

Chapter 2. Digital Image Fundamentals

Bin Liu
BIN's BIN

1 A Simple Image Formation Model

Recall that images are denoted by two-dimensional functions of the form $f(x, y)$. $f(x, y)$ is nonzero and finite; that is,

$$0 < f(x, y) < \infty \quad (1)$$

Let the intensity (gray level) of a monochrome image at any coordinates (x_0, y_0) be denoted by

$$l = f(x_0, y_0) \quad (2)$$

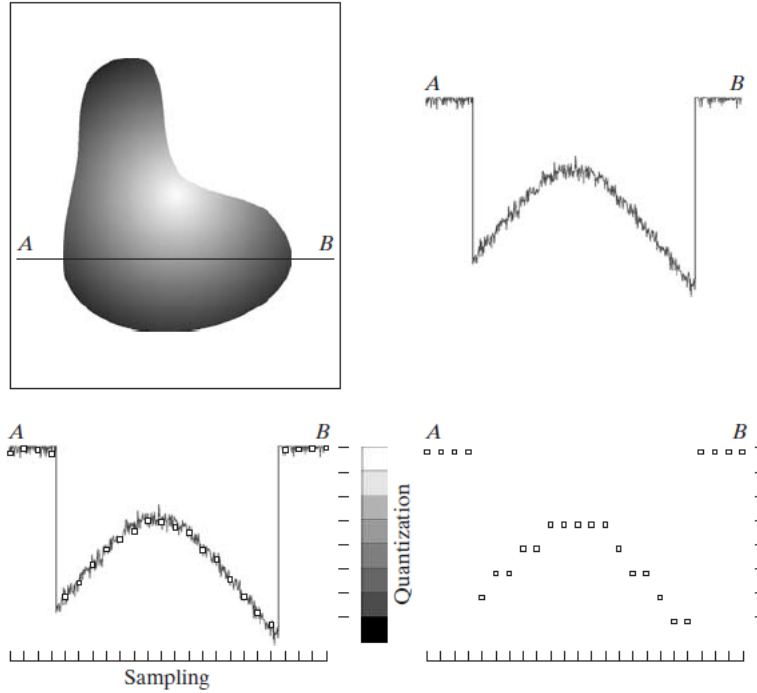
which lies in the range

$$L_{\min} \leq l \leq L_{\max} \quad (3)$$

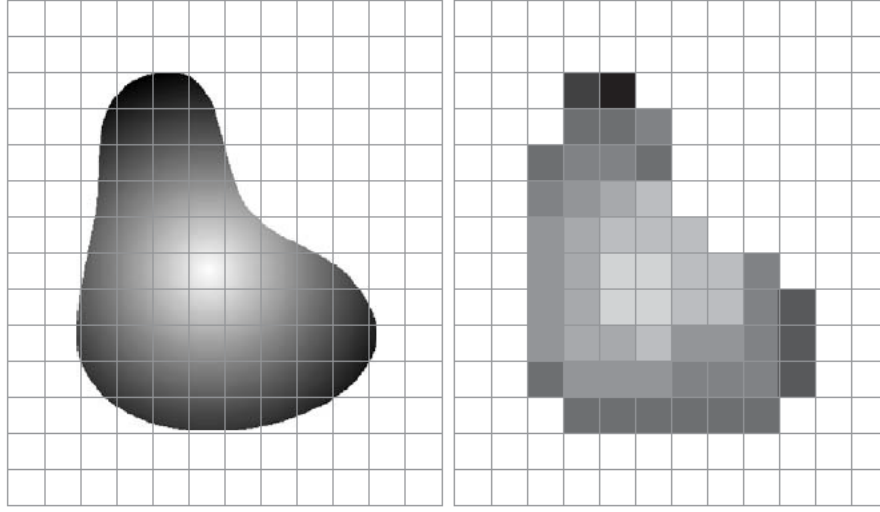
2 Image Sampling and Quantization

2.1 Basic Concepts in Sampling and Quantization

The following figure illustrates the basic concepts. The upper left is a continuous image f . The upper right is a plot of amplitude values of the continuous image along the line segment AB . The lower left shows the sampling process, and the lower right shows the result after sampling and quantization.

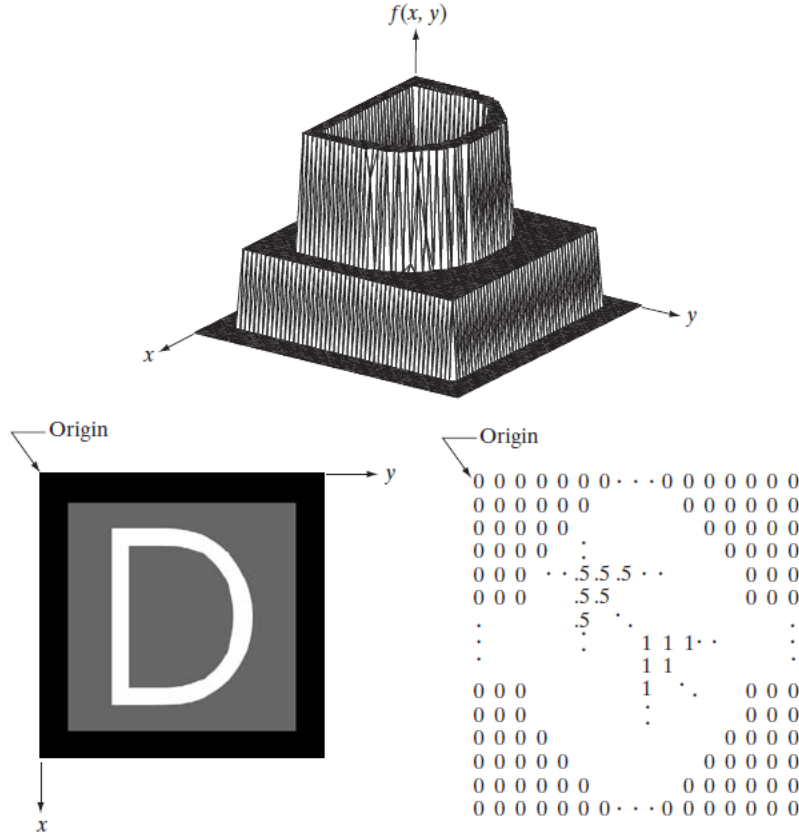


The following figure show another example of sampling and quantization when a sensing array is used for image acquisition.



2.2 Representing Digital Images

Let $f(s, t)$ represent a continuous image function of two continuous variables, s and t . Suppose the continuous image is sampled into a 2D array, $f(x, y)$, containing M rows and N columns, where (x, y) are discrete coordinates, that is, $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$. The following figure shows three different ways to represent $f(x, y)$.



The numerical arrays are used for processing and algorithm development. The representation of an $M \times N$ numerical array is written as

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1, 0) & f(M-1, 1) & \cdots & f(M-1, N-1) \end{bmatrix} \quad (4)$$

The first element of a matrix is by convention at the top left of the array as shown above. This representation is the standard right-handed Cartesian coordinate system. The axes point downward and to the right.

Each element of this matrix is called an *image element*, *picture element*, *pixel*, or *pel*.

A more traditional matrix notation is used to denote a digital image and its elements:

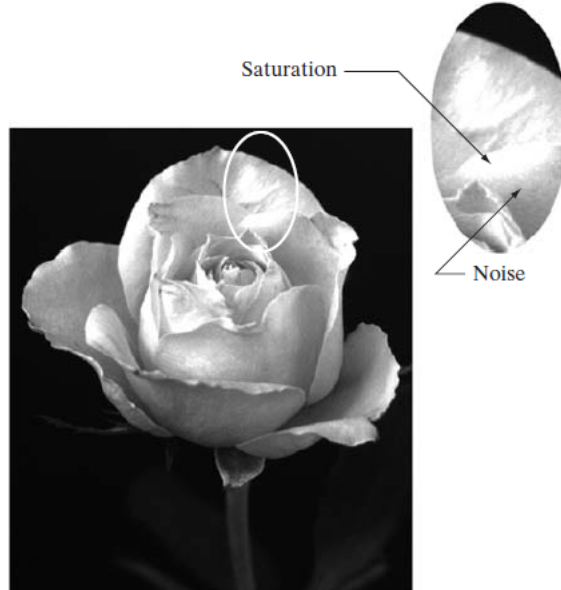
$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad (5)$$

Let Z and R denote the set of integers and the set of real numbers, respectively. The sampling process may be viewed as partitioning the xy -plane into a grid, with the coordinates of the center of each cell in the grid being a pair of elements from the Cartesian product Z^2 , which is the set of all ordered pairs of elements (z_i, z_j) , with z_i and z_j being integers from Z .

The digitization process requires that decisions be made regarding the values of M , N , and for the number L of discrete intensity levels. The number of intensity levels typically is an integer power of 2:

$$L = 2^k \quad (6)$$

assuming that the discrete levels are equally spaced and that they are integers in the interval $[0, L - 1]$. The *dynamic range* of an imaging system is defined to be the ratio of the maximum measurable intensity to the minimum detectable intensity level in the system. The upper limit is determined by *saturation* and the lower limit by *noise*, as shown below.



The dynamic range establishes the lowest and highest intensity levels that a system can represent and an image can have. The *image contrast* is defined as the difference in intensity between the highest and lowest intensity levels in an image. An image with high dynamic range typically has high contrast.

The number, b , of bits required to store a digitized image is

$$b = M \times N \times k \quad (7)$$

When $M = N$, this equation becomes

$$b = N^2 k \quad (8)$$

The table below shows the number of bits required to store square images with various values of N and k . The number of intensity levels corresponding to each value of k is shown in parentheses. When an image can have 2^k intensity levels, the image is referred to as a " k -bit image".

Number of storage bits for various values of N and k . L is the number of intensity levels.

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

2.3 Spatial and Intensity Resolution

Spatial resolution is a measure of the smallest discernible detail in an image. Dots per unit distance is a measure of image resolution used commonly in the printing and publishing industry. In the U.S., this measure usually is expressed as *dots per inch* (dpi).

Intensity resolution is a measure of the smallest discernible change in intensity level. It is referred to as the number of bits used to quantize intensity.

3 Some Basic Relationships between Pixels

3.1 Neighbors of a Pixel

A pixel p at coordinate (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$$

This set of pixels, called the 4-*neighbors* of p , is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y) .

The 4 *diagonal* neighbors of p have coordinates

$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1)$$

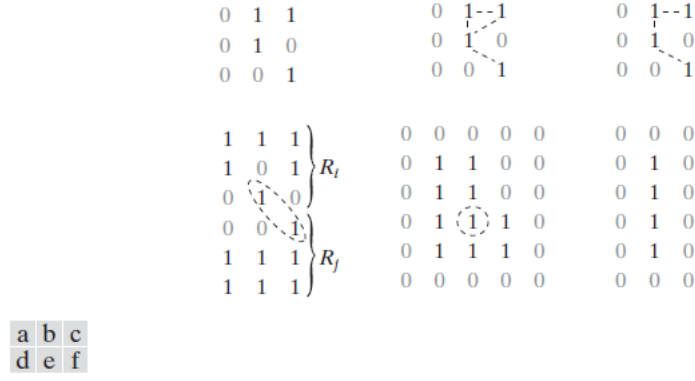
which are denoted by $N_D(p)$. These points, together with the 4-neighbors, are called the 8-*neighbors* of p , denoted by $N_8(p)$.

3.2 Adjacency, Connectivity, Regions, and Boundaries

Let V be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if the adjacency of pixels is referred to with value 1. In a gray-scale image, V is set to contains more elements. Three types of adjacency are considered:

- 4-*adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
- 8-*adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.
- *m-adjacency* (mixed adjacency). Two pixels p and q with values from V are *m*-adjacent if
 - q is in $N_4(p)$, or
 - q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V .

Mixed adjacency is a modification of 8-adjacency. Consider the pixel arrangement shown below (a) for $V = \{1\}$. The three pixels at the top of figure (b) below show multiple (ambiguous) 8-adjacency, as indicated by the dashed lines. This ambiguity is removed by using *m*-adjacency, as shown in figure (c) below.



A (*digital*) *path* (or *curve*) from pixel p with coordinate (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$, and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$. In this case, n is the *length* of the path. If $(x_0, y_0) = (x_n, y_n)$, the path is *closed* path.

Let S represent a subset of pixels in an image. Two pixels p and q are said to be *connected* in S if there exists a path between them consisting entirely of pixels in S . For any pixel p in S , the *set* of pixels that are connected to it in S is called a *connected component* of S . If it only has one connected component, then set S is called a *connected set*.

Let R be a subset of pixels in an image. R is called a *region* of the image if R is a connected set. Two regions, R_i and R_j , are said to be *adjacent* if their union form a connected set. Regions that are not adjacent are said to be *disjoint*. Regions are considered for 4- and 8-adjacency as shown above in figure (d).

Suppose that an image contains K disjoint regions, $R_k, k = 1, 2, \dots, K$, none of which touches the image border. Let R_u denote the union of all the K regions, and let $(R_u)^c$ denote its complement. All the points in R_u are called the *foreground*, and all the points in $(R_u)^c$ are the *background* of the image.

The *boundary* (*border* or *contour*) of a region R is the set of points that are adjacent to points in the complement of R . The border of a region is the set of pixels in the region that have at least one background neighbor. The points circled in the above figure (e) is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background.

3.3 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

- $D(p, q) \geq 0$ ($D(p, q) = 0$ iff $p = q$),
- $D(p, q) = D(q, p)$, and
- $D(p, z) \leq D(p, q) + D(q, z)$

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

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2} \quad (9)$$

The D_4 *distance* (called the *city-block distance*) between p and q is defined as

$$D_4(p, q) = |x - s| + |y - t| \quad (10)$$

The pixels with D_4 distance ≤ 2 from (x, y) (the center point) form the following contours of constant distance:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

The pixels with $D_4 = 1$ are the 4-neighbors of (x, y) .

The D_8 *distance* (the *chessboard distance*) between p and q is defined as

$$D_8(p, q) = \max(|x - s|, |y - t|) \quad (11)$$

The pixels with D_8 distance ≤ 2 from (x,y) (the center point) form the following contours of constant distance:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

The pixel with $D_8 = 1$ are the 8-neighbors of (x, y) .

The D_m distance for m -adjacency between two points is defined as the shortest m -path between the points.

4 An Introduction to the Mathematical Tools Used in Digital Image Processing

4.1 Array versus Matrix Operations

An *array* operation involving one or more images is carried out on a *pixel-by-pixel* basis. Consider the following 2×2 images:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The *array product* of these two images is

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

On the other hand, the *matrix product* is given by

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{12}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Assuming the array operations throughout all chapters here, unless stated otherwise.

4.2 Linear versus Nonlinear Operations

Consider a general operator H that produces an output image $g(x, y)$ for a given input image $f(x, y)$:

$$H[f(x, y)] = g(x, y) \quad (12)$$

H is a *linear operator* if

$$\begin{aligned} H[a_i f_i(x, y) + a_j f_j(x, y)] &= a_i H[f_i(x, y)] + a_j H[f_j(x, y)] \\ &= a_i g_i(x, y) + a_j g_j(x, y) \end{aligned} \quad (13)$$

where $a_i, a_j, f_i(x, y), f_j(x, y)$ are arbitrary constants and images, respectively. The linear operators have properties of *additivity* and *homogeneity*.

4.3 Arithmetic Operations

Arithmetic operations between images are array operations. The four arithmetic operations are:

$$\begin{aligned} s(x, y) &= f(x, y) + g(x, y) \\ d(x, y) &= f(x, y) - g(x, y) \\ p(x, y) &= f(x, y) \times g(x, y) \\ v(x, y) &= f(x, y) \div g(x, y) \end{aligned} \quad (14)$$

Addition for noise reduction

Let $g(x, y)$ denote a corrupted image formed by the addition of noise $\eta(x, y)$ to a noiseless image $f(x, y)$:

$$g(x, y) = f(x, y) + \eta(x, y) \quad (15)$$

assuming that the noise is uncorrelated and has zero average value at every pair of coordinates (x, y) .

Image enhancement: If the noise satisfies the constraints defined above, an image $\bar{g}(x, y)$ is formed by averaging K different noisy images:

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y) \quad (16)$$

then,

$$E\{\bar{g}(x, y)\} = f(x, y) \quad (17)$$

and

$$\sigma_{\bar{g}(x, y)}^2 = \frac{1}{K} \sigma_{\eta(x, y)}^2 \quad (18)$$

where $E\{\bar{g}(x, y)\}$ is the expected value of \bar{g} , and $\sigma_{\bar{g}(x, y)}^2$ and $\sigma_{\eta(x, y)}^2$ are the variances of \bar{g} and η , respectively, all at coordinates (x, y) . The standard deviation at any point in the average image is

$$\sigma_{\bar{g}(x, y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x, y)} \quad (19)$$

As K increases, the variances of the pixel values at each location (x, y) decreases.

Subtraction for enhancing differences

Consider image differences of the form

$$g(x, y) = f(x, y) - h(x, y) \quad (20)$$

$h(x, y)$ is the *mask*. In the medical imaging field, $h(x, y)$ is an X-ray image of region of a patient's body. $f(x, y)$ is a series of images of the same anatomical region. The net effect of subtracting the mask from each sample live image is that the areas that are different between $f(x, y)$ and $h(x, y)$ appear in the output image, $g(x, y)$, as enhanced detail.

Multiplication and Division for shading correction

Suppose that an imaging sensor producing images that can be modeled as the product of a "perfect image", denoted by $f(x, y)$, times a shading function $h(x, y)$; that is, $g(x, y) = f(x, y)h(x, y)$. If $h(x, y)$ is known, $f(x, y)$ is obtained by multiplying the sensed image by the inverse of $h(x, y)$ (i.e., dividing g by h). If $h(x, y)$ is not known, but access to the imaging system is possible, an approximation can be obtained to the shading function by imaging a target of constant intensity.

In practice, most images are displayed using 8 bits. Thus, the image values are expected to be in the range from 0 to 255. When images are saved in a standard format, such as TIFF and JPEG, conversion to this range is automatic. However, the approach used for the conversion depends on the system used. Given an image f , an approach that guarantees that the full range of an arithmetic operation between images is "captured" into a fixed number of bits is as follows. First,

$$f_m = f - \min(f) \quad (21)$$

which creates an image whose minimum value is 0. Then,

$$f_s = K[f_m / \max(f_m)] \quad (22)$$

which creates a scaled image f_s whose values are in the range $[0, K]$.

4.4 Set and Logical Operations

Basic set operations

Let A be a set composed of *ordered pairs* of real numbers. If $a = (a_1, a_2)$ is an element of A , then

$$a \in A \quad (23)$$

Similarly, if a is not an element of A ,

$$a \notin A \quad (24)$$

The set with no elements is the *null* or *empty set* and denoted by \emptyset .

If every element of a set A is also an element of a set B , then A is a subset of B ,

$$A \subseteq B \quad (25)$$

The *union* of two sets A and B ,

$$C = A \cup B \quad (26)$$

is the set of elements belonging to either A , B , or both. Similarly, the *intersection* of two sets A and B ,

$$D = A \cap B \quad (27)$$

is the set of elements belonging to both A and B . Two sets A and B are *disjoint* or *mutually exclusive* if they have no common elements,

$$A \cap B = \emptyset \quad (28)$$

The *set universe* U is the set of all elements in a given application. All set elements in a given application are members of the universe defined for that application.

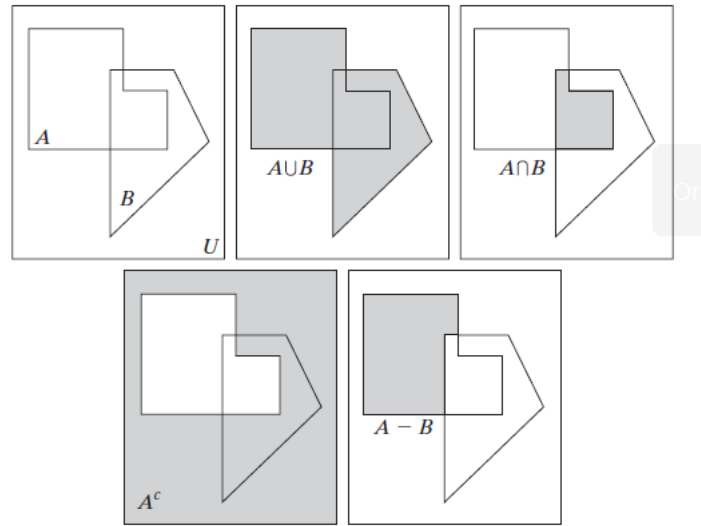
The *complement* of a set A is the set of elements that are not in A :

$$A^c = \{w | w \notin A\} \quad (29)$$

The *difference* of two sets A and B is defined as

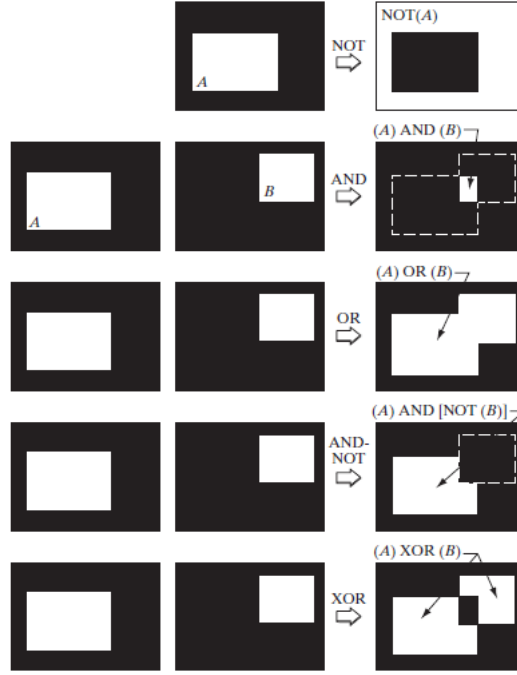
$$A - B = \{w | w \in A, w \notin B\} = A \cap B^c \quad (30)$$

The preceding concepts are shown in the figure below.



Logial operations

When dealing with binary images, the *foreground* is a set of 1-valued pixels and the *background* is a set of 0-valued pixels. Consider two regions (sets) A and B composed of foreground pixels. The following figure illustrates the logical operations of OR, AND, and NOT operations.



Fuzzy sets

The preceding set and logical results are *crisp* concepts, in the sense that elements either are or are not members of a set. The theory of *fuzzy sets* utilizes membership functions that provide a gradual transition between two extremes.

4.5 Spatial Operations

Spatial operations are performed directly on the pixels of a given image.

Single-pixel operations

To alter the values of its individual pixels based on their intensity, apply a transformation function T ,

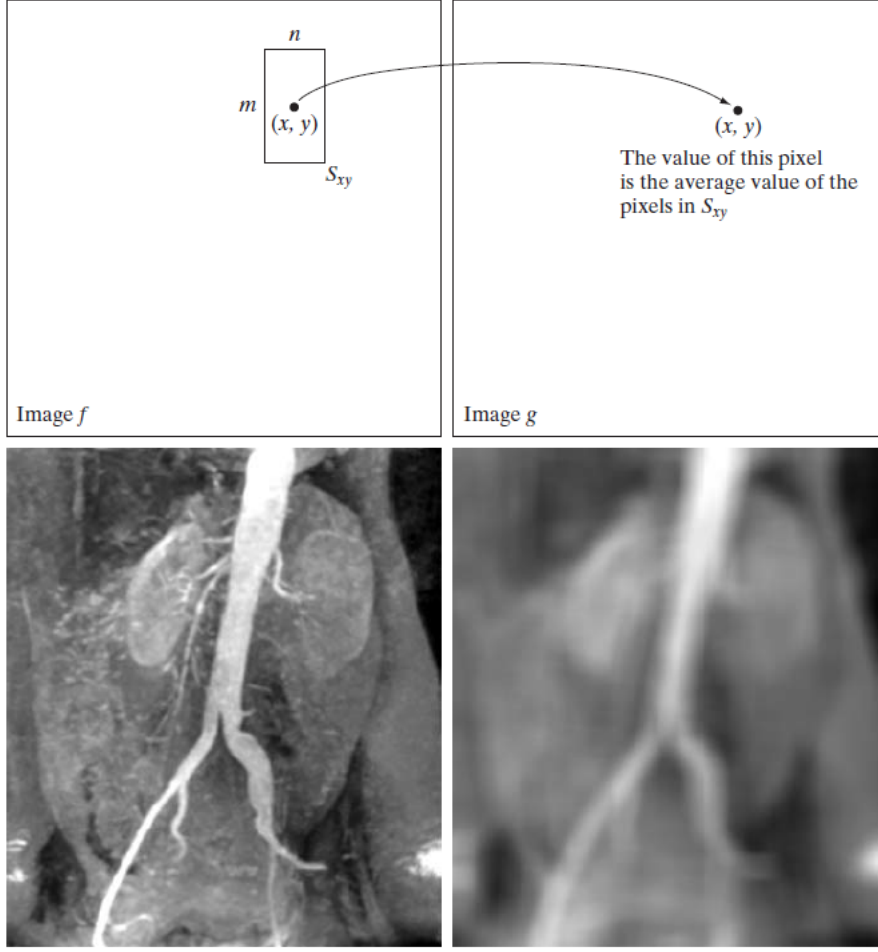
$$s = T(z) \quad (31)$$

where z is the intensity of a pixel in the original image and s is the (mapped) intensity of the corresponding pixel in the processed image.

Neighborhood operations

Let S_{xy} denote the set of coordinates of a neighborhood centered on an arbitrary point (x, y) in an image f . Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image g , such that the value of that pixel is determined by a specified operation involving the pixels in the input image with coordinates in S_{xy} .

Suppose that the specified operation is to compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on (x, y) . The locations of pixels in this region constitute the set S_{xy} , as shown below.



This can be expressed as

$$g(x, y) = \frac{1}{mn} \sum_{(r, c) \in S_{xy}} f(r, c) \quad (32)$$

where r and c are the row and column coordinates of the pixels whose coordinates are members of the set S_{xy} . Image g is created by varying the coordinates (x, y) so that the center of the neighborhood moves from pixel to pixel in image f , and repeating the neighborhood operation at each new location. The net effect is to perform local blurring in the original image.

Geometric spatial transformations and image registration

A geometric transformation consists of two basic operations:

- a spatial transformation of coordinates and
- intensity interpolation that assigns intensity values to the spatially transformed pixels.

The transformation of coordinates may be expressed as

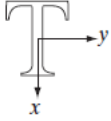

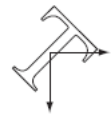
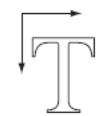
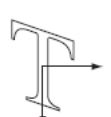

$$(x, y) = T\{(v, w)\} \quad (33)$$

where (v, w) are pixel coordinates in the original image and (x, y) are the corresponding pixel coordinates in the transformed image.

The *affine transform* has the general form

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} v & w & 1 \end{bmatrix} \mathbf{T} = \begin{bmatrix} v & w & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \quad (34)$$

This transformation can scale, rotate, translate, or sheer a set of coordinate points, depending on the value chosen for the elements of matrix \mathbf{T} shown below.

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v \cos \theta - w \sin \theta$ $y = v \sin \theta + w \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Equation 34 can be used in two ways. The first is a *forward mapping*, consisting of scanning the pixels of the input image and, at each location (v, w) , computing the spatial location (x, y) of the corresponding pixel in the output image using Equation 34 directly. The second approach, called *inverse mapping*, scans the output pixel locations and, at each location (x, y) , computes the corresponding location in the input image using $(v, w) = T^{-1}(x, y)$. Then it interpolates among the nearest input pixels to determine the intensity of the output pixel value.

Image registration is an important application of digital image processing used to align two or more images of the same scene. In image registration, the specific transformation in the preceding section is unknown. The problem is to estimate the transformation function and then use it to register the two images. The input image is the image to be transformed. The *reference* image is the image against which is registered the input.

It may be of interest to align (register) two or more images taken at approximately the same time, but using different imaging systems, such as an MRI (magnetic resonance imaging) scanner and a PET (positron emission tomography) scanner.

One of the principal approaches for solving the problem is to use *tie points* (also called *control points*), which are corresponding points whose locations are known precisely in the input and reference images.

Suppose a set of four tie points is obtained, each in an input and a reference image. A simple model based on a bilinear approximation is given by

$$x = c_1 v + v_2 w + c_3 v w + c_4 \quad (35)$$

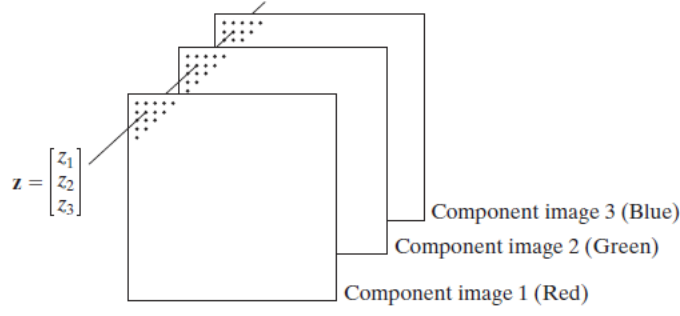
and

$$y = c_5 v + c_6 w + c_7 v w + c_8 \quad (36)$$

where, during the estimation phase, (v, w) and (x, y) are the coordinates of tie points in the input and reference images, respectively. If four pairs of corresponding tie points in both images are available, the equations above can be used to solve for the unknown coefficients, c_1, c_2, \dots, c_8 .

4.6 Vector and Matrix Operations

Multispectral image processing is a typical area in which vector and matrix operations are used routinely. The color images shown below are the examples.



Each pixel of an RGB image has three components, which can be organized in the form of a *column vector*

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \quad (37)$$

where z_1 is the intensity of the pixel in the red image, and the other two elements are the corresponding pixel intensities in the green and blue images, respectively. Thus an RGB color image of size $M \times N$ can be represented by three component images of this size, or by a total of MN 3-D vectors.

Once pixels have been represented as vectors, an *Euclidean distance* D , between a pixel vector \mathbf{z} and an arbitrary point \mathbf{a} in n -dimensional space is defined as the vector product

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= [(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})]^{1/2} \\ &= [(z_1 - a_1)^2 + (z_2 - a_2)^2 + \dots + (z_n - a_n)^2]^{1/2} \end{aligned} \quad (38)$$

The equation above is referred to as a *vector norm*, denoted by $\|\mathbf{z} - \mathbf{a}\|$.

The linear transformations can be represented as

$$\mathbf{w} = \mathbf{A}(\mathbf{z} - \mathbf{a}) \quad (39)$$

where \mathbf{A} is a matrix of size $m \times n$ and \mathbf{z} and \mathbf{a} are column vectors of size $n \times 1$.

An image of size $M \times N$ can be expressed as a vector of dimension $MN \times 1$ by letting the first row of the image be the first N elements of the vectors, second row the next N elements, and so on. A broad range of linear processes applied to an image by using

$$\mathbf{g} = \mathbf{H}\mathbf{f} + \mathbf{n} \quad (40)$$

where \mathbf{f} is an $MN \times 1$ vector representing an input image, \mathbf{n} is an $MN \times 1$ vector representing an $M \times N$ noise pattern, \mathbf{g} is an $MN \times 1$ vector representing a process image, and \mathbf{H} is an $MN \times MN$ matrix representing a linear process applied to the input image.

4.7 Image Transforms

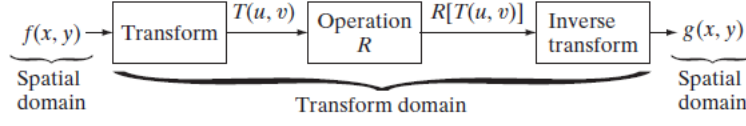
Image processing tasks are, in some cases, best formulated by transforming the input images, carrying the specified task in a *transform domain*, and applying the inverse transform to return to the spatial domain. A particularly important class of 2-D linear transform, denoted $T(u, v)$, can be expressed in the general form

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v) \quad (41)$$

where $f(x, y)$ is the input image, $r(x, y, u, v)$ is called the *forward transformation kernel*, and for $u = 0, 1, 2, \dots, M-1$ and $v = 0, 1, 2, \dots, N-1$. x and y are spatial variables, while M and N are the row and column dimensions of f . u and v are the *transform variables*. $T(u, v)$ is called the *forward transform* of $f(x, y)$. Given $T(u, v)$, $f(x, y)$ is recovered using the *inverse transform* of $T(u, v)$,

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) s(x, y, u, v) \quad (42)$$

for $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$, where $s(x, y, u, v)$ is the *inverse transformation kernel*. Then, Equations 41 and 42 are called a *transform pair*.



The forward transformation kernel is *separable* if

$$r(x, y, u, v) = r_1(x, u)r_2(y, v) \quad (43)$$

The kernel is *symmetric* if $r_1(x, y)$ is functionally equal to $r_2(x, y)$, so that

$$r(x, y, u, v) = r_1(x, u)r_1(y, v) \quad (44)$$

Identical comments apply to the inverse kernel by replacing r with s in the preceding equations.

The 2-D Fourier transform has the following forward and inverse kernels:

$$r(x, y, u, v) = e^{-j2\pi(ux/M+vy/N)} \quad (45)$$

and

$$s(x, y, u, v) = \frac{1}{MN} e^{j2\pi(ux/M+vy/N)} \quad (46)$$

respectively, where $j = \sqrt{-1}$. Substituting these into Equations 41 and 42 gives the *discrete Fourier transform pair*:

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)} \quad (47)$$

and

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) e^{j2\pi(ux/M+vy/N)} \quad (48)$$

The Fourier kernels are separable and symmetric, and the separable and symmetric kernels allow 2-D transforms to be computed using 1-D transforms. When the forward and inverse kernels of a transform pair satisfy these two conditions, and $f(x, y)$ is square image of size $M \times M$, Equations 41 and 42 can be expressed in matrix form:

$$\mathbf{T} = \mathbf{AFA} \quad (49)$$

where \mathbf{F} is an $M \times M$ matrix containing the elements of $f(x, y)$, \mathbf{A} is an $M \times M$ matrix with elements $a_{ij} = r_1(i, j)$, and \mathbf{T} is the resulting $M \times M$ transform, with values $T(u, v)$ for $u, v = 0, 1, 2, \dots, M-1$.

To obtain the inverse transform, Equation 49 is transformed by an inverse transformation matrix \mathbf{B} :

$$\mathbf{BTB} = \mathbf{BAFAB} \quad (50)$$

If $\mathbf{B} = \mathbf{A}^{-1}$,

$$\mathbf{F} = \mathbf{BTB} \quad (51)$$

indicating that \mathbf{F} (whose elements are equal to image $f(x, y)$) can be recovered completely from its forward transform. If \mathbf{B} is not equal to \mathbf{A}^{-1} , then an approximation is applied:

$$\hat{\mathbf{F}} = \mathbf{BAFAB} \quad (52)$$

4.8 Probabilistic Methods

Let $z_i, i = 0, 1, 2, \dots, L-1$, denote the values of all possible intensities in an $M \times N$ digital image. The probability $p(z_k)$ of intensity level z_k occurring in a given image is estimated as

$$p(z_k) = \frac{n_k}{MN} \quad (53)$$

where n_k is the number of times that intensity z_k occurs in the image and MN is the total number of pixels. Clearly,

$$\sum_{k=0}^{L-1} p(z_k) = 1 \quad (54)$$

Once $p(z_k)$ is obtained, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k p(z_k) \quad (55)$$

The variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \quad (56)$$

The variance is a measure of the spread of the values of z about the mean, so it is a useful measure of image contrast. In general, the n th moment of random variable z about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \quad (57)$$