } f_2 = \begin{pmatrix} 1 \vee 1 & 0 \vee 1 & 0 \vee 1 \\ 1 \vee 1 & 1 \vee 1 & 1 \vee 1 \\ 0 \vee 1 & 0 \vee 1 & 1 \vee 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

NOT

$$\text{NOT}(f_1) = \begin{pmatrix} \neg 1 & \neg 0 & \neg 0 \\ \neg 1 & \neg 1 & \neg 1 \\ \neg 0 & \neg 0 & \neg 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Difference

$$f_1 \text{ AND } (\neg f_2) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and

$$f_2 \text{ AND } (\neg f_1) = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

3.2.3 Geometrical Operations

Let us now discuss the geometrical operations used in image processing.

3.2.3.1 *Translation*

Translation is the movement of an image to a new position. Let us assume that the point at the coordinate position $X = (x, y)$ of the matrix $\boldsymbol{F}$ is moved to the new position X' whose coordinate position is (x', y'). Mathematically, this can be stated as a translation of a point X to the new position X'. The translation is represented as

$$x' = x + \delta x$$

$$y' = y + \delta y$$

The translation of the floor image by (25, 25) is shown in Fig. 3.21. In vector notation, this is represented as $\boldsymbol{F}' = \boldsymbol{F} + T$, where δx and δy are translations parallel to the x and y axes. Here, $\boldsymbol{F}$ and $\boldsymbol{F}'$ are the original and the translated images, respectively. However, other transformations such as scaling and rotation are multiplicative in nature. The transformation process for rotation is given as $\boldsymbol{F}' = \boldsymbol{RF}$, where $\boldsymbol{R}$ is the transform matrix for performing rotation, and the transformation process for scaling is given as $\boldsymbol{F}' = \boldsymbol{SF}$. Here, $\boldsymbol{S}$ is the scaling transformation matrix.

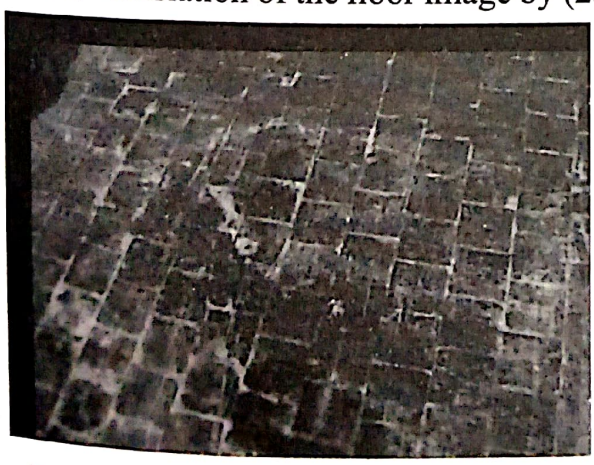


Fig. 3.21 Result of translation by 50 units

To create uniformity and consistency, it is necessary to use a homogeneous coordinate system where all transformations are treated as multiplications. A point (x, y) in 2D space is expressed as (wx, wy, w) for $w \neq 0$. The properties of homogeneous coordinates are as follows:

1. In homogeneous coordinates, at least one point should be non-zero. Thus (0, 0, 0) does not exist in the homogeneous coordinate system.
2. If one point is multiplicative of the other point, they are same. Thus, the points (1, 3, 5) and (3, 9, 15) are same as the second point is $3 \times (1, 3, 5)$.
3. The point (x, y, w) in the homogenous coordinate system corresponds to the point $\left(\frac{x}{w}, \frac{y}{w}\right)$ in 2D space.

In the homogeneous coordinate system, the translation process of point (x, y) of image $\mathbf{F}'$ to the new point (x', y') of the image $\mathbf{F}'$ is described as

$$x' = x + \delta x$$

and

$$y' = y + \delta y$$

In matrix form, this can be stated as

$$[x', y', 1] = \begin{pmatrix} 1 & 0 & \delta x \\ 0 & 1 & \delta y \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^{\mathrm{T}}$$

Sometimes, the image may not be present at the origin. In that case, a suitable negative translational value can be used to bring the image to align with the origin.

3.2.3.2 Scaling

Depending on the requirement, the object can be scaled. *Scaling* means enlarging and shrinking. In the homogeneous coordinate system, the scaling of the point (x, y) of the image $\mathbf{F}$ to the new point (x', y') of the image $\mathbf{F}'$ is described as

$$x' = x \times S_x$$
$$y' = y \times S_y$$

$$[x', y'] = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} [x, y]$$

S_x and S_y are called scaling factors along the x and y axes, respectively. If the scale factor is 1, the object would appear larger. If the scaling factors are fractions, the object would shrink. Similarly, if S_x and S_y are equal, scaling is uniform. This is known as *isotropic scaling*. Otherwise, it is called *differential scaling*. In the homogeneous coordinates system, it is represented as

$$[x', y', 1] = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

The matrix $S = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$ is called scaling matrix.

3.2.3.3 Mirror or reflection operation

This function creates the reflection of the object in a plane mirror. The function returns an image in which the pixels are reversed. This operation is useful in creating an image in the desired order and for making comparisons. The reflected object is of the same size as the original object, but the object is in the opposite quadrant. Reflection is also described as rotation by 180°. The reflection along the x-axis is given by

$$F' = [-x, y] = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \times [x, y]^T$$

Similarly, the reflection along the y-axis is given by

$$F' = [x, -y] = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \times [x, y]^T$$

Similarly, the reflection about the line $y = x$ is given as

$$F' = [x, -y] = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times [x, y]^T$$

(a) (b)

Fig. 3.22 Mirror operation (a) Original image (b) Reflection along the y-axis

The reflection about $y = -x$ is given as

$$F' = [x, -y] = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \times [x, y]^T$$

The reflection operation is illustrated in Figs 3.22(a) and 3.22(b). In the homogeneous coordinate system, the matrices for reflection can be given as

$$R_{y\text{-axis}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; R_{x\text{-axis}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; R_{\text{origin}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Similarly, the reflection along a line can be given as

$$R_{y=x} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}; R_{y=-x} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.2.3.4 Shearing

Shearing is a transformation that produces a distortion of shape. This can be applied either in the x-direction or the y-direction. In this transformation, the parallel and opposite layers of the object are simply sided with respect to each other.

Shearing can be done using the following calculation and can be represented in the matrix form as

$$x' = x + ay$$
$$y' = y$$
$$X_{\text{shear}} = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ (where } a = sh_x)$$

Similarly, Y_{shear} can be given as

$$x' = x$$
$$y' = y + bx$$
$$Y_{\text{shear}} = \begin{pmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ (where } b = sh_y)$$

where sh_x and sh_y are shear factors in the x and y directions, respectively.

3.2.3.5 Rotation

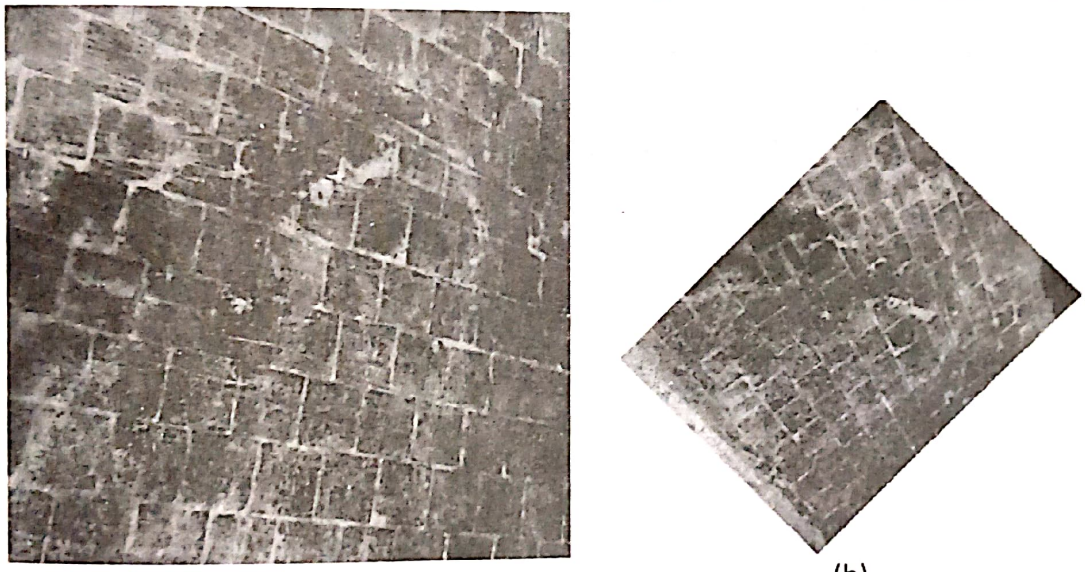
An image can be rotated by various degrees such as 90°, 180°, or 270°. In the matrix form, it is given as

$$[x', y'] = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} [x, y]^T$$

This can be represented as $\boldsymbol{F'} = \boldsymbol{RA}$. The parameter θ is the angle of rotation with respect to the x-axis. It is assumed that the object rotation is about the origin. The value of θ can be positive or negative. A positive angle represents counter clockwise rotation and a negative angle represents clockwise rotation. The rotation of an image is shown in Figs 3.23(a) and 3.23(b). In the homogeneous coordinate system, rotation can be represented as

$$[x', y', 1] = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

If θ is substituted with $-\theta$, this matrix rotates the image in the clockwise direction.



(a) (b)

Fig. 3.23 Rotation (a) Original image (b) Result of rotation by 45°

3.2.3.6 *Affine transform*

The transformation that maps the pixel at the coordinates (x, y) to a new coordinate position is given as a pair of transformation equations. In this transform, straight lines are preserved and parallel lines remain unchanged. It is described mathematically as

$$x' = T_x(x, y)$$

$$y' = T_y(x, y)$$

T_x and T_y are expressed as polynomials. The linear equation gives an affine transform.

$$x' = a_0 x + a_1 y + a_2$$

$$y' = b_0 x + b_1 y + b_2$$

This is expressed in matrix form as

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

The affine transform is a compact way of representing all transformations. The given equation represents all transformations. Translation is the situation where $a_0 = 1$, $a_1 = 0$, and $a_2 = \delta_x$; scaling transformation is a situation where $a_0 = s_x$ and $b_1 = s_y$ and $a_1 = 0$, $a_2 = 0$, $b_0 = 0$, and $b_2 = 0$; rotation is a situation where $a_0 = \cos\theta$, $a_1 = -\sin\theta$, $b_0 = \sin\theta$, $b_1 = \cos\theta$, $a_2 = 0$, and $b_2 = 0$; and horizontal shear is performed when $a_0 = 1$, $b_0 = 1$, $a_1 = Sh_x$, $a_2 = 0$, $b_1 = Sh_y$ and $b_2 = 0$.

3.2.3.7 *Inverse transformation*

The purpose of inverse transformation is to restore the transformed object to its original form and position. The inverse or backward transformation matrices are given as follows:

$$\text{Inverse transform for translation} = \begin{pmatrix} 1 & 0 & -\delta x \\ 0 & 1 & -\delta y \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{Inverse transform for scaling} = \begin{pmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Inverse transform for rotation can be obtained by changing the sign of the transform term. For example, the following matrix performs inverse transform.

$$\begin{pmatrix} \cos\theta & +\sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Example 3.6 Consider an image point [2, 2]. Perform the following operations and show the results of these transforms.

(a) Translate the image right by 3 units.

(b) Perform a scaling operation in both x-axis and y-axis by 3 units.

(c) Rotate the image in x-axis by 45°.

(d) Perform horizontal skewing by 45°.

(e) Perform mirroring about x-axis.

(f) Perform shear in y-direction by 30 units.

Solution

(a) Translation of the image right by 3 units means that

$$\delta_x = 3 \text{ and } \delta_y = 0$$

So the translation matrix is given as

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore,

$$x' = \boldsymbol{T} \times x$$

$$= \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2, 2, 1] = [5 \quad 2 \quad 1]^{\mathrm{T}}$$

(b) Scaling by 3 units in both the directions means that

$$\boldsymbol{S} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{S} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2, 2, 1]^{\mathrm{T}} = [6 \quad 6 \quad 1]^{\mathrm{T}}$$

(c) Rotating the image in x-axis by 45°

$$\boldsymbol{R} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{R} = \begin{pmatrix} 0.707 & -0.707 & 0 \\ 0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2, 2, 1]^{\mathrm{T}} = [0 \quad 2.828 \quad 1]^{\mathrm{T}}$$

The fraction 2.828 will be rounded to 3. This is determined by the interpolation technique used.

(d) Performing horizontal skewing by 45°

$$Skew = \begin{pmatrix} 1 & 0 & 0 \\ \tan\varphi & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \tan 45 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Skew = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2, 2, 1]^{\mathrm{T}} = [2, 4, 1]^{\mathrm{T}}$$

(e) Performing mirroring about x-axis

$$M_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Mirroring = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2,2,1] = [2 \quad -2 \quad 1]^T$$

(f) Performing shear in y-direction by 30 units

$$Shear_y = \begin{pmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 30 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Mirroring = \begin{pmatrix} 1 & 30 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times [2, 2, 1]^T = [62 \quad 2 \quad 1]^T$$

3.2.3.8 3D Transforms

Some medical images such as computerized tomography (CT) and magnetic resonance imaging (MRI) images are three-dimensional images. To apply translation, rotation, and scaling on 3D images, 3D transformations are required. 3D transformations are logical extensions of 2D transformations. These are summarized and described as follows:

$$\text{Translation} = \begin{pmatrix} 0 & 0 & 0 & \delta x \\ 0 & 1 & 0 & \delta y \\ 0 & 0 & 1 & \delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Scaling} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation along z-axis

$$R_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation along x-axis

$$\boldsymbol{R}_{x,\theta} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation along y-axis

$$\boldsymbol{R}_{y,\theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Compactly, these can be represented as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = C \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Similarly, the reflection transform matrices can be given as

$$\text{Reflection}_{xy\text{-plane}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Reflection}_{xz\text{-plane}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Reflection}_{yz\text{-plane}} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The shear matrices for the shear quantities a and b can be given as

$$\text{Shear}_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Shear}_x = \begin{pmatrix} 1 & a & b & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Shear}_y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ a & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.2.4 Image Interpolation Techniques

What is the need for interpolation techniques? Transforms can be of two types. Affine transforms often produce pixels of the resultant image that cannot be fit as some of the pixel values are non-integers and often go beyond the acceptable range. This results in gaps (or holes) and issues related to number of pixels and range. So interpolation techniques are required to solve these issues.

3.2.4.1 Forward mapping

Forward mapping is the process of applying transformations iteratively to every pixel in the image, yielding a new coordinate position and copying the values of the pixel to a new position.

3.2.4.2 Backward mapping

Similarly, backward mapping is the process of checking the pixels of the output image to determine the position of the pixels in the input image. This is used to guarantee that all the pixels of the input image are processed.

During the process of both forward and backward mapping, it may happen that pixels cannot be fitted into the new coordinates. For example, consider the process of rotation of a point, (10, 5), by 45°. This yields

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ &= 10\cos(45°) - 5\sin(45°) \\ &= 10(0.707) - 5(0.707) \\ &= 3.535 \\ y' &= x\sin\theta + y\cos\theta \\ &= 10\sin(45°) + 5\cos(45°) \\ &= 10(0.707) + 5(0.707) \\ &= 10.605 \end{aligned}$$

Since these new coordinate positions are not integers, the rotation process cannot be carried out. Thus, the process may leave a gap in the new coordinate position, which creates poor quality output. Therefore, whenever a geometric transformation is performed, a resampling process should be carried out so that the desirable quality is achieved in the resultant image. The resampling process creates new pixels so that the quality of the output

is maintained. In addition, the rounding off of the new coordinate position (3.535, 10.605) should be carried out as (4, 11). This process of fitting the output to the new coordinates is called interpolation.

Interpolation is the method of calculating the expected values for a function with known pixels. Some of the popular interpolation techniques are

1. Nearest neighbour technique
2. Bilinear technique
3. Bicubic technique

The most elementary form of interpolation is nearest neighbour interpolation or zero-order interpolation. This technique determines the closest pixel and assigns it to every pixel in the new image matrix, that is, the brightness of the pixels is equal to the closest neighbour. Sometimes, this may result in pixel blocking and can degrade the resulting image, which may appear spatially distorted. These distortions are called aliasing.

A more accurate interpolation scheme is bilinear interpolation. This is called first-order interpolation. Here, four neighbours of the transformed original pixels that surround the new pixel are obtained and are used to calculate the new pixel value. Linear interpolation is used in both the directions. Weights are assigned based on the proximity. Then the process takes the weighted average of the brightness of the four pixels that surround the pixels of interest.

$$g(x, y) = (1 - a)(1 - b) f(x', y') + (1 - a)\, b f(x', y' + 1) + a\,(1 - b) f(x' + 1, y') + a\, b f(x' + 1, y' + 1)$$

Here, g(x, y) is the output image and f(x, y) is the image that undergoes the interpolation operation. If the desired pixel is very close to one of the four nearest neighbour pixels, its weight will be much higher. This technique leads to blurring of the edges. However, it reduces aliasing artefacts.

High-order interpolation scheme takes more pixels into account. Second-order interpolation is known as cubic interpolation; it uses a neighbourhood of 16 pixels. Then it fits two polynomials to the 16 pixels of the transformed original matrix and the new image pixel. This technique is very effective and produces images that are very close to the original. In extreme cases, more than 64 neighbouring pixels can be used. However, as the number of pixels increases, the computational complexity also increases. The results of interpolation are shown in Fig. 3.24.

(a) (b) (c)

Fig. 3.24 Result of interpolation (a) Nearest neighbour (b) Bilinear (c) Bicubic

3.2.5 Set Operations

An image can be visualized as a set. For example, the following binary image (Fig. 3.25) can be visualized as a set A, where $A = \{(0, 0), (0, 2), (2, 2)\}$. The coordinates' values represent the value of 1. Set operators can then be applied to the set to get the resultant, which is useful for image analysis.

$$F = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 \end{array}$$

Fig. 3.25 Sample binary image

The complement of set A can be defined as the set of pixels that does not belong to the set A.

$$A^c = \{c/c \notin A\}$$

The reflection of the set is defined as

$$A = \{c = -a, a \in A\}$$

The union of two sets, A and B, can be represented as

$$A \cup B = \{c/(c \in A) \vee (c \in B)\}$$

where the pixel c belongs to A, B, or both.

The intersection of two sets is given as $A \cap B = \{c/(c \in A) \wedge (c \in B)\}$. The pixel c belongs to A, B, or both.

The difference can be expressed as

$$A - B = \{c/(c \in A) \wedge (c \notin B)\}$$

which is equivalent to $A \cap B^c$.

Morphology is a collection of operations based on set theory, to accomplish various tasks such as extracting boundaries, filling small holes present in the image, and removing noise present in the image.

Mathematical morphology is a very powerful tool for analysing the shapes of the objects that are present in the images. The theory of mathematical morphology is based on set theory. One can visualize a binary object as a set. Set theory can then be applied to the sample set. Morphological operators often take a binary image and a mask known as structuring element as input. The set operators such as intersection, union, inclusion, and complement can then be applied to the images. Dilation is one of the two basic operators. It can be applied to binary as well as grey scale images. The basic effect of this operator on a binary image is that it gradually increases the boundaries of the region, while the small holes that are present in the images become smaller.

Let us assume that A and B are a set of pixel coordinates. The dilation of A by B can be denoted as

$$A \oplus B = \{(x, y) + (u, v): (x, y) \in A, (u, v) \in B\}$$

where x and y correspond to the set A, and u and v correspond to the set B. The coordinates are added and the union is carried out to create the resultant set. These kinds of operations are based on Minkowski algebra.

Example 3.7 Consider the following binary image. Show the results of the dilation and erosion operations.

$$F = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 \end{array}$$

Let the structured element S be [1 1] with coordinates $\{(0, 0), (0, 1)\}$. Show the results of dilation and erosion.

Solution The image $\boldsymbol{F}$ can be written as

$$\boldsymbol{F} = \{(0, 2), (1, 2), (2, 1), (2, 2)\}$$
$$S = \{(0, 0), (0, 1)\}$$

The dilation operation is done as follows:
First add the coordinates (0, 0) of S to all the coordinate points of the image set $\boldsymbol{F}$, followed by the second point of the set S.

$$\boldsymbol{F} \text{ Dilation } S = \{(0, 2), (1, 2), (2, 1), (2, 2), (0, 3), (1, 3), (2, 2), (2, 3)\}$$

Remove the repetitions; the union of these sets results in dilation. This results in

$$S = \{(0, 2), (0, 3), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3)\}$$

The erosion is the intersection of these sets.
First subtract the coordinates (0, 0) of S from all the coordinate points of the image set $\boldsymbol{F}$, followed by the second point of the set S.

$$\boldsymbol{F} \text{ Erosion } S = \{(0, 2), (1, 2), (2, 1), (2, 2), (0, 1), (1, 1), (2, 0), (2, 1)\}$$

The erosion is the intersection operation. Find the common element. This results in

$$S = (2, 1)$$

If the coordinates are (x, y) and (s, t), the result would be $(x + s, y + t)$ for dilation and $(x - s, y - t)$ for erosion.
The result of this numerical calculation is shown in Fig. 3.26(a).

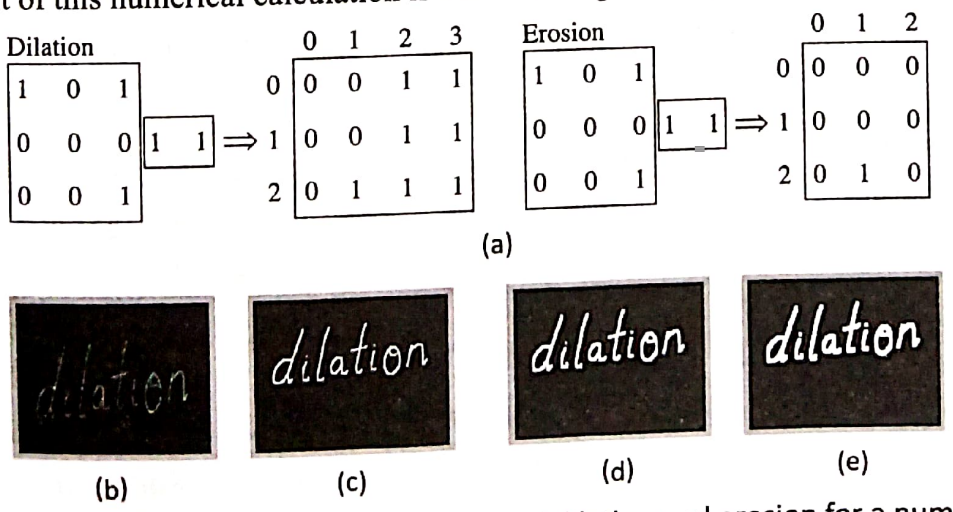


Fig. 3.26 Dilation operation (a) Effects of dilation and erosion for a numerical example (b) Original large image (c) Dilation operation with structural element (of order 3 × 3) (d) Dilation operation with structural element (of order 9 × 9) (e) Dilation operation with structural element (of order 13 × 13)

A dilation process for the bigger images requires a lot of computational effort. For example, the image shown in Fig. 3.26(b) illustrates one application of the dilation operator. The dilation operation with three sets of structural elements is used in Figs 3.26(c)–3.26(e). It can be observed that the brightness of the image increases.

The morphological operators are covered in detail in Chapter 9.

3.2.6 Statistical Operations

Statistics play an important role in image processing. An image can be assumed to be a set of discrete points. Statistical operations can be applied to the image to get the desired results such as manipulation of brightness and contrast. Some of the very useful statistical operations include mean, median, mode, and mid-range. These measures are useful in image processing. The measures of data dispersion also include quartiles, inter-quartile range, and variance.

Some of the frequently used statistical measures are the following:

Mean Mean is the average of all the values in the sample (population) and is denoted as $\bar{X}$.

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

The overall brightness of the grey scale image is measured using the mean. This is calculated by summing all the values of the pixels of an image and dividing it by the number of pixels in the image.

$$\mu = \frac{1}{n}\sum_{i=0}^{n-1} I_i$$

Sometimes the data is associated with a weight. This is called weighted mean. The problem of mean is its extreme sensitivity to noise. Even small changes in the input affect the mean drastically.

Median Median is the value where the given X_i is divided into two equal halves, with half of the values being lower than the median and the other half higher. The procedure for obtaining the median is to sort the values of the given X_i in ascending order. If the given sequence has an odd number of values, the middle value is the median. Otherwise, the median is the arithmetic mean of the two middle values.

Mode Mode is the value that occurs most frequently in the dataset. The procedure for finding the mode is to calculate the frequencies for all of the values in the data. The mode is the value (or values) with the highest frequency. Normally, based on the mode, the dataset is classified as unimodal, bimodal, and trimodal. Any dataset that has two modes is called bimodal.

Percentile Percentiles are data that are less than the coordinate by some percentage of the total value. For example, the median is the 50th percentile and can be denoted as $Q_{0.50}$. The 25th percentile is called the first quartile and the 75th percentile is called third quartile. Another measure that is useful to measure dispersion is the inter-quartile range. The inter-quartile is defined as $Q_{0.75}$ $Q_{0.25}$. Semi quartile range is = 0.5 × IQR.

Unimodal curves are slightly skewed and the empirical relation is

$$\text{Mean} - \text{Mode} = 3 \times (\text{Mean} - \text{Median})$$

The interpretation of the formula is that the mode for the unimodal frequency curve is moderately skewed. The mid-range is also used to assess the central tendency of the dataset. In a normal distribution, the mean, median, and mode are the same. In symmetrical distributions, it is possible for the mean and median to be the same even though there may be several modes. By contrast, in asymmetrical distributions, the mean and median are not the same. These distributions are said to be skewed data where more than half the cases are either above or below the mean.

Standard deviation and variance By far, the most commonly used measures of dispersion are variance and standard deviation. The mean does not convey much more than a middle point. For example, the following datasets, {10, 20, 30} and {10, 50, 0}, both have a mean of 20. The difference between these two sets is the spread of data. Standard deviation is the average distance from the mean of the dataset to each point. The formula for standard deviation is given by

$$\sigma = \sqrt{\sum_{i=1}^{N} (Xi - \mu)^2 / N - 1}$$

Sometimes, we divide the value by $N - 1$ instead of N. The reason is that in a larger, real-world scenario, division by $N - 1$ gives an answer that is closer to the actual value. In image processing, it is a measure of how much a pixel varies from the mean value of the image. The mean value and the standard deviation characterize the perceived brightness and contrast of the image. Variance is another measure of the spread of the data. It is the square of standard deviation. While standard deviation is a more common measure, variance also indicates the spread of the data effectively.

Entropy This is the measure of the amount of orderliness that is present in the image. The entropy can be calculated by assuming that the pixels are totally uncorrelated. An organized system has low entropy and a complex system has a very high entropy. Entropy also indicates the average global information content. Its unit is bits per pixel. It can be computed using the formula

$$\text{Entropy } H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the prior probability of the occurrence of the message. Let us consider a binary image, where the pixel assumes only two possible states, 0 or 1 and the occurrence of each

state is equally likely. Hence, the probability is 1/2. Therefore, the entropy is $H = -[1/2 \log (1/2) + 1/2 \log (1/2)] = 1$ bit. Therefore, 1 bit is sufficient to store the intensity of the pixel. Therefore, binary images are less complex.

Example 3.8 Calculate the entropy for the symbols given in Table 3.6.

Table 3.6 Symbols and their probability

Symbols	1	2	3	4	5
Probability	0.4	0.3	0.1	0.1	0.1

$$\text{Entropy } H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

$$H = -(0.4\log_2 0.4 + 0.3\log_2 0.3 + 0.1\log_2 0.1 + 0.1\log_2 0.1 + 0.1\log_2 0.1)$$

$$H = -(-0.5288 - 0.5211 - 0.3322 - 0.3322 - 0.3322)$$

$$= 2.0465$$

Example 3.9 Calculate the entropy of the given image

1	2	3	5
3	4	4	6
9	9	8	7
1	3	5	6

Solution The formula for computing the entropy is

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log p_i \cdot$$

The probability of all the pixels are computed and shown in the following table.

Symbol in image	Probability of occurrence (p_i)
1	2/16
2	1/16
3	3/16
4	2/16
5	2/16
6	2/16
7	1/16
8	1/16
9	2/16

Therefore, the entropy is computed as follows:

$$\text{Entropy} = -\left\{\frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{3}{16}\log\left(\frac{3}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right)\right\}$$

state is equally likely. Hence, the probability is 1/2. Therefore, the entropy is $H = -[1/2 \log (1/2) + 1/2 \log (1/2)] = 1$ bit. Therefore, 1 bit is sufficient to store the intensity of the pixel. Therefore, binary images are less complex.

Example 3.8 Calculate the entropy for the symbols given in Table 3.6.

Table 3.6 Symbols and their probability

Symbols	1	2	3	4	5
Probability	0.4	0.3	0.1	0.1	0.1

$$\text{Entropy } H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

$$H = -(0.4\log_2 0.4 + 0.3\log_2 0.3 + 0.1\log_2 0.1 + 0.1\log_2 0.1 + 0.1\log_2 0.1)$$

$$H = -(-0.5288 - 0.5211 - 0.3322 - 0.3322 - 0.3322)$$

$$= 2.0465$$

Example 3.9 Calculate the entropy of the given image

1	2	3	5
3	4	4	6
9	9	8	7
1	3	5	6

Solution The formula for computing the entropy is

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log p_i \cdot$$

The probability of all the pixels are computed and shown in the following table.

Symbol in image	Probability of occurrence (p_i)
1	2/16
2	1/16
3	3/16
4	2/16
5	2/16
6	2/16
7	1/16
8	1/16
9	2/16

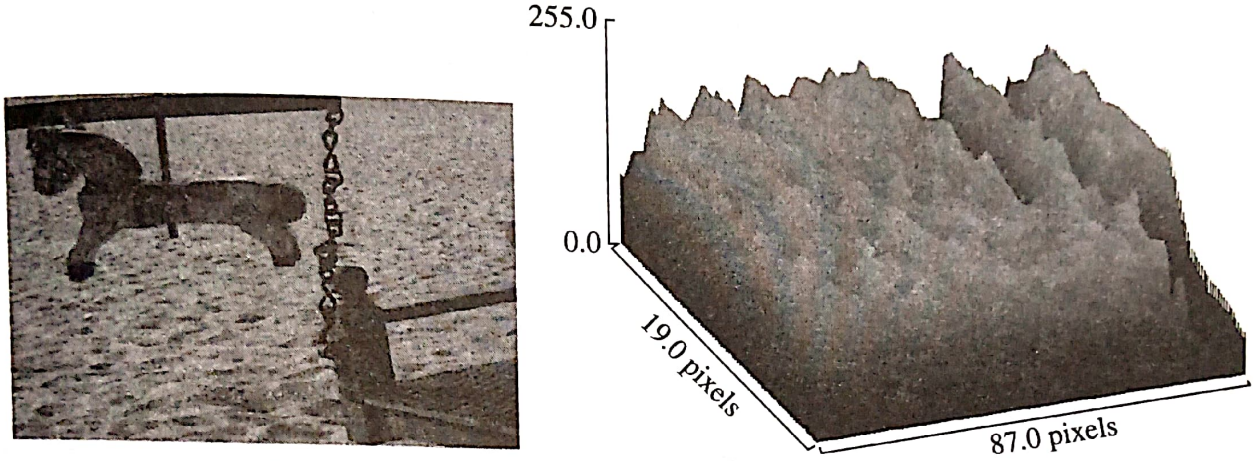
Therefore, the entropy is computed as follows:

$$\text{Entropy} = -\left\{\frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{3}{16}\log\left(\frac{3}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{1}{16}\log\left(\frac{1}{16}\right) + \frac{2}{16}\log\left(\frac{2}{16}\right)\right\}$$

$$= -\{-0.375 - 0.25 - 0.45 - 0.375 - 0.375 - 0.375 - 0.25 - 0.25 - 0.375\}$$
$$= -\{-3.075\}$$

Therefore, the entropy of the image is 3.075.

Thus, entropy indicates the information richness of the image. This can be seen visually using a surface plot where pixel values are plotted as a function of pixel position. One such plot is shown in Figs 3.27(a) and 3.27(b). Entropy can also be used to characterize the texture of the images.



(a) (b)

Fig. 3.27 Surface plot (a) Original image (b) Surface plot

Applications of statistical operations The brightness and contrast can be defined as the mean grey level and the mean absolute deviation (MAD), respectively. It is given as

$$\text{MAD} = 1/MN \times \sum_{i=1}^{M} \sum_{i=1}^{N} |(x, y) = mean|$$

Then the equation

$$g(x, y) = \left(f(x, y)/\frac{\sigma}{\sigma_n} \right) + \mu_n$$

can be used with σ_n and μ_n, which are the standard deviation and mean of the desired new image. M and N are the dimensions of the image. Generally, this operation increases the brightness and contrast.

3.2.7 Convolution and Correlation Operations

The imaging system can be modelled as a 2D linear system. Let $f(x, y)$ and $g(x, y)$ represent the input and output images, respectively. Then, they can be written as $g(x, y) = t * (f(x, y)$. Convolution is a group process, that is, unlike point operations, group processes operate on a group of input pixels to yield the result. Spatial convolution is a method of taking a group of pixels in the input image and computing the resultant output image. This is also known as a finite impulse response (FIR) filter. Spatial convolution moves across pixel by pixel and produces the output image. Each pixel of the resultant image is dependent on a group of pixels (called kernel).

The one-dimensional convolution formula is as follows:

$$g(x) = t * f(x)$$

$$= \sum_{i=-n}^{n} t(i) f(x-i)$$

The convolution window is a sliding window that centres on each pixel of the image to generate the resultant image. The resultant pixel is thus calculated by multiplying the weight of the convolution mask by pixel values and summing these values. Thus, the sliding window is moved for every corresponding pixel in the image in both directions. Therefore, the convolution is called 'shift–add–multiply' operation.

To carry out the process of convolution, the template or mask is first rotated by 180°. Then the convolution process is carried out. Consider the process of convolution of two sequences—***F***, whose dimension is 1 × 5 and a kernel or template ***T***, whose dimension is 1 × 3.

Let ***F*** = {0, 0, 2, 0, 0} and the kernel be {7 5 1}. As mentioned, the template has to be rotated by 180°. The rotated mask of this original mask [7 5 1] is a convolution template whose dimension is 1 × 3 with value {1, 5, 7}.

To carry out the convolution process, first, the process of zero padding should be carried out. Zero padding is the process of creating more zeros and is done as shown in Table 3.7. Added zeros are underlined.

Table 3.7 Zero padding process for convolution

1	**5**	**7**				**1**	**5**	**7**
<u>0</u>	<u>0</u>	0	0	2	0	0	<u>0</u>	<u>0</u>

Convolution is the process of shifting and adding the sum of the product of mask coefficients and the image to give the centre value. This process is shown in Table 3.8.

Table 3.8 Convolution process

(a) Initial position

Template

1	5	7						
0	0	0	0	2	0	0	0	0
	0							

Output is produced in the centre pixel.

(b) Position after one shift

Template is shifted by one bit.

	1	5	7					
0	0	0	0	2	0	0	0	0
	0	0						

Output produced is zero.

(c) Position after two shifts

Template is shifted again.

		1	5	7				
0	0	0	0	2	0	0	0	0
	0	0	14					

Output produced is 14.

(d) Position after three shifts

Template is shifted again.

			1	5	7			
0	0	0	0	2	0	0	0	0
	0	0	14	10				

Output produced is 10.

(e) Position after four shifts

Template is shifted again.

				1	5	7		
0	0	0	0	2	0	0	0	0
	0	0	14	10	2			

Output produced is 2.

(f) Position after five shifts

Template is shifted again.

					1	5	7	
0	0	0	0	2	0	0	0	0
	0	0	14	10	2	0		

Output produced is 0.

(g) Final position

Template is shifted again.

						1	5	7
0	0	0	0	2	0	0	0	0
	0	0	14	10	2	0	0	

Output produced is 0. Further shift crosses the range.
Hence the process is stopped.

So in the final position, the output produced is [0 0 14 10 2 0 0].

Correlation is similar to the convolution operation and is very useful in recognizing the basic shapes in the image. Correlation reduces to convolution if the kernels are symmetric. The difference between the correlation and convolution processes is that the mask or template is applied directly without any prior rotation, as in the convolution process.

The correlation of these sequences is carried out to observe the difference between these processes. The correlation process also involves the zero padding process, as shown in Table 3.9.

Table 3.9 Zero padding process for correlation

7	**5**	**1**				**7**	**5**	**1**
0	0	0	0	2	0	0	0	0

The padded zeros are underlined. The correlation process is similar to the convolution process described. This process is shown in Table 3.10.

Table 3.10 Correlation process

(a) Initial position

Template

7	5	1						
0	0	0	0	2	0	0	0	0
	0							

Output produced is 0.

(b) Position after one shift

Template is shifted by one bit.

	7	5	1					
0	0	0	0	2	0	0	0	0
	0	0						

Output produced is 0.

(c) Position after two shifts

Template is shifted again.

		7	5	1				
0	0	0	0	2	0	0	0	0
	0	0	2					

Output produced is 2.

(d) Position after three shifts

Template is shifted again.

			7	5	1			
0	0	0	0	2	0	0	0	0
	0	0	2	10				

Output produced is 10.

(e) Position after four shifts

Template is shifted again.

				7	5	1		
0	0	0	0	2	0	0	0	0
	0	0	2	10	14			

Output produced is 14.

(f) Position after five shifts

Template is shifted again.

					7	5	1	
0	0	0	0	2	0	0	0	0
	0	0	2	10	14	0		

Output produced is 0.

(g) Final position

Template is shifted again.

						7	5	1
0	0	0	0	2	0	0	0	0
	0	0	2	10	14	0	0	

Output produced is 0.

So in the final position, the output produced is [0 0 2 10 14 0 0].

The processes of convolution and correlation can be extended easily to 2D signals such as images. Kernels are usually square, with an odd number of cells. The smallest kernel is 1 × 1; it can also be 3 × 3, 5 × 5, 7 × 7, etc. As the numbers increase, the flexibility and precision increase. For a 1D signal, the convolution process is as follows:

If $T(i, j)$ is a template of dimension $n \times m$ and the image $f(i, j)$ is the input image, the convolution of f with T is written as

$$F * T = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} t(i, j) \times f(x - i, y - j)$$

Similarly, the formula for correlation can be derived. If the image is $f(x, y)$ and the template, whose size is $n \times m$, is $t(i, j)$, the correlation is given as

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (t(i, j) - f(x + i, y + j))^2$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} [(t(i, j)^2) - 2t(i, j) f(x + i, y + j) + (f(x + i, y + j)^2)]$$

Factors such as $(t(i, j)^2)$ and $(f(x + i, y + j)^2)$ are constant and hence can be ignored. This reduces the correlation to just multiplication, addition, and convolution. However, in

image processing, convolution is loosely interpreted to mean cross correlation. Correlation is written as

$$F \circ T = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} T(i, j) \times f(x+i, y+j)$$

The dimensions of the resultant image will be $(n + N - 1)$ and $(n + M - 1)$.
The convolution process on a 2D image can be shown as follows. Consider an image

$$F = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \text{ and the template or mask is } t = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

The algorithm for the convolution process is as follows. The mask is rotated by 180°. Prior to the convolution process, the operation of zero padding is carried out so that the mask or template can fully align with the image. Zero padding is done such that the template will be aligned with the leftmost corner, to check the number of zeros to be inserted. In the given example, the zeros should be inserted on the left, right, top, and bottom.

This is shown in the following steps. The mask coefficients are shown in subscripts, and the sum of the products of the mask and image coefficients is found to produce the result.

In the first step, the mask is positioned and the result is as shown in Fig. 3.28(a).

The mask coefficients are shown in subscript.

In the second step, the mask is moved horizontally by one position. The mask coefficients are shown in subscript. The result is shown in Fig. 3.28(b).

In the third step, the mask is moved again by one position and the result can be seen in Fig. 3.28(c).

Further, the mask cannot be moved in this direction as the mask range crosses the range of the image. Therefore, it will be zeros or uncertain values. So the mask is moved down and the steps are repeated and are shown in Figs 3.28(d)–3.28(f).

Again, the mask cannot be moved further. So the mask is shifted down and the process is repeated.

After the last vertical shift, the positions of the mask are as shown in Figs 3.28(g)–3.28(i).

$$\begin{matrix} \boxed{\begin{matrix} 0_1 & 0_1 \\ 0_1 & 2_1 \end{matrix}} & \begin{matrix} 0 & 0 \\ 2 & 0 \end{matrix} \\ \begin{matrix} 0 & 2 \\ 0 & 0 \end{matrix} & \begin{matrix} 2 & 0 \\ 0 & 0 \end{matrix} \end{matrix} \Rightarrow \begin{matrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 2 & \boxed{\begin{matrix} 0_1 & 0_1 \\ 2_1 & 2_1 \end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & & \\ 0 & 2 & 2 \\ 0 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(a) (b)

$$\begin{matrix} 2 & 4 & \boxed{\begin{matrix} 0_1 & 0_1 \\ 2_1 & 0_1 \end{matrix}} \\ 0 & 2 & \\ 0 & 2 & 2 \\ 0 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 2 & 4 & 2 \\ \boxed{\begin{matrix} 0_1 & 2_1 \\ 0_1 & 2_1 \end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(c) (d)

$$\begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 2_1 & 2_1 & 0 \\ 0 & 2_1 & 2_1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(e)

$$\begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 2_1 & 0_1 \\ 0 & 2 & 2_1 & 0_1 \\ 0 & 0 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(f)

$$\begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 0_1 & 2_1 & 2 & 0 \\ 0_1 & 0_1 & 0 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(g)

$$\begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 2 & 2_1 & 2_1 & 0 \\ 0 & 0_1 & 0_1 & 0 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 2 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(h)

$$\begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 2 & 4 & 2_1 & 0_1 \\ 0 & 0 & 0_1 & 0_1 \end{matrix} \Rightarrow \begin{matrix} 2 & 4 & 2 & 0 \\ 4 & 8 & 4 & 0 \\ 2 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

(i)

Fig. 3.28 Convolution process (a) Initial position (b) Position after first shift (c) Position after second shift (d) Position after first vertical shift (e) Position after third shift (f) Position after fourth shift (g) Position after second vertical shift (h) Position after fifth shift (i) Position after sixth shift

The mask cannot be moved further and thus the process stops.

The process described here is applicable to correlation also. In image processing, the calculations in the frequency domain involve only real convolution. However, in the spatial domain, the correlation process is often referred to as convolution process as the masks involved in image processing are generally symmetric. Hence the processes of convolution and correlation are technically the same and there is not much distinction between the two processes.

Some of the properties of convolution that are useful in image processing are as follows. Here f and T are input images.

1. Convolution is separable.
2. Convolution is commutative.

$$f * T = T * f$$

3. Convolution is associative.

$$(f * T) * h = f * (T * h)$$

4. Convolution follows the principle of superposition.

$$(f + g) * T = f * T + g * T$$

This chapter focused on image operations. The knowledge of these operations is important for implementing applications that involve images. Some of these operations

can be done in frequency domain also. This requires knowledge of image transforms, which are discussed in the next chapter.

SUMMARY

- A digital image is characterized by the image coordinate system and the neighbourhood concept.
- The distance between the pixels p and q in an image can be given by distance measures such as Euclidian distance, D_4 distance, and D_8 distance.
- Point operations are operations whose output value at a specific coordinate is dependent only on the input value.
- A local operation is an operation whose output value at a specific coordinate is dependent on the input values in the neighbourhood of that pixel.
- Global operations are operations whose output value at a specific coordinate is dependent on all the values in the input image.
- Some of the widely used image operations are arithmetic, logical, geometrical, statistical, and spatial operations.
- Arithmetic operations include image addition, subtraction, multiplication, and division
- Logical operations include AND, OR, NOT, and XOR,
- Geometrical operations include scaling, rotation, translation, mirror, and shear.
- Whenever a geometric transformation is performed, a resampling process should be carried out so that the desirable quality is maintained in the resultant image.
- Statistical operations include finding maximum, minimum, standard deviation, quartiles, mean, and mode.
- Image morphology operations include set operations.
- Image convolution and correlation are spatial operations that are often used to implement image filters.

KEY TERMS

Affine transform It is a transformation that includes translation, scaling, rotation, and shear that transforms an object and the resultant image preserves the straight and parallel lines of the original.

Boundary or contour A boundary is a set of pixels covering a region that has one or more neighbours outside the region.

Brightness The brightness of an image is the average pixel intensity of an image.

Connected component If the path between pixels x and y lies through pixels that are within the set, it is called a connected component.

Distance function It is a mathematical measure that specifies the distance between the given two pixels or objects.

Downsampling or subsampling This is a spatial resolution technique where the image is scaled down by half by reducing the sampling rate.

Edges Edges are intensity changes among the pixels.

Equivalence relation A binary relation that is reflexive, symmetric, and transitive is called an equivalence relation.

Global operations These are operations whose output value at a specific coordinate is dependent on all the values in the input image.

Image entropy It is the measure of the amount of orderliness that is present in an image.

Image geometry It is the study of the neighbourhood, paths among pixels, boundary, and connected components. Properties are topological as the change of shape of the image as a rubber sheet does not affect the property.

Image morphology It is a collection of set operations based on the assumption that the image is a set of coordinate points.

Isotropic scaling If the scaling factors are the same in both x-and y-directions, it is called isotropic scaling.

Linear operator An operator is called a linear operator if it obeys the rules of additivity and

homogeneity. Otherwise the operator is called a non-linear operator.

Local operations These are operations whose output value at a specific coordinate is dependent on the input values in the neighbourhood of that pixel.

Logical pixels Logical pixels specify the points of a continuous 2D function. They specify a location but occupy no physical area.

Neighbours Pixels that share edges and corners with a given pixel are called neighbours.

Physical pixels Physical pixels specify a point in the device and occupy space.

Point operations These are operations whose output value at a specific coordinate is dependent only on the input value.

Spatial convolution It is a group process that operates on a group of input pixels that involve shift and add operations to yield the result.

Upsampling or interpolation This is a process of increasing the spatial resolution of an image.

REVIEW QUESTIONS

1. Define connectivity.
2. What is the difference between 8-connectivity and *m*-connectivity?
3. Define Euclidean, D_4, and D_8 distances.
4. What are the image arithmetic operations?
5. List some of the applications of image arithmetic operations.
6. What is the need for interpolation techniques?
7. What is the significance of image entropy?
8. What is the difference between image convolution and correlation? Is there any difference between them in image processing?
9. What is the role of a data structure in imaging applications?

NUMERICAL PROBLEMS

1. Consider two pixels p and q whose coordinates are (0, 0) and (6, 3). Calculate the D_e, D_4, and D_8 distances between the pixels p and q.
2. Consider the following adjacent regions

$$\underset{R_1}{\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{matrix}} \quad \underset{R_2}{\begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}}$$

What is the connectivity between these two regions?

3. For the values of $a_1 = -1$ and $a_2 = 1$, check whether the median operation is a linear operation or not.
4. Consider the following images of uint 8 type.

$$F_1 = \begin{pmatrix} 10 & 40 & 30 \\ 40 & 100 & 90 \\ 90 & 80 & 70 \end{pmatrix}$$

$$F_2 = \begin{pmatrix} 40 & 140 & 90 \\ 140 & 100 & 90 \\ 90 & 80 & 190 \end{pmatrix}$$

Perform the image addition, multiplication, and division operations. Suppose the image is changed to type uint16, will the results change?

5. Consider the image in question 4. What is the result of image subtraction and image absolute difference? Is there any difference between them?
6. Consider the following logical images:

$$F_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$F_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Perform the logical operations AND, OR, XOR, and NAND.

Suppose the second row of the image F_1 is to be highlighted, what is the mask that is needed?

7. Derive the inverse scaling transformation matrix.
8. Show that inverse transformations restore the original object.
9. Consider the image $F = \begin{pmatrix} 2 & 1 \\ 3 & 1 \end{pmatrix}$. Apply replication and interpolation techniques and obtain the result.
10. Consider the image $F = \begin{pmatrix} 2 & 2 \\ 3 & 2 \end{pmatrix}$. Calculate the mean and entropy of the image.
11. Find the convolution and correlation of the following streams of data:
    (a) {1 7 9 6} and {1 3 5}
    (b) {1 2 3 4} and {1 1}
12. Find the convolution and correlation of the following streams of data:

    (a) $F = \begin{pmatrix} 7 & 3 & 3 \\ 2 & 2 & 2 \\ 2 & 2 & 1 \end{pmatrix}$ and $I = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

    (b) $F = \begin{pmatrix} 7 & 2 & 3 \\ 2 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix}$ and $I = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

13. A triangle is marked by the points (1, 1), (5, 5), and (10, 10).
    (a) Apply translation of $\delta x = 3$ and $\delta y = 4$.
    (b) Apply rotation with degrees 45, 90, 180, and 270.
14. Consider the image

$$F = \begin{pmatrix} 130 & 11 & 67 \\ 10 & 10 & 50 \\ 80 & 90 & 100 \end{pmatrix}$$

    What is the mean, median, mode, IQR, standard deviation, and variance of the image?
15. Consider the following symbol and its probability.

Symbols	1	2	3	4
Probability	0.4	0.3	0.2	0.1

    What is the image entropy?