Example

The pixel values of the following 5×5 image are represented by 8-bit integers:

$$f = \begin{bmatrix} 123 & 162 & 200 & 147 & 93 \\ 137 & 157 & 165 & 232 & 189 \\ 151 & 155 & 152 & 141 & 130 \\ 205 & 101 & 100 & 193 & 115 \\ 250 & 50 & 75 & 88 & 100 \end{bmatrix}$$

Determine $\boldsymbol{f}$ with a gray-level resolution of 2^k for (i) k=5 and (ii) k=3.

Solution:

Dividing the image by 2 will reduce its gray level resolution by 1 bit.

Hence to reduce the gray level resolution from 8-bit to 5-bit,

8 bits – 5 bits = 3 bits will be reduced

Thus, we divide the 8-bit image by 8 (2^3) to get the following 5-bit image:

$$f = \begin{bmatrix} 15 & 20 & 25 & 18 & 11 \\ 17 & 19 & 20 & 29 & 23 \\ 18 & 19 & 19 & 17 & 16 \\ 25 & 12 & 12 & 24 & 14 \\ 31 & 6 & 9 & 10 & 12 \end{bmatrix}$$

Similarly, to obtain 3-bit image, we divide the 8-bit image by 2^5 (32) to get:

$$f = \begin{bmatrix} 3 & 5 & 6 & 4 & 2 \\ 4 & 4 & 5 & 7 & 5 \\ 4 & 4 & 4 & 4 & 4 \\ 6 & 3 & 3 & 6 & 3 \\ 7 & 1 & 2 & 2 & 3 \end{bmatrix}$$

Basic relationships between pixels

1. Neighbors of a Pixel

A pixel p at coordinates (x ,y) has the following neighbors:

- **4-neighbors**
  - four horizontal and vertical neighbors whose coordinates are
    $(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$
  - denoted by $N_4(p)$
- **diagonal neighbors**
  - $(x + l , y + l),(x+ l , y- l),(x-l , y + l),(x- l , y -1)$
  - denoted by $N_D(p)$
- **8-neighbors**
  - both 4-neighbors and diagonal neighbors
  - denoted by $N_8(p)$

Note: some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image.

2. Distance Measures

For pixels p, q, and z, with coordinates (x, y), (s, t), and (v, w), respectively, D is a *distance function* or *metric* if

a) $D(p, q) >= 0$ ($D(p,q)=0$ iff $p = q$),

b) $D(p,q) = D(q,p)$, *and*

c) $D(p,z) <= D(p,q) + D(q,z)$.

- **Euclidean distance** between p and q is defined as

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

- **City-block distance** between p and q is defined as

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

- **<u>Chessboard distance</u>** between p and q is defined as

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

Zooming and Shrinking Digital Images

A) Zooming

Zooming may be viewed as oversampling. It is the scaling of an image area ***A*** of ***w×h*** pixels by a factor ***s*** while maintaining spatial resolution (i.e. output has ***sw×sh*** pixels). Zooming requires two steps:

- creation of new pixel locations
- assignment of gray levels to those new locations

There are many methods of gray-level assignments, for example *nearest neighbor interpolation* and *bilinear interpolation*.

<u>Nearest neighbor interpolation (Zero-order hold)</u>

is performed by repeating pixel values, thus creating a checkerboard effect. **Pixel replication** (a special case of nearest neighbor interpolation) is used to increase the size of an image an integer number of times. The example below shows 8-bit image zooming by 2x (2 times) using nearest neighbor interpolation:

$$\begin{bmatrix} 69 & 50 & 80 \\ 45 & 60 & 66 \\ 30 & 55 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 69 & 50 & 50 & 80 & 80 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 30 & 30 & 55 & 55 & 80 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 69 & 50 & 50 & 80 & 80 \\ 69 & 69 & 50 & 50 & 80 & 80 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 30 & 30 & 55 & 55 & 80 & 80 \\ 30 & 30 & 55 & 55 & 80 & 80 \end{bmatrix}$$

Original image   image with rows expanded   image with rows and columns expanded

<u>Bilinear interpolation (First-order hold)</u>

is performed by finding linear interpolation between adjacent pixels, thus creating a blurring effect. This can be done by finding the average gray value between two pixels and use that as the pixel value between those two. We can do this for the rows first, and then we take that result and expand the columns in the same way. The example below shows 8-bit image zooming by 2x (2 times) using bilinear interpolation:

$$\begin{bmatrix} 69 & 50 & 80 \\ 45 & 60 & 66 \\ 30 & 55 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 59 & 50 & 65 & 80 \\ 45 & 52 & 60 & 63 & 66 \\ 30 & 42 & 55 & 67 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 59 & 50 & 65 & 80 \\ 57 & 55 & 55 & 64 & 73 \\ 45 & 52 & 60 & 63 & 66 \\ 37 & 47 & 57 & 65 & 73 \\ 30 & 42 & 55 & 67 & 80 \end{bmatrix}$$

Original image   image with rows expanded   image with rows and columns expanded

Note that the zoomed image has size 2M-1 × 2N-1. However, we can use techniques such as padding which means adding new columns and/or rows to the original image in order to perform bilinear interpolation to get zoomed image of size 2M × 2N.

The figure below shows image zooming using nearest neighbor interpolation and bilinear interpolation. We can clearly see the checkerboard and blurring effects. However, the improvements using bilinear interpolation in overall appearance are clear, especially in the 128×128 and 64×64 cases. The bilinear interpolated images are smoother than those resulted from nearest neighbor interpolation.

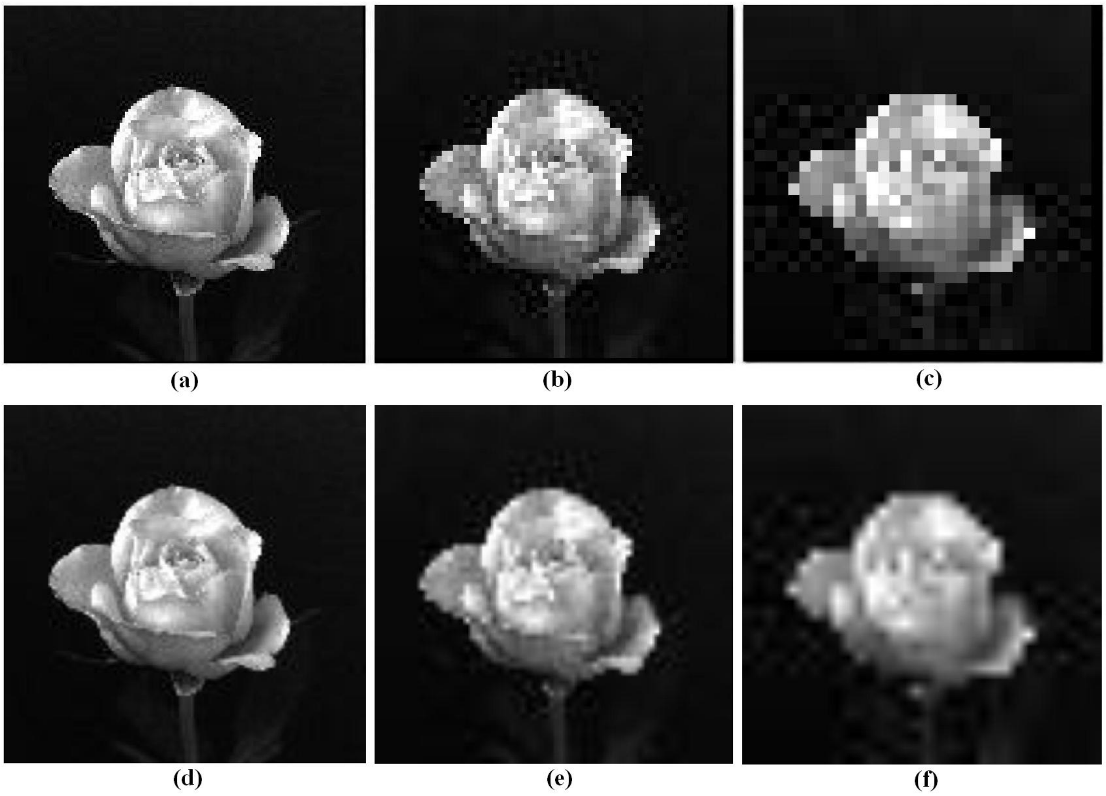


Figure 3.1 Top row: images zoomed from 128×128, 64×64, and 32×32 pixels to 1024×1024 pixels susing nearest neighbor interpolation. Bottom row, same sequence, but using bilinear interpolation

B) Shrinking

Shrinking may be viewed as undersampling. Image shrinking is performed by **row-column deletion**. For example, to shrink an image by one-half, we delete every other row and column.

$$\begin{bmatrix} 69 & 69 & 50 & 50 & 80 & 80 \\ 69 & 69 & 50 & 50 & 80 & 80 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 30 & 30 & 55 & 55 & 80 & 80 \\ 30 & 30 & 55 & 55 & 80 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 69 & 50 & 50 & 80 & 80 \\ 45 & 45 & 60 & 60 & 66 & 66 \\ 30 & 30 & 55 & 55 & 80 & 80 \end{bmatrix} = \begin{bmatrix} 69 & 50 & 80 \\ 45 & 60 & 66 \\ 30 & 55 & 80 \end{bmatrix}$$

Original image   image with rows deleted   image with rows and columns deleted

Image algebra

There are two categories of algebraic operations applied to images:

- Arithmetic
- Logic

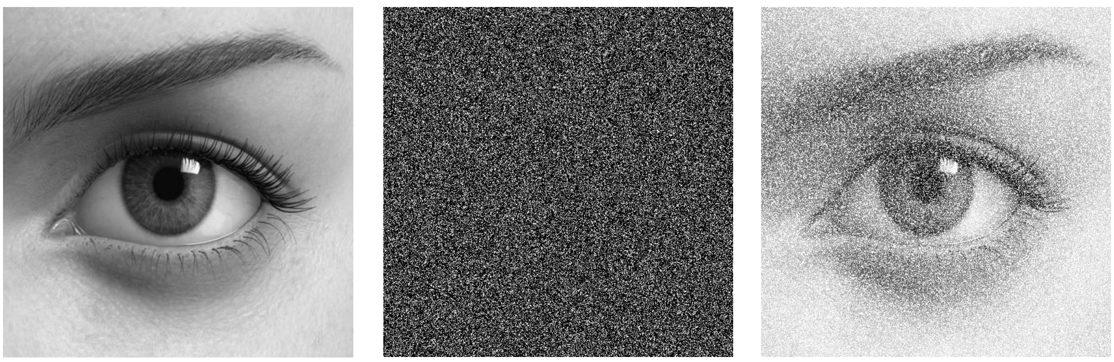
These operations are performed on a pixel-by-pixel basis between two or more images, except for the NOT logic operation which requires only one image. For example, to add images I_1 and I_2 to create I_3:

$$I_3(x,y) = I_1(x,y) + I_2(x,y)$$

$$I_1 = \begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 5 \\ 2 & 4 & 6 \end{bmatrix} \qquad I_2 = \begin{bmatrix} 6 & 6 & 6 \\ 4 & 2 & 6 \\ 3 & 5 & 5 \end{bmatrix}$$

$$I_3 = \begin{bmatrix} 3+6 & 4+6 & 7+6 \\ 3+4 & 4+2 & 5+6 \\ 2+3 & 4+5 & 6+5 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 13 \\ 7 & 6 & 11 \\ 5 & 9 & 11 \end{bmatrix}$$

- **Addition** is used to combine the information in two images. Applications include development of image restoration algorithms for modeling additive noise and special effects such as image morphing in motion pictures as shown in the figures below.



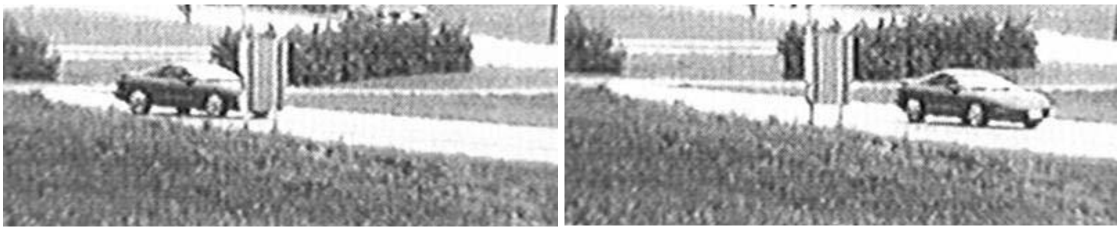
(a) Original image (b) Gaussian noise (c) Addition of images (a) and (b)

Figure 3.2 Image addition (adding noise to the image)

(a) First Original (b) Second Original (c) Addition of images (a) and (b)

Figure 3.3 Image addition (image morphing example)

- **Subtraction** of two images is often used to detect motion. For example, in a scene when nothing has changed, the image resulting from the subtraction is filled with zeros(black image). If something has changed in the scene, subtraction produces a nonzero result at the location of movement as shown in the figure below.



(a) Original scene (b) Same scene at a later time

(c) Subtracting image (b) from (a). Only moving objects appear in the resulting image

Figure 3.4 Image subtraction

- **Multiplication and division** are used to adjust the brightness of an image. Multiplying the pixel values by a number greater than one will brighten the image, and dividing the pixel values by a factor greater than one will darken the image. An example of brightness adjustment is shown in the figure below.

(a) Original image

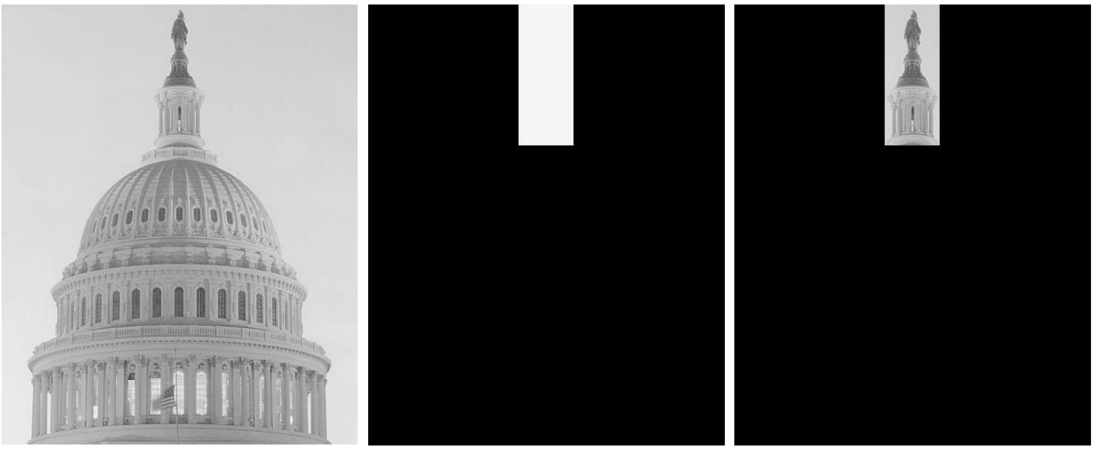
(b) Image multiplied by 2

(c) Image divided by 2

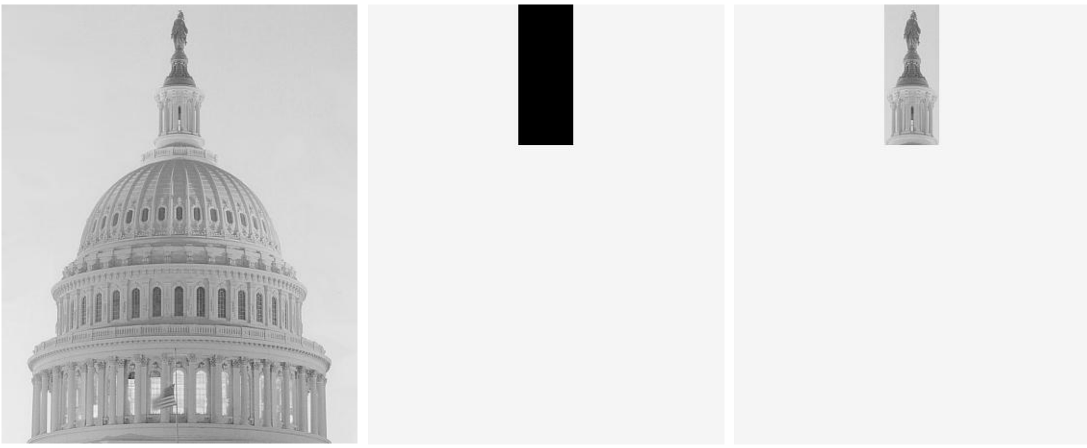
Figure 3.5 Image multiplication and division

The logic operations AND, OR, and NOT form a complete set, meaning that any other logic operation (XOR, NOR, NAND) can be created by a combination of these basic elements. They operate in a bit-wise fashion on pixel data.

The AND and OR operations are used to perform *masking* operation; that is; for selecting subimages in an image, as shown in the figure below. Masking is also called *Region of Interest (ROI)* processing.



(a) Original image (b) AND image mask (c) Resulting image, (a) AND (b)



(d) Original image (e) OR image mask (f) Resulting image, (d) OR (e)

Figure 3.6 Image masking

The NOT operation creates a negative of the original image ,as shown in the figure below, by inverting each bit within each pixel value.

(a) Original image

(b) NOT operator applied to image (a)

Figure 3.7 Complement image

Image Histogram

The histogram of a digital image is a plot that records the frequency distribution of gray levels in that image. In other words, the histogram is a plot of the gray-level values versus the number of pixels at each gray value. The shape of the histogram provides us with useful information about the nature of the image content.

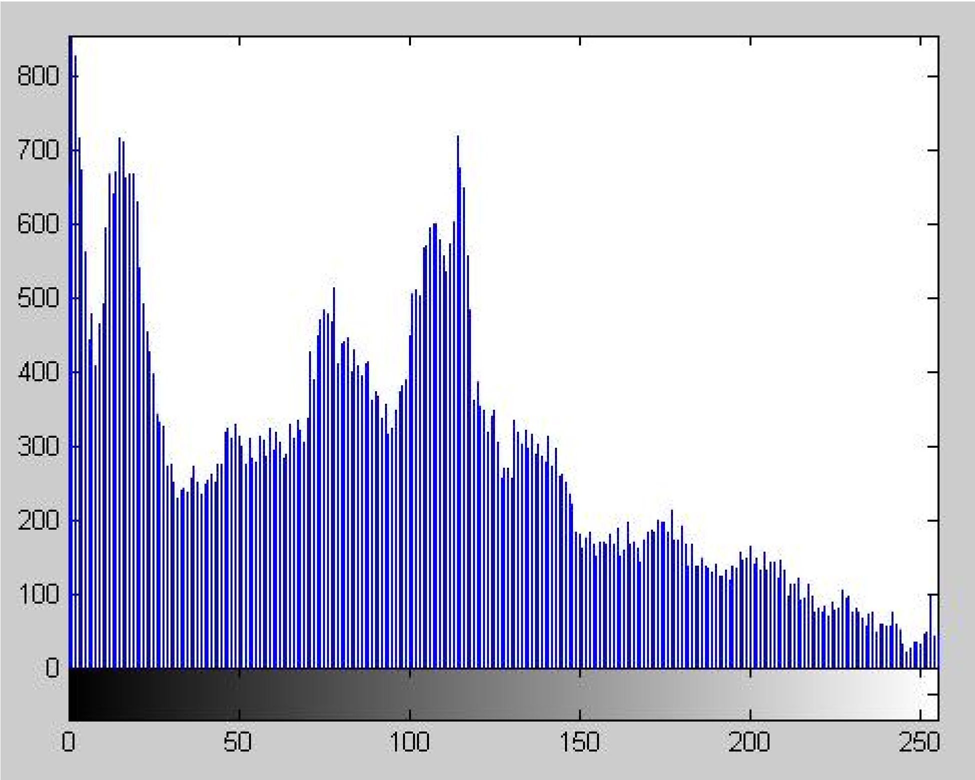
The histogram $\boldsymbol{h}$ of a digital image $\boldsymbol{f}$ of size $M \times N$ and gray levels in the range $[0, L\text{-}1]$ is a discrete function

$$\boldsymbol{h}(k) = n_k$$

where k is the kth gray level and n_k is the number of pixels in the image having gray level k .

The next figure shows an image and its histogram.

(a) 8-bit image

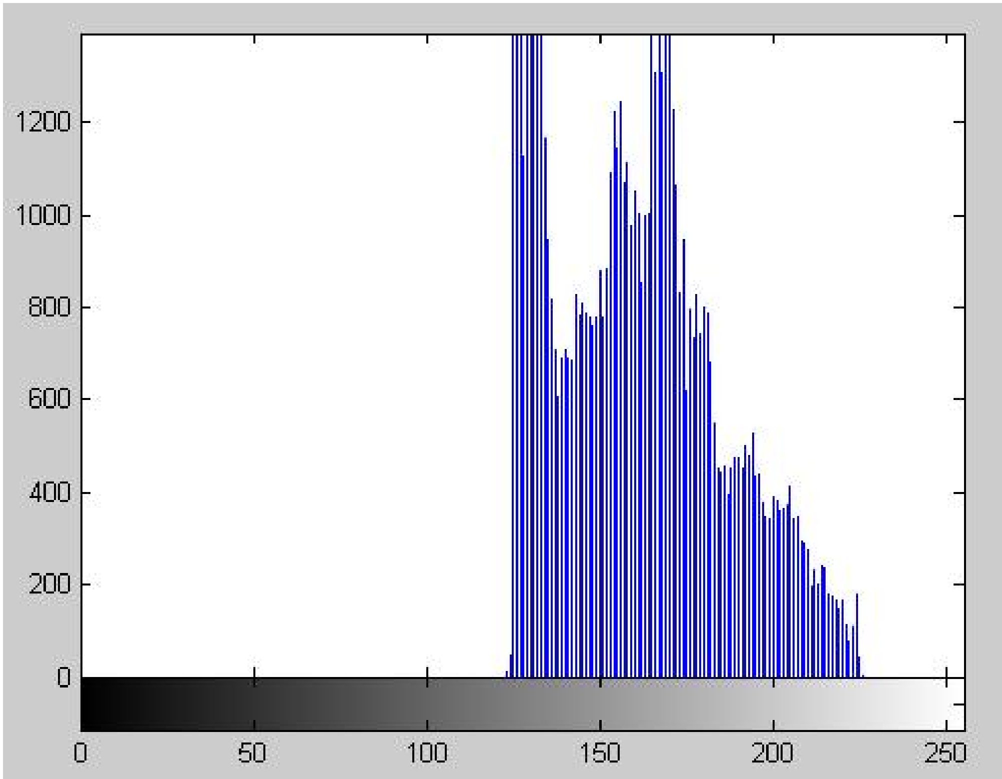


(b) Histogram of image (a)

Figure 3.8 Image histogram

Note that the horizontal axis of the histogram plot (Figure 3.8(b)) represents gray level values, k, from 0 to 255. The vertical axis represents the values of $h(k)$ i.e. the number of pixels which have the gray level k. The next figure shows another image and its histogram.

(a) 8-bit image



(b) Histogram of image (a)

Figure 3.9 Another image histogram

It is customary to "normalize" a histogram by dividing each of its values by the total number of pixels in the image, i.e. use the probability distribution:

$$p(k) = \frac{h(k)}{M \times N}$$

Thus, $p(k)$ represents the probability of occurrence of gray level k.

As with any probability distribution:

- all the values of a normalized histogram $p(k)$ are less than or equal to 1
- the sum of all $p(k)$ values is equal to 1

Histograms are used in numerous image processing techniques, such as image enhancement, compression and segmentation.