## Single-Pixel Operations

The simplest operation we perform on a digital image is to alter the intensity of its pixels individually using a transformation function, $T$, of the form:

$$s = T(z) \tag{2-42}$$

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. For example, Fig. 2.38 shows the transformation used to obtain the *negative* (sometimes called the *complement*) of an 8-bit image. This transformation could be used, for example, to obtain the negative image in Fig. 2.36, instead of using sets.

Our use of the word "negative" in this context refers to the digital equivalent of a photographic negative, not to the numerical negative of the pixels in the image.

## Neighborhood Operations

Let $S_{xy}$ denote the set of coordinates of a neighborhood (see Section 2.5 regarding neighborhoods) 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 on the neighborhood of pixels in the input image with coordinates in the set $S_{xy}$. For example, 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 coordinates of pixels in this region are the elements of set $S_{xy}$. Figures 2.39(a) and (b) illustrate the process. We can express this averaging operation as

$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c) \tag{2-43}$$

where $r$ and $c$ are the row and column coordinates of the pixels whose coordinates are in 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 then repeating the neighborhood operation at each new location. For instance, the image in Fig. 2.39(d) was created in this manner using a neighborhood of size $41 \times 41$. The

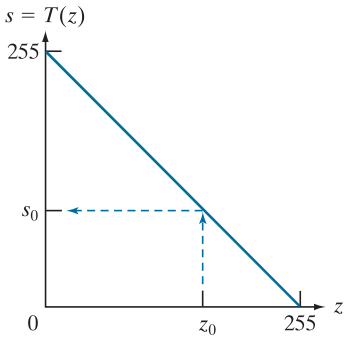
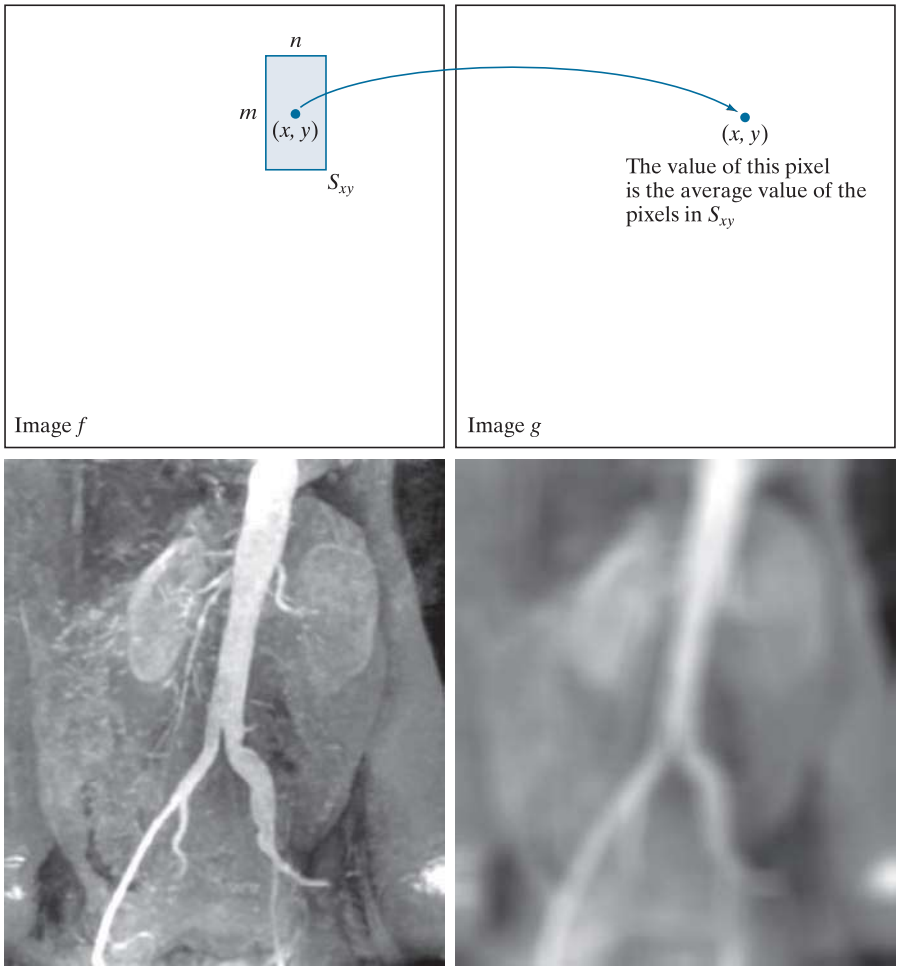



**FIGURE 2.38** Intensity transformation function used to obtain the digital equivalent of photographic negative of an 8-bit image..




a b
c d

**FIGURE 2.39** Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood. (c) An aortic angiogram (see Section 1.3). (d) The result of using Eq. (2-43) with $m = n = 41$. The images are of size $790 \times 686$ pixels. (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

net effect is to perform local blurring in the original image. This type of process is used, for example, to eliminate small details and thus render "blobs" corresponding to the largest regions of an image. We will discuss neighborhood processing in Chapters 3 and 5, and in several other places in the book.

## Geometric Transformations

We use geometric transformations modify the spatial arrangement of pixels in an image. These transformations are called *rubber-sheet transformations* because they may be viewed as analogous to "printing" an image on a rubber sheet, then stretching or shrinking the sheet according to a predefined set of rules. Geometric transformations of digital images consist of two basic operations:

1. Spatial transformation of coordinates.
2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

The transformation of coordinates may be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2-44}$$

where $(x, y)$ are pixel coordinates in the original image and $(x', y')$ are the corresponding pixel coordinates of the transformed image. For example, the transformation $(x', y') = (x/2, y/2)$ shrinks the original image to half its size in both spatial directions.

Our interest is in so-called *affine transformations*, which include scaling, translation, rotation, and shearing. The key characteristic of an affine transformation in 2-D is that it preserves points, straight lines, and planes. Equation (2-44) can be used to express the transformations just mentioned, except translation, which would require that a constant 2-D vector be added to the right side of the equation. However, it is possible to use homogeneous coordinates to express all four affine transformations using a single $3 \times 3$ matrix in the following general form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2-45}$$

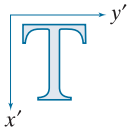
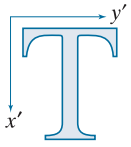
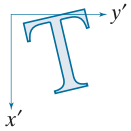
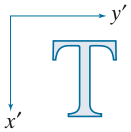
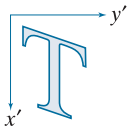
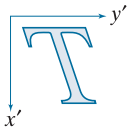
This transformation can *scale*, *rotate*, *translate*, or *sheer* an image, depending on the values chosen for the elements of matrix **A**. Table 2.3 shows the matrix values used to implement these transformations. A significant advantage of being able to perform all transformations using the unified representation in Eq. (2-45) is that it provides the framework for concatenating a sequence of operations. For example, if we want to resize an image, rotate it, and move the result to some location, we simply form a $3 \times 3$ matrix equal to the product of the scaling, rotation, and translation matrices from Table 2.3 (see Problems 2.36 and 2.37).

The preceding transformation moves the coordinates of pixels in an image to new locations. To complete the process, we have to assign intensity values to those locations. This task is accomplished using *intensity interpolation*. We already discussed this topic in Section 2.4. We began that discussion with an example of zooming an image and discussed the issue of intensity assignment to new pixel locations. Zooming is simply scaling, as detailed in the second row of Table 2.3, and an analysis similar to the one we developed for zooming is applicable to the problem of assigning intensity values to the relocated pixels resulting from the other transformations in Table 2.3. As in Section 2.4, we consider nearest neighbor, bilinear, and bicubic interpolation techniques when working with these transformations.

We can use Eq. (2-45) in two basic ways. The first, is a *forward mapping*, which consists of scanning the pixels of the input image and, at each location $(x, y)$, com-

puting the spatial location $(x', y')$ of the corresponding pixel in the output image using Eq. (2-45) directly. A problem with the forward mapping approach is that two or more pixels in the input image can be transformed to the same location in the output image, raising the question of how to combine multiple output values into a single output pixel value. In addition, it is possible that some output locations may not be assigned a pixel at all. 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 $(x, y) = A^{-1}(x', y')$. It then interpolates (using one of the techniques discussed in Section 2.4) among the nearest input pixels to determine the intensity of the output pixel value. Inverse mappings are more efficient to implement than forward mappings, and are used in numerous commercial implementations of spatial transformations (for example, MATLAB uses this approach).

**TABLE 2.3** Affine transformations based on Eq. (2-45).

| Transformation Name | Affine Matrix, A | Coordinate Equations | Example |
|---|---|---|---|
| Identity | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = x$<br>$y' = y$ |  |
| Scaling/Reflection (For reflection, set one scaling factor to −1 and the other to 0) | $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = c_x x$<br>$y' = c_y y$ |  |
| Rotation (about the origin) | $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = x\cos\theta - y\sin\theta$<br>$y' = x\sin\theta + y\cos\theta$ |  |
| Translation | $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = x + t_x$<br>$y' = y + t_y$ |  |
| Shear (vertical) | $\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = x + s_v y$<br>$y' = y$ |  |
| Shear (horizontal) | $\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x' = x$<br>$y' = s_h x + y$ |  |

**EXAMPLE 2.9: Image rotation and intensity interpolation.**

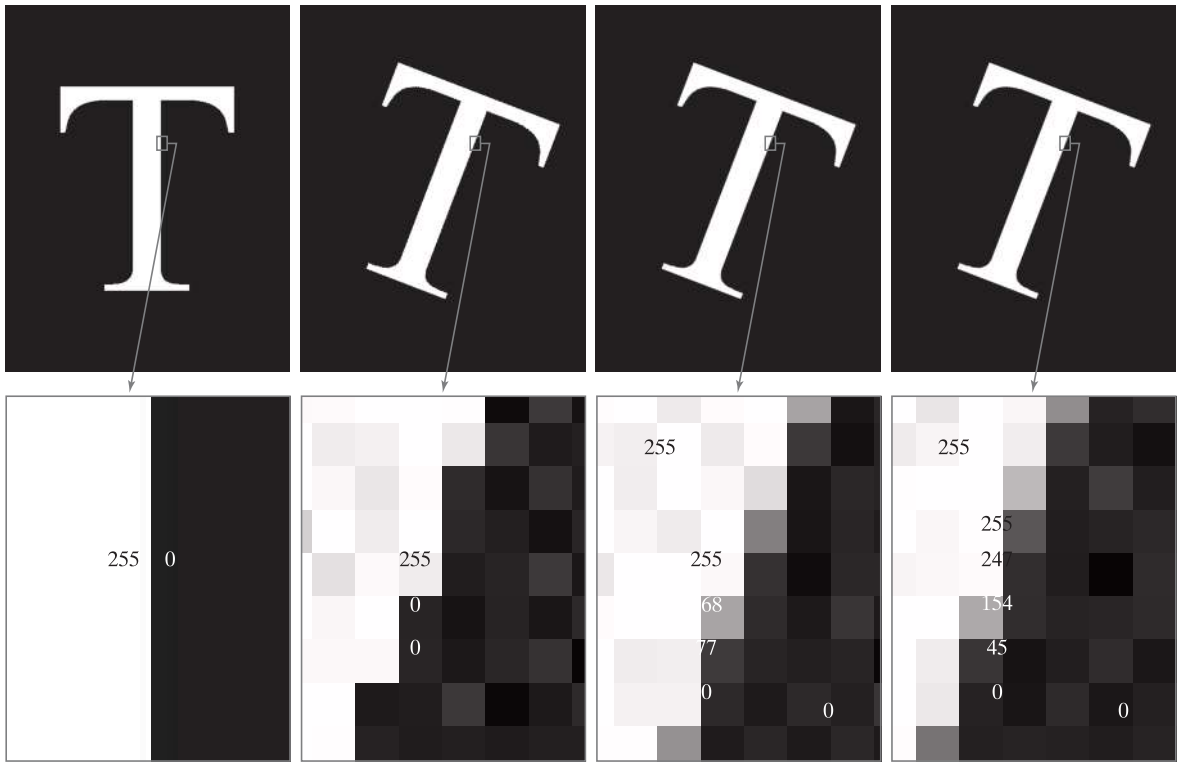
The objective of this example is to illustrate image rotation using an affine transform. Figure 2.40(a) shows a simple image and Figs. 2.40(b)–(d) are the results (using inverse mapping) of rotating the original image by –21° (in Table 2.3, clockwise angles of rotation are negative). Intensity assignments were computed using nearest neighbor, bilinear, and bicubic interpolation, respectively. A key issue in image rotation is the preservation of straight-line features. As you can see in the enlarged edge sections in Figs. 2.40(f) through (h), nearest neighbor interpolation produced the most jagged edges and, as in Section 2.4, bilinear interpolation yielded significantly improved results. As before, using bicubic interpolation produced slightly better results. In fact, if you compare the progression of enlarged detail in Figs. 2.40(f) to (h), you can see that the transition from white (255) to black (0) is smoother in the last figure because the edge region has more values, and the distribution of those values is better balanced. Although the small intensity differences resulting from bilinear and bicubic interpolation are not always noticeable in human visual analysis, they can be important in processing image data, such as in automated edge following in rotated images.

The size of the spatial rectangle needed to contain a rotated image is larger than the rectangle of the original image, as Figs. 2.41(a) and (b) illustrate. We have two options for dealing with this: (1) we can crop the rotated image so that its size is equal to the size of the original image, as in Fig. 2.41(c), or we can keep the larger image containing the full rotated original, an Fig. 2.41(d). We used the first option in Fig. 2.40 because the rotation did not cause the object of interest to lie outside the bounds of the original rectangle. The areas in the rotated image that do not contain image data must be filled with some value, 0 (black) being the most common. Note that counterclockwise angles of rotation are considered positive. This is a result of the way in which our image coordinate system is set up (see Fig. 2.19), and the way in which rotation is defined in Table 2.3.

### Image Registration

Image registration is an important application of digital image processing used to align two or more images of the same scene. In image registration, we have available an *input* image and a *reference* image. The objective is to transform the input image geometrically to produce an output image that is aligned (registered) with the reference image. Unlike the discussion in the previous section where transformation functions are known, the geometric transformation needed to produce the output, registered image generally is not known, and must be estimated.

Examples of image registration include aligning 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. Or, perhaps the images were taken at different times using the same instruments, such as satellite images of a given location taken several days, months, or even years apart. In either case, combining the images or performing quantitative analysis and comparisons between them requires compensating for geometric distortions caused by differences in viewing angle, distance, orientation, sensor resolution, shifts in object location, and other factors.




a b c d
e f g h

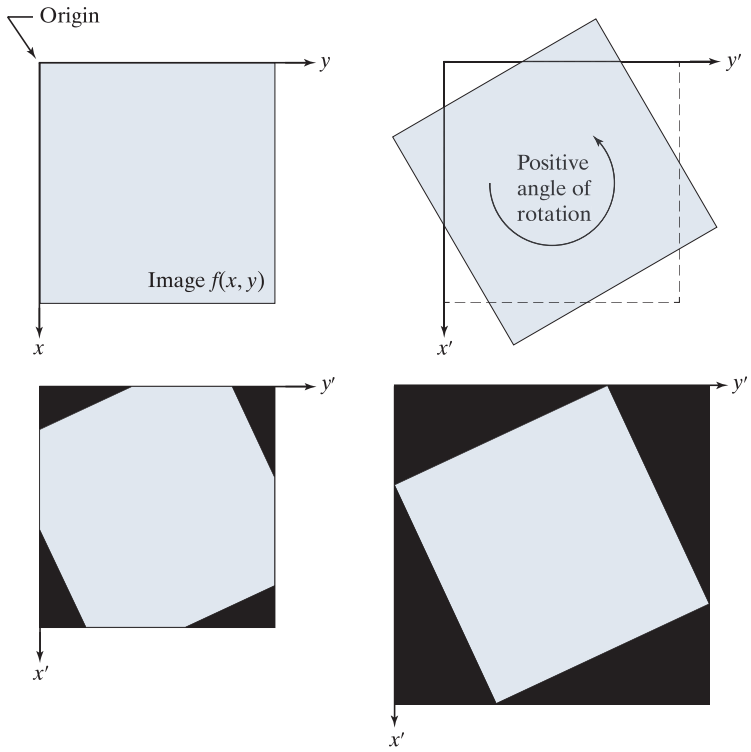
**FIGURE 2.40** (a) A $541 \times 421$ image of the letter T. (b) Image rotated –21° using nearest-neighbor interpolation for intensity assignments. (c) Image rotated –21° using bilinear interpolation. (d) Image rotated –21° using bicubic interpolation. (e)-(h) Zoomed sections (each square is one pixel, and the numbers shown are intensity values).

One of the principal approaches for solving the problem just discussed is to use *tie points* (also called *control points*). These are corresponding points whose locations are known precisely in the input and reference images. Approaches for selecting tie points range from selecting them interactively to using algorithms that detect these points automatically. Some imaging systems have physical artifacts (such as small metallic objects) embedded in the imaging sensors. These produce a set of known points (called *reseau marks* or *fiducial marks*) directly on all images captured by the system. These known points can then be used as guides for establishing tie points.

The problem of estimating the transformation function is one of modeling. For example, suppose that we have a set of four tie points each in an input and a reference image. A simple model based on a bilinear approximation is given by

$$x = c_1 v + c_2 w + c_3 v w + c_4 \tag{2-46}$$

and




a b
c d

**FIGURE 2.41**
(a) A digital image.
(b) Rotated image (note the counterclockwise direction for a positive angle of rotation).
(c) Rotated image cropped to fit the same area as the original image.
(d) Image enlarged to accommodate the entire rotated image.

$$y = c_5 v + c_6 w + c_7 vw + c_8 \tag{2-47}$$

During the estimation phase, $(v, w)$ and $(x, y)$ are the coordinates of tie points in the input and reference images, respectively. If we have four pairs of corresponding tie points in both images, we can write eight equations using Eqs. (2-46) and (2-47) and use them to solve for the eight unknown coefficients, $c_1$ through $c_8$.

Once we have the coefficients, Eqs. (2-46) and (2-47) become our vehicle for transforming all the pixels in the input image. The result is the desired registered image. After the coefficients have been computed, we let $(v, w)$ denote the coordinates of each pixel in the input image, and $(x, y)$ become the corresponding coordinates of the output image. The same set of coefficients, $c_1$ through $c_8$, are used in computing all coordinates $(x, y)$; we just step through all $(v, w)$ in the input image to generate the corresponding $(x, y)$ in the output, registered image. If the tie points were selected correctly, this new image should be registered with the reference image, within the accuracy of the bilinear approximation model.

In situations where four tie points are insufficient to obtain satisfactory registration, an approach used frequently is to select a larger number of tie points and then treat the quadrilaterals formed by groups of four tie points as subimages. The subimages are processed as above, with all the pixels within a quadrilateral being transformed using the coefficients determined from the tie points corresponding to that quadrilateral. Then we move to another set of four tie points and repeat the

procedure until all quadrilateral regions have been processed. It is possible to use more complex regions than quadrilaterals, and to employ more complex models, such as polynomials fitted by least squares algorithms. The number of control points and sophistication of the model required to solve a problem is dependent on the severity of the geometric distortion. Finally, keep in mind that the transformations defined by Eqs. (2-46) and (2-47), or any other model for that matter, only map the spatial coordinates of the pixels in the input image. We still need to perform intensity interpolation using any of the methods discussed previously to assign intensity values to the transformed pixels.

**EXAMPLE 2.10: Image registration.**

Figure 2.42(a) shows a reference image and Fig. 2.42(b) shows the same image, but distorted geometrically by vertical and horizontal shear. Our objective is to use the reference image to obtain tie points and then use them to register the images. The tie points we selected (manually) are shown as small white squares near the corners of the images (we needed only four tie points because the distortion is linear shear in both directions). Figure 2.42(c) shows the registration result obtained using these tie points in the procedure discussed in the preceding paragraphs. Observe that registration was not perfect, as is evident by the black edges in Fig. 2.42(c). The difference image in Fig. 2.42(d) shows more clearly the slight lack of registration between the reference and corrected images. The reason for the discrepancies is error in the manual selection of the tie points. It is difficult to achieve perfect matches for tie points when distortion is so severe.

## VECTOR AND MATRIX OPERATIONS

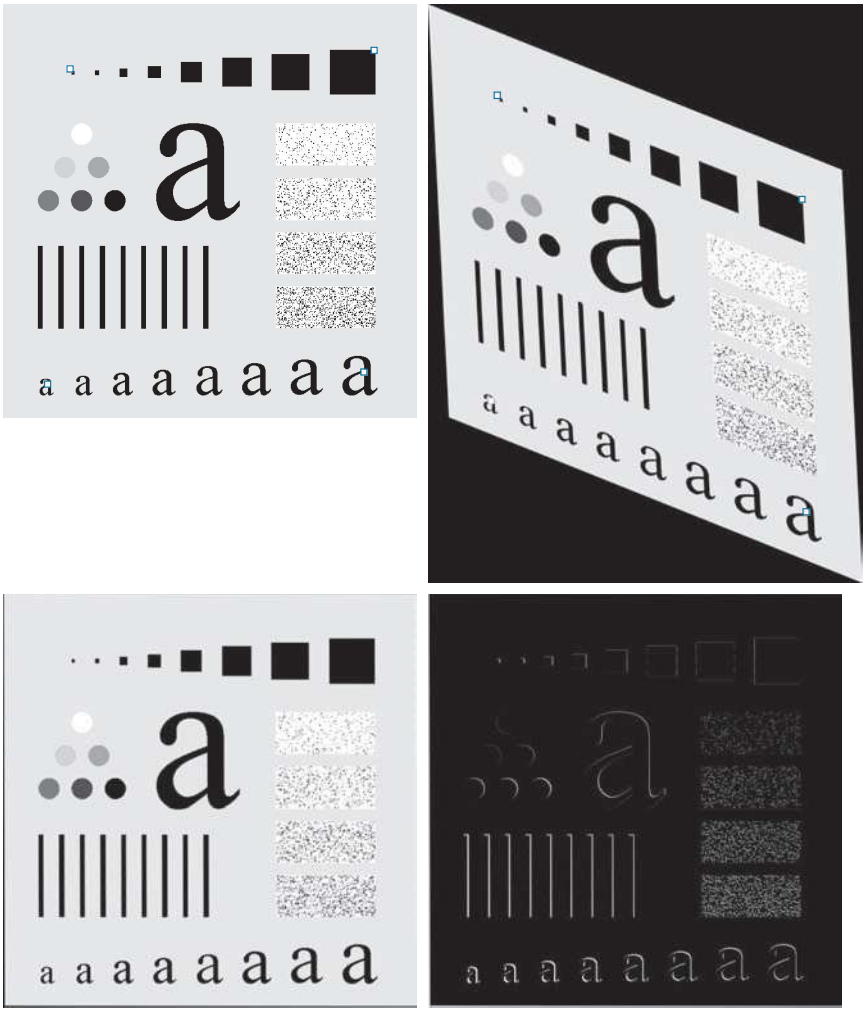
Multispectral image processing is a typical area in which vector and matrix operations are used routinely. For example, you will learn in Chapter 6 that color images are formed in RGB color space by using red, green, and blue component images, as Fig. 2.43 illustrates. Here we see that *each* pixel of an RGB image has three components, which can be organized in the form of a column vector

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2-48}$$

where $z_1$ is the intensity of the pixel in the red image, and $z_2$ and $z_3$ 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 $MN$ vectors of size $3 \times 1$. A general multispectral case involving $n$ component images (e.g., see Fig. 1.10) will result in $n$-dimensional vectors:

Recall that an $n$-dimensional vector can be thought of as a point in $n$-dimensional Euclidean space.

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \tag{2-49}$$



a b
c d

**FIGURE 2.42** Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered (output) image (note the errors in the border). (d) Difference between (a) and (c), showing more registration errors.

We will use this type of vector representation throughout the book.

The product $\mathbf{ab}^T$ is called the *outer product* of **a** and **b**. It is a matrix of size $n \times n$.

The *inner product* (also called the *dot product*) of two $n$-dimensional column vectors **a** and **b** is defined as

$$\begin{aligned}\mathbf{a} \bullet \mathbf{b} &\triangleq \mathbf{a}^T\mathbf{b} \\ &= a_1b_1 + a_2b_2 + \cdots + a_nb_n \\ &= \sum_{i=1}^{n} a_i b_i\end{aligned} \tag{2-50}$$

where $T$ indicates the transpose. The *Euclidean vector norm*, denoted by $\|\mathbf{z}\|$, is defined as the square root of the inner product:

$$\|\mathbf{z}\| = \left(\mathbf{z}^T\mathbf{z}\right)^{\frac{1}{2}} \tag{2-51}$$

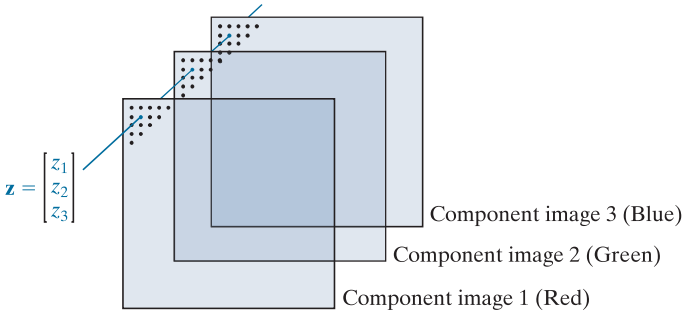



**FIGURE 2.43** Forming a vector from corresponding pixel values in three RGB component images.

We recognize this expression as the length of vector $\mathbf{z}$.

We can use vector notation to express several of the concepts discussed earlier. For example, the Euclidean distance, $D(\mathbf{z}, \mathbf{a})$, between points (vectors) $\mathbf{z}$ and $\mathbf{a}$ in $n$-dimensional space is defined as the Euclidean vector norm:

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= \|\mathbf{z} - \mathbf{a}\| = \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}} \\ &= \left[(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots + (z_n - a_n)^2\right]^{\frac{1}{2}} \end{aligned} \tag{2-52}$$

This is a generalization of the 2-D Euclidean distance defined in Eq. (2-19).

Another advantage of pixel vectors is in linear transformations, represented as

$$\mathbf{w} = \mathbf{A}(\mathbf{z} - \mathbf{a}) \tag{2-53}$$

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

As noted in Eq. (2-10), entire images can be treated as matrices (or, equivalently, as vectors), a fact that has important implication in the solution of numerous image processing problems. For example, we can express an image of size $M \times N$ as a column vector of dimension $MN \times 1$ by letting the first $M$ elements of the vector equal the first column of the image, the next $M$ elements equal the second column, and so on. With images formed in this manner, we can express a broad range of linear processes applied to an image by using the notation

$$\mathbf{g} = \mathbf{Hf} + \mathbf{n} \tag{2-54}$$

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 processed image, and $\mathbf{H}$ is an $MN \times MN$ matrix representing a linear process applied to the input image (see the discussion earlier in this chapter regarding linear processes). It is possible, for example, to develop an entire body of generalized techniques for image restoration starting with Eq. (2-54), as we discuss in Section 5.9. We will mention the use of matrices again in the following section, and show other uses of matrices for image processing in numerous chapters in the book.

## IMAGE TRANSFORMS

All the image processing approaches discussed thus far operate directly on the pixels of an input image; that is, they work directly in the spatial domain. In some cases, image processing tasks are 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. You will encounter a number of different transforms as you proceed through the book. A particularly important class of 2-D *linear transforms*, denoted $T(u,v)$, can be expressed in the general form

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)r(x,y,u,v) \tag{2-55}$$

where $f(x,y)$ is an input image, $r(x,y,u,v)$ is called a *forward transformation kernel*, and Eq. (2-55) is evaluated for $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$. As before, $x$ and $y$ are spatial variables, while $M$ and $N$ are the row and column dimensions of $f$. Variables $u$ and $v$ are called the *transform variables*. $T(u,v)$ is called the *forward transform* of $f(x,y)$. Given $T(u,v)$, we can recover $f(x,y)$ using the *inverse transform* of $T(u,v)$:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)s(x,y,u,v) \tag{2-56}$$

for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$, where $s(x,y,u,v)$ is called an *inverse transformation kernel*. Together, Eqs. (2-55) and (2-56) are called a *transform pair*.

Figure 2.44 shows the basic steps for performing image processing in the linear transform domain. First, the input image is transformed, the transform is then modified by a predefined operation and, finally, the output image is obtained by computing the inverse of the modified transform. Thus, we see that the process goes from the spatial domain to the transform domain, and then back to the spatial domain.

The forward transformation kernel is said to be *separable* if

$$r(x,y,u,v) = r_1(x,u)r_2(y,v) \tag{2-57}$$

In addition, the kernel is said to be *symmetric* if $r_1(x,u)$ is functionally equal to $r_2(y,v)$, so that

$$r(x,y,u,v) = r_1(x,u)r_1(y,v) \tag{2-58}$$

Identical comments apply to the inverse kernel.

The nature of a transform is determined by its kernel. A transform of particular importance in digital image processing is the *Fourier transform*, which has the following forward and inverse kernels:

$$r(x,y,u,v) = e^{-j2\pi(ux/M + vy/N)} \tag{2-59}$$

and

$$s(x,y,u,v) = \frac{1}{MN}e^{j2\pi(ux/M + vy/N)} \tag{2-60}$$

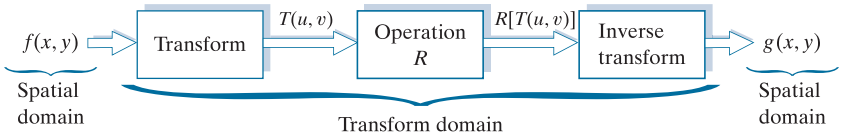



**FIGURE 2.44** General approach for working in the linear transform domain.

respectively, where $j = \sqrt{-1}$, so these kernels are complex functions. Substituting the preceding kernels into the general transform formulations in Eqs. (2-55) and (2-56) gives us the *discrete Fourier transform pair*:

The exponential terms in the Fourier transform kernels can be expanded as sines and cosines of various frequencies. As a result, the domain of the Fourier transform is called the *frequency domain*.

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M + vy/N)} \tag{2-61}$$

and

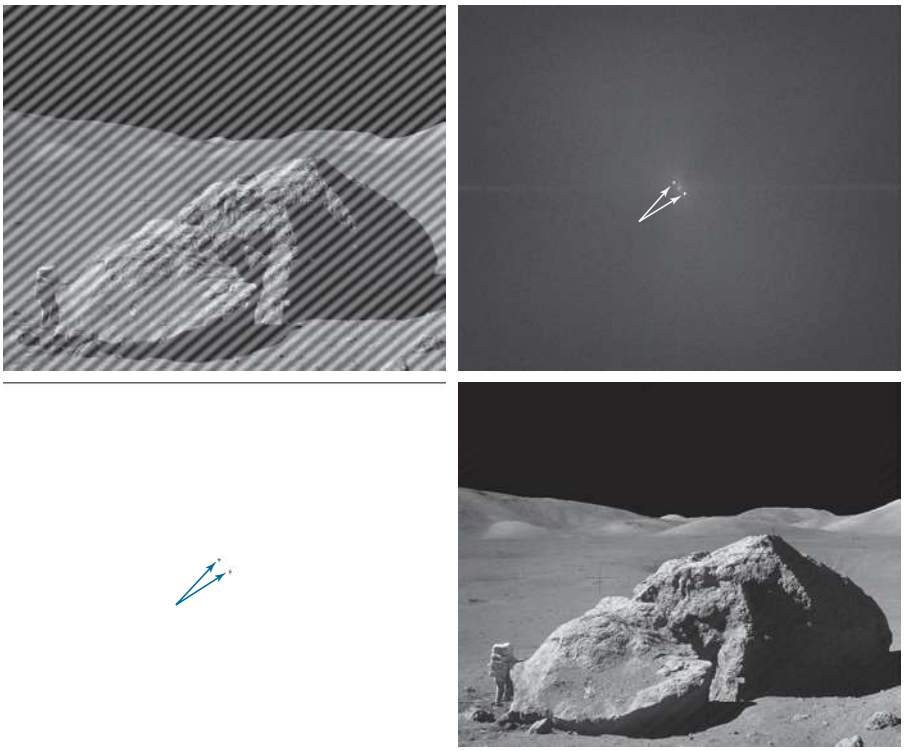
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)e^{j2\pi(ux/M + vy/N)} \tag{2-62}$$

It can be shown that the Fourier kernels are separable and symmetric (Problem 2.39), and that separable and symmetric kernels allow 2-D transforms to be computed using 1-D transforms (see Problem 2.40). The preceding two equations are of fundamental importance in digital image processing, as you will see in Chapters 4 and 5.

**EXAMPLE 2.11: Image processing in the transform domain.**

Figure 2.45(a) shows an image corrupted by periodic (sinusoidal) interference. This type of interference can be caused, for example, by a malfunctioning imaging system; we will discuss it in Chapter 5. In the spatial domain, the interference appears as waves of intensity. In the frequency domain, the interference manifests itself as bright bursts of intensity, whose location is determined by the frequency of the sinusoidal interference (we will discuss these concepts in much more detail in Chapters 4 and 5). Typically, the bursts are easily observable in an image of the magnitude of the Fourier transform, $|T(u,v)|$. With reference to the diagram in Fig. 2.44, the corrupted image is $f(x,y)$, the transform in the leftmost box is the Fourier transform, and Fig. 2.45(b) is $|T(u,v)|$ displayed as an image. The bright dots shown are the bursts of intensity mentioned above. Figure 2.45(c) shows a mask image (called a *filter*) with white and black representing 1 and 0, respectively. For this example, the operation in the second box of Fig. 2.44 is to multiply the filter by the transform to remove the bursts associated with the interference. Figure 2.45(d) shows the final result, obtained by computing the inverse of the modified transform. The interference is no longer visible, and previously unseen image detail is now made quite clear. Observe, for example, the fiducial marks (faint crosses) that are used for image registration, as discussed earlier.

When the forward and inverse kernels of a transform are separable and symmetric, and $f(x,y)$ is a square image of size $M \times M$, Eqs. (2-55) and (2-56) can be expressed in matrix form:



a b
c d

**FIGURE 2.45**
(a) Image corrupted by sinusoidal interference. (b) Magnitude of the Fourier transform showing the bursts of energy caused by the interference (the bursts were enlarged for display purposes). (c) Mask used to eliminate the energy bursts. (d) Result of computing the inverse of the modified Fourier transform. (Original image courtesy of NASA.)

$$\mathbf{T} = \mathbf{AFA} \tag{2-63}$$

where $\mathbf{F}$ is an $M \times M$ matrix containing the elements of $f(x, y)$ [see Eq. (2-9)], $\mathbf{A}$ is an $M \times M$ matrix with elements $a_{ij} = r_1(i, j)$, and $\mathbf{T}$ is an $M \times M$ transform matrix with elements $T(u, v)$, for $u, v = 0, 1, 2, \ldots, M - 1$.

To obtain the inverse transform, we pre- and post-multiply Eq. (2-63) by an inverse transformation matrix $\mathbf{B}$:

$$\mathbf{BTB} = \mathbf{BAFAB} \tag{2-64}$$

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

$$\mathbf{F} = \mathbf{BTB} \tag{2-65}$$

indicating that $\mathbf{F}$ or, equivalently, $f(x, y)$, can be recovered completely from its forward transform. If $\mathbf{B}$ is not equal to $\mathbf{A}^{-1}$, Eq. (2-65) yields an approximation:

$$\hat{\mathbf{F}} = \mathbf{BAFAB} \tag{2-66}$$

In addition to the Fourier transform, a number of important transforms, including the *Walsh*, *Hadamard*, *discrete cosine*, *Haar*, and *slant* transforms, can be expressed in the form of Eqs. (2-55) and (2-56), or, equivalently, in the form of Eqs. (2-63) and (2-65). We will discuss these and other types of image transforms in later chapters.

## IMAGE INTENSITIES AS RANDOM VARIABLES

You may find it useful to consult the tutorials section in the book website for a brief review of probability.

We treat image intensities as random quantities in numerous places in the book. For example, let $z_i, i = 0,1,2,\ldots,L-1$, 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 the image is estimated as

$$p(z_k) = \frac{n_k}{MN} \tag{2-67}$$

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,

$$\sum_{k=0}^{L-1} p(z_k) = 1 \tag{2-68}$$

Once we have $p(z_k)$, we can determine a number of important image characteristics. For example, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k p(z_k) \tag{2-69}$$

Similarly, the variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \tag{2-70}$$

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 central moment of random variable $z$ about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \tag{2-71}$$

We see that $\mu_0(z) = 1$, $\mu_1(z) = 0$, and $\mu_2(z) = \sigma^2$. Whereas the mean and variance have an immediately obvious relationship to visual properties of an image, higher-order moments are more subtle. For example, a positive third moment indicates that the intensities are biased to values higher than the mean, a negative third moment would indicate the opposite condition, and a zero third moment would tell us that the intensities are distributed approximately equally on both sides of the mean. These features are useful for computational purposes, but they do not tell us much about the appearance of an image in general.

As you will see in subsequent chapters, concepts from probability play a central role in a broad range of image processing applications. For example, Eq. (2-67) is utilized in Chapter 3 as the basis for image enhancement techniques based on histograms. In Chapter 5, we use probability to develop image restoration algorithms, in Chapter 10 we use probability for image segmentation, in Chapter 11 we use it to describe texture, and in Chapter 12 we use probability as the basis for deriving optimum pattern recognition algorithms.

## Summary, References, and Further Reading

The material in this chapter is the foundation for the remainder of the book. For additional reading on visual perception, see Snowden et al. [2012], and the classic book by Cornsweet [1970]. Born and Wolf [1999] discuss light in terms of electromagnetic theory. A basic source for further reading on image sensing is Trussell and Vrhel [2008]. The image formation model discussed in Section 2.3 is from Oppenheim et al. [1968]. The IES Lighting Handbook [2011] is a reference for the illumination and reflectance values used in that section. The concepts of image sampling introduced in Section 2.4 will be covered in detail in Chapter 4. The discussion on experiments dealing with the relationship between image quality and sampling is based on results from Huang [1965]. For further reading on the topics discussed in Section 2.5, see Rosenfeld and Kak [1982], and Klette and Rosenfeld [2004].

See Castleman [1996] for additional reading on linear systems in the context of image processing. The method of noise reduction by image averaging was first proposed by Kohler and Howell [1963]. See Ross [2014] regarding the expected value of the mean and variance of the sum of random variables. See Schröder [2010] for additional reading on logic and sets. For additional reading on geometric spatial transformations see Wolberg [1990] and Hughes and Andries [2013]. For further reading on image registration see Goshtasby [2012]. Bronson and Costa [2009] is a good reference for additional reading on vectors and matrices. See Chapter 4 for a detailed treatment of the Fourier transform, and Chapters 7, 8, and 11 for details on other image transforms. For details on the software aspects of many of the examples in this chapter, see Gonzalez, Woods, and Eddins [2009].

## Problems

*Solutions to the problems marked with an asterisk (*) are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).*

**2.1** If you use a sheet of white paper to shield your eyes when looking directly at the sun, the side of the sheet facing you appears black. Which of the visual processes discussed in Section 2.1 is responsible for this?

**2.2*** Using the background information provided in Section 2.1, and thinking purely in geometrical terms, estimate the diameter of the smallest printed dot that the eye can discern if the page on which the dot is printed is 0.2 m away from the eyes. Assume for simplicity that the visual system ceases to detect the dot when the image of the dot on the fovea becomes smaller than the diameter of one receptor (cone) in that area of the retina. Assume further that the fovea can be modeled as a square array of dimension 1.5 mm on the side, and that the cones and spaces between the cones are distributed uniformly throughout this array.

**2.3** Although it is not shown in Fig. 2.10, alternating current is part of the electromagnetic spectrum. Commercial alternating current in the United States has a frequency of 60 Hz. What is the wavelength in kilometers of this component of the spectrum?

**2.4** You are hired to design the front end of an imaging system for studying the shapes of cells, bacteria, viruses, and proteins. The front end consists in this case of the illumination source(s) and corresponding imaging camera(s). The diameters of circles required to fully enclose individual specimens in each of these categories are 50, 1, 0.1, and 0.01 $\mu$m, respectively. In order to perform automated analysis, the smallest detail discernible on a specimen must be 0.001 $\mu$m.

**(a)*** Can you solve the imaging aspects of this problem with a single sensor and camera? If your answer is yes, specify the illumination wavelength band and the type of camera needed. By "type," we mean the band of the electromagnetic spectrum to which the camera is most sensitive (e.g., infrared).

**(b)** If your answer in (a) is no, what type of illumination sources and corresponding imaging sensors would you recommend? Specify the light sources and cameras as requested in part (a). Use the minimum number of illumination sources and cameras needed to solve the problem. (*Hint:* From the discussion in

Section 2.2, the illumination required to "see" an object must have a wavelength the same size or smaller than the object.)

**2.5** You are preparing a report and have to insert in it an image of size $2048 \times 2048$ pixels.

**(a)*** Assuming no limitations on the printer, what would the resolution in line pairs per mm have to be for the image to fit in a space of size $5 \times 5$ cm?

**(b)** What would the resolution have to be in dpi for the image to fit in $2 \times 2$ inches?

**2.6*** A CCD camera chip of dimensions $7 \times 7$ mm and $1024 \times 1024$ sensing elements, is focused on a square, flat area, located 0.5 m away. The camera is equipped with a 35-mm lens. How many line pairs per mm will this camera be able to resolve? (*Hint:* Model the imaging process as in Fig. 2.3, with the focal length of the camera lens substituting for the focal length of the eye.)

**2.7** An automobile manufacturer is automating the placement of certain components on the bumpers of a limited-edition line of sports cars. The components are color-coordinated, so the assembly robots need to know the color of each car in order to select the appropriate bumper component. Models come in only four colors: blue, green, red, and white. You are hired to propose a solution based on imaging. How would you solve the problem of determining the color of each car, keeping in mind that cost is the most important consideration in your choice of components?

**2.8*** Suppose that a given automated imaging application requires a minimum resolution of 5 line pairs per mm to be able to detect features of interest in objects viewed by the camera. The distance between the focal center of the camera lens and the area to be imaged is 1 m. The area being imaged is $0.5 \times 0.5$ m. You have available a 200 mm lens, and your job is to pick an appropriate CCD imaging chip. What is the minimum number of sensing elements and square size, $d \times d$, of the CCD chip that will meet the requirements of this application? (*Hint:* Model the imaging process as in Fig. 2.3, and assume for simplicity that the imaged area is square.)

**2.9** A common measure of transmission for digital data is the *baud rate*, defined as symbols (bits in our case) per second. As a minimum, transmission is accomplished in packets consisting of a start bit, a byte (8 bits) of information, and a stop bit. Using these facts, answer the following:

**(a)*** How many seconds would it take to transmit a sequence of 500 images of size $1024 \times 1024$ pixels with 256 intensity levels using a 3 M-baud ($10^6$ bits/sec) baud modem? (This is a representative medium speed for a DSL (Digital Subscriber Line) residential line.

**(b)** What would the time be using a 30 G-baud ($10^9$ bits/sec) modem? (This is a representative medium speed for a commercial line.)

**2.10*** High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (i.e., where every other line is "painted" on the screen in each of two fields, each field being 1/60th of a second in duration). The width-to-height aspect ratio of the images is 16:9. The fact that the number of horizontal lines is fixed determines the vertical resolution of the images. A company has designed a system that extracts digital images from HDTV video. The resolution of each horizontal line in their system is proportional to vertical resolution of HDTV, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity, 8 bits each for a red, a green, and a blue component image. These three "primary" images form a color image. How many bits would it take to store the images extracted from a two-hour HDTV movie?

**2.11** When discussing linear indexing in Section 2.4, we arrived at the linear index in Eq. (2-14) by inspection. The same argument used there can be extended to a 3-D array with coordinates $x$, $y$, and $z$, and corresponding dimensions $M$, $N$, and $P$. The linear index for any $(x, y, z)$ is

$$s = x + M(y + Nz)$$

Start with this expression and

**(a)*** Derive Eq. (2-15).

**(b)** Derive Eq. (2-16).

**2.12*** Suppose that a flat area with center at $(x_0, y_0)$ is

illuminated by a light source with intensity distribution

$$i(x,y) = Ke^{-[(x-x_0)^2 + (y-y_0)^2]}$$

Assume for simplicity that the reflectance of the area is constant and equal to 1.0, and let $K = 255$. If the intensity of the resulting image is quantized using $k$ bits, and the eye can detect an abrupt change of eight intensity levels between adjacent pixels, what is the highest value of $k$ that will cause visible false contouring?

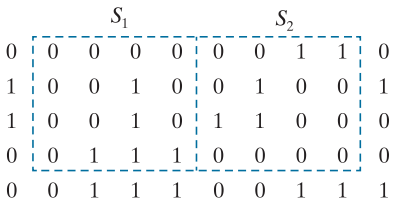
**2.13** Sketch the image in Problem 2.12 for $k = 2$.

**2.14** Consider the two image subsets, $S_1$ and $S_2$ in the following figure. With reference to Section 2.5, and assuming that $V = \{1\}$, determine whether these two subsets are:

**(a)*** 4-adjacent.

**(b)** 8-adjacent.

**(c)** $m$-adjacent.




**2.15*** Develop an algorithm for converting a one-pixel-thick 8-path to a 4-path.

**2.16** Develop an algorithm for converting a one-pixel-thick $m$-path to a 4-path.

**2.17** Refer to the discussion toward the end of Section 2.5, where we defined the background of an image as $(R_u)^c$, the complement of the union of all the regions in the image. In some applications, it is advantageous to define the background as the subset of pixels of $(R_u)^c$ that are not *hole* pixels (informally, think of holes as sets of background pixels surrounded by foreground pixels). How would you modify the definition to exclude hole pixels from $(R_u)^c$? An answer such as "the background is the subset of pixels of $(R_u)^c$ that are not hole pixels" is not acceptable. (*Hint:* Use the concept of connectivity.)

**2.18** Consider the image segment shown in the figure that follows.

**(a)*** As in Section 2.5, let $V = \{0,1\}$ be the set of intensity values used to define adjacency. Compute the lengths of the shortest 4-, 8-, and $m$-path between $p$ and $q$ in the following image. If a particular path does not exist between these two points, explain why.

| | | | |
|---|---|---|---|
| 3 | 1 | 2 | 1 ($q$) |
| 2 | 2 | 0 | 2 |
| 1 | 2 | 1 | 1 |
| ($p$) 1 | 0 | 1 | 2 |

**(b)** Repeat (a) but using $V = \{1,2\}$.

**2.19** Consider two points $p$ and $q$.

**(a)*** State the condition(s) under which the $D_4$ distance between $p$ and $q$ is equal to the shortest 4-path between these points.

**(b)** Is this path unique?

**2.20** Repeat problem 2.19 for the $D_8$ distance.

**2.21** Consider two *one-dimensional* images $f$ and $g$ of the same size. What has to be true about the orientation of these images for the elementwise and matrix products discussed in Section 2.6 to make sense? Either of the two images can be first in forming the product.

**2.22*** In the next chapter, we will deal with operators whose function is to compute the sum of pixel values in a small subimage area, $S_{xy}$, as in Eq. (2-43). Show that these are linear operators.

**2.23** Refer to Eq. (2-24) in answering the following:

**(a)*** Show that image summation is a linear operation.

**(b)** Show that image subtraction is a linear operation.

**(c)*** Show that image multiplication in a nonlinear operation.

**(d)** Show that image division is a nonlinear operation.

**2.24** The median, $\zeta$, of a set of numbers is such that

half the values in the set are below $\zeta$ and the other half are above it. For example, the median of the set of values $\{2,3,8,20,21,25,31\}$ is 20. Show that an operator that computes the median of a subimage area, $S$, is nonlinear. (*Hint:* It is sufficient to show that $\zeta$ fails the linearity test for a simple numerical example.)

**2.25*** Show that image averaging can be done recursively. That is, show that if $a(k)$is the average of $k$ images, then the average of $k+1$ images can be obtained from the already-computed average, $a(k)$, and the new image, $f_{k+1}$.

**2.26** With reference to Example 2.5:

**(a)*** Prove the validity of Eq. (2-27).

**(b)** Prove the validity of Eq. (2-28).

For part (b) you will need the following facts from probability: (1) the variance of a constant times a random variable is equal to the constant squared times the variance of the random variable. (2) The variance of the sum of uncorrelated random variables is equal to the sum of the variances of the individual random variables.

**2.27** Consider two 8-bit images whose intensity levels span the full range from 0 to 255.

**(a)*** Discuss the limiting effect of repeatedly subtracting image (2) from image (1). Assume that the results have to be represented also in eight bits.

**(b)** Would reversing the order of the images yield a different result?

**2.28*** Image subtraction is used often in industrial applications for detecting missing components in product assembly. The approach is to store a "golden" image that corresponds to a correct assembly; this image is then subtracted from incoming images of the same product. *Ideally*, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the area where they differ from the golden image. What conditions do you think have to be met in practice for this method to work?

**2.29** With reference to Eq. (2-32),

**(a)*** Give a general formula for the value of $K$ as a function of the number of bits, $k$, in an image, such that $K$ results in a scaled image whose intensities span the full $k$-bit range.

**(b)** Find $K$ for 16- and 32-bit images.

**2.30** Give Venn diagrams for the following expressions:

**(a)*** $(A \cap C) - (A \cap B \cap C)$.

**(b)** $(A \cap C) \cup (B \cap C)$.

**(c)** $B - [(A \cap B) - (A \cap B \cap C)]$

**(d)** $B - B \cap (A \cup C)$; Given that $A \cap C = \varnothing$.

**2.31** Use Venn diagrams to prove the validity of the following expressions:

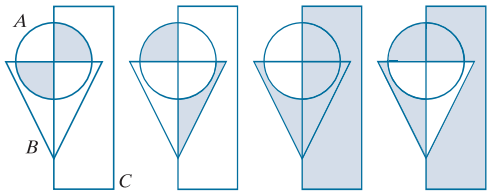
**(a)*** $(A \cap B) \cup [(A \cap C) - A \cap B \cap C] = A \cap (B \cup C)$

**(b)** $(A \cup B \cup C)^c = A^c \cap B^c \cap C^c$

**(c)** $(A \cup C)^c \cap B = (B - A) - C$

**(d)** $(A \cap B \cap C)^c = A^c \cup B^c \cup C^c$

**2.32** Give expressions (in terms of sets $A$, $B$, and $C$) for the sets shown shaded in the following figures. The shaded areas in each figure constitute one set, so give only one expression for each of the four figures.




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

**2.33** With reference to the discussion on sets in Section 2.6, do the following:

**(a)*** Let $S$ be a set of real numbers ordered by the relation "less than or equal to" $(\leq)$. Show that $S$ is a partially ordered set; that is, show that the reflexive, transitive, and antisymmetric properties hold.

**(b)*** Show that changing the relation "less than or equal to" to "less than" $(<)$ produces a strict ordered set.

**(c)** Now let $S$ be the set of lower-case letters in the English alphabet. Show that, under $(<)$, $S$ is a strict ordered set.

**2.34** For any nonzero integers $m$ and $n$, we say that $m$

is divisible by $n$, written $m/n$, if there exists an integer $k$ such that $kn = m$. For example, 42 ($m$) is divisible by 7 ($n$) because there exists an integer $k = 6$ such that $kn = m$. Show that the set of positive integers is a partially ordered set under the relation "divisible by." In other words, do the following:

**(a)*** Show that the property of reflectivity holds under this relation.

**(b)** Show that the property of transitivity holds.

**(c)** Show that anti symmetry holds.

**2.35** In general, what would the resulting image, $g(x, y)$, look like if we modified Eq. (2-43), as follows:

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

where $T$ is the intensity transformation function in Fig. 2.38(b)?

**2.36** With reference to Table 2.3, provide single, composite transformation functions for performing the following operations:

**(a)*** Scaling and translation.

**(b)*** Scaling, translation, and rotation.

**(c)** Vertical shear, scaling, translation, and rotation.

**(d)** Does the order of multiplication of the individual matrices to produce a single transformations make a difference? Give an example based on a scaling/translation transformation to support your answer.

**2.37** We know from Eq. (2-45) that an affine transformation of coordinates is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $(x', y')$ are the transformed coordinates, $(x, y)$ are the original coordinates, and the elements of $\mathbf{A}$ are given in Table 2.3 for various types of transformations. The inverse transformation, $\mathbf{A}^{-1}$, to go from the transformed back to the original coordinates is just as important for performing inverse mappings.

**(a)*** Find the inverse scaling transformation.

**(b)** Find the inverse translation transformation.

**(c)** Find the inverse vertical and horizontal shearing transformations.

**(d)*** Find the inverse rotation transformation.

**(e)*** Show a composite inverse translation/rotation transformation.

**2.38** What are the equations, analogous to Eqs. (2-46) and (2-47), that would result from using triangular instead of quadrilateral regions?

**2.39** Do the following.

**(a)*** Prove that the Fourier kernel in Eq. (2-59) is separable and symmetric.

**(b)** Repeat (a) for the kernel in Eq. (2-60).

**2.40*** Show that 2-D transforms with separable, symmetric kernels can be computed by: (1) computing 1-D transforms along the individual rows (columns) of the input image; and (2) computing 1-D transforms along the columns (rows) of the result from step (1).

**2.41** A plant produces miniature polymer squares that have to undergo 100% visual inspection. Inspection is semi-automated. At each inspection station, a robot places each polymer square over an optical system that produces a magnified image of the square. The image completely fills a viewing screen of size $80 \times 80$ mm. Defects appear as dark circular blobs, and the human inspector's job is to look at the screen and reject any sample that has one or more dark blobs with a diameter of 0.8 mm or greater, as measured on the scale of the screen. The manufacturing manager believes that if she can find a way to fully automate the process, profits will increase by 50%, and success in this project will aid her climb up the corporate ladder. After extensive investigation, the manager decides that the way to solve the problem is to view each inspection screen with a CCD TV camera and feed the output of the camera into an image processing system capable of detecting the blobs, measuring their diameter, and activating the accept/reject button previously operated by a human inspector. She is able to find a suitable system, provided that the smallest defect occupies an area of at least $2 \times 2$ pixels in the digital image. The manager hires you to help her specify the camera and lens

system to satisfy this requirement, using off-the-shelf components. Available off-the-shelf lenses have focal lengths that are integer multiples of 25 mm or 35 mm, up to 200 mm. Available cameras yield image sizes of $512 \times 512$, $1024 \times 1024$, or $2048 \times 2048$ pixels. The *individual* imaging elements in these cameras are squares measuring $8 \times 8$ $\mu$m, and the spaces between imaging elements are 2 $\mu$m. For this application, the cameras cost much more than the lenses, so you should use the lowest-resolution camera possible, consistent with a suitable lens. As a consultant, you have to provide a written recommendation, showing in reasonable detail the analysis that led to your choice of components. Use the imaging geometry suggested in Problem 2.6.