

## Chapter 2

# Image Enhancement in the Spatial Domain

**Abstract** Although the transform domain processing is essential, as the images naturally occur in the spatial domain, image enhancement in the spatial domain is presented first. Point operations, histogram processing, and neighborhood operations are presented. The convolution operation, along with the Fourier analysis, is essential for any form of signal processing. Therefore, the 1-D and 2-D convolution operations are introduced. Linear and nonlinear filtering of images is described next.

An image is enhanced to increase the amount of information that can be interpreted visually. Image enhancement improves the quality of an image for a specific purpose. The process depends up on the characteristics of the image and whether it is required for human perception or machine vision. Some features are enhanced to suit human or machine vision. For example, the spot noise is reduced in median filtering so that a better viewing of the original image is obtained. Edges are enhanced by highpass filtering and the output image is a step in computer vision. In this chapter, we present three types of operations. The simplest and yet very useful image enhancement process is point operation. The output pixel is a function of the corresponding input pixels of one or more images. Thresholding is an important operation in processing images. Another type is intensity transformations to contrast enhancement, called histogram processing. Linear and nonlinear filtering is a major type of processing in which the output pixel is a function of the pixels in a small neighborhood of the input pixel. An operation is linear, if the output to a linear combination of input signals is the same linear combination of the outputs to the individual signals.

### 2.1 Point Operations

In point processing, the new value of a pixel is a function of the corresponding values of one or more images. Let  $x(m, n)$  and  $y(m, n)$  be two images of the same size. Then, pointwise arithmetic operations of corresponding pixel values of the two images are given as

$$z(m, n) = x(m, n) + y(m, n)$$

$$z(m, n) = x(m, n) - y(m, n)$$

$$z(m, n) = x(m, n) * y(m, n)$$

$$z(m, n) = x(m, n) / y(m, n)$$

One of the operands in these operations can be a constant. For example,  $z(m, n) = Cx(m, n)$  and  $z(m, n) = C + x(m, n)$ , where  $C$  is a constant. Logical operations AND (&), OR (|) and NOT (~) are also used in a similar way on binary images.

### 2.1.1 Image Complement

The complement of an image is its photographic negative obtained by subtracting the pixel values from their maximum range. In a 8-bit gray-level image, the complement,  $\tilde{x}(m, n)$ , of the image  $x(m, n)$  is given by

$$\tilde{x}(m, n) = 255 - x(m, n)$$

The new pixel value is obtained by subtracting the current value from 255. For example,

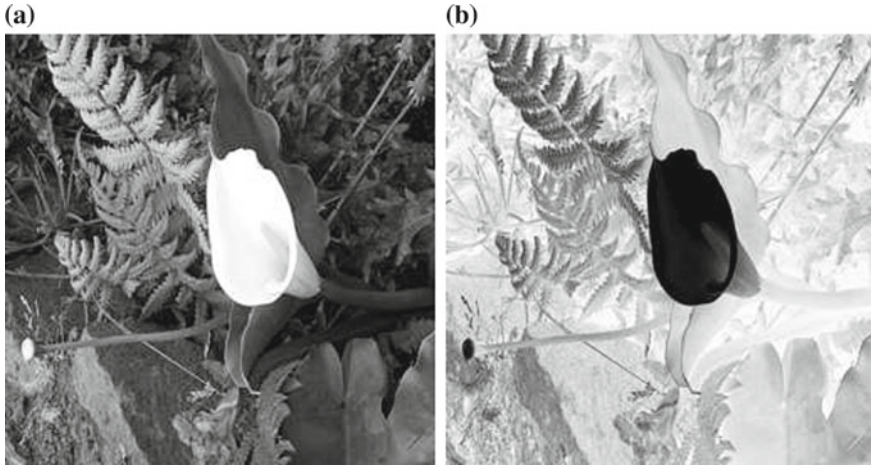
$$x(m, n) = \begin{bmatrix} 101 & 104 & 110 & 134 \\ 96 & 103 & 100 & 126 \\ 98 & 99 & 106 & 98 \\ 100 & 93 & 107 & 90 \end{bmatrix} \quad \tilde{x}(m, n) = \begin{bmatrix} 154 & 151 & 145 & 121 \\ 159 & 152 & 155 & 129 \\ 157 & 156 & 149 & 157 \\ 155 & 162 & 148 & 165 \end{bmatrix}$$

Figure 2.1a, b show, respectively, a  $256 \times 256$  8-bit gray level image and its complement. The flower in the middle is white in (a) and it has become black in (b), as expected. The dark areas have become white and vice versa. Sometimes, the complement brings out certain features better. For a binary image, the complement is given by

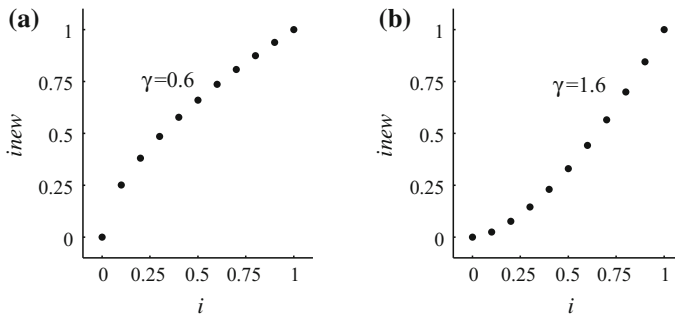
$$\tilde{x}(m, n) = 1 - x(m, n)$$

### 2.1.2 Gamma Correction

Image sensors and display devices often have nonlinear intensity characteristics. Since the nonlinearity is characterized by a power law and  $\gamma$  is the symbol used for the exponent, this operation is called gamma correction. To compensate such nonlinearity, an inverse transformation has to be applied to individual pixels of the image.



**Fig. 2.1** **a** A  $256 \times 256$  8-bit gray level image and **b** its complement



**Fig. 2.2** Intensity transformation in  $\gamma$  correction **a**  $\gamma = 0.6$ ; **b**  $\gamma = 1.6$

In gamma correction, the new intensity value  $inew$  of a pixel is its present value  $i$  raised to the power of  $\gamma$ .

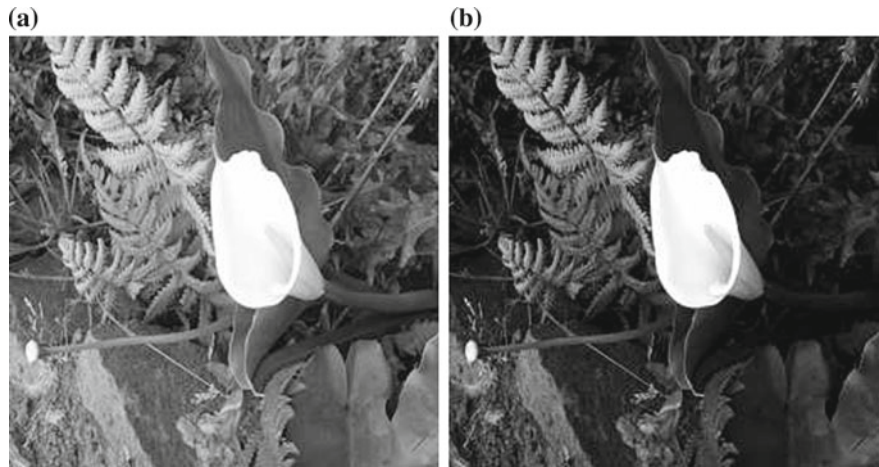
$$inew = i^\gamma \quad (2.1)$$

Let the maximum intensity value be 255. Then, all the pixel values are first divided by 255 to map the intensity values into the range 0–1. This step ensures that the pixel values stay in the range 0–255. Then, Eq. (2.1) is applied. The resulting values are multiplied by 255 and rounded to get the processed values.

Figure 2.2a, b show, respectively, the intensity mapping for values of  $\gamma = 0.6$  and  $\gamma = 1.6$ . The pixel values are also tabulated in Table 2.1. For  $\gamma < 1$ , the intensity values are scaled up and the output image gets brighter. For  $\gamma > 1$ , the intensity values are scaled down. Figure 2.3a, b, show, respectively, the versions of the image in Fig. 2.1a after gamma correction with  $\gamma = 0.8$  and  $\gamma = 1.6$ , respectively. The image is brighter in (a) and dimmer in (b). In addition to correcting nonlinearity of devices, this transformation can also be used for contrast manipulation of images.

**Table 2.1** Gamma correction

$i$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$i^{0.6}$	0	0.2512	0.3807	0.4856	0.5771	0.6598	0.7360	0.8073	0.8747	0.9387	1
$i^{1.6}$	0	0.0251	0.0761	0.1457	0.2308	0.3299	0.4416	0.5651	0.6998	0.8449	1



**Fig. 2.3** Versions of the image in Fig. 2.1 a after gamma correction with  $\gamma = 0.8$  (a) and  $\gamma = 1.6$  (b)

2.2 Histogram Processing

The histogram, which is an important entity in image processing, depicts the number of occurrences of each possible gray level in an image. Consider the  $4 \times 4$  8-bit gray level image shown in Table 2.2 (left). In order to find the histogram of the image, the histogram vector is initialized to zero. Its length is 256 since the range of gray levels is 0–255. All the pixel values of the image are scanned. Depending on the pixel value, the corresponding element in the histogram vector is incremented by 1. For example, the first pixel value is 249 and it occurs only once as indicated in the last column of the middle row of the histogram, shown in Table 2.3. The pixels with zero occurrences are not shown in the table.

**Table 2.2** Pixel values of a  $4 \times 4$  8-bit image (*left*) and its contrast-stretched version (*right*)

249	108	110	113	255	201	219	245
10	98	108	114	0	114	201	254
85	100	96	104	1	131	96	166
85	87	95	98	1	18	88	114

**Table 2.3** Histograms of the input image and its contrast-stretched version. Pixels with zero occurrences are not shown

Gray level	10	85	87	95	96	98	100	104	108	110	113	114	249
Count	1	2	1	1	1	2	1	1	2	1	1	1	1
Gray level	0	1	18	88	96	114	131	166	201	219	245	254	255

Two images can have the same histogram. By modifying the histogram suitably, the image can be enhanced. While it is a simple process to construct a histogram of an image, it is very useful in several image processing tasks such as enhancement and segmentation. It is also a feature of an image. The distribution of the gray levels of an image gives useful information. Then, the histogram is used as such or modified to suit the requirements. Large number of pixels with values at the lower end of the gray level range indicates that the image is dark. Large number of pixels with values at the upper end indicates that the image is too bright. If most of the pixels have values in the middle, then the image contrast will not be good. In all these cases, contrast stretching or histogram equalization is possible for improving the image quality. The point is that a well spread out histogram over most of the range gives a better image. Contrast stretching increases the contrast, while histogram equalization enhances the contrast. The shape of the histogram remains the same in contrast stretching and it changes in histogram equalization. As in the case of any processing, the enhancement ability of these processes varies depending on the characteristics of the histogram of the input image.

### 2.2.1 Contrast Stretching

Let the range of gray levels before and after the transformation be the same, for example 0–255. Contrast is the difference between the maximum and minimum of the gray level range of the image. A higher difference results in a better contrast. Due to the limited dynamic range of the image recording device or underexposure, the gray levels of pixels may be concentrated only at some part of the allowable range. In general, some gray levels will lie outside the range intended for stretching. Let  $i$  and  $i_{new}$  are the gray levels before and after contrast stretching. In this case, using the transformation

$$i_{new} = \left\lfloor \frac{(I_{max} - I_{min} - 2)}{(M - L)}(i - L) \right\rfloor + 1, \quad L \leq i \leq M$$

$$i_{new} = I_{min}, \quad i < L$$

$$i_{new} = I_{max}, \quad i > M$$

the contrast of the image can be enhanced, where  $I_{min}$  and  $I_{max}$  are the values of the minimum and maximum of the allowable gray level range, and  $L$  and  $M$  are the values of the minimum and maximum of the part of the gray level range to be stretched. The gray levels outside the main range are given only single values.

Consider the  $4 \times 4$  8-bit image shown in Table 2.2 (left). The histogram is shown in Table 2.3 (first 2 rows). The range of the gray levels is 0–255. With only 16 pixels in the image, most of the entries in the histogram are zero and they are not shown in the table. The point is that the histogram is concentrated in the range 85–114. Gray levels 10 and 249 are extreme values. As only a small part of the range of gray levels is used, the contrast of this type of images is poor. Contrast stretching is required to enhance the quality of the image. Now, the scale factor is computed as

$$\frac{255 - 0 - 2}{114 - 85} = 8.7241$$

For all those gray levels in the range 0–84, we assign the new gray level 0. For all those gray levels in the range 115–255, we assign the new gray level 255. For those gray levels in the range 85–114, the new value *inew* is computed from *i* as

$$inew = \lfloor 8.7241(i - 85) \rfloor + 1$$

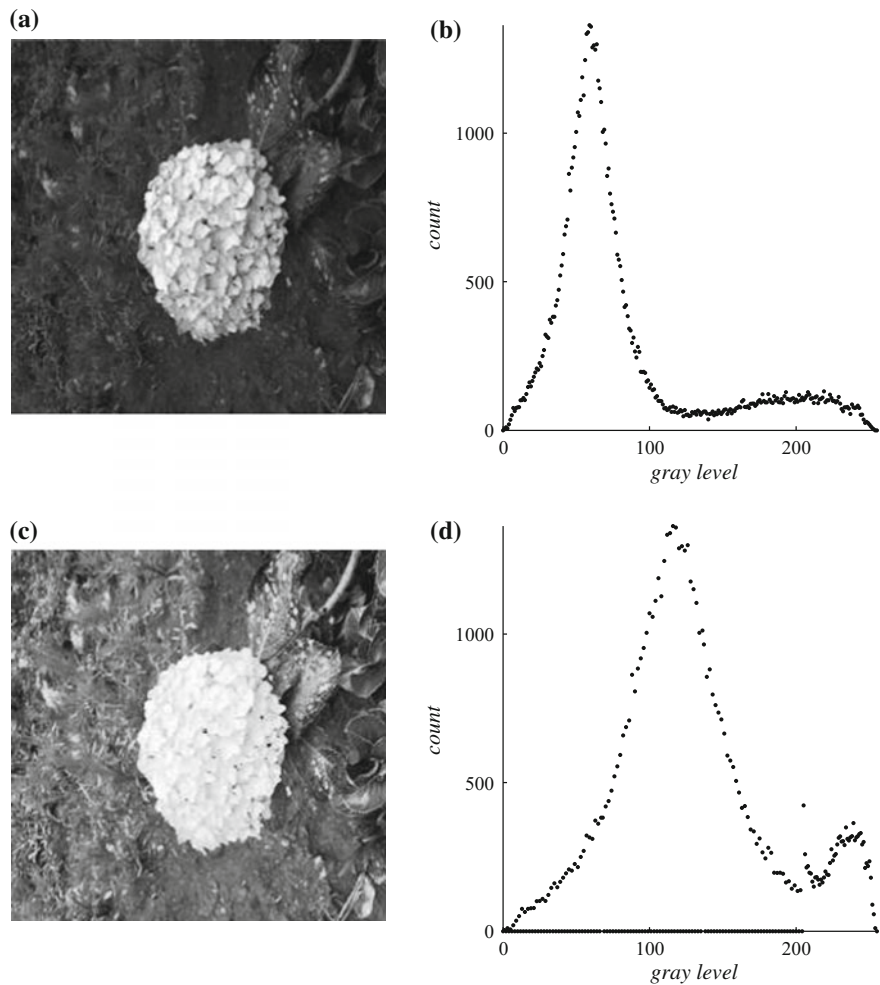
The computation involves the *floor* function which rounds the numbers to the nearest integer towards minus infinity. For example, gray level 114 is mapped to

$$inew = \lfloor 8.7241(114 - 85) \rfloor + 1 = 253 + 1 = 254$$

The contrast stretched image is shown in Table 2.2 (right). The new histogram, which is well spread out, is also shown in Table 2.3 (last 2 rows).

While we have presented the basic procedure, the algorithm can be modified to suit the specific requirements. For example, selection of the range to be stretched and the handling of the other values have to be suitably decided.

Figure 2.4a shows a  $256 \times 256$  8-bit image and (b) shows its histogram. The horizontal axis shows the gray levels and the vertical axis shows the count of the occurrence of the corresponding gray levels. The distribution of pixels is very heavy in the first half of the histogram. Therefore, the range of the histogram 0–104 is stretched and the rest compressed. The resulting image is shown in Fig. 2.4c and its histogram is shown in (d). While the dark areas got enhanced, the contrast of the brighter areas got deteriorated. Ideally, the pixels outside the range of stretching should have zero occurrences. Since it is unlikely in practical images, judgment is required to select the part to be stretched.



**Fig. 2.4** a A  $256 \times 256$  8-bit image; b its histogram; c the histogram-stretched image; d its histogram

### 2.2.2 Histogram Equalization

In both contrast stretching and histogram equalization, the objective is to spread the gray levels over the entire allowable gray level range. While stretching is a linear process and is reversible, equalization is a nonlinear process and is irreversible. Histogram equalization tries to redistribute about the same number of pixels for each gray level and it is automatic.

Consider the  $4 \times 4$  4-bit image shown in Table 2.4 (left). The gray levels are in the range 0–15. The histogram of the image is shown in Table 2.5 (second row, *count<sub>in</sub>*). It is more usually presented in a graphic form, as shown in Fig. 2.5a.

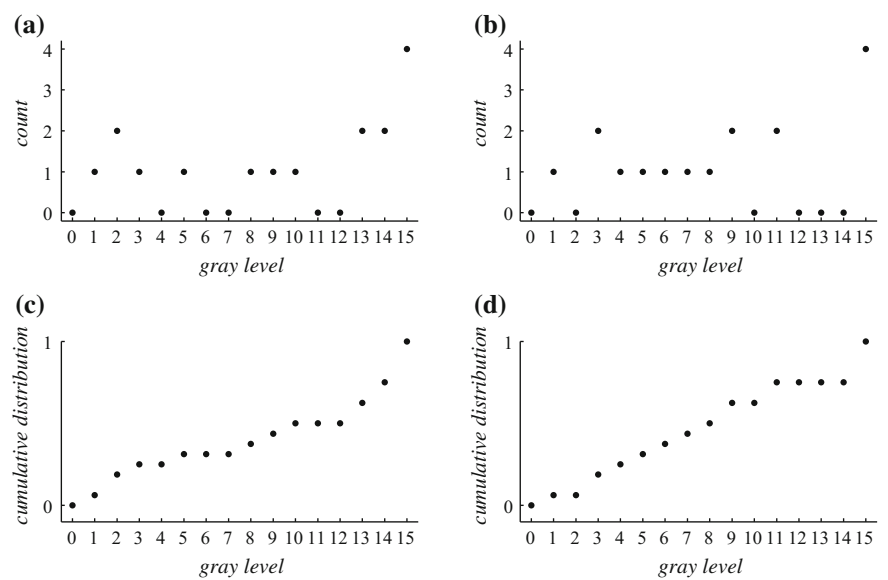
**Table 2.4** A  $4 \times 4$  4-bit image (*left*) and its histogram-equalized version (*right*)

13	14	2	14
10	2	5	9
15	15	3	15
15	8	13	1

9	11	3	11
8	3	5	7
15	15	4	15
15	6	9	1

**Table 2.5** Histogram of the image and its equalized version

Gray level	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<i>count_in</i>	0	1	2	1	0	1	0	0	1	1	1	0	0	2	2	4
<i>count_eq</i>	0	1	0	2	1	1	1	1	1	2	0	2	0	0	0	4



**Fig. 2.5** **a** The histogram of the image shown in Table 2.4 (*left*); **b** the histogram of the histogram-equalized image shown in Table 2.4 (*right*); **c** the cumulative distribution of the image; **d** the cumulative distribution of the histogram-equalized image

The sum of the number of occurrences of all the gray levels must be equal to the number of pixels in the image. The histogram is normalized by dividing the number of occurrences by the total number of pixels. The normalized histogram of the image is obtained by dividing by 16 (the number of pixels in the image) as

$$\{0, 0.0625, 0.125, 0.0625, 0, 0.0625, 0, 0, 0.0625, 0.0625, 0.0625, 0, 0, 0.125, 0.125, 0.25\}$$

This is also the probability distribution of the gray levels. Often, the histograms of images are not evenly spread over the entire intensity range. The contrast of an image can be improved by making the histogram more uniformly spread. The more the number of occurrence of a gray level, the wider the spread it gets in the



equalized histogram. For a  $N \times N$  image with  $L$  gray levels  $\{u = 0, 1, \dots, L - 1\}$ , the probability of occurrence of the  $u$ th gray level is

$$p(u) = \frac{n_u}{N^2}$$

where  $n_u$  is the number of occurrences of the pixel with gray level  $u$ . The equalization process for a gray level  $u$  of the input image is given by

$$v = (L - 1) \sum_{n=0}^u p(n), \quad u = 0, 1, \dots, L - 1$$

where  $v$  is the corresponding gray level in the histogram equalized image. The justification for the process is as follows. The cumulative histogram value, up to gray level  $u$ , in the histogram of the input image should be covered up to gray level  $v$  in the histogram after equalization.

$$\sum_{n=0}^u hist(n) = \sum_{n=0}^v hist\_eq(n)$$

Since the new histogram is to be flat, for a  $N \times N$  image with gray level values  $0 - (L - 1)$ , the number of pixels for each gray level range is

$$\frac{N^2}{L - 1}$$

The new cumulative histogram is

$$v \frac{N^2}{L - 1}$$

Since

$$\sum_{n=0}^u hist(n) = v \frac{N^2}{L - 1}, \quad v = (L - 1) \frac{\sum_{n=0}^u hist(n)}{N^2} = (L - 1) \sum_{n=0}^u p(n)$$

For the example image, the cumulative distribution of the pixel values are

$$\{0, 0.0625, 0.1875, 0.25, 0.25, 0.3125, 0.3125, 0.3125, \\ 0.375, 0.4375, 0.5, 0.5, 0.5, 0.625, 0.75, 1\}$$

obtained by computing the cumulative sum of the probability distribution computed earlier and it is shown in Fig. 2.5c. These values, multiplied by  $L - 1 = 15$ , are

{0, 0.9375, 2.8125, 3.75, 3.75, 4.6875, 4.6875, 4.6875,  
5.625, 6.5625, 7.5, 7.5, 7.5, 9.375, 11.25, 15}

The rounding of these values yields the equalized gray levels.

{0, 1, 3, 4, 4, 5, 5, 5, 6, 7, 8, 8, 8, 9, 11, 15}

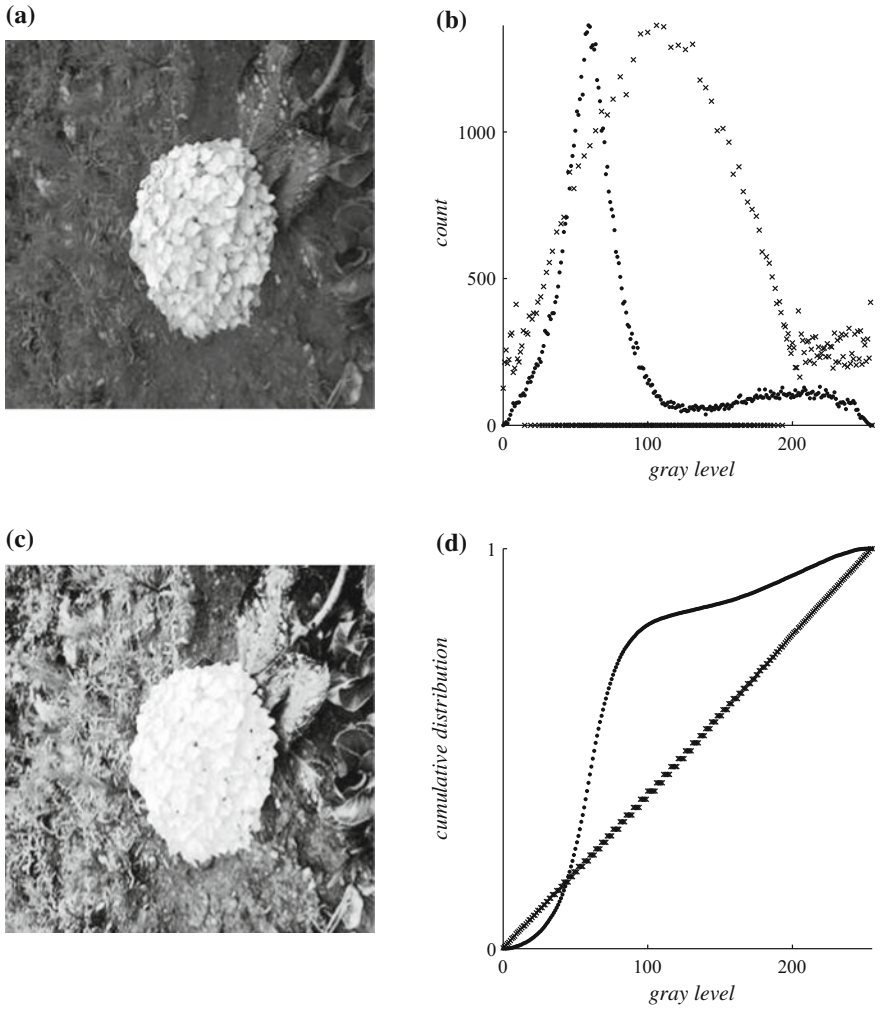
Mapping the input image, using these values, we get the histogram equalized image shown in Table 2.4 (right). The equalized histogram of the image is shown in Fig. 2.5b and in Table 2.5 (third row, *count\_eq*). The cumulative distribution of the gray levels of the image is shown in Fig. 2.5d. It is clear from Fig. 2.5c, d that the gray level values are more evenly distributed in (d). In histogram equalization, the densely populated areas of the histogram are stretched and the sparsely populated areas are compressed. Overall, the contrast of the image is enhanced. So far, we considered the distribution of the pixels over the whole image. Of course, histogram processing can also be applied to sections of the image if it suits the purpose.

Figure 2.6a shows a  $256 \times 256$  8-bit image and (b) shows the histograms of the image and its equalized version (c). Figure 2.6d shows the corresponding cumulative distributions of the gray levels. The cumulative distribution of the gray levels is a straight line for the histogram-equalized image. It is clear that equalization results in the even distribution of the gray levels. The histogram-equalized image looks better than that of the histogram-stretched image, shown in Fig. 2.4c.

As always, the effectiveness of an algorithm to do the required processing for the given data has to be checked out. Blind application of an algorithm for all data types is not recommended. For example, histogram equalization may or may not be effective for a certain image. If the number of pixels at either or both the ends of the histogram is large, equalization may not enhance the image. In these cases, an algorithm has to be modified or a new algorithm is used. The point is that the suitability of the characteristics of the image for the effective application of an algorithm is an important criterion in the selection of the algorithm.

### 2.2.3 Histogram Specification

In histogram equalization, the gray levels of the input image is redistributed in the equalized image so that its histogram approximates a uniform distribution. The distribution can be other than uniform. In certain cases where equalization algorithm is not effective, using a suitable distribution may become effective in enhancing the image. The histogram  $a(n)$  of a reference image  $A$  is specified and the histogram  $b(n)$  of the input image  $B$  is to be modified to produce an image  $C$  so that its distribution of pixels (histogram  $c(n)$ ) is as similar to that of image  $A$  as possible. This process is useful in restoring an image from its modified version, if its original histogram is known. The steps of the algorithms are:



**Fig. 2.6** **a** A  $256 \times 256$  8-bit image; **b** the histograms of the image (*dot*) and its equalized version (*cross*) (**c**); **d** the corresponding cumulative distributions of the gray levels

1. Compute the cumulative distribution,  $cum\_a(n)$ , of the reference image  $A$ .
2. Compute the cumulative distribution,  $cum\_b(k)$ , of the input image  $B$ .
3. For each value in  $cum\_b(k)$ , find the minimum value in  $cum\_a(n)$  that is greater than or equal to the current value in  $cum\_b(k)$ . That  $n$  is the new gray level in the image  $C$  corresponding to  $k$  in image  $B$ .

Consider the  $4 \times 4$  4-bit reference (left) and input (right) images shown in Table 2.6. The histogram of the reference and input images, respectively, are

**Table 2.6**  $4 \times 4$  4-bit reference (*left*) and input (*right*) images

8	8	8	8
8	8	8	8
8	8	8	8
8	8	8	8

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$$\{0, 0, 0, 0, 0, 0, 0, 0, 16, 0, 0, 0, 0, 0, 0, 0\} \quad \text{and} \\ \{16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

The cumulative distribution,  $cum\_a(n)$ , of the reference image and the cumulative distribution,  $cum\_b(k)$ , of the input image, respectively, are

$$\{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1\} \quad \text{and} \\ \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$$

All the values in  $cum\_b(k)$  map to  $cum\_a(8)$  and all the pixels in the input image map to 8 in the output image. That is, the histograms of the reference and output images are the same. Let us interchange the reference and input images. Then, all the values in  $cum\_b(k)$  map to  $cum\_a(0)$  and all the pixels in the input image map to 0 in the output image.

As this problem is a generalization of the histogram equalization problem, let us do that example again following the 3 steps given above. In histogram equalization, the reference cumulative distribution values are those of the uniform probability distribution. Therefore, the values of  $cum\_a(n)$  are

$$\{0, 0.0667, 0.1333, 0.2, 0.2667, 0.3333, 0.4, 0.4667, 0.5333, \\ 0.6, 0.6667, 0.7333, 0.8, 0.8667, 0.9333, 1\}$$

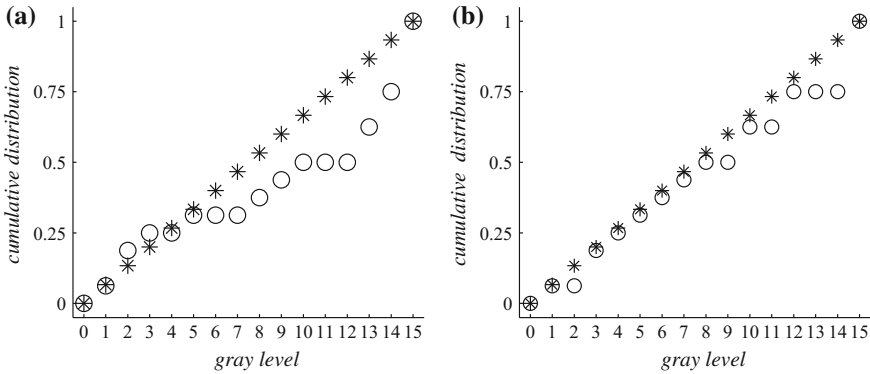
From the equalization example, the values of  $cum\_b(k)$  are

$$\{0, 0.0625, 0.1875, 0.25, 0.25, 0.3125, 0.3125, 0.3125, \\ 0.375, 0.4375, 0.5, 0.5, 0.5, 0.625, 0.75, 1\}$$

The first value in  $cum\_b(k)$  is zero. The minimum value greater than or equal to it in  $cum\_a(n)$  is 0 and gray level value 0 maps to 0. Carrying out this process for all the values in  $cum\_b(k)$ , we get the equalized gray levels.

$$\{0, 1, 3, 4, 4, 5, 5, 5, 6, 7, 8, 8, 8, 10, 12, 15\}$$

These are about the same values obtained by equalization algorithm. Using these values the output image is created. Figure 2.7a shows the cumulative distributions of



**Fig. 2.7** **a** The cumulative distributions of the reference (\*) and input images (o); **b** The cumulative distributions of the reference (\*) and output images (o)

**Table 2.7**  $4 \times 4$  reference, input and output images, respectively, from *left*

13	14	2	14
10	2	5	9
15	15	3	15
15	8	13	1

11	13	0	13
7	0	2	5
15	15	1	15
15	4	11	0

13	14	2	14
10	2	5	9
15	15	3	15
15	8	13	2

the reference (\*) and input images (o). Figure 2.7b shows the cumulative distributions of the reference (\*) and output images (o). The cumulative distribution of the output image is close to that of the uniform distribution.

Example images *A*, *B* and *C* are shown in Table 2.7. The normalized histogram of the reference image is

$$\{0, 0.0625, 0.1250, 0.0625, 0, 0.0625, 0, 0, 0.0625, 0.0625, 0.0625, 0, 0, 0.1250, 0.1250, 0.25\}$$

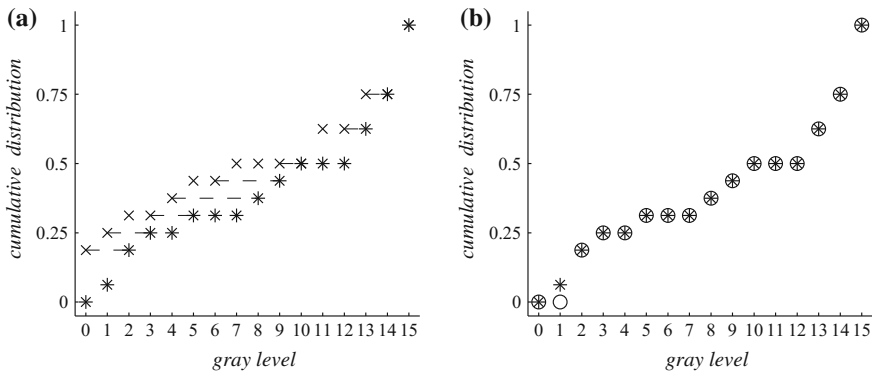
The normalized histogram of the input image is

$$\{0.1875, 0.0625, 0.0625, 0, 0.0625, 0.0625, 0, 0.0625, 0, 0, 0, 0.1250, 0, 0.1250, 0, 0.25\}$$

The cumulative distribution,  $cum\_a(n)$ , of the reference image is

$$\{0, 0.0625, 0.1875, 0.25, 0.25, 0.3125, 0.3125, 0.3125, 0.3750, 0.4375, 0.5, 0.5, 0.5, 0.6250, 0.75, 1\}$$

The cumulative distribution,  $cum\_b(k)$ , of the input image is



**Fig. 2.8** **a** The cumulative distributions of the input ( $\times$ ) and reference ( $*$ ) images; **b** the cumulative distributions of the output ( $o$ ) and reference ( $*$ ) images

$$\{0.1875, 0.25, 0.3125, 0.3125, 0.3750, 0.4375, 0.4375, \\ 0.5, 0.5, 0.5, 0.5, 0.6250, 0.6250, 0.75, 0.75, 1\}$$

The cumulative distributions of the reference and input images are shown in Fig. 2.8a. Each value in  $cum\_b(k)$  has to be mapped to the minimum value of  $cum\_a(n)$  that is greater than or equal to  $cum\_b(k)$ . For example, the first value of  $cum\_b(k)$  is 0.1875. The corresponding value is  $cum\_a(2)$ . That is, gray level 0 is mapped to 2 in the output image. Gray level with value 1 is mapped to 3 and so on. In Fig. 2.8a, the mappings are shown by dashed lines. Pixels of the input image in the range 0–15 are mapped to

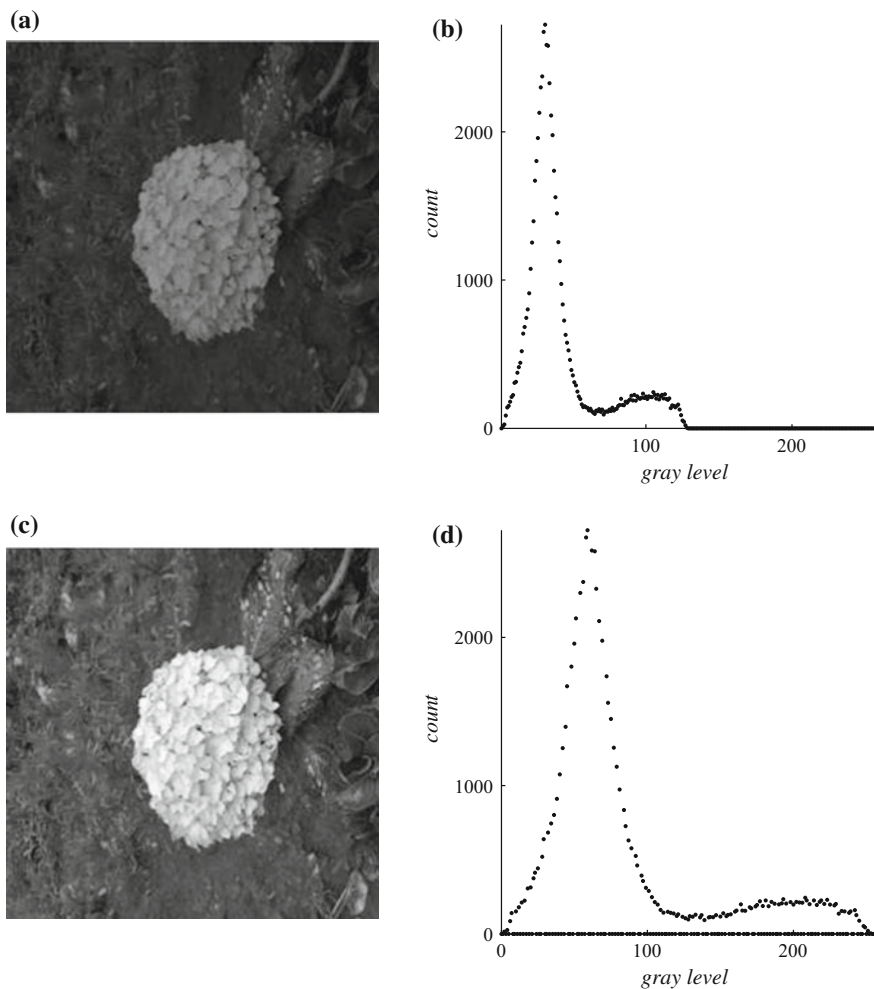
$$\{2, 3, 5, 5, 8, 9, 9, 10, 10, 10, 10, 13, 13, 14, 14, 15\}$$

in the output image. Using these mappings, the output image is reconstructed (the rightmost in Table 2.7). The cumulative distribution of the output image is

$$\{0, 0, 0.1875, 0.25, 0.25, 0.3125, 0.3125, 0.3125, 0.3750, 0.4375, \\ 0.5, 0.5, 0.5, 0.6250, 0.75, 1\}$$

The cumulative distributions of the reference and output images are almost the same, as shown in Fig. 2.8b.

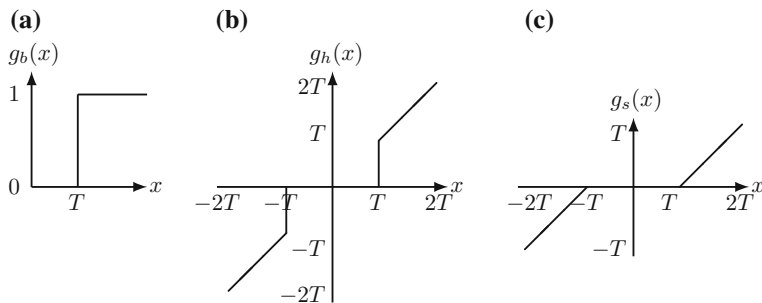
Figure 2.9a, b show, respectively, a  $256 \times 256$  8-bit image and its histogram. Figure 2.9c, d show, respectively, the restored image using histogram specification algorithm and its histogram.



**Fig. 2.9** **a** A  $256 \times 256$  8-bit image and **b** its histogram; **c** the restored image using histogram specification algorithm and **d** its histogram

## 2.3 Thresholding

Thresholding operation is frequently used in image processing. It is used in tasks such as enhancement, segmentation and compression. A threshold indicates an intensity level of some significance. There are several variations of thresholding used in image processing. The first type is to threshold a gray level image to get a binary image. A threshold  $T > 0$  is specified and all the gray levels with magnitude less than or equal to  $T$  are set to zero and the rest are set to 1.



**Fig. 2.10** **a** Binary thresholding; **b** Hard thresholding; **c** Soft thresholding

$$g_b(x) = \begin{cases} 0 & \text{if } x \leq T \\ 1, & \text{otherwise} \end{cases}$$

This type of thresholding is shown in Fig. 2.10a.

In another type of thresholding, all the gray levels with magnitude less than or equal to  $T$  are set to zero and the rest are unaltered or set to the difference between the input values and the threshold. Hard thresholding, shown in Fig. 2.10b, is defined as

$$g_h(x) = \begin{cases} 0 & \text{if } |x| \leq T \\ x, & \text{if } |x| > T \end{cases}$$

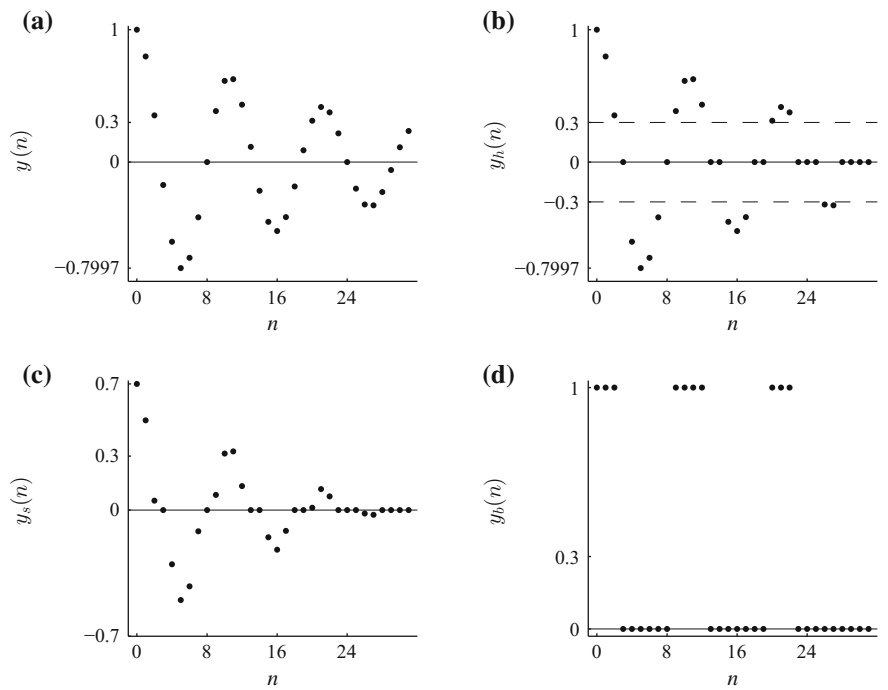
In hard thresholding, the value of the function is retained, if its magnitude is greater than a chosen threshold value. Otherwise, the value of the function is set to zero. Typical application of this type of thresholding is in lossy image compression. A higher threshold gets a higher compression ratio at the cost of image quality. Soft thresholding, shown in Fig. 2.10c, is defined as

$$g_s(x) = \begin{cases} 0, & \text{if } |x| \leq T \\ x - T, & \text{if } x > T \\ x + T, & \text{if } x < -T \end{cases}$$

The difference in soft thresholding is that the value of the function is made closer to zero by adding or subtracting the chosen threshold value from it, if its magnitude is greater than the threshold. A typical application of soft thresholding is in denoising. Thresholding is easily extended to multiple levels.

Figure 2.11a shows a damped sinusoid. Figure 2.11b shows the damped sinusoid hard thresholded with level  $T = 0.3$ . Values less than or equal to 0.3 have been assigned the value zero. Figure 2.11c shows the damped sinusoid soft thresholded with level  $T = 0.3$ . Values less than or equal to 0.3 have been assigned the value zero and values greater than 0.3 have been assigned values closer to zero by 0.3. Figure 2.11d shows the damped sinusoid binary thresholded with level  $T = 0.3$ .





**Fig. 2.11** **a** A damped sinusoid; **b** hard, **c** soft and **d** binary thresholding of the sinusoid with  $T = 0.3$

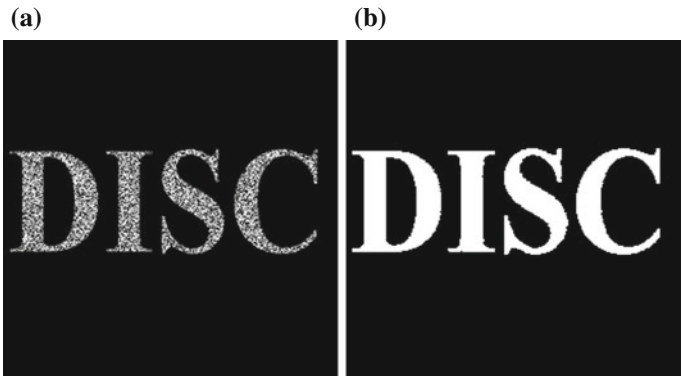
Values less than or equal to 0.3 have been assigned the value zero and values greater than 0.3 have been assigned the value 1.

Consider the  $8 \times 8$  8-bit gray level image shown by the left matrix.

117	170	130	54	84	209	164	148
135	151	137	96	56	157	225	189
136	152	174	146	64	84	146	90
123	139	182	133	51	71	56	74
119	137	172	146	119	67	65	70
90	123	166	184	203	101	49	64
85	102	162	194	164	80	38	56
73	84	155	185	147	163	87	57

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

The result of binary thresholding with  $T = 120$  is shown in the right matrix. The results of hard and soft thresholding with  $T = 120$  are shown in the left and right matrices, respectively.



**Fig. 2.12** **a** A  $256 \times 256$  image and **b** its threshold version with  $T = 1$

0	170	130	0	0	209	164	148		0	50	10	0	0	89	44	28
135	151	137	0	0	157	225	189		15	31	17	0	0	37	105	69
136	152	174	146	0	0	146	0		16	32	54	26	0	0	26	0
123	139	182	133	0	0	0	0		3	19	62	13	0	0	0	0
0	137	172	146	0	0	0	0		0	17	52	26	0	0	0	0
0	123	166	184	203	0	0	0		0	3	46	64	83	0	0	0
0	0	162	194	164	0	0	0		0	0	42	74	44	0	0	0
0	0	155	185	147	163	0	0		0	0	35	65	27	43	0	0

Figure 2.12a shows a  $256 \times 256$  image. The image is corrupted with noise and the letters are not clear. The white pixels showing the letters have values varying from 2 to 255. Therefore, with the threshold  $T = 1$ , setting all the pixels greater than 1 to 255 with the rest set to 0 enhances the image, as shown in Fig. 2.12b.

## 2.4 Neighborhood Operations

In this type of processing, called neighborhood operation, each pixel value is replaced by another, which is a linear or nonlinear function of the values of the pixels in its neighborhood. The area of a square or rectangle or circle (sometimes of other shapes) forming the neighborhood is called a window. Typical window sizes vary from  $3 \times 3$  to  $11 \times 11$ . If the window size is  $1 \times 1$  (the neighborhood consists of the pixel itself), then the operation is called the point operation. The window is moved over the image row by row and column by column and the same operation is carried out for each pixel.

A  $3 \times 3$  window of the pixel  $x(m, n)$  is

$$\begin{bmatrix} x(m-1, n-1) & x(m-1, n) & x(m-1, n+1) \\ x(m, n-1) & x(m, n) & x(m, n+1) \\ x(m+1, n-1) & x(m+1, n) & x(m+1, n+1) \end{bmatrix}$$

The set of pixels (strong neighbors)

$$\{x(m-1, n), x(m, n+1), x(m+1, n), x(m, n-1)\}$$

is called the 4-neighbors of  $x(m, n)$ .

$$\begin{bmatrix} & x(m-1, n) & \\ x(m, n-1) & x(m, n) & x(m, n+1) \\ & x(m+1, n) & \end{bmatrix}$$

The distance between these pixels and  $x(m, n)$  is 1. The other 4 pixels (weak neighbors) are diagonal neighbors of  $x(m, n)$ . All the neighbors in the window are called the 8-neighbors of  $x(m, n)$ .

### Border Extension

If the complete window is to overlap the image pixels, then the output image after a neighborhood operation will be smaller. This is due to the fact that the required pixels are not defined at the borders. Then, we have to accept a smaller output image or extend the input image at the borders suitably. For example, many operations are based on convolving an image with the impulse response or coefficient matrix. When trying to find the convolution output corresponding to the pixels located in the vicinity of the borders, some of the required pixels are not available. Obviously, we can assume that the values are zero. This method of border extension is called zero-padding. Of course, when this method is not suitable, there are other possibilities. Consider the  $4 \times 4$  image

23	51	23	32
32	44	44	23
23	23	44	32
44	44	23	23

Some of the commonly used image extensions are given below. Any other suitable extension can also be used.

The symmetric extension of the image by 2 rows and 2 columns on all sides yields

44	32	32	44	44	23	23	44
51	23	23	51	23	32	32	23
51	23	23	51	23	32	32	23
44	32	32	44	44	23	23	44
23	23	23	23	44	32	32	44
44	44	44	44	23	23	23	23
44	44	44	44	23	23	23	23
23	23	23	23	44	32	32	44

The extension is the mirror image of itself at the borders.

The replication method of extension of the image by 2 rows and 2 columns on all sides yields

23	23	23	51	23	32	32	32
23	23	23	51	23	32	32	32
23	23	23	51	23	32	32	32
32	32	32	44	44	23	23	23
23	23	23	23	44	32	32	32
44	44	44	44	23	23	23	23
44	44	44	44	23	23	23	23
44	44	44	44	23	23	23	23

Border values are repeated.

The periodic extension of the image by 2 rows and 2 columns on all sides yields

44	32	23	23	44	32	23	23
23	23	44	44	23	23	44	44
23	32	23	51	23	32	23	51
44	23	32	44	44	23	32	44
44	32	23	23	44	32	23	23
23	23	44	44	23	23	44	44
23	32	23	51	23	32	23	51
44	23	32	44	44	23	32	44

This extension considers the image as one period of a 2-D periodic signal. The top and bottom edges are considered adjacent and so are the right and left edges.

### 2.4.1 Linear Filtering

A filter, in general, is a device that passes the desirable part of its input. In the context of image processing, a filter modifies the spectrum of an image in a specified manner. This modification can be done either in the spatial domain or frequency domain.

The choice primarily depends of the size of the filter among other considerations. A linear filter is characterized by its impulse response, which is its response for a unit-impulse input with zero initial conditions. For enhancement purposes, a filter is used to improve the quality of an image for human or machine perception. The improvement in the quality of an image is evaluated subjectively. Two types of filters, lowpass and highpass, are often used to improve the quality. A lowpass filter is essentially an integrator, passing the low frequency components and suppressing the high frequency components. For example, the integral of  $\cos(\omega t)$  is  $\sin(\omega t)/\omega$ . The higher the frequency, the higher is the attenuation of the frequency component after integration. A highpass filter is essentially a differentiator that suppresses the low frequency components. The derivative of  $\sin(\omega t)$  is  $\omega \cos(\omega t)$ . The higher the frequency, the higher is the amplification of the frequency component after differentiation.

In linear filtering, convolution operation is a convenient system model. It relates the input and output of a system through its impulse response. Although the image is a 2-D signal, its processing can often be carried out using the corresponding 1-D operations repeatedly over the rows and columns. Conceptually, 1-D operations are easier to understand. Further, 2-D convolution is a straightforward extension of that of the 1-D. Therefore, we present the 1-D convolution briefly. First, as it is so important (along with Fourier analysis), we present a simple example to explain the concept.

Consider the problem of finding the amount in our bank account for the deposits on a yearly basis. We are familiar that, for compound interest, the amount of interest paid increases from year to year. Let the annual interest rate be 10%. Then, an amount of \$1 will be \$1 at the time of deposit, \$1.1 after 1 year, \$1.21 after 2 years and so on, as shown in Fig. 2.13a. Let our current deposit be \$200, \$300 a year before and \$100 two years before, as shown in Fig. 2.13b. The problem is to find the current balance in the account. From Fig. 2.13a, b, it is obvious that if we reverse the order of numbers in (a), shift and multiply with the corresponding numbers in (b) and sum the products, we get the current balance \$651, as shown in Fig. 2.13c.

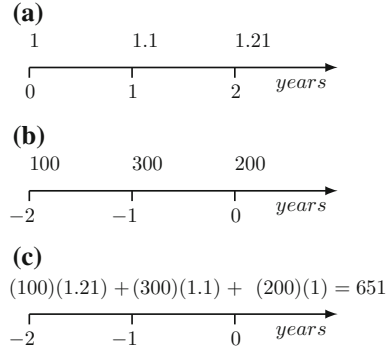
Of course, we could have reversed the order of the numbers in (b) either. For longer sets of numbers, we can repeat the operation. This is convolution operation and it is simple. It is basically a sum of products of two sequences, after either one (not both) is time-reversed. In formal description, the set of interest rates is called as the system impulse response. The set of deposits is called the input to the system. The set of balances at different time periods is called the system output. Convolution relates the input and the impulse response of a system to its output.

### 1-D Linear Convolution

The 1-D linear convolution of two aperiodic sequences  $x(n)$  and  $h(n)$  is defined as

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = x(n) * h(n) = h(n) * x(n)$$

**Fig. 2.13** Basics of linear convolution. **a** annual interest rate; **b** deposits; **c** computation of current balance



The convolution operation relates the input  $x(n)$ , the output  $y(n)$  and the impulse response  $h(n)$  of a system. The impulse response, which characterizes the system in the time-domain, is the response of a relaxed (initial conditions are zero) system for the unit-impulse  $\delta(n)$ . A discrete unit-impulse signal is defined as

$$\delta(n) = \begin{cases} 1, & \text{for } n = 0 \\ 0, & \text{for } n \neq 0 \end{cases}$$

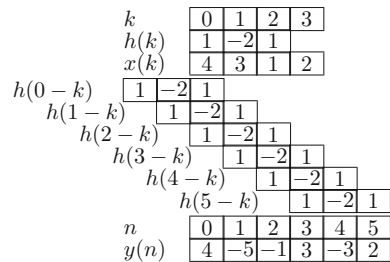
It is an all-zero sequence, except that its value is one when its argument  $n$  is equal to zero. The input  $x(n)$  is decomposed into a sum of scaled and delayed unit-impulses. The response to each impulse is found and the superposition summation of all the responses is the system output. It can also be considered as the weighted average of sections of the input with the weighting sequence being the impulse response.

Figure 2.14 shows the convolution of the signal  $\{x(0) = 4, x(1) = 3, x(2) = 1, x(3) = 2\}$  and  $\{h(0) = 1, h(1) = -2, h(2) = 1\}$ . The output  $y(0)$ , from the definition, is

$$y(0) = x(k)h(0 - k) = (4)(1) = 4,$$

where  $h(0 - k)$  is the time-reversal of  $h(k)$ . Shifting  $h(0 - k)$  to the right, we get the remaining outputs as

**Fig. 2.14** 1-D linear convolution



$$\begin{aligned}
y(1) &= x(k)h(1-k) = (4)(-2) + (3)(1) = -5 \\
y(2) &= x(k)h(2-k) = (4)(1) + (3)(-2) + (1)(1) = -1 \\
y(3) &= x(k)h(3-k) = (3)(1) + (1)(-2) + (2)(1) = 3 \\
y(4) &= x(k)h(4-k) = (1)(1) + (2)(-2) = -3 \\
y(5) &= x(k)h(5-k) = (2)(1) = 2
\end{aligned}$$

Outside the defined values of  $x(n)$ , we have assumed zero values. As mentioned earlier, a suitable extension of the input, to get a convolution output of the same length, should be made to suit the requirements of the problem. The six convolution output values are called the full convolution output. Most of the times, the central part of the output, of the same size as the input, is required. If the window is to be confined inside the input data, the size of the output will be smaller than that of the input.

## 2-D Linear Convolution

In the 2-D convolution, a 2-D window is moved over the image. The convolution of images  $x(m, n)$  and  $h(m, n)$  is defined as

$$\begin{aligned}
y(m, n) &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k, l)h(m-k, n-l) \\
&= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k, l)x(m-k, n-l) = h(m, n) * x(m, n)
\end{aligned}$$

Four operations, similar to those of the 1-D convolution, are repeatedly executed in carrying out the 2-D convolution.

1. One of the images, say  $h(k, l)$ , is rotated in the  $(k, l)$  plane by  $180^\circ$  about the origin to get  $h(-k, -l)$ . The same effect is achieved by folding the image about the  $k$  axis to get  $h(k, -l)$  and then, folding the resulting image about the  $l$  axis.
2. The rotated image is shifted by  $(m, n)$  to get  $h(m-k, n-l)$ .
3. The products  $x(k, l)h(m-k, n-l)$  of all the overlapping samples are found.
4. The sum of all the products yields the convolution output  $y(m, n)$  at  $(m, n)$ .

Consider the convolution of the  $3 \times 3$  image  $h(k, l)$  and the  $4 \times 4$  image  $x(k, l)$

$$h(k, l) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{and} \quad x(k, l) = \begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 1 & 2 & 4 \\ 1 & -1 & 2 & -2 \\ 3 & 1 & 2 & 2 \end{bmatrix}$$

shown in Fig. 2.15. Four examples of computing the convolution output are shown. For example, with a shift of  $(0-k, 0-l)$ , there is only one overlapping pair  $(1, -1)$ . The product of these numbers is the output  $y(0, 0) = -1$ . The process is repeated to

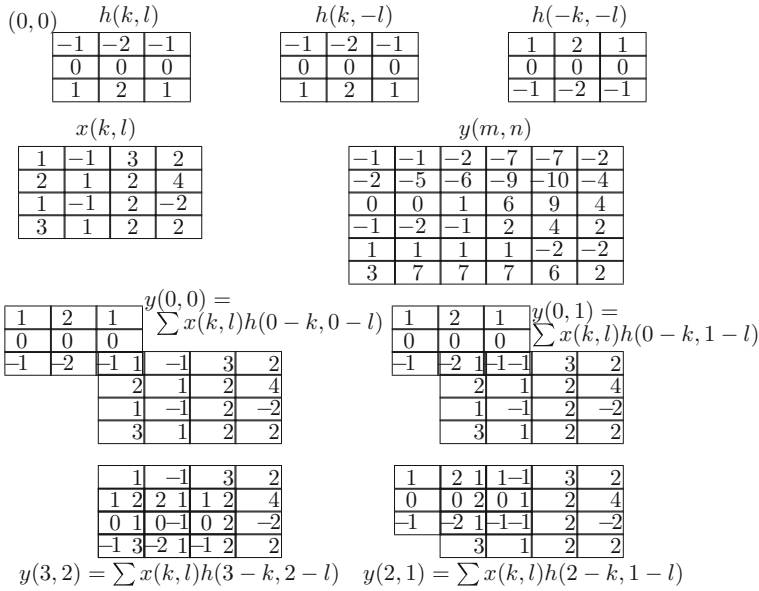


Fig. 2.15 2-D linear convolution

get the complete convolution output  $y(m,n)$  shown in the figure. We assumed that the pixel values outside the defined region of the image are zero. This assumption may or may not be suitable. Some other commonly used border extensions are based on periodicity, symmetry or replication, as presented earlier.

### Lowpass Filtering

The output of convolution for a given input depends on the impulse response of the system. In lowpass filtering, the frequency response corresponding to the impulse response will be of lowpass nature. The system readily passes the low frequency components of the signal and suppresses the high frequency components. Low frequency components vary slowly compared with the bumpy nature of the high frequency components. Lowpass filtering is typically used for deliberate blurring to remove unwanted details of an image and reduce the noise content of the image. The impulse response of the simplest and widely used  $3 \times 3$  lowpass filter, called the averaging filter, is

$$h(m,n) = \frac{1}{9} \begin{bmatrix} 1 & \mathbf{1} & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad m = -1, 0, 1, \quad n = -1, 0, 1$$

The origin of the filter is shown in boldface. All the coefficient values are the same. Other filters produce weighted average outputs. This filter, when applied to an image, replaces each pixel in the input by the average of the values of a set of its neighboring pixels. Pixel  $x(m,n)$  is replaced by the value



$$y(m, n) = \frac{1}{9}(x(m-1, n-1) + x(m-1, n) + x(m-1, n+1) + x(m, n-1) + x(m, n) + x(m, n+1) + x(m+1, n-1) + x(m+1, n) + x(m+1, n+1))$$

The bumps are smoothed out due to averaging. Blurring will proportionally increase with larger filters. This filter is separable. Multiplying the  $3 \times 1$  column filter  $h_c(m) = \{1, 1, 1\}^T/3$  with the  $1 \times 3$  row filter  $h_r(n) = \{1, 1, 1\}/3$ , which is the transpose of the column filter, we obtain the  $3 \times 3$  averaging filter.

$$h(m, n) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = h_c(m)h_r(n)$$

This implies that the computational complexity is reduced by convolving each row of the input image with the row filter first and then convolving each column of the result with the column filter or vice versa. With the 2-D filter  $h(m, n)$  separable,  $h(m, n) = h_c(m)h_r(n)$  and, with input  $x(m, n)$ ,

$$\begin{aligned} h(m, n) * x(m, n) &= (h_c(m)h_r(n)) * x(m, n) \\ &= (h_c(m) * x(m, n)) * h_r(n) = h_c(m) * (x(m, n) * h_r(n)) \end{aligned}$$

$$y(k, l) = \sum_m h_c(m) \sum_n h_r(n) x(k-m, l-n) = \sum_n h_r(n) \sum_m h_c(m) x(k-m, l-n)$$

Whenever a filter is separable, it is advantageous to decompose a 2-D operation into a pair of 1-D operations.

Let the input be

$$x(m, n) = \begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 1 & 2 & 4 \\ 1 & -1 & 2 & -2 \\ 3 & 1 & 2 & 2 \end{bmatrix}$$

Assuming zero-padding at the borders, the output of 1-D filtering of the rows of the input and the output of 1-D filtering of the columns of the partial output are, respectively,

$$y_r(m, n) = \frac{1}{3} \begin{bmatrix} 0 & 3 & 4 & 5 \\ 3 & 5 & 7 & 6 \\ 0 & 2 & -1 & 0 \\ 4 & 6 & 5 & 4 \end{bmatrix} \quad y(m, n) = \frac{1}{9} \begin{bmatrix} 3 & 8 & 11 & 11 \\ 3 & 10 & 10 & 11 \\ 7 & 13 & 11 & 10 \\ 4 & 8 & 4 & 4 \end{bmatrix}$$

Assuming replication at the borders, the extended input and the output are, respectively,

$$xe(m, n) = \begin{bmatrix} 1 & 1 & -1 & 3 & 2 & 2 \\ 1 & 1 & -1 & 3 & 2 & 2 \\ 2 & 2 & 1 & 2 & 4 & 4 \\ 1 & 1 & -1 & 2 & -2 & -2 \\ 3 & 3 & 1 & 2 & 2 & 2 \\ 3 & 3 & 1 & 2 & 2 & 2 \end{bmatrix} \quad y(m, n) = \frac{1}{9} \begin{bmatrix} 7 & 11 & 15 & 24 \\ 7 & 10 & 10 & 15 \\ 13 & 13 & 11 & 14 \\ 15 & 14 & 9 & 10 \end{bmatrix}$$

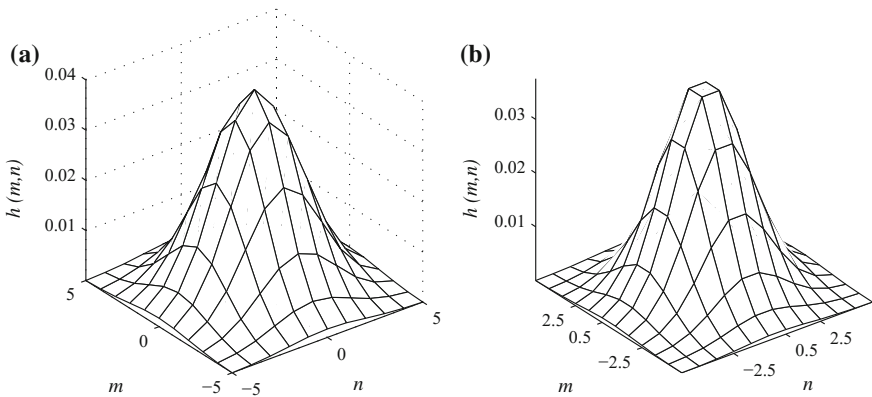
Only the output at the borders differ with different border extensions. The central part of the output is the same.

### Gaussian Lowpass Filter

The 2-D Gaussian function is a lowpass filter, with a bell-shaped impulse response (frequency response) in the spatial domain (frequency domain). The Gaussian lowpass filters are based on Gaussian probability distribution function. The impulse response  $h(m, n)$  of the Gaussian  $N \times N$  lowpass filter, with the standard deviation  $\sigma$ , is given by

$$h(m, n) = \frac{e^{-\frac{(m^2+n^2)}{(2\sigma^2)}}}{K}, \quad K = \sum_{m=-(N-1)/2}^{(N-1)/2} \sum_{n=-(N-1)/2}^{(N-1)/2} e^{-\frac{(m^2+n^2)}{(2\sigma^2)}}$$

assuming  $N$  is odd. The larger the value of the standard deviation  $\sigma$ , the flatter is the filter impulse response. For very large value of  $\sigma$ , as it appears squared in the denominator of the exponent of the exponential function of the defining equation, it tends to the averaging filter in the limit. The impulse response of the Gaussian lowpass filters with  $\sigma = 2$ , of size  $11 \times 11$  and  $12 \times 12$ , are shown in Fig. 2.16a, b, respectively. The impulse response of the Gaussian  $3 \times 3$  lowpass filter, with  $\sigma = 0.5$ , is



**Fig. 2.16** The impulse response of the Gaussian lowpass filters with  $\sigma = 2$ . **a**  $11 \times 11$ ; **b**  $12 \times 12$

$$h(m, n) = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & \mathbf{0.6193} & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}, \quad m = -1, 0, 1, \quad n = -1, 0, 1$$

The origin of the filter is shown in boldface. For example, let  $m = n = 0$  in the defining equation for  $h(m, n)$ . Then, the numerator is 1.

$$\begin{aligned} K &= (e^{-2(1+1)} + e^{-2(0+1)} + e^{-2(1+1)} + e^{-2(1+0)} + e^{-2(0+0)} \\ &\quad + e^{-2(1+0)} + e^{-2(1+1)} + e^{-2(0+1)} + e^{-2(1+1)}) \\ &= e^{-4} + e^{-2} + e^{-4} + e^{-2} + 1 + e^{-2} + e^{-4} + e^{-2} + e^{-4} \\ &= 4e^{-4} + 4e^{-2} + 1 = 1.6146 \end{aligned}$$

The inverse of 1.6146 is  $0.6193 = h(0, 0)$ . This filter is also separable. Multiplying the  $3 \times 1$  column filter  $\{0.1065, 0.7870, 0.1065\}^T$  with the  $1 \times 3$  row filter  $\{0.1065, 0.7870, 0.1065\}$ , which is the transpose of the column filter, we obtain the  $3 \times 3$  Gaussian filter.

The Gaussian filter is widely used. The features of this filter include:

1. There is no directional bias, since it is symmetric.
2. By varying the value of the standard deviation  $\sigma$ , the conflicting requirement of less blurring and more noise removal is controlled.
3. The filter is separable.
4. The coefficients fall off to negligible levels at the edges.
5. The Fourier transform of a Gaussian function is another Gaussian function.
6. The convolution of two Gaussian functions is another Gaussian function.

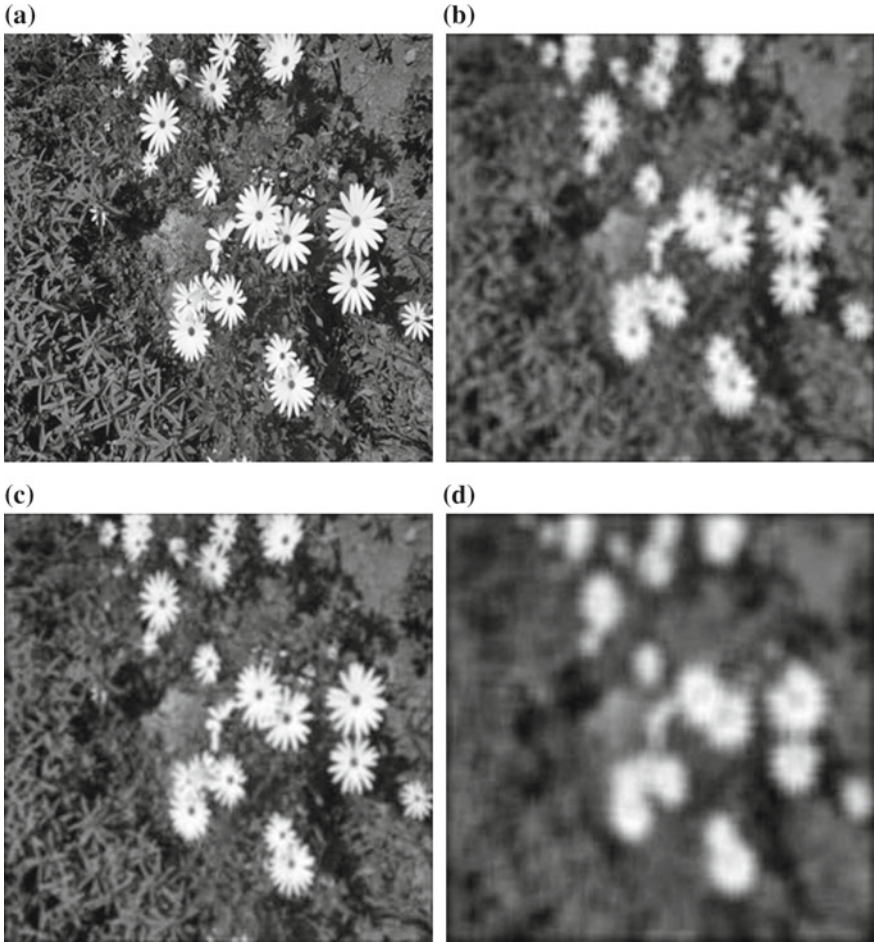
Let

$$x(m, n) = \begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 1 & 2 & 4 \\ 1 & -1 & 2 & -2 \\ 3 & 1 & 2 & 2 \end{bmatrix}$$

Assuming zero-padding at the borders, the output of 1-D filtering of the rows of the input and the output of 1-D filtering of the columns of the partial output are, respectively,

$$\begin{bmatrix} 0.6805 & -0.3610 & 2.4675 & 1.8935 \\ 1.6805 & 1.2130 & 2.1065 & 3.3610 \\ 0.6805 & -0.4675 & 1.2545 & -1.3610 \\ 2.4675 & 1.3195 & 1.8935 & 1.7870 \end{bmatrix} y(m, n) = \begin{bmatrix} 0.7145 & -0.1549 & 2.1663 & 1.8481 \\ 1.4675 & 0.8664 & 2.0542 & 2.7018 \\ 0.9773 & -0.0982 & 1.4133 & -0.5228 \\ 2.0144 & 0.9887 & 1.6238 & 1.2614 \end{bmatrix}$$

Assuming periodicity at the borders, the extended input and the output are, respectively,



**Fig. 2.17** **a** A  $256 \times 256$  8-bit image; **b** filtered image with  $5 \times 5$  averaging filter; **c** filtered image with  $5 \times 5$  Gaussian filter with  $\sigma = 1$ ; **d** filtered image with  $11 \times 11$  averaging filter

$$xe(m, n) = \begin{bmatrix} 2 & 3 & 1 & 2 & 2 & 3 \\ 2 & 1 & -1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 2 & 4 & 2 \\ -2 & 1 & -1 & 2 & -2 & 1 \\ 2 & 3 & 1 & 2 & 2 & 3 \\ 2 & 1 & -1 & 3 & 2 & 1 \end{bmatrix} \quad y(m, n) = \begin{bmatrix} 1.2130 & -0.0143 & 2.3679 & 2.1790 \\ 1.8027 & 0.8664 & 2.0542 & 2.8921 \\ 0.8777 & -0.0982 & 1.4133 & -0.3822 \\ 2.2545 & 0.9502 & 1.8866 & 1.7372 \end{bmatrix}$$

Figure 2.17a shows a  $256 \times 256$  8-bit gray level image. Figure 2.17b, d show the filtered images with  $5 \times 5$  and  $11 \times 11$  averaging filters, respectively. Obviously, the blurring of the image is more with the larger filter. Figure 2.17c shows the filtered image with  $5 \times 5$  Gaussian filter with  $\sigma = 1$ . As the passband spectrum of the

averaging filter, due to sharp transition at the borders, is relatively narrow, the blurring is more for the same size window. As the Gaussian filter is smooth, it has a relatively wider spectrum and the blurring is less.

### Highpass Filtering

Frequency, in image processing, is the rate of change of gray levels of an image with respect to distance. A high frequency component is characterized by large changes in gray levels over short distances and vice versa. Highpass filters pass high frequency components and suppress low frequency components. This type of filters are used for sharpening images and edge detection. Images often get blurred and may require sharpening. Blurring corresponds to integration and sharpening corresponds to differentiation and they undo the effects of the other. High frequency components may have to be enhanced by suppressing low frequency components.

### Laplacian Highpass Filter

While the first-order derivative is also a highpass filter, the Laplacian filter is formed using the second-order derivative. A peak is the indicator of an edge by the first-order derivative and it is the zero-crossing by the second-order derivative. The Laplacian operator of a function  $f(x, y)$

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

is an often used linear derivative operator. It is isotropic (invariant with respect to direction). Consider the 4-neighborhood

$$\begin{bmatrix} & x(m-1, n) & \\ x(m, n-1) & x(m, n) & x(m, n+1) \\ & x(m+1, n) & \end{bmatrix}$$

For discrete signals, differencing approximates differentiation. At the point  $x(m, n)$ , the first differences along the horizontal and vertical directions,  $\Delta_h(m, n)$  and  $\Delta_v(m, n)$ , are defined as

$$\Delta_h x(m, n) = x(m, n) - x(m, n-1) \quad \text{and} \quad \Delta_v x(m, n) = x(m, n) - x(m-1, n)$$

Using the first differences again, we get the second differences.

$$\begin{aligned}
\Delta_v^2 x(m, n) &= \Delta_v x(m+1, n) - \Delta_v x(m, n) \\
&= (x(m+1, n) - x(m, n)) - (x(m, n) - x(m-1, n)) \\
&= x(m+1, n) + x(m-1, n) - 2x(m, n) \\
\Delta_h^2 x(m, n) &= \Delta_h x(m, n+1) - \Delta_h x(m, n) \\
&= (x(m, n+1) - x(m, n)) - (x(m, n) - x(m, n-1)) \\
&= x(m, n+1) + x(m, n-1) - 2x(m, n)
\end{aligned}$$

Summing the two second differences, we get the discrete approximation of the Laplacian as

$$\begin{aligned}
\nabla^2 x(m, n) &= \Delta_h^2 x(m, n) + \Delta_v^2 x(m, n) \\
&= x(m, n+1) + x(m, n-1) + x(m+1, n) + x(m-1, n) - 4x(m, n)
\end{aligned}$$

The filter coefficients  $h(m, n)$  are

$$h(m, n) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.2)$$

By adding this mask by its 45° rotated version, we get the filter coefficients  $h(m, n)$  for 8-neighborhood

$$h(m, n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.3)$$

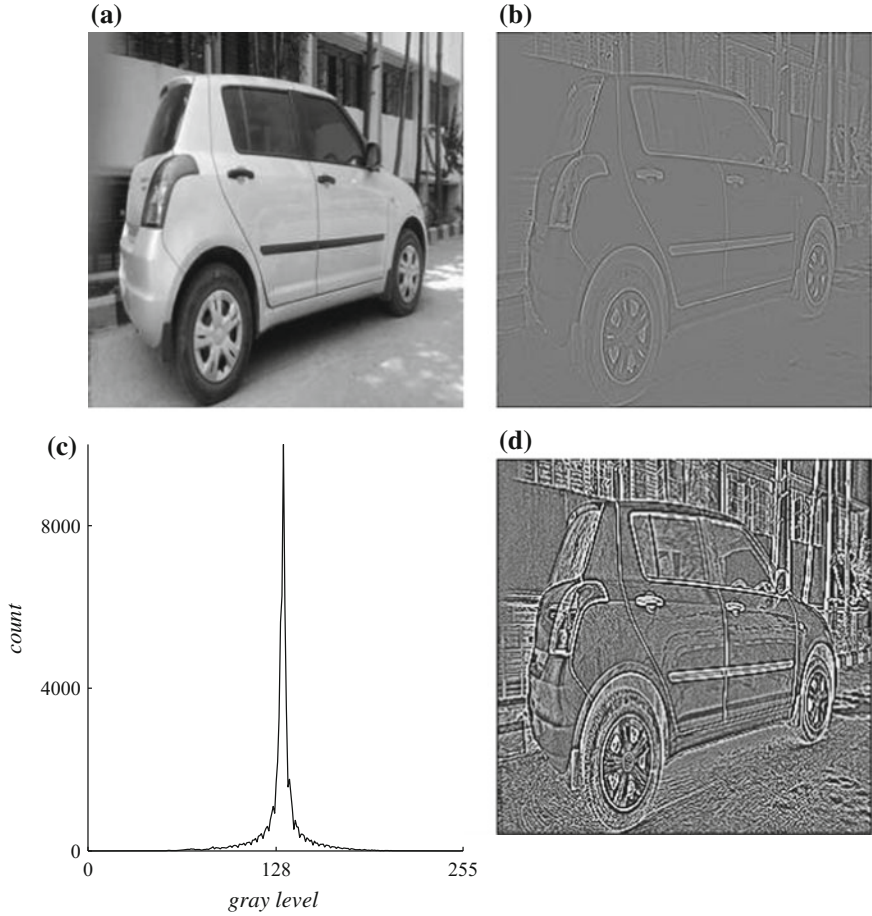
Let the input be the same used for lowpass filtering. With zero-padded and replicated inputs, the outputs of applying the Laplacian mask (Eq. 2.2) are, respectively,

$$y(m, n) = \begin{bmatrix} -3 & 9 & -9 & -1 \\ -5 & -2 & 2 & -14 \\ 0 & 9 & -7 & 16 \\ -10 & 0 & -3 & -8 \end{bmatrix} \quad y(m, n) = \begin{bmatrix} -1 & 8 & -6 & 3 \\ -3 & -2 & 2 & -10 \\ 1 & 9 & -7 & 14 \\ -4 & 1 & -1 & -4 \end{bmatrix}$$

The output has large number of negative values. For proper display of the output, scaling is required. With 256 gray levels,

$$y_s(m, n) = \frac{(y(m, n) - y_{min})}{(y_{max} - y_{min})} 255$$

Figure 2.18a show a  $256 \times 256$  8-bit image. Figure 2.18b shows the image after the application of the Laplacian filter (Eq. (2.2)). The low contrast of the image is



**Fig. 2.18** **a** A  $256 \times 256$  8-bit image; **b** the image after application of the Laplacian filter (Eq. (2.2)); **c** its scaled histogram; **d** the histogram equalized image

due to the concentration of the pixel values in the middle of the scaled histogram (Fig. 2.18c). The histogram equalized image is shown in Fig. 2.18d.

Subtracting the Laplacian from the image sharpens the image. Using the first mask,

$$\begin{aligned}
 x(m, n) - \nabla^2 x(m, n) &= 5x(m, n) - (x(m, n+1) + x(m, n-1) + x(m+1, n) + x(m-1, n)) \\
 &= x(m, n) + 5(x(m, n) \\
 &\quad - \frac{1}{5}(x(m, n+1) + x(m, n-1) + x(m, n) + x(m+1, n) + x(m-1, n)))
 \end{aligned}$$

The third line is a blurred and scaled version of the image  $x(m, n)$ . The high frequency components are suppressed. When the blurred version is subtracted from the input

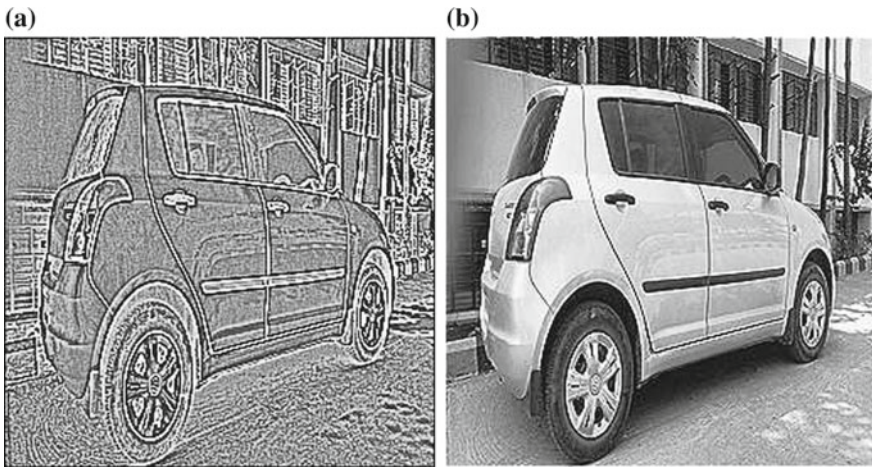
image (called unsharp masking), the resulting image is composed of strong high frequency components and weak low frequency components. When this version is multiplied by the factor 5 and added to the image, the high frequency components are boosted (high-emphasis filtering) and the low frequency components remain about the same. The corresponding Laplacian sharpening filter is deduced from the last equation as

$$h(m, n) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (2.4)$$

Using this filter, with the same input used for lowpass filtering, the outputs with the input zero-padded and replicated are, respectively,

$$y(m, n) = \begin{bmatrix} 4 & -10 & 12 & 3 \\ 7 & 3 & 0 & 18 \\ 1 & -10 & 9 & -18 \\ 13 & 1 & 5 & 10 \end{bmatrix} \quad y(m, n) = \begin{bmatrix} 2 & -9 & 9 & -1 \\ 5 & 3 & 0 & 14 \\ 0 & -10 & 9 & -16 \\ 7 & 0 & 3 & 6 \end{bmatrix}$$

Figure 2.19a shows the image in Fig. 2.18a after application of the Laplacian filter (Eq. (2.3)). The edges at the diagonal directions are sharper compared with Fig. 2.18b. Figure 2.19b shows the image in Fig. 2.18a after application of the Laplacian sharpening filter (Eq. (2.4)). The edges are sharper compared with Fig. 2.18a.



**Fig. 2.19** **a** Image in Fig. 2.18a after application of the Laplacian filter (Eq. (2.3)); **b** Image in Fig. 2.18a after application of the Laplacian sharpening filter (Eq. (2.4))



### 2.4.2 Median Filtering

Some measures of the distribution of the pixel values in an image are the mean, the median, the standard deviation and the histogram. The mean,  $\bar{x}$ , of a  $M \times N$  image  $x(m, n)$  is given by

$$\bar{x} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m, n)$$

The median of a list of  $N$  numbers  $x(n)$

$$\{x(0), x(1), \dots, x(N-1)\}$$

is defined as the middle number of the sorted list of  $x(n)$ , if  $N$  is odd. If  $N$  is even, the median is defined as the mean of the two middle numbers of the sorted list. For 2-D data, all the samples in the window are listed as 1-D data for median computation. The mean and median gives an indication of the center of the data. The spread of the data is given by the variance and the standard deviation. The variance is a measure of the spread of each pixel from the mean of an image. A variance value of zero indicates that all the pixels are the same as the mean. A small variance value indicates that pixel values are distributed close to the mean and close to themselves and vice versa. It is a positive value. The variance  $\sigma^2$  of a  $M \times N$  image  $x(m, n)$  is given by

$$\sigma^2 = \frac{1}{(M)(N)} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (x(m, n) - \bar{x})^2$$

(Sometimes, the divisor  $((M-1)(N-1))$  is used in the definition of  $\sigma^2$ .) The variance is the mean of the squared differences between each value and the mean of the data. The standard deviation  $\sigma$  is the square root of the variance. Consider the  $4 \times 4$  image

23	51	23	32
32	44	44	23
23	23	44	32
44	44	23	23

The mean, variance and standard deviation are 33, 102 and 10.0995, respectively.

Median filtering, which is nonlinear, replaces a pixel by the median of a window of pixels in its neighborhood. It involves sorting the pixels in the window in ascending or descending order and selecting the middle value, if the number of pixels is odd. Otherwise, the average of the two middle values is the median. In this case, if the input is integer-valued then the output can be of the same type by using truncation or rounding. The window sizes used typically are  $3 \times 3$ ,  $5 \times 5$  and  $7 \times 7$ .

Consider the  $4 \times 4$  image and its boundary replicated version

				23	23	51	23	32	32
23	51	23	32	23	23	51	23	32	32
32	44	44	23	32	32	44	44	23	23
23	23	44	32	23	23	23	44	32	32
44	44	23	23	44	44	44	23	23	23
				44	44	44	23	23	23

The image after median filtering with a  $3 \times 3$  window is

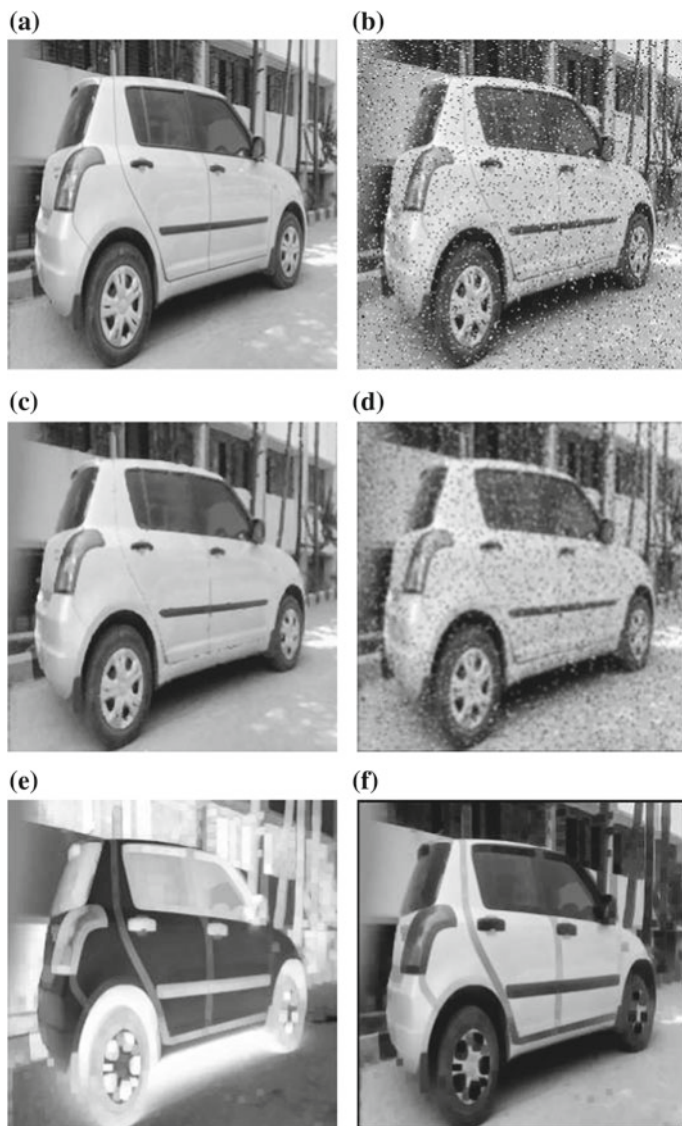
32	32	32	32
23	32	32	32
32	44	32	23
44	44	23	23

Median filtering is effective in reducing the spot (or impulse or salt-and-pepper) noise, characterized by the random occurrence of black and white pixels. The probability distribution of this noise is given by

$$p(x) = \begin{cases} p_1, & \text{for } x = 1 \\ p_0, & \text{for } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

Pixel value 1 indicates that it will be white and zero indicates that the pixel will be black. If the probabilities of the occurrence of the black and white pixels are about equal, then the effect of this noise is to look like flecks of salt and pepper spread all over the image. Hence, it is called as salt-and-pepper noise.

A pixel with a value that is much larger than those of its neighbors is probably a noise pixel. The image is enhanced if such pixels are replaced by the median in their neighborhood. On the other hand, if the pixel value is valid then median filtering will degrade the image quality. In any image processing, the most suitable operators with respect to size and response, and algorithms should be used. This requires some trial and error. While median filtering is commonly used, a pixel can be replaced by any other pixel in the sorted list of its neighborhood, such as the maximum and minimum values. Figure 2.20a, b show a  $256 \times 256$  8-bit image and the image with spot noise, respectively. Figure 2.20c shows the median filtered image with a  $3 \times 3$  window. The noise has been removed. Figure 2.20d shows the lowpass filtered image with a  $3 \times 3$  window. Lowpass filtering is not effective to reduce the spot noise. Figure 2.20e shows the image with each pixel in the complement of input image replaced by the maximum value in its  $5 \times 5$  neighborhood. It highlights the brightest parts of the image. The image has become brighter. Figure 2.20f shows the image with each pixel replaced by the minimum value in its  $5 \times 5$  neighborhood. It highlights the darkest parts of the image.



**Fig. 2.20** **a** A  $256 \times 256$  8-bit image and **b** the image with spot noise; **c** median filtered image with a  $3 \times 3$  window; **d** lowpass filtered image with a  $3 \times 3$  window; **e** image with each pixel in the complement of the input image replaced by the maximum value in its  $5 \times 5$  neighborhood; **f** image with each pixel replaced by the minimum value in its  $5 \times 5$  neighborhood

## 2.5 Summary

- Image enhancement involves modifying the pixel values to improve the quality of the image with some respect for human or machine vision.
- The simplest type is that in which each output pixel is a function of the input pixel only. Typical operations include complementing and gamma correction. Pointwise arithmetic and logical operations are carried out with corresponding pixels in two or more images.
- Histogram is the count of the number of occurrences of each gray level in the image. In addition to enhancement, histograms are useful for other operations such as segmentation.
- In histogram stretching, a part of the range of gray levels is stretched to enhance the image.
- In histogram equalization, the gray levels are redistributed uniformly over the gray level range to enhance the image.
- In histogram specification, the gray levels are redistributed, according to the specified histogram, over the gray level range to restore the image, with a knowledge of its original histogram.
- Thresholding is choosing a gray level of some significance and using it to do processing such as segmentation, compression and denoising.
- In neighborhood operations, the output value of a pixel is a linear or nonlinear function of a set of pixels in its neighborhood.
- In linear filtering, the spectrum of the image is modified in a desired way. This includes operations such as lowpass and highpass filtering.
- Typical lowpass filters are averaging and Gaussian. Laplacian filter is of highpass type.
- Filtering is implemented by the convolution operation in the spatial domain. Although an image is a 2-D function, 1-D convolution is also used for filtering with separable filters.
- In nonlinear filtering, the output value of a pixel is a nonlinear function of a set of pixels in its neighborhood.
- Typical example of nonlinear filtering is median filtering, in which the output pixel value corresponding to a pixel is the median of a set of pixels in its neighborhood.

## Exercises

**2.1** Find the complement of the  $4 \times 4$  8-bit gray level image and verify that the image can be restored by complementing the complemented image.

(i)

$$\begin{bmatrix} 112 & 148 & 72 & 153 \\ 120 & 125 & 30 & 99 \\ 95 & 120 & 89 & 33 \\ 170 & 99 & 109 & 40 \end{bmatrix}$$

(ii)

$$\begin{bmatrix} 164 & 127 & 117 & 59 \\ 154 & 122 & 104 & 83 \\ 129 & 136 & 100 & 60 \\ 117 & 128 & 80 & 48 \end{bmatrix}$$

(iii)

$$\begin{bmatrix} 46 & 48 & 46 & 45 \\ 42 & 49 & 46 & 45 \\ 64 & 73 & 60 & 43 \\ 94 & 69 & 63 & 37 \end{bmatrix}$$

**2.2** Find the complement of the  $4 \times 4$  binary image and verify that the image can be restored by complementing the complemented image.

(i)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

(ii)

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(iii)

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

**2.3** For the list of gray levels, apply gamma correction and find the corresponding new gray levels. Apply the inverse transformation to the new gray levels and verify that the given gray levels are obtained.

$$\{0, 25, 50, 100, 150, 200, 250, 255\}$$

- (i)  $\gamma = 0.8$ .
- (ii)  $\gamma = 1.1$ .
- (iii)  $\gamma = 1.8$ .

**2.4** Given a  $4 \times 4$  4-bit image, find the histogram equalized version of it.

\* (i)

$$\begin{bmatrix} 4 & 4 & 3 & 3 \\ 4 & 4 & 4 & 3 \\ 5 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

(ii)

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 0 & 2 \end{bmatrix}$$

(iii)

$$\begin{bmatrix} 3 & 5 & 5 & 3 \\ 4 & 4 & 4 & 3 \\ 4 & 2 & 3 & 4 \\ 4 & 2 & 2 & 3 \end{bmatrix}$$

**2.5** Given  $4 \times 4$  4-bit reference and input images, use histogram matching to restore the input image.

\* (i) The reference and input images, respectively, are

$$\begin{bmatrix} 4 & 4 & 3 & 3 \\ 4 & 4 & 4 & 3 \\ 5 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix} \quad \begin{bmatrix} 15 & 15 & 0 & 0 \\ 15 & 15 & 15 & 0 \\ 15 & 15 & 15 & 15 \\ 15 & 15 & 15 & 15 \end{bmatrix}$$

(ii) The reference and input images, respectively, are

$$\begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 2 & 2 & 3 \\ 2 & 2 & 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 15 & 15 & 15 & 15 \\ 15 & 15 & 15 & 15 \\ 15 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(iii) The reference and input images, respectively, are

$$\begin{bmatrix} 3 & 5 & 5 & 3 \\ 4 & 4 & 4 & 3 \\ 4 & 2 & 3 & 4 \\ 4 & 2 & 2 & 3 \end{bmatrix} \quad \begin{bmatrix} 0 & 15 & 15 & 0 \\ 15 & 15 & 15 & 0 \\ 15 & 0 & 0 & 15 \\ 15 & 0 & 0 & 0 \end{bmatrix}$$

**2.6** Given a  $8 \times 8$  8-bit image, find the binary, hard and soft thresholded versions with the threshold  $T = 160$ .

$$\begin{bmatrix} 255 & 255 & 255 & 117 & 50 & 39 & 50 & 56 \\ 255 & 255 & 255 & 194 & 45 & 26 & 48 & 54 \\ 255 & 255 & 255 & 241 & 61 & 25 & 53 & 57 \\ 255 & 255 & 255 & 255 & 104 & 32 & 64 & 64 \\ 255 & 255 & 255 & 255 & 154 & 37 & 59 & 61 \\ 255 & 255 & 255 & 255 & 199 & 54 & 55 & 61 \\ 255 & 255 & 255 & 255 & 230 & 71 & 59 & 64 \\ 255 & 255 & 255 & 255 & 250 & 95 & 60 & 68 \end{bmatrix}$$

**2.7** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  averaging filter with the borders zero-padded.

$$x(m, n) = \begin{bmatrix} 70 & 62 & 51 & 45 \\ 71 & 62 & 57 & 55 \\ 73 & 65 & 56 & 60 \\ 68 & 69 & 63 & 66 \end{bmatrix}$$

**2.8** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  averaging filter with the borders replicated.

$$x(m, n) = \begin{bmatrix} 41 & 43 & 45 & 43 \\ 40 & 41 & 42 & 41 \\ 42 & 38 & 39 & 42 \\ 39 & 33 & 37 & 36 \end{bmatrix}$$

**2.9** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  averaging filter with the borders periodically extended.

$$x(m, n) = \begin{bmatrix} 45 & 78 & 87 & 51 \\ 59 & 56 & 62 & 49 \\ 59 & 39 & 44 & 57 \\ 56 & 36 & 35 & 51 \end{bmatrix}$$

**2.10** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  Gaussian filter ( $\sigma = 0.5$ ) with the borders symmetrically extended.

$$x(m, n) = \begin{bmatrix} 202 & 195 & 192 & 191 \\ 216 & 211 & 200 & 209 \\ 224 & 212 & 215 & 227 \\ 224 & 205 & 227 & 230 \end{bmatrix}$$

**2.11** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  Gaussian filter ( $\sigma = 0.5$ ) with the borders periodically extended.

$$x(m, n) = \begin{bmatrix} 202 & 195 & 192 & 191 \\ 216 & 211 & 200 & 209 \\ 224 & 212 & 215 & 227 \\ 224 & 205 & 227 & 230 \end{bmatrix}$$

**2.12** Given a  $4 \times 4$  image, find the  $4 \times 4$  lowpass filtered output using the  $3 \times 3$  Gaussian filter ( $\sigma = 0.5$ ) with the borders zero-padded.

$$x(m, n) = \begin{bmatrix} 95 & 82 & 54 & 33 \\ 84 & 78 & 56 & 64 \\ 73 & 71 & 53 & 60 \\ 73 & 73 & 54 & 36 \end{bmatrix}$$

**2.13** Given a  $4 \times 4$  image, find the  $4 \times 4$  highpass filtered output using the  $3 \times 3$  Laplacian filter

$$h(m, n) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

with the borders zero-padded.

$$x(m, n) = \begin{bmatrix} 45 & 52 & 56 & 52 \\ 49 & 60 & 55 & 55 \\ 47 & 55 & 53 & 46 \\ 45 & 48 & 51 & 40 \end{bmatrix}$$

**2.14** Given a  $4 \times 4$  image, find the  $4 \times 4$  highpass filtered output using the  $3 \times 3$  Laplacian filter with the borders symmetrically extended.

$$x(m, n) = \begin{bmatrix} 64 & 62 & 62 & 68 \\ 68 & 66 & 58 & 64 \\ 75 & 70 & 60 & 58 \\ 72 & 69 & 59 & 60 \end{bmatrix}$$

**2.15** Given a  $4 \times 4$  image, find the  $4 \times 4$  highpass filtered output using the  $3 \times 3$  Laplacian filter with the borders replicated.

$$x(m, n) = \begin{bmatrix} 39 & 40 & 35 & 33 \\ 31 & 40 & 39 & 37 \\ 34 & 38 & 41 & 43 \\ 37 & 39 & 42 & 43 \end{bmatrix}$$



**\*2.16** Given a  $4 \times 4$  image, find the  $4 \times 4$  enhanced output using the  $3 \times 3$  Laplacian sharpening filter

$$h(m, n) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

with the borders replicated.

$$x(m, n) = \begin{bmatrix} 190 & 206 & 228 & 238 \\ 180 & 205 & 227 & 219 \\ 182 & 203 & 211 & 159 \\ 184 & 212 & 206 & 177 \end{bmatrix}$$

**2.17** Given a  $4 \times 4$  image, find the  $4 \times 4$  enhanced output using the  $3 \times 3$  Laplacian sharpening filter with the borders periodically extended.

$$x(m, n) = \begin{bmatrix} 138 & 163 & 162 & 177 \\ 148 & 157 & 167 & 175 \\ 153 & 165 & 160 & 178 \\ 157 & 162 & 164 & 188 \end{bmatrix}$$

**2.18** Given a  $4 \times 4$  image, find the  $4 \times 4$  enhanced output using the  $3 \times 3$  Laplacian sharpening filter with the borders zero-padded.

$$x(m, n) = \begin{bmatrix} 201 & 195 & 191 & 169 \\ 210 & 201 & 181 & 157 \\ 213 & 207 & 190 & 166 \\ 204 & 204 & 197 & 159 \end{bmatrix}$$

**2.19** Given a  $4 \times 4$  image, find the  $4 \times 4$  median filtered output using the  $3 \times 3$  window with the borders zero-padded.

(i)

$$x(m, n) = \begin{bmatrix} 201 & 195 & 191 & 169 \\ 210 & 201 & 181 & 157 \\ 213 & 207 & 190 & 166 \\ 204 & 204 & 197 & 159 \end{bmatrix}$$

(ii)

$$x(m, n) = \begin{bmatrix} 138 & 163 & 162 & 177 \\ 148 & 157 & 167 & 175 \\ 153 & 165 & 160 & 178 \\ 157 & 162 & 164 & 188 \end{bmatrix}$$

(iii)

$$x(m, n) = \begin{bmatrix} 190 & 206 & 228 & 238 \\ 180 & 205 & 227 & 219 \\ 182 & 203 & 211 & 159 \\ 184 & 212 & 206 & 177 \end{bmatrix}$$

Digital Image Processing

A Signal Processing and Algorithmic Approach

Sundararajan, D.

2017, XVII, 468 p. 182 illus., 17 illus. in color.,

Hardcover

ISBN: 978-981-10-6112-7