# ROBOTICS:
## Control, Sensing, Vision, and Intelligence

**K. S. Fu**
*School of Electrical Engineering*
*Purdue University*

**R. C. Gonzalez**
*Department of Electrical Engineering*
*University of Tennessee*
*and*
*Perceptics Corporation*
*Knoxville, Tennessee*

**C. S. G. Lee**
*School of Electrical Engineering*
*Purdue University*

**McGraw-Hill Book Company**

New York St. Louis San Francisco Auckland Bogotá
Hamburg Johannesburg London Madrid Mexico Milan Montreal New Delhi
Panama Paris São Paulo Singapore Sydney Tokyo Toronto

CHAPTER
# SEVEN

# LOW-LEVEL VISION

Where there is no vision, the people perish.
*Proverbs*

## 7.1 INTRODUCTION

As is true in humans, vision capabilities endow a robot with a sophisticated sensing mechanism that allows the machine to respond to its environment in an "intelligent" and flexible manner. The use of vision and other sensing schemes, such as those discussed in Chap. 6, is motivated by the continuing need to increase the flexibility and scope of applications of robotic systems. While proximity, touch, and force sensing play a significant role in the improvement of robot performance, vision is recognized as the most powerful of robot sensory capabilities. As might be expected, the sensors, concepts, and processing hardware associated with machine vision are considerably more complex than those associated with the sensory approaches discussed in Chap. 6.

Robot vision may be defined as the process of extracting, characterizing, and interpreting information from images of a three-dimensional world. This process, also commonly referred to as *machine* or *computer vision*, may be subdivided into six principal areas: (1) sensing, (2) preprocessing, (3) segmentation, (4) description, (5) recognition, and (6) interpretation. Sensing is the process that yields a visual image. Preprocessing deals with techniques such as noise reduction and enhancement of details. Segmentation is the process that partitions an image into objects of interest. Description deals with the computation of features (e.g., size, shape) suitable for differentiating one type of object from another. Recognition is the process that identifies these objects (e.g., wrench, bolt, engine block). Finally, interpretation assigns meaning to an ensemble of recognized objects.

It is convenient to group these various areas according to the sophistication involved in their implementation. We consider three levels of processing: low-, medium-, and high-level vision. While there are no clearcut boundaries between these subdivisions, they do provide a useful framework for categorizing the various processes that are inherent components of a machine vision system. For instance, we associate with low-level vision those processes that are primitive in the sense that they may be considered "automatic reactions" requiring no intelligence on the part of the vision system. In our discussion, we shall treat sensing and preprocessing as low-level vision functions. This will take us from the image formation process itself to compensations such as noise reduction, and finally to the extraction of

primitive image features such as intensity discontinuities. This range of processes may be compared with the sensing and adaptation process a human goes through in trying to find a seat in a dark theater immediately after walking in during a bright afternoon. The intelligent process of finding an unoccupied space cannot begin until a suitable image is available.

We will associate with medium-level vision those processes that extract, characterize, and label components in an image resulting from low-level vision. In terms of our six subdivisions, we will treat segmentation, description, and recognition of individual objects as medium-level vision functions. High-level vision refers to processes that attempt to emulate cognition. While algorithms for low- and medium-level vision encompass a reasonably well-defined spectrum of activities, our knowledge and understanding of high-level vision processes is considerably more vague and speculative. As discussed in Chap. 8, these limitations lead to the formulation of constraints and idealizations intended to reduce the complexity of this task.

The categories and subdivisions discussed above are suggested to a large extent by the way machine vision systems are generally implemented. It is not implied that these subdivisions represent a model of human vision nor that they are carried out independently of each other. We know, for example, that recognition and interpretation are highly interrelated functions in a human. These relationships, however, are not yet understood to the point where they can be modeled analytically. Thus, the subdivision of functions addressed in this discussion may be viewed as a practical approach for implementing state-of-the-art machine vision systems, given our level of understanding and the analytical tools currently available in this field.

The material in this chapter deals with sensing, preprocessing, and with concepts and techniques required to implement low-level vision functions. Topics in higher-level vision are discussed in Chap. 8. Although true vision is inherently a three-dimensional activity, most of the work in machine vision is carried out using images of a three-dimensional scene, with depth information being obtained by special imaging techniques, such as the structured-lighting approach discussed in Sec. 7.3, or by the use of stereo imaging, as discussed in Sec. 7.4.

## 7.2 IMAGE ACQUISITION

Visual information is converted to electrical signals by visual sensors. When sampled spatially and quantized in amplitude, these signals yield a *digital image*. In this section we are interested in three main topics: (1) the principal imaging techniques used for robotic vision, (2) the effects of sampling on spatial resolution, and (3) the effects of amplitude quantization on intensity resolution. The mathematics of image formation are discussed in Sec. 7.4.

The principal devices used for robotic vision are television cameras, consisting either of a tube or solid-state imaging sensor, and associated electronics. Although an in-depth treatment of these devices is beyond the scope of the present discus-

sion, we will consider the principles of operation of the vidicon tube, a commonly used representative of the tube family of TV cameras. Solid-state imaging sensors will be introduced via a brief discussion of charge-coupled devices (CCDs), which are one of the principal exponents of this technology. Solid-state imaging devices offer a number of advantages over tube cameras, including lighter weight, smaller size, longer life, and lower power consumption. However, the resolution of certain tubes is still beyond the capabilities of solid-state cameras.

As shown schematically in Fig. 7.1*a*, the vidicon camera tube is a cylindrical glass envelope containing an electron gun at one end, and a faceplate and target at the other. The beam is focused and deflected by voltages applied to the coils shown in Fig. 7.1*a*. The deflection circuit causes the beam to scan the inner surface of the target in order to "read" the image, as explained below. The inner surface of the glass faceplate is coated with a transparent metal film which forms an electrode from which an electrical video signal is derived. A thin photosensi-

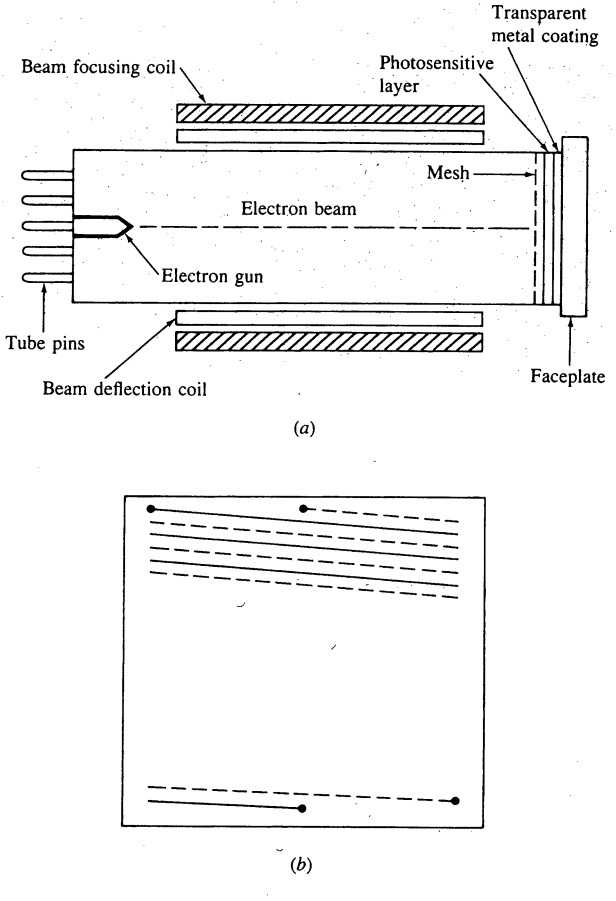


**Figure 7.1** (*a*) Schematic of a vidicon tube. (*b*) Electron beam scanning pattern.

tive "target" layer is deposited onto the metal film; this layer consists of very small resistive globules whose resistance is inversely proportional to light intensity. Behind the photosensitive target there is a positively charged fine wire mesh which decelerates electrons emitted by the gun so that they reach the target surface with essentially zero velocity.

In normal operation, a positive voltage is applied to the metal coating of the faceplate. In the absence of light, the photosensitive material behaves as a dielectric, with the electron beam depositing a layer of electrons on the inner surface of the target surface to balance the positive charge on the metal coating. As the electron beam scans the surface of the target layer, the photosensitive layer thus becomes a capacitor with negative charge on the inner surface and positive charge on the other side. When light strikes the target layer, its resistance is reduced and electrons are allowed to flow and neutralize the positive charge. Since the amount of electronic charge that flows is proportional to the amount of light in any local area of the target, this effect produces an image on the target layer that is identical to the light image on the faceplate of the tube; that is, the remaining concentration of electron charge is high in dark areas and lower in light areas. As the beam again scans the target it replaces the lost charge, thus causing a current to flow in the metal layer and out one of the tube pins. This current is proportional to the number of electrons replaced and, therefore, to the light intensity at a particular location of the scanning beam. This variation in current during the electron beam scanning motion produces, after conditioning by the camera circuitry, a video signal proportional to the intensity of the input image.

The principal scanning standard used in the United States is shown in Fig. 7.1*b*. The electron beam scans the entire surface of the target 30 times per second, each complete scan (called a *frame*) consisting of 525 lines of which 480 contain image information. If the lines were scanned sequentially and the result shown on a TV monitor, the image would flicker perceptibly. This phenomenon is avoided by using a scan mechanism in which a frame is divided into two *interlaced fields*, each consisting of 262.5 lines and scanned 60 times each second, or twice the frame rate. The first field of each frame scans the odd lines (shown dashed in Fig. 7.1*a*), while the second field scans the even lines. This scanning scheme, called the RETMA (Radio-Electronics-Television Manufacturers Association) scanning convention, is the standard used for broadcast television in the United States. Other standards exist which yield higher line rates per frame, but their principle of operation is essentially the same. For example, a popular scanning approach in computer vision and digital image processing is based on 559 lines, of which 512 contain image data. Working with integer powers of 2 has a number of advantages for both hardware and software implementations.

When discussing CCD devices, it is convenient to subdivide sensors into two categories: line scan sensors and area sensors. The basic component of a line scan CCD sensor is a row of silicon imaging elements called *photosites*. Image photons pass through a transparent polycrystalline silicon gate structure and are absorbed in the silicon crystal, thus creating electron-hole pairs. The resulting photoelectrons are collected in the photosites, with the amount of charge collected

at each photosite being proportional to the illumination intensity at that location. As shown in Fig. 7.2*a*, a typical line scan sensor is composed of a row of the imaging elements just discussed, two transfer gates used to clock the contents of the imaging elements into so-called transport registers, and an output gate used to clock the contents of the transport registers into an amplifier whose output is a voltage signal proportional to the contents of the row of photosites.

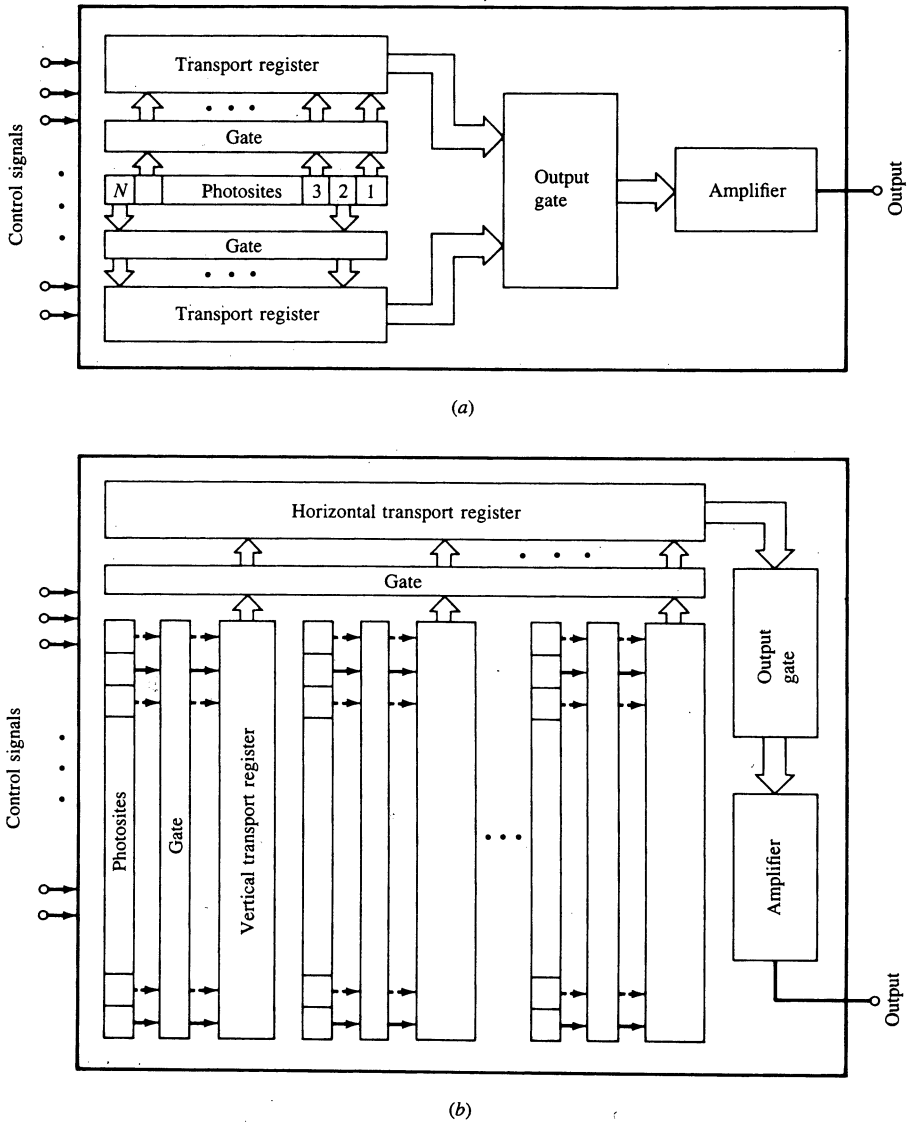


**Figure 7.2** (*a*) CCD line scan sensor. (*b*) CCD area sensor.

Charge-coupled area arrays are similar to the line scan sensors, with the exception that the photosites are arranged in a matrix format and there is a gate-transport register combination between columns of photosites, as shown in Fig. 7.2*b*. The contents of odd-numbered photosites are sequentially gated into the vertical transport registers and then into the horizontal transport register. The content of this register is fed into an amplifier whose output is a line of video. Repeating this procedure for the even-numbered lines completes the second field of a TV frame. This "scanning" mechanism is repeated 30 times per second.

Line scan cameras obviously yield only one line of an input image. These devices are ideally suited for applications in which objects are moving past the sensor (as in conveyor belts). The motion of an object in the direction perpendicular to the sensor produces a two-dimensional image. Line scan sensors with resolutions ranging between 256 and 2048 elements are not uncommon. The resolutions of area sensors range between 32 × 32 at the low end to 256 × 256 elements for a medium resolution sensor. Higher-resolution devices presently in the market have a resolution on the order of 480 × 380 elements, and experimental CCD sensors are capable of achieving a resolution of 1024 × 1024 elements or higher.

Throughout this book, we will use $f(x, y)$ to denote the two-dimensional image out of a TV camera or other imaging device, where $x$ and $y$ denote spatial (i.e., image plane) coordinates, and the value of $f$ at any point $(x, y)$ is proportional to the brightness (intensity) of the image at that point. Figure 7.3 illustrates this concept, as well as the coordinate convention on which all subsequent discussions will be based. We will often use the variable $z$ to denote intensity variations in an image when the spatial location of these variations is of no interest.

In order to be in a form suitable for computer processing, an image function $f(x, y)$ must be digitized both spatially and in amplitude (intensity). Digitization of the spatial coordinates $(x, y)$ will be referred to as *image sampling*, while amplitude digitization will be called *intensity* or *gray-level quantization*. The latter term is applicable to monochrome images and reflects the fact that these images vary from black to white in shades of gray. The terms intensity and gray level will be used interchangeably.

Suppose that a continuous image is sampled uniformly into an array of $N$ rows and $M$ columns, where each sample is also quantized in intensity. This array, called a *digital image*, may be represented as

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, M-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, M-1) \\ \cdots & \cdots & \cdots & \cdots \\ f(N-1, 0) & f(N-1, 1) & \cdots & f(N-1, M-1) \end{bmatrix} \tag{7.2-1}$$

where $x$ and $y$ are now discrete variables: $x = 0, 1, 2, \ldots, N-1$; $y = 0, 1, 2, \ldots, M-1$. Each element in the array is called an *image element, picture element*, or *pixel*. With reference to Fig. 7.3, it is noted that $f(0, 0)$ represents the pixel at the origin of the image, $f(0, 1)$ the pixel to its right, and so

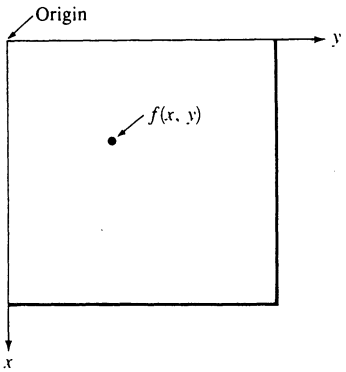


**Figure 7.3** Coordinate convention for image representation. The value of any point $(x, y)$ is given by the value (intensity) of $f$ at that point.

on. It is common practice to let $N$, $M$, and the number of discrete intensity levels of each quantized pixel be integer powers of 2.

In order to gain insight into the effect of sampling and quantization, consider Fig. 7.4. Part (*a*) of this figure shows an image sampled into an array of $N \times N$ pixels with $N = 512$; the intensity of each pixel is quantized into one of 256 discrete levels. Figure 7.4*b* to *e* shows the same image, but with $N = 256$, 128, 64, and 32. In all cases the number of allowed intensity levels was kept at 256. Since the display area used for each image was the same ($512 \times 512$ display points), pixels in the lower resolution images were duplicated in order to fill the entire display field. This produced a checkerboard effect that is particularly visible in the low-resolution images. It is noted that the $256 \times 256$ image is reasonably close to Fig. 7.4*a*, but image quality deteriorated rapidly for the other values of $N$.

Figure 7.5 illustrates the effect produced by reducing the number of intensity levels while keeping the spatial resolution constant at $512 \times 512$. The 256-, 128-, and 64-level images are of acceptable quality. The 32-level image, however, shows a slight degradation (particularly in areas of nearly constant intensity) as a result of using too few intensity levels to represent each pixel. This effect is considerably more visible as ridgelike structures (called *false contours*) in the image displayed with 16 levels, and increases sharply thereafter.

The number of samples and intensity levels required to produce a useful (in the machine vision sense) reproduction of an original image depends on the image itself and on the intended application. As a basis for comparison, the requirements to obtain quality comparable to that of monochrome TV pictures are on the order of $512 \times 512$ pixels with 128 intensity levels. As a rule, a minimum system for general-purpose vision work should have spatial resolution capabilities on the order of $256 \times 256$ pixels with 64 levels.

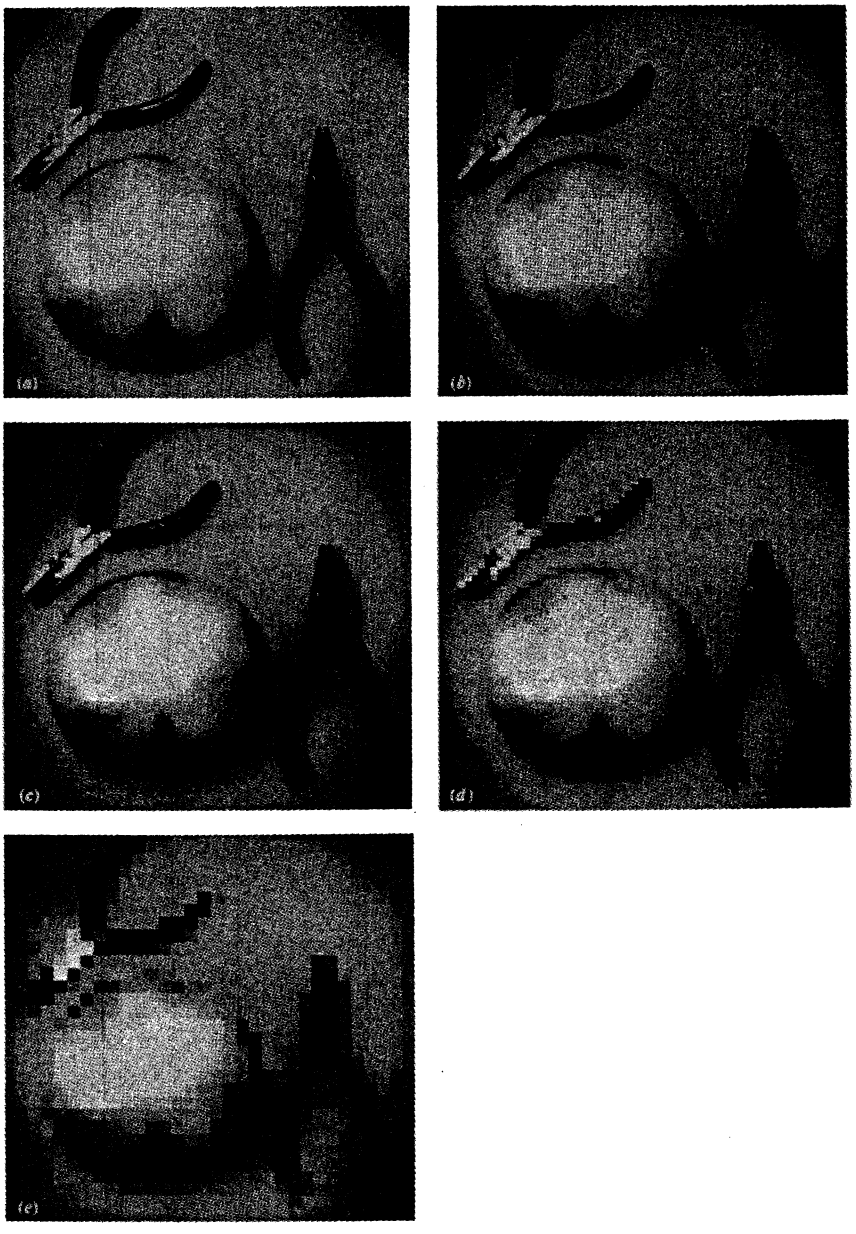

**Figure 7.4** Effects of reducing sampling-grid size. (*a*) 512 × 512. (*b*) 256 × 256. (*c*) 128 × 128. (*d*) 64 × 64. (*e*) 32 × 32.

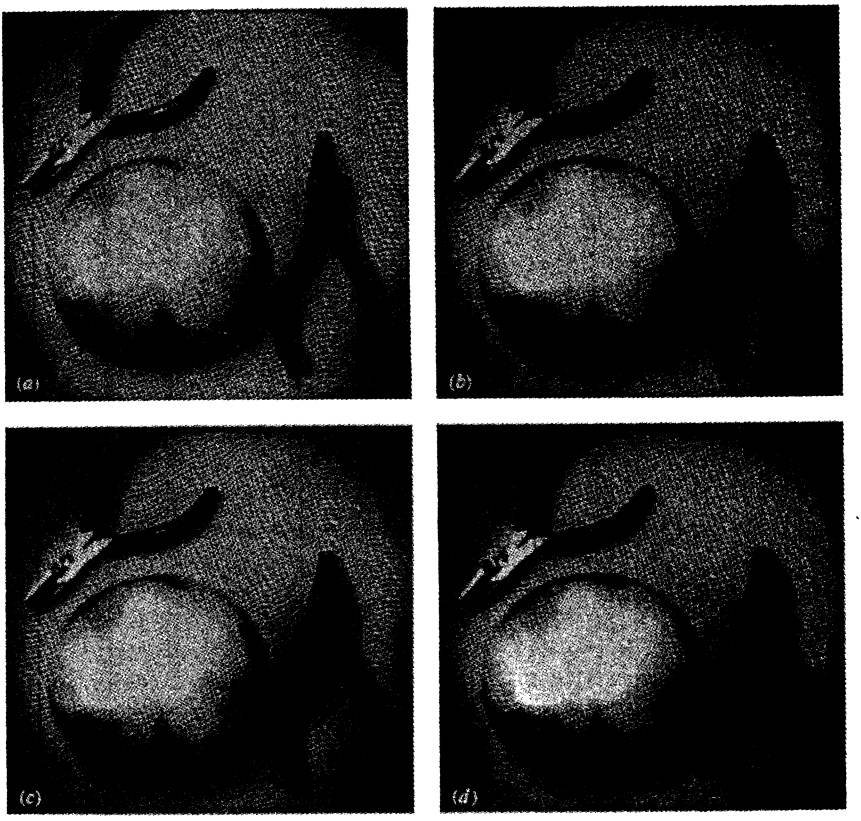

**Figure 7.5** A 512 × 512 image displayed with 256, 128, 64, 32, 16, 8, 4, and 2 levels.

## 7.3 ILLUMINATION TECHNIQUES

Illumination of a scene is an important factor that often affects the complexity of vision algorithms. Arbitrary lighting of the environment is often not acceptable because it can result in low-contrast images, specular reflections, shadows, and extraneous details. A well-designed lighting system illuminates a scene so that the complexity of the resulting image is minimized, while the information required for object detection and extraction is enhanced.

Figure 7.6 shows four of the principal schemes used for illuminating a robot work space. The diffuse-lighting approach shown in Fig. 7.6*a* can be employed for objects characterized by smooth, regular surfaces. This lighting scheme is generally employed in applications where surface characteristics are important. An example is shown in Fig. 7.7. Backlighting, as shown in Fig. 7.6*b*, produces a black and white (binary) image. This technique is ideally suited for applications in which silhouettes of objects are sufficient for recognition or other measurements. An example is shown in Fig. 7.8.

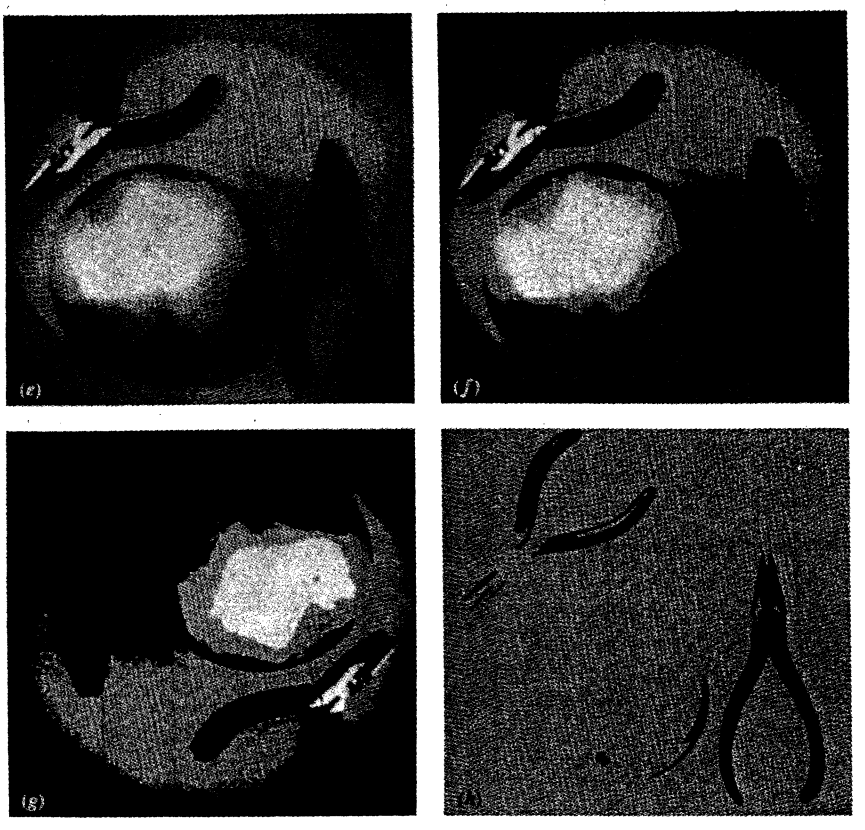

**Figure 7.5** (continued)

The structured-lighting approach shown in Fig. 7.6*c* consists of projecting points, stripes, or grids onto the work surface. This lighting technique has two important advantages. First, it establishes a known light pattern on the work space, and disturbances of this pattern indicate the presence of an object, thus simplifying the object detection problem. Second, by analyzing the way in which the light pattern is distorted, it is possible to gain insight into the three-dimensional characteristics of the object. Two examples of the structured-lighting approach are shown in Fig. 7.9. The first shows a block illuminated by parallel light planes which become light stripes upon intersecting a flat surface. The example shown in Fig. 7.9*b* consists of two light planes projected from different directions, but converging on a single stripe on the surface, as shown in Fig. 7.10*a*. A line scan camera, located above the surface and focused on the stripe would see a continuous line of light in the absence of an object. This line would be interrupted by an object which breaks both light planes simultaneously. This particular approach is ideally suited for objects moving on a conveyor belt past the camera. As shown in Fig. 7.10*b*, two light sources are used to guarantee that the object will break the

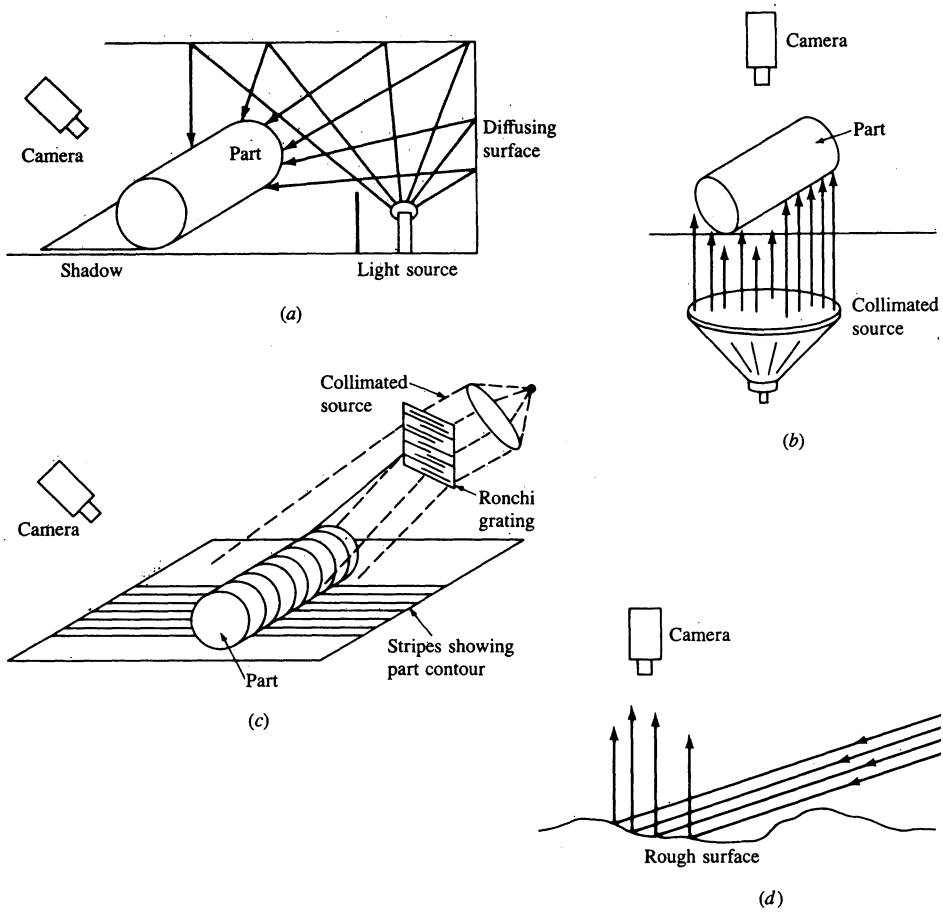


**Figure 7.6** Four basic illumination schemes. (From Mundy [1977], © IEEE.)

light stripe only when it is directly below the camera. It is of interest to note that the line scan camera sees *only* the line on which the two light planes converge, but two-dimensional information can be accumulated as the object moves past the camera.

The directional-lighting approach shown in Fig. 7.6*d* is useful primarily for inspection of object surfaces. Defects on the surface, such as pits and scratches, can be detected by using a highly directed light beam (e.g., a laser beam) and measuring the amount of scatter. For flaw-free surfaces little light is scattered upward to the camera. On the other hand, the presence of a flaw generally increases the amount of light scattered to the camera, thus facilitating detection of a defect. An example is shown in Fig. 7.11.

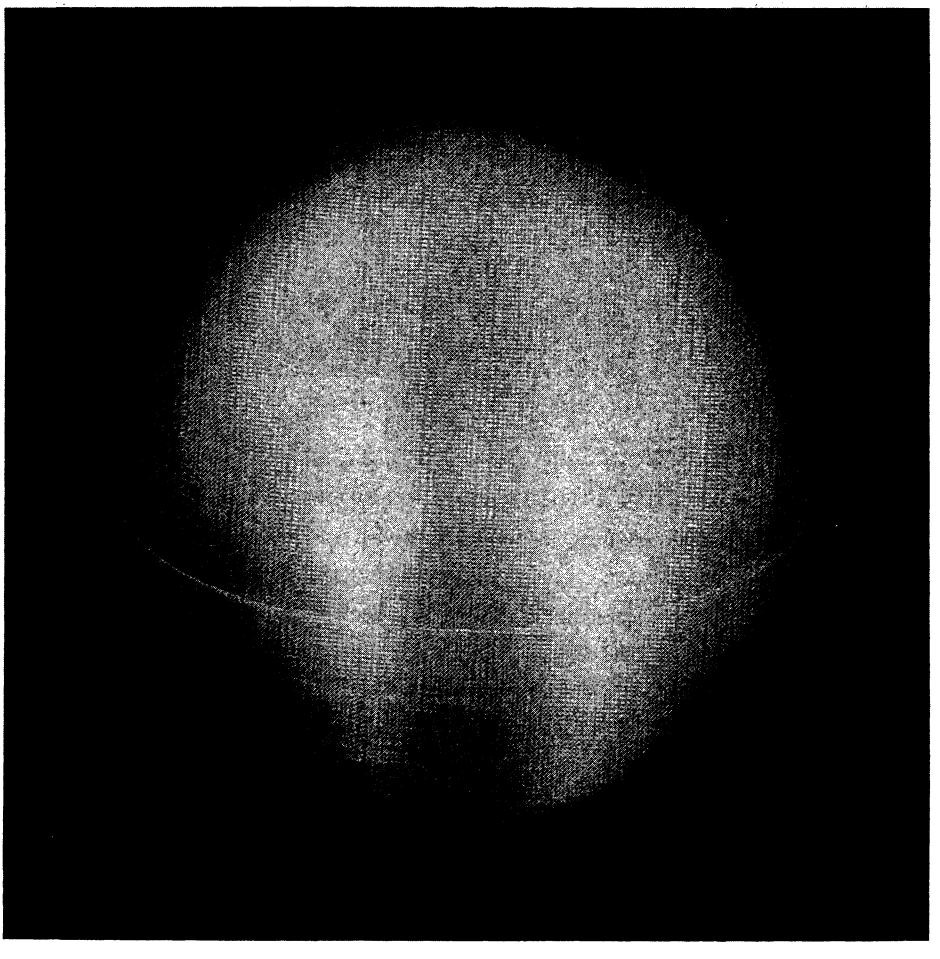

**Figure 7.7** Example of diffuse lighting.

## 7.4 IMAGING GEOMETRY

In the following discussion we consider several important transformations used in imaging, derive a camera model, and treat the stereo imaging problem in some detail. Some of the transformations discussed in the following section were already introduced in Chap. 2 in connection with robot arm kinematics. Here, we consider a similar problem, but from the point of view of imaging.

### 7.4.1 Some Basic Transformations

The material in this section deals with the development of a unified representation for problems such as image rotation, scaling, and translation. All transformations

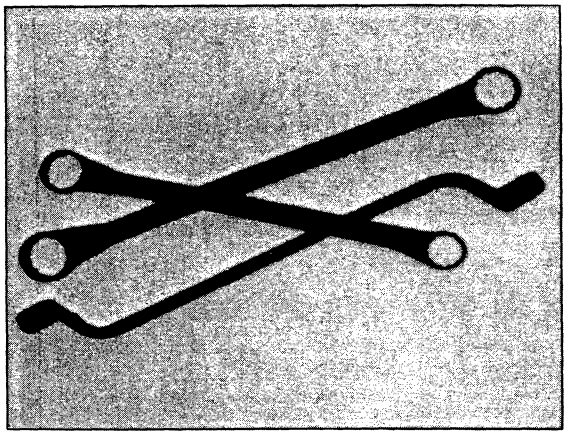

**Figure 7.8** Example of backlighting.

are expressed in a three-dimensional (3D) cartesian coordinate system in which a point has coordinates denoted by $(X, Y, Z)$. In cases involving two-dimensional images, we will adhere to our previous convention of using the lowercase representation $(x, y)$ to denote the coordinates of a pixel. It is common terminology to refer to $(X, Y, Z)$ as the *world coordinates* of a point.

**Translation.** Suppose that we wish to translate a point with coordinates $(X, Y, Z)$ to a new location by using displacements $(X_0, Y_0, Z_0)$. The translation is easily accomplished by using the following equations:

$$\begin{aligned} X^* &= X + X_0 \\ Y^* &= Y + Y_0 \\ Z^* &= Z + Z_0 \end{aligned} \tag{7.4.1}$$

where $(X^*, Y^*, Z^*)$ are the coordinates of the new point. Equation (7.4-1) can be expressed in matrix form by writing:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7.4-2}$$

As indicated later in this section, it is often useful to concatenate several transformations to produce a composite result, such as translation, followed by scaling, and then rotation. The notational representation of this process is

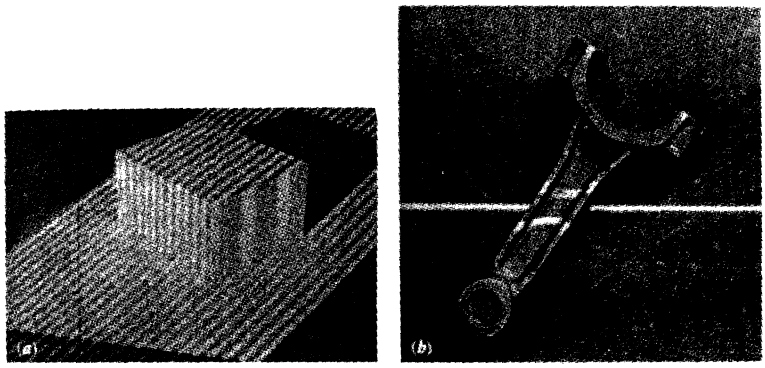

**Figure 7.9** Two examples of structured lighting. (Part (*a*) is from Rocher and Keissling [1975], © Kaufmann, Inc.; part (*b*) is from Myers [1980], © IEEE.)

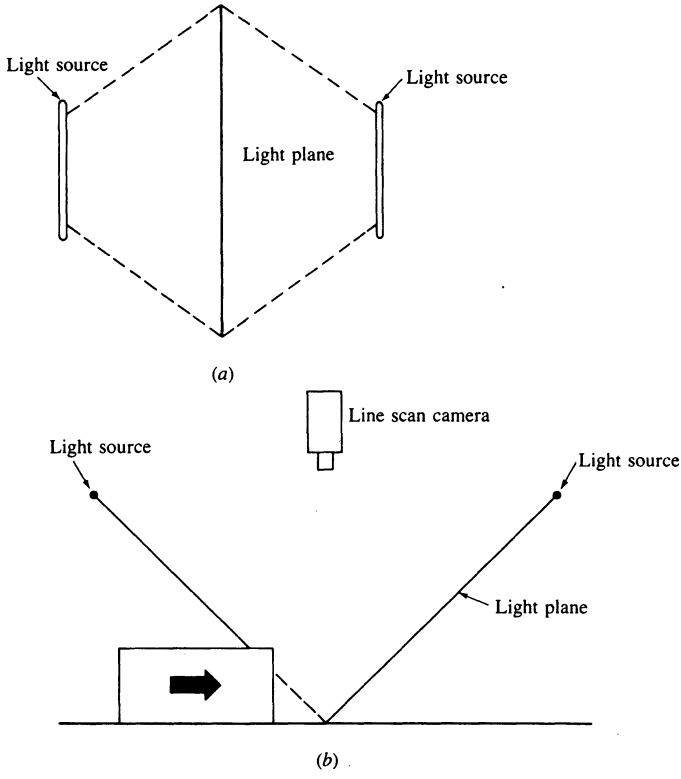


**Figure 7.10** (*a*) Top view of two light planes intersecting in a line of light. (*b*) Object will be seen by the camera only when it interrupts both light planes. (Adapted from Holland [1979], © Plenum.)

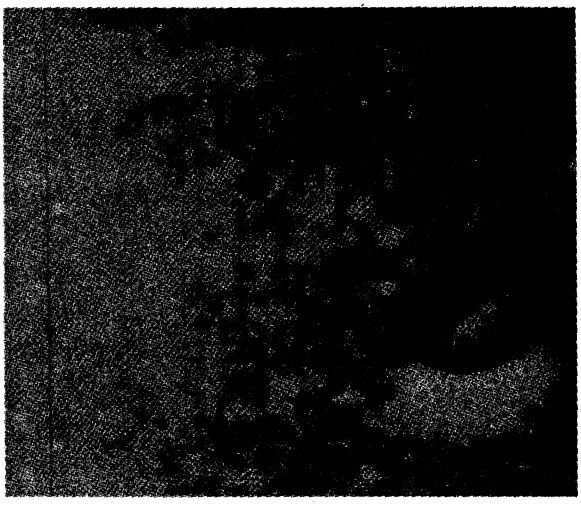

**Figure 7.11** Example of directional lighting. (From Mundy [1977], © IEEE.)

simplified considerably by using square matrices. With this in mind, we write Eq. (7.4-2) in the following form:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7.4-3}$$

In terms of the values of $X^*$, $Y^*$, and $Z^*$, Eqs. (7.4-2) and (7.4-3) are clearly equivalent.

Throughout this section, we will use the unified matrix representation

$$\mathbf{v}^* = \mathbf{A}\mathbf{v} \tag{7.4-4}$$

where $\mathbf{A}$ is a $4 \times 4$ transformation matrix, $\mathbf{v}$ is a column vector containing the original coordinates:

$$\mathbf{v} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7.4-5}$$

and $\mathbf{v}^*$ is a column vector whose components are the transformed coordinates:

$$\mathbf{v}^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} \tag{7.4-6}$$

Using this notation, the matrix used for translation is given by

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-7}$$

and the translation process is accomplished by using Eq. (7.4-4), so that $\mathbf{v}^* = \mathbf{T}\mathbf{v}$.

**Scaling**. Scaling by factors $S_x$, $S_y$, and $S_z$ along the $X$, $Y$, and $Z$ axes is given by the transformation matrix

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-8}$$

**Rotation**. The transformations used for three-dimensional rotation are inherently more complex than the transformations discussed thus far. The simplest form of these transformations is for rotation of a point about the coordinate axes. To rotate a given point about an arbitrary point in space requires three transformations: The first translates the arbitrary point to the origin, the second performs the rotation, and the third translates the point back to its original position.

With reference to Fig. 7.12, rotation of a point about the $Z$ coordinate axis by an angle $\theta$ is achieved by using the transformation

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-9}$$

The rotation angle $\theta$ is measured clockwise when looking at the origin from a point on the $+Z$ axis. It is noted that this transformation affects only the values of $X$ and $Y$ coordinates.

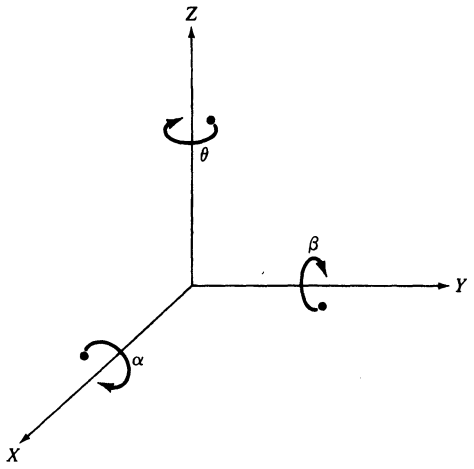


**Figure 7.12** Rotation of a point about each of the coordinate axes. Angles are measured clockwise when looking along the rotation axis toward the origin.

Rotation of a point about the $X$ axis by an angle $\alpha$ is performed by using the transformation

$$\mathbf{R}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-10}$$

Finally, rotation of a point about the $Y$ axis by an angle $\beta$ is achieved by using the transformation

$$\mathbf{R}_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-11}$$

**Concatenation and Inverse Transformations.** The application of several transformations can be represented by a single $4 \times 4$ transformation matrix. For example, translation, scaling, and rotation about the $Z$ axis of a point $\mathbf{v}$ is given by

$$\mathbf{v}^* = \mathbf{R}_\theta[\mathbf{S}(\mathbf{T}\mathbf{v})] = \mathbf{A}\mathbf{v} \tag{7.4-12}$$

where $\mathbf{A}$ is the $4 \times 4$ matrix $\mathbf{A} = \mathbf{R}_\theta\mathbf{S}\mathbf{T}$. It is important to note that these matrices generally do not commute, and so the order of application is important.

Although our discussion thus far has been limited to transformations of a single point, the same ideas extend to transforming a set of $m$ points simultaneously by using a single transformation. With reference to Eq. (7.4-5), let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ represent the coordinates of $m$ points. If we form a $4 \times m$ matrix $\mathbf{V}$ whose columns are these column vectors, then the simultaneous transformation of all these points by a $4 \times 4$ transformation matrix $\mathbf{A}$ is given by

$$\mathbf{V}^* = \mathbf{AV} \tag{7.4-13}$$

The resulting matrix $\mathbf{V}^*$ is $4 \times m$. Its $i$th column, $\mathbf{v}_i^*$, contains the coordinates of the transformed point corresponding to $\mathbf{v}_i$.

Before leaving this section, we point out that many of the transformations discussed above have inverse matrices that perform the opposite transformation and can be obtained by inspection. For example, the inverse translation matrix is given by

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-14}$$

Similarly, the inverse rotation matrix $\mathbf{R}_\theta^{-1}$ is given by

$$\mathbf{R}_\theta^{-1} = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) & 0 & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-15}$$

The inverse of more complex transformation matrices is usually obtained by numerical techniques.

### 7.4.2 Perspective Transformations

A *perspective transformation* (also called an *imaging transformation*) projects 3D points onto a plane. Perspective transformations play a central role in image processing because they provide an approximation to the manner in which an image is formed by viewing a three-dimensional world. Although perspective transformations will be expressed later in this section in a $4 \times 4$ matrix form, these transformations are fundamentally different from those discussed in the previous section because they are nonlinear in the sense that they involve division by coordinate values.

A model of the image formation process is shown in Fig. 7.13. We define the camera coordinate system $(x, y, z)$ as having the image plane coincident with the $xy$ plane, and optical axis (established by the center of the lens) along the $z$

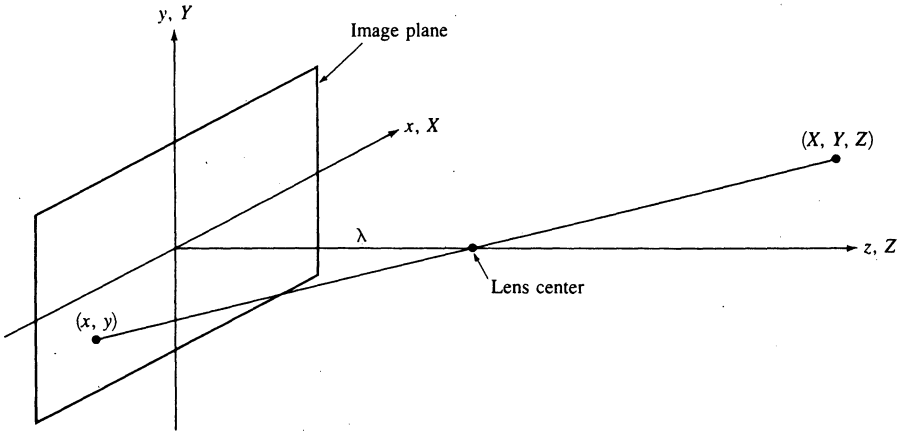


**Figure 7.13** Basic model of the imaging process. The camera coordinate system $(x, y, z)$ is aligned with the world coordinate system $(X, Y, Z)$.

axis. Thus, the center of the image plane is at the origin, and the center of the lens is at coordinates $(0, 0, \lambda)$. If the camera is in focus for distant objects, $\lambda$ is the *focal length* of the lens. In this section, it is assumed that the camera coordinate system is aligned with the world coordinate system $(X, Y, Z)$. This restriction will be removed in the following section.

Let $(X, Y, Z)$ be the world coordinates of any point in a 3D scene, as shown in Fig. 7.13. It will be assumed throughout the following discussion that $Z > \lambda$, that is, all points of interest lie in front of the lens. What we wish to do first is obtain a relationship that gives the coordinates $(x, y)$ of the projection of the point $(X, Y, Z)$ onto the image plane. This is easily accomplished by the use of similar triangles. With reference to Fig. 7.13, it follows that

$$\frac{x}{\lambda} = -\frac{X}{Z - \lambda} = \frac{X}{\lambda - Z} \tag{7.4-16}$$

and

$$\frac{y}{\lambda} = -\frac{Y}{Z - \lambda} = \frac{Y}{\lambda - Z} \tag{7.4-17}$$

where the negative signs in front of $X$ and $Y$ indicate that image points are actually inverted, as can be seen from the geometry of Fig. 7.13.

The image-plane coordinates of the projected 3D point follow directly from Eqs. (7.4-16) and (7.4-17):

$$x = \frac{\lambda X}{\lambda - Z} \tag{7.4-18}$$

and

$$y = \frac{\lambda Y}{\lambda - Z} \tag{7.4-19}$$

It is important to note that these equations are nonlinear because they involve division by the variable $Z$. Although we could use them directly as shown above, it is often convenient to express these equations in matrix form as we did in the previous section for rotation, translation, and scaling. This can be accomplished easily by using homogeneous coordinates.

The homogeneous coordinates of a point with cartesian coordinates $(X, Y, Z)$ are defined as $(kX, kY, kZ, k)$, where $k$ is an arbitrary, nonzero constant. Clearly, conversion of homogeneous coordinates back to cartesian coordinates is accomplished by dividing the first three homogeneous coordinates by the fourth. A point in the cartesian world coordinate system may be expressed in vector form as

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{7.4-20}$$

and its homogeneous counterpart is given by

$$\mathbf{w}_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{7.4-21}$$

If we define the *perspective transformation matrix*

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \tag{7.4-22}$$

Then the product $\mathbf{P}\mathbf{w}_h$ yields a vector which we shall denote by $\mathbf{c}_h$:

$$\mathbf{c}_h = \mathbf{P}\mathbf{w}_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ \frac{-kZ}{\lambda} + k \end{bmatrix} \tag{7.4-23}$$

The elements of $\mathbf{c}_h$ are the camera coordinates in homogeneous form. As indi-

cated above, these coordinates can be converted to cartesian form by dividing each of the first three components of $\mathbf{c}_h$ by the fourth. Thus, the cartesian coordinates of any point in the camera coordinate system are given in vector form by

$$\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \dfrac{\lambda X}{\lambda - Z} \\ \dfrac{\lambda Y}{\lambda - Z} \\ \dfrac{\lambda Z}{\lambda - Z} \end{bmatrix} \tag{7.4-24}$$

The first two components of $\mathbf{c}$ are the $(x, y)$ coordinates in the image plane of a projected 3D point $(X, Y, Z)$, as shown earlier in Eqs. (7.4-18) and (7.4-19). The third component is of no interest to us in terms of the model in Fig. 7.13. As will be seen below, this component acts as a free variable in the inverse perspective transformation.

The inverse perspective transformation maps an image point back into 3D. Thus, from Eq. (7.4-23),

$$\mathbf{w}_h = \mathbf{P}^{-1}\mathbf{c}_h \tag{7.4-25}$$

where $\mathbf{P}^{-1}$ is easily found to be

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \dfrac{1}{\lambda} & 1 \end{bmatrix} \tag{7.4-26}$$

Suppose that a given image point has coordinates $(x_0, y_0, 0)$, where the 0 in the $z$ location simply indicates the fact that the image plane is located at $z = 0$. This point can be expressed in homogeneous vector form as

$$\mathbf{c}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \tag{7.4-27}$$

Application of Eq. (7.4-25) then yields the homogeneous world coordinate vector

$$\mathbf{w}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \tag{7.4-28}$$

or, in cartesian coordinates,

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix} \tag{7.4-29}$$

This is obviously not what one would expect since it gives $Z = 0$ for *any* 3D point. The problem here is caused by the fact that mapping a 3D scene onto the image plane is a many-to-one transformation. The image point $(x_0, y_0)$ corresponds to the set of colinear 3D points which lie on the line that passes through $(x_0, y_0, 0)$ and $(0, 0, \lambda)$. The equations of this line in the world coordinate system are obtained from Eqs. (7.4-18) and (7.4-19); that is,

$$X = \frac{x_0}{\lambda}(\lambda - Z) \tag{7.4-30}$$

and

$$Y = \frac{y_0}{\lambda}(\lambda - Z) \tag{7.4-31}$$

These equations show that, unless we know something about the 3D point which generated a given image point (for example, its $Z$ coordinate), we cannot completely recover the 3D point from its image. This observation, which is certainly not unexpected, can be used as a way to formulate the inverse perspective transformation simply by using the $z$ component of $\mathbf{c}_h$ as a free variable instead of 0. Thus, letting

$$\mathbf{c}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ k \end{bmatrix} \tag{7.4-32}$$

we now have from Eq. (7.4-25) that

$$\mathbf{w}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ \frac{kz}{\lambda} + k \end{bmatrix} \tag{7.4-33}$$

which, upon conversion to cartesian coordinates, yields

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \dfrac{\lambda x_0}{\lambda + z} \\ \dfrac{\lambda y_0}{\lambda + z} \\ \dfrac{\lambda z}{\lambda + z} \end{bmatrix} \tag{7.4-34}$$

In other words, treating $z$ as a free variable yields the equations

$$\begin{aligned} X &= \frac{\lambda x_0}{\lambda + z} \\ Y &= \frac{\lambda y_0}{\lambda + z} \\ Z &= \frac{\lambda z}{\lambda + z} \end{aligned} \tag{7.4-35}$$

Solving for $z$ in terms of $Z$ in the last equation and substituting in the first two expressions yields

$$X = \frac{x_0}{\lambda}(\lambda - Z) \tag{7.4-36}$$

$$Y = \frac{y_0}{\lambda}(\lambda - Z) \tag{7.4-37}$$

which agrees with the above observation that recovering a 3D point from its image by means of the inverse perspective transformation requires knowledge of at least one of the world coordinates of the point. This problem will be addressed again in Sec. 7.4.5.

### 7.4.3 Camera Model

Equations (7.4-23) and (7.4-24) characterize the formation of an image via the projection of 3D points onto an image plane. These two equations thus constitute a basic mathematical model of an imaging camera. This model is based on the assumption that the camera and world coordinate systems are coincident. In this section we consider a more general problem in which the two coordinate systems are allowed to be separate. However, the basic objective of obtaining the image-plane coordinates of any given world point remains the same.

The situation is depicted in Fig. 7.14, which shows a world coordinate system $(X, Y, Z)$ used to locate both the camera and 3D points (denoted by $\mathbf{w}$). This

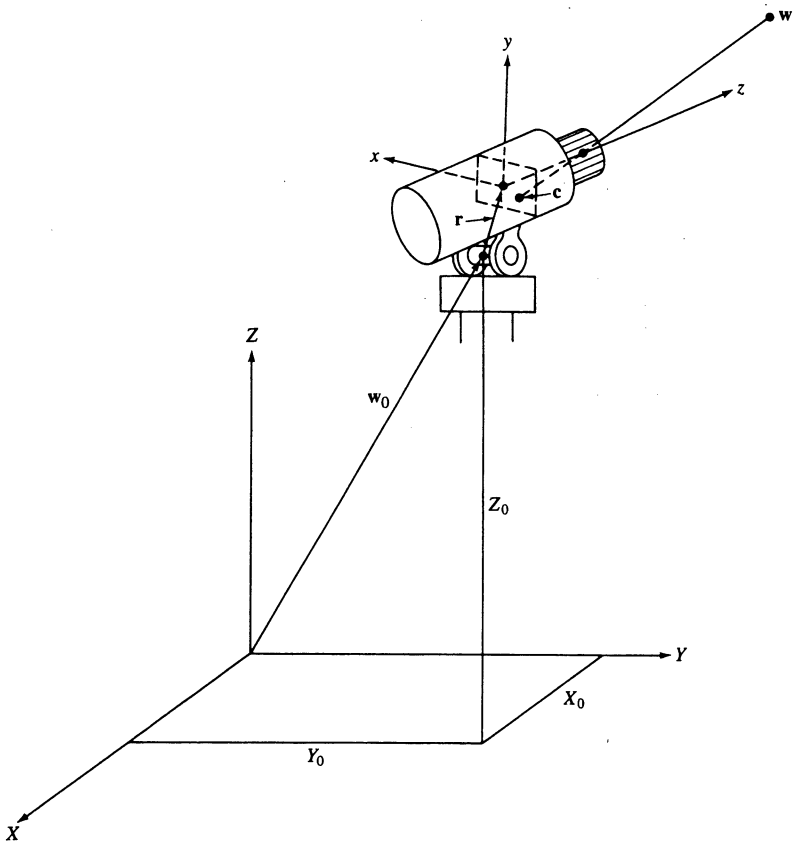


**Figure 7.14** Imaging geometry with two coordinate systems.

figure also shows the camera coordinate system $(x, y, z)$ and image points (denoted by **c**). It is assumed that the camera is mounted on a gimbal which allows pan through an angle $\theta$ and tilt through an angle $\alpha$. In this discussion, pan is defined as the angle between the $x$ and $X$ axes, and tilt as the angle between the $z$ and $Z$ axes. The offset of the center of the gimbal from the origin of the world coordinate system is denoted by vector $\mathbf{w}_0$, and the offset of the center of the imaging plane with respect to the gimbal center is denoted by a vector **r**, with components $(r_1, r_2, r_3)$.

The concepts developed in the last two sections provide all the necessary tools to derive a camera model based on the geometrical arrangement of Fig. 7.14. The approach is to bring the camera and world coordinate systems into alignment by applying a set of transformations. After this has been accomplished, we simply apply the perspective transformation given in Eq. (7.4-22) to obtain the image-plane coordinates of any given world point. In other words, we first reduce the

problem to the geometrical arrangement shown in Fig. 7.13 before applying the perspective transformation.

Suppose that, initially, the camera was in *normal position*, in the sense that the gimbal center and origin of the image plane were at the origin of the world coordinate system, and all axes were aligned. Starting from normal position, the geometrical arrangement of Fig. 7.14 can be achieved in a number of ways. We assume the following sequence of steps: (1) displacement of the gimbal center from the origin, (2) pan of the $x$ axis, (3) tilt of the $z$ axis, and (4) displacement of the image plane with respect to the gimbal center.

The sequence of *mechanical* steps just discussed obviously does not affect the world points since the set of points seen by the camera after it was moved from normal position is quite different. However, we can achieve normal position again simply by applying exactly the same sequence of steps to all world points. Since a camera in normal position satisfies the arrangement of Fig. 7.13 for application of the perspective transformation, our problem is thus reduced to applying to every world point a set of transformations which correspond to the steps given above.

Translation of the origin of the world coordinate system to the location of the gimbal center is accomplished by using the following transformation matrix:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-38}$$

In other words, a homogeneous world point $\mathbf{w}_h$ that was at coordinates $(X_0, Y_0, Z_0)$ is at the origin of the new coordinate system after the transformation $\mathbf{G}\mathbf{w}_h$.

As indicated earlier, the pan angle is measured between the $x$ and $X$ axes. In normal position, these two axes are aligned. In order to pan the $x$ axis through the desired angle, we simply rotate it by $\theta$. The rotation is with respect to the $z$ axis and is accomplished by using the transformation matrix $\mathbf{R}_\theta$ given in Eq. (7.4-9). In other words, application of this matrix to all points (including the point $\mathbf{G}\mathbf{w}_h$) effectively rotates the $x$ axis to the desired location. When using Eq. (7.4-9), it is important to keep clearly in mind the convention established in Fig. 7.12. That is, angles are considered positive when points are rotated clockwise, which implies a counterclockwise rotation of the camera about the $z$ axis. The unrotated (0°) position corresponds to the case when the $x$ and $X$ axes are aligned.

At this point in the development the $z$ and $Z$ axes are still aligned. Since tilt is the angle between these two axes, we tilt the camera an angle $\alpha$ by rotating the $z$ axis by $\alpha$. The rotation is with respect to the $x$ axis and is accomplished by applying the transformation matrix $\mathbf{R}_\alpha$ given in Eq. (7.4-10) to all points (including the point $\mathbf{R}_\theta\mathbf{G}\mathbf{w}_h$). As above, a counterclockwise rotation of the camera implies positive angles, and the 0° mark is where the $z$ and $Z$ axes are aligned.†

† A useful way to visualize these transformations is to construct an axis system (e.g., with pipe cleaners), label the axes $x$, $y$, and $z$, and perform the rotations manually, one axis at a time.

According to the discussion in Sec. 7.4.4, the two rotation matrices can be concatenated into a single matrix, $\mathbf{R} = \mathbf{R}_\alpha \mathbf{R}_\theta$. It then follows from Eqs. (7.4-9) and (7.4-10) that

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & 0 \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-39}$$

Finally, displacement of the origin of the image plane by vector $\mathbf{r}$ is achieved by the transformation matrix

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4-40}$$

Thus, by applying to $\mathbf{w}_h$ the series of transformations $\mathbf{CRGw}_h$ we have brought the world and camera coordinate systems into coincidence. The image-plane coordinates of a point $\mathbf{w}_h$ are finally obtained by using Eq. (7.4-22). In other words, a homogeneous world point which is being viewed by a camera satisfying the geometrical arrangement shown in Fig. 7.14 has the following homogeneous representation in the camera coordinate system:

$$\mathbf{c}_h = \mathbf{PCRGw}_h \tag{7.4-41}$$

This equation represents a perspective transformation involving two coordinate systems.

As indicated in Sec. 7.4.2, we obtain the cartesian coordinates $(x, y)$ of the imaged point by dividing the first and second components of $\mathbf{c}_h$ by the fourth. Expanding Eq. (7.4-41) and converting to cartesian coordinates yields

$$x = \lambda \frac{(X-X_0)\cos\theta + (Y-Y_0)\sin\theta - r_1}{-(X-X_0)\sin\theta\sin\alpha + (Y-Y_0)\cos\theta\sin\alpha - (Z-Z_0)\cos\alpha + r_3 + \lambda} \tag{7.4-42}$$

and

$$y = \lambda \frac{-(X-X_0)\sin\theta\cos\alpha + (Y-Y_0)\cos\theta\cos\alpha + (Z-Z_0)\sin\alpha - r_2}{-(X-X_0)\sin\theta\sin\alpha + (Y-Y_0)\cos\theta\sin\alpha - (Z-Z_0)\cos\alpha + r_3 + \lambda} \tag{7.4-43}$$

which are the image coordinates of a point $\mathbf{w}$ whose world coordinates are $(X, Y, Z)$. It is noted that these equations reduce to Eqs. (7.4-18) and (7.4-19) when $X_0 = Y_0 = Z_0 = 0$, $r_1 = r_2 = r_3 = 0$, and $\alpha = \theta = 0°$.

**Example:** As an illustration of the concepts just discussed, suppose that we wish to find the image coordinates of the corner of the block shown in Fig.

7.15. The camera is offset from the origin and is viewing the scene with a pan of 135° and a tilt of 135°. We will follow the convention established above that transformation angles are positive when the camera rotates in a counterclockwise manner when viewing the origin along the axis of rotation.

Let us examine in detail the steps required to move the camera from normal position to the geometry shown in Fig. 7.15. The camera is shown in normal position in Fig. 7.16*a*, and displaced from the origin in Fig. 7.16*b*. It is important to note that, after this step, the world coordinate axes are used *only* to establish angle references. That is, after displacement of the world-coordinate origin, all rotations take place about the new (camera) axes. Figure 7.16*c* shows a view along the $z$ axis of the camera to establish pan. In this case the rotation of the camera about the $z$ axis is counterclockwise so world points are rotated about this axis in the opposite direction, which makes $\theta$ a positive angle. Figure 7.16*d* shows a view after pan, along the $x$ axis of the camera to establish tilt. The rotation about this axis is counterclockwise, which makes $\alpha$ a positive angle. The world coordinate axes are shown dashed in the latter two figures to emphasize the fact that their only use is to establish the zero reference for the pan and tilt angles. We do not show in this figure the final step of displacing the image plane from the center of the gimbal.

The following parameter values apply to the problem:

$$X_0 = 0\,\text{m}$$

$$Y_0 = 0\,\text{m}$$

$$Z_0 = 1\,\text{m}$$

$$\alpha = 135°$$

$$\theta = 135°$$

$$r_1 = 0.03\,\text{m}$$

$$r_2 = r_3 = 0.02\,\text{m}$$

$$\lambda = 35\,\text{mm} = 0.035\,\text{m}$$

The corner in question is at coordinates $(X, Y, Z) = (1, 1, 0.2)$.

To compute the image coordinates of the block corner, we simply substitute the above parameter values into Eqs. (7.4-42) and (7.4-43); that is,

$$x = \lambda \frac{-0.03}{-1.53 + \lambda}$$

and

$$y = \lambda \frac{-0.42}{-1.53 + \lambda}$$

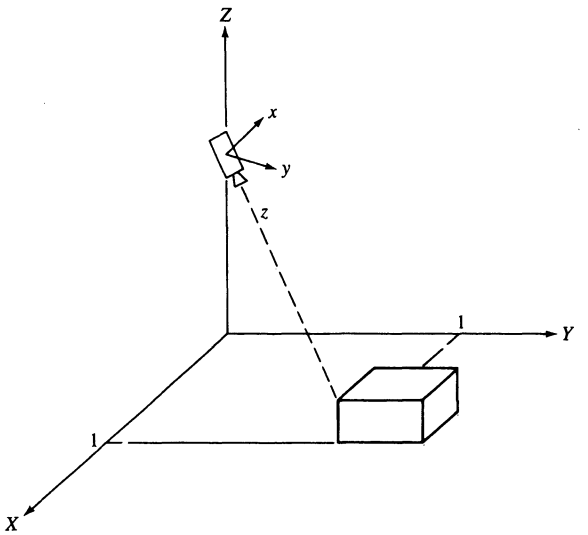


**Figure 7.15** Camera viewing a 3D scene.

Substituting $\lambda = 0.035$ yields the image coordinates

$$x = 0.0007\,\text{m}$$

and

$$y = 0.009\,\text{m}$$

It is of interest to note that these coordinates are well within a $1 \times 1$ inch ($0.025 \times 0.025$ m) imaging plane. If, for example, we had used a lens with a 200-mm focal length, it is easily verified from the above results that the corner of the block would have been imaged outside the boundary of a plane with these dimensions (i.e., it would have been outside the effective field of view of the camera).

Finally, we point out that all coordinates obtained via the use of Eqs. (7.4-42) and (7.4-43) are with respect to the center of the image plane. A change of coordinates would be required to use the convention established earlier, in which the origin of an image is at its top left corner. □

## 7.4.4 Camera Calibration

In Sec. 7.4.3 we obtained explicit equations for the image coordinates $(x, y)$ of a world point $\mathbf{w}$. As shown in Eqs. (7.4-42) and (7.4-43), implementation of these equations requires knowledge of the focal length, camera offsets, and angles of pan and tilt. While these parameters could be measured directly, it is often more convenient (e.g., when the camera moves frequently) to determine one or more of the

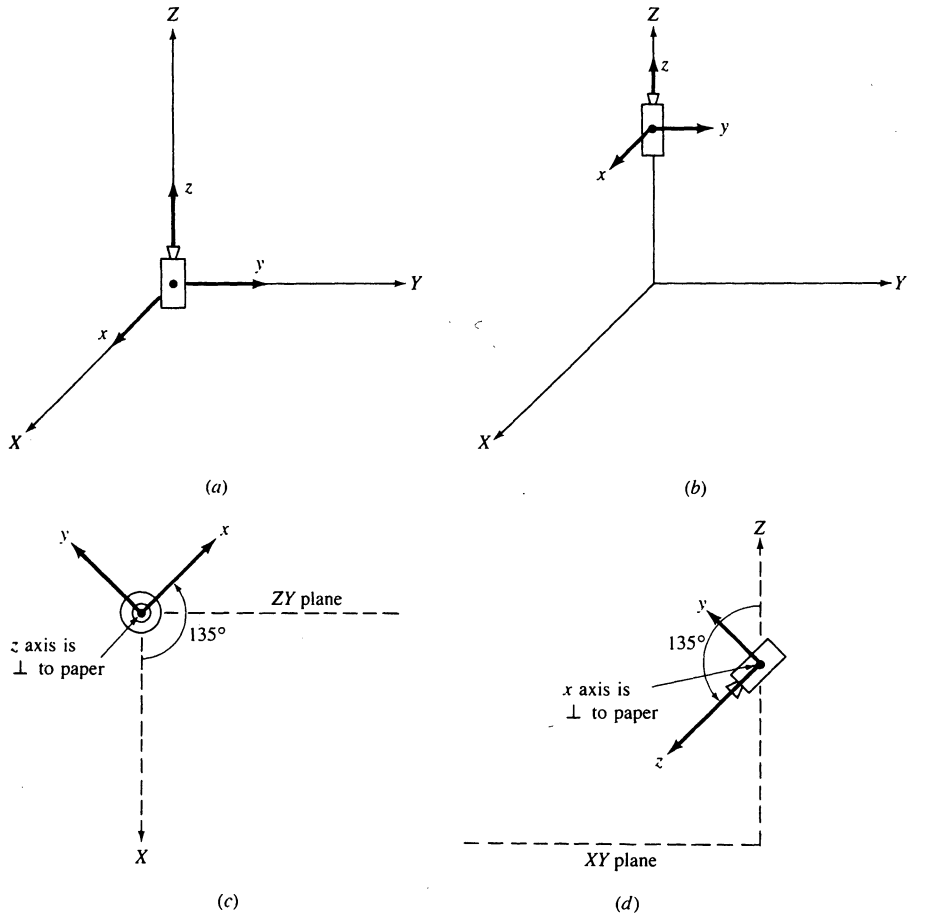


**Figure 7.16** (*a*) Camera in normal position. (*b*) Gimbal center displaced from origin. (*c*) Observer view of rotation about $z$ axis to determine pan angle. (*d*) Observer view of rotation about $x$ axis for tilt.

parameters by using the camera itself as a measuring device. This requires a set of image points whose world coordinates are known, and the computational procedure used to obtain the camera parameters using these known points is often referred to as *camera calibration.*

With reference to Eq. (7.4-41), let $\mathbf{A} = \mathbf{PCRG}$. The elements of $\mathbf{A}$ contain all the camera parameters, and we know from Eq. (7.4-41) that $\mathbf{c}_h = \mathbf{A}\mathbf{w}_h$. Letting $k = 1$ in the homogeneous representation, we may write

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7.4-44}$$

From the discussion in the previous two sections we know that the camera coordinates in cartesian form are given by

$$x = \frac{c_{h1}}{c_{h4}} \tag{7.4-45}$$

and

$$y = \frac{c_{h2}}{c_{h4}} \tag{7.4-46}$$

Substituting $c_{h1} = xc_{h4}$ and $c_{h2} = yc_{h4}$ in Eq. (7.6-44) and expanding the matrix product yields

$$\begin{aligned} xc_{h4} &= a_{11}X + a_{12}Y + a_{13}Z + a_{14} \\ yc_{h4} &= a_{21}X + a_{22}Y + a_{23}Z + a_{24} \\ c_{h4} &= a_{41}X + a_{42}Y + a_{43}Z + a_{44} \end{aligned} \tag{7.4-47}$$

where expansion of $c_{h3}$ has been ignored because it is related to $z$.

Substitution of $c_{h4}$ in the first two equations of (7.4-47) yields two equations with twelve unknown coefficients:

$$a_{11}X + a_{12}Y + a_{13}Z - a_{41}xX - a_{42}xY - a_{43}xZ - a_{44}x + a_{14} = 0 \tag{7.4-48}$$

$$a_{21}X + a_{22}Y + a_{23}Z - a_{41}yX - a_{42}yY - a_{43}yZ - a_{44}y + a_{24} = 0 \tag{7.4-49}$$

The calibration procedure then consists of (1) obtaining $m \geqslant 6$ world points with known coordinates $(X_i, Y_i, Z_i)$, $i = 1, 2, \ldots, m$ (there are *two* equations involving the coordinates of these points, so at least six points are needed), (2) imaging these points with the camera in a given position to obtain the corresponding image points $(x_i, y_i)$, $i = 1, 2, \ldots, m$, and (3) using these results in Eqs. (7.4-48) and (7.4-49) to solve for the unknown coefficients. There are many numerical techniques for finding an optimal solution to a linear system of equations such as (7.4-48) and (7.4-49) (see, for example, Noble [1969]).

### 7.4.5 Stereo Imaging

It was noted in Sec. 7.4.2 that mapping a 3D scene onto an image plane is a many-to-one transformation. That is, an image point does not uniquely determine the location of a corresponding world point. It is shown in this section that the missing *depth* information can be obtained by using stereoscopic (*stereo* for short) imaging techniques.

As shown in Fig. 7.17, stereo imaging involves obtaining two separate image views of an object of interest (e.g., a world point w). The distance between the centers of the two lenses is called the *baseline*, and the objective is to find the coordinates $(X, Y, Z)$ of a point w given its image points $(x_1, y_1)$ and $(x_2, y_2)$. It is assumed that the cameras are identical and that the coordinate systems of both

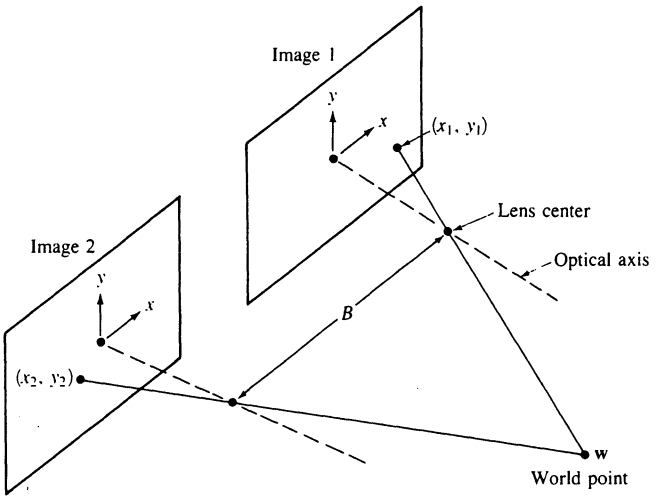


**Figure 7.17** Model of the stereo imaging process.

cameras are perfectly aligned, differing only in the location of their origins, a condition usually met in practice. Recall our convention that, after the camera and world coordinate systems have been brought into coincidence, the $xy$ plane of the image is aligned with the $XY$ plane of the world coordinate system. Then, under the above assumption, the $Z$ coordinate of **w** is exactly the same for both camera coordinate systems.

Suppose that we bring the first camera into coincidence with the world coordinate system, as shown in Fig. 7.18. Then, from Eq. (7.4-31), **w** lies on the line with (partial) coordinates

$$X_1 = \frac{x_1}{\lambda}(\lambda - Z_1) \tag{7.4-50}$$

where the subscripts on $X$ and $Z$ indicate that the first camera was moved to the origin of the world coordinate system, with the second camera and **w** following, but keeping the relative arrangement shown in Fig. 7.17. If, instead, the second camera had been brought to the origin of the world coordinate system, then we would have that **w** lies on the line with (partial) coordinates

$$X_2 = \frac{x_2}{\lambda}(\lambda - Z_2) \tag{7.4-51}$$

However, due to the separation between cameras and the fact that the $Z$ coordinate of **w** is the same for both camera coordinate systems, it follows that

$$X_2 = X_1 + B \tag{7.4-52}$$

and

$$Z_2 = Z_1 = Z \tag{7.4-53}$$

where, as indicated above, $B$ is the baseline distance.

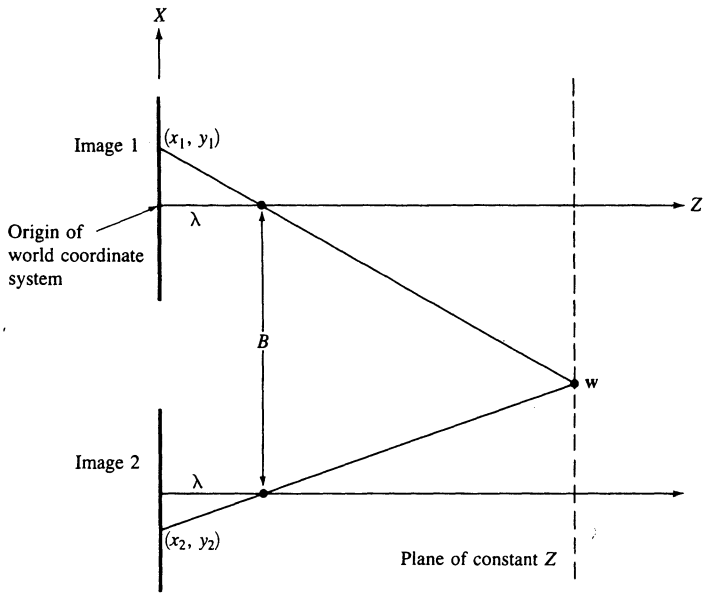


**Figure 7.18** Top view of Fig. 7.17 with the first camera brought into coincidence with the world coordinate system.

Substitution of Eqs. (7.4-52) and (7.4-53) into Eqs. (7.4-50) and (7.4-51) results in the following equations:

$$X_1 + B = \frac{x_2}{\lambda}(\lambda - Z) \tag{7.4-54}$$

and

$$X_1 = \frac{x_1}{\lambda}(\lambda - Z) \tag{7.4-55}$$

Subtracting Eq. (7.4-55) from (7.4-54) and solving for $Z$ yields the expression

$$Z = \lambda - \frac{\lambda B}{x_2 - x_1} \tag{7.4-56}$$

which indicates that if the difference between the corresponding image coordinates $x_2$ and $x_1$ can be determined, and the baseline and focal length are known, calculating the $Z$ coordinate of **w** is a simple matter. The $X$ and $Y$ world coordinates then follow directly from Eqs. (7.4-30) and (7.4-31) using either $(x_1, y_1)$ or $(x_2, y_2)$.

The most difficult task in using Eq. (7.4-56) to obtain $Z$ is to actually find two corresponding points in different images of the same scene. Since these points are generally in the same vicinity, a frequently used approach is to select a point within a small region in one of the image views and then attempt to find the best matching region in the other view by using correlation techniques, as discussed in

Chap. 8. When the scene contains distinct features, such as prominent corners, a feature-matching approach will generally yield a faster solution for establishing correspondence.

Before leaving this discussion, we point out that the calibration procedure developed in the previous section is directly applicable to stereo imaging by simply treating the cameras independently.

## 7.5 SOME BASIC RELATIONSHIPS BETWEEN PIXELS

In this section we consider several primitive, but important relationships between pixels in a digital image. As in the previous sections, an image will be denoted by $f(x, y)$. When referring to a particular pixel, we will use lower-case letters, such as $p$ and $q$. A subset of pixels of $f(x, y)$ will be denoted by $S$.

### 7.5.1 Neighbors of a Pixel

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

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

This set of pixels, called the 4-*neighbors* of $p$, will be denoted by $N_4(p)$. It is noted that each of these pixels is a unit distance from $(x, y)$ and also that some of the neighbors of $p$ will be outside the digital image if $(x, y)$ is on the border of the image.

The four *diagonal* neighbors of $p$ have coordinates

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

and will be denoted $N_D(p)$. These points, together with the 4-neighbors defined above, are called the 8-*neighbors* of $p$, denoted $N_8(p)$. As before, some of the points in $N_D(p)$ and $N_8(p)$ will be outside the image if $(x, y)$ is on the border to the image.

### 7.5.2 Connectivity

Let $V$ be the set of intensity values of pixels which are allowed to be connected; for example, if only connectivity of pixels with intensities of 59, 60, and 61 is desired, then $V = \{59, 60, 61\}$. We consider three types of connectivity:

1. 4-*connectivity*. Two pixels $p$ and $q$ with values from $V$ are 4-connected if $q$ is in the set $N_4(p)$.
2. 8-*connectivity*. Two pixels $p$ and $q$ with values from $V$ are 8-connected if $q$ is in the set $N_8(p)$.
3. m-*connectivity* (mixed connectivity). Two pixels $p$ and $q$ with values from $V$ are m-connected if

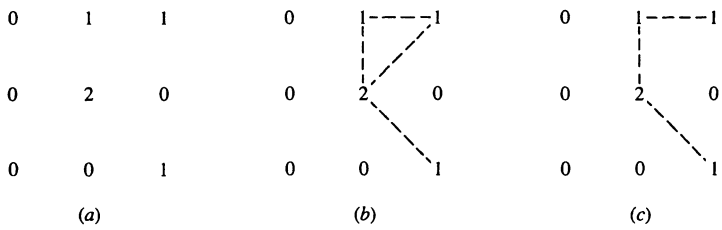


**Figure 7.19** (*a*) Arrangement of pixels. (*b*) 8-neighbors of the pixel labeled "2." (*c*) m-neighbors of the same pixel.

(*a*) $q$ is in $N_4(p)$, *or*
(*b*) $q$ is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ is empty. (This is the set of pixels that are 4-neighbors of both $p$ and $q$ and whose values are from $V$.)

Mixed connectivity is a modification of 8-connectivity and is introduced to eliminate the multiple connections which often cause difficulty when 8-connectivity is used. For example, consider the pixel arrangement shown in Fig. 7.19*a*. Assuming $V = \{1, 2\}$, the 8-neighbors of the pixel with value 2 are shown by dashed lines in Fig. 7.19*b*. It is important to note the ambiguity that results from multiple connections to this pixel. This ambiguity is removed by using m-connectivity, as shown in Fig. 7.19*c*.

A pixel $p$ is *adjacent* to a pixel $q$ if they are connected. We may define 4-, 8-, or m-adjacency, depending on the type of connectivity specified. Two image subsets $S_1$ and $S_2$ are adjacent if some pixel in $S_1$ is adjacent to some pixel in $S_2$.

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

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

where $(x_0, y_0) = (x, y)$ and $(x_n, y_n) = (s, t)$, $(x_i, y_i)$ is adjacent to $(x_{i-1}, y_{i-1})$, $1 \leqslant i \leqslant n$, and $n$ is the *length* of the path. We may define 4-, 8-, or m-paths, depending on the type of adjacency used.

If $p$ and $q$ are pixels of an image subset $S$, then $p$ is *connected* to $q$ in $S$ if there is a path from $p$ to $q$ consisting entirely of pixels in $S$. For any pixel $p$ in $S$, the set of pixels in $S$ that are connected to $p$ is called a *connected component* of $S$. It then follows that any two pixels of a connected component are connected to each other, and that distinct connected components are disjoint.

### 7.5.3 Distance Measures

Given pixels $p$, $q$, and $z$, with coordinates $(x, y)$, $(s, t)$, and $(u, v)$, respectively, we call $D$ a *distance function* or *metric* if

1. $D(p, q) \geqslant 0 \quad [D(p, q) = 0 \text{ iff } p = q]$
2. $D(p, q) = D(q, p)$
3. $D(p, z) \leqslant D(p, q) + D(q, z)$

The *euclidean distance* between $p$ and $q$ is defined as

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2} \tag{7.5-1}$$

For this distance measure, the pixels having a distance less than or equal to some value $r$ from $(x, y)$ are the points contained in a disk of radius $r$ centered at $(x, y)$.

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

$$D_4(p, q) = |x - s| + |y - t| \tag{7.5-2}$$

In this case the pixels having a $D_4$ distance less than or equal to some value $r$ from $(x, y)$ form a diamond centered at $(x, y)$. For example, the pixels with $D_4$ distance $\leqslant 2$ from $(x, y)$ (the center point) form the following contours of constant distance:

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

It is noted that the pixels with $D_4 = 1$ are the 4-neighbors of $(x, y)$.

The $D_8$ *distance* (also called *chessboard distance*) between $p$ and $q$ is defined as

$$D_8(p, q) = \max(|x - s|, |y - t|) \tag{7.5-3}$$

In this case the pixels with $D_8$ distance less than or equal to some value $r$ form a square centered at $(x, y)$. For example, the pixels with $D_8$ distance $\leqslant 2$ 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 pixels with $D_8 = 1$ are the 8-neighbors of $(x, y)$.

It is of interest to note that the $D_4$ distance between two points $p$ and $q$ is equal to the length of the shortest 4-path between these two points. Similar comments apply to the $D_8$ distance. In fact, we can consider both the $D_4$ and $D_8$ distances between $p$ and $q$ regardless of whether or not a connected path exists between them, since the definition of these distances involve only the coordinates of these points. When dealing with m-connectivity, however, the value of the distance (length of the path) between two pixels depends on the values of the pixels

along the path as well as their neighbors. For instance, consider the following arrangement of pixels, where it is assumed that $p$, $p_2$, and $p_4$ are valued 1 and $p_1$ and $p_3$ may be valued 0 or 1:

$$\begin{matrix} & p_3 & p_4 \\ p_1 & p_2 & \\ p & & \end{matrix}$$

If we only allow connectivity of pixels valued 1, and $p_1$ and $p_3$ are 0, the m-distance between $p$ and $p_4$ is 2. If either $p_1$ or $p_3$ is 1, the distance is 3. If both $p_1$ and $p_3$ are 1, the distance is 4.

## 7.6 PREPROCESSING

In this section we discuss several preprocessing approaches used in robotic vision systems. Although the number of techniques available for preprocessing general image data is significant, only a subset of these methods satisfies the requirements of computational speed and low implementation cost, which are essential elements of an industrial vision system. The range of preprocessing approaches discussed in this section are typical of methods that satisfy these requirements.

### 7.6.1 Foundation

In this section we consider two basic approaches to preprocessing. The first is based on spatial-domain techniques and the second deals with frequency-domain concepts via the Fourier transform. Together, these approaches encompass most of the preprocessing algorithms used in robot vision systems.

**Spatial-Domain Methods.** The *spatial domain* refers to the aggregate of pixels composing an image, and spatial-domain methods are procedures that operate directly on these pixels. Preprocessing functions in the spatial domain may be expressed as

$$g(x, y) = h[f(x, y)] \tag{7.6-1}$$

where $f(x, y)$ is the input image, $g(x, y)$ is the resulting (preprocessed) image, and $h$ is an operator on $f$, defined over some neighborhood of $(x, y)$. It is also possible to let $h$ operate on a *set* of input images, such as performing the pixel-by-pixel sum of $K$ images for noise reduction, as discussed in Sec. 7.6.2.

The principal approach used in defining a neighborhood about $(x, y)$ is to use a square or rectangular subimage area centered at $(x, y)$, as shown in Fig. 7.20. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner, and applying the operator at each location $(x, y)$ to yield $g(x, y)$. Although other neighborhood shapes, such as a circle, are sometimes used, square arrays are by far the most predominant because of their ease of implementation.

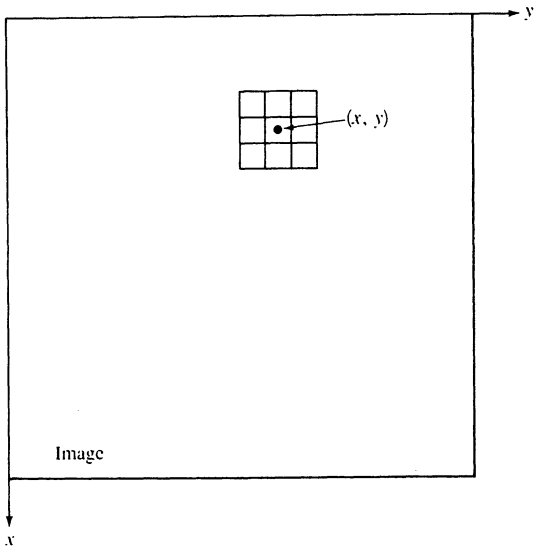


**Figure 7.20** A 3 × 3 neighborhood about a point $(x, y)$ in an image.

The simplest form of $h$ is when the neighborhood is $1 \times 1$ and, therefore, $g$ depends only on the value of $f$ at $(x, y)$. In this case $h$ becomes an *intensity mapping* or *transformation* $T$ of the form

$$s = T(r) \tag{7.6-2}$$

where, for simplicity, we have used $s$ and $r$ as variables denoting, respectively, the intensity of $f(x, y)$ and $g(x, y)$ at any point $(x, y)$. This type of transformation is discussed in more detail in Sec. 7.6.3.

One of the spatial-domain techniques used most frequently is based on the use of so-called *convolution masks* (also referred to as *templates*, *windows*, or *filters*). Basically, a mask is a small (e.g., 3 × 3) two-dimensional array, such as the one shown in Fig. 7.20, whose coefficients are chosen to detect a given property in an image. As an introduction to this concept, suppose that we have an image of constant intensity which contains widely isolated pixels whose intensities are different from the background. These points can be detected by using the mask shown in Fig. 7.21. The procedure is as follows: The center of the mask (labeled 8) is moved around the image, as indicated above. At each pixel position in the image, we multiply every pixel that is contained within the mask area by the corresponding mask coefficient; that is, the pixel in the center of the mask is multiplied by 8, while its 8-neighbors are multiplied by $-1$. The results of these nine multiplications are then summed. If all the pixels within the mask area have the same value (constant background), the sum will be zero. If, on the other hand, the center of the mask is located at one of the isolated points, the sum will be different

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

**Figure 7.21** A mask for detecting isolated points different from a constant background.

from zero. If the isolated point is in an off-center position, the sum will also be different from zero, but the magnitude of the response will be weaker. These weaker responses can be eliminated by comparing the sum against a threshold.

As shown in Fig. 7.22, if we let $w_1, w_2, \ldots, w_9$ represent mask coefficients and consider the 8-neighbors of $(x, y)$, we may generalize the preceding discussion as that of performing the following operation:

$$\begin{aligned} h[f(x, y)] = {} & w_1 f(x-1, y-1) + w_2 f(x-1, y) + w_3 f(x-1, y+1) \\ & + w_4 f(x, y-1) + w_5 f(x, y) + w_6 f(x, y+1) \\ & + w_7 f(x+1, y-1) + w_8 f(x+1, y) \\ & + w_9 f(x+1, y+1) \end{aligned} \tag{7.6-3}$$

on a $3 \times 3$ neighborhood of $(x, y)$.

| | | |
|---|---|---|
| $w_1$ $(x-1, y-1)$ | $w_2$ $(x-1, y)$ | $w_3$ $(x-1, y+1)$ |
| $w_4$ $(x, y-1)$ | $w_5$ $(x, y)$ | $w_6$ $(x, y+1)$ |
| $w_7$ $(x+1, y-1)$ | $w_8$ $(x+1, y)$ | $w_9$ $(x+1, y+1)$ |

**Figure 7.22** A general $3 \times 3$ mask showing coefficients and corresponding image pixel locations.

Before leaving this section, we point out that the concept of neighborhood processing is not limited to $3 \times 3$ areas nor to the cases treated thus far. For instance, we will use neighborhood operations in subsequent discussions for noise reduction, to obtain variable image thresholds, to compute measures of texture, and to obtain the skeleton of an object.

**Frequency-Domain Methods.** The *frequency domain* refers to an aggregate of complex pixels resulting from taking the Fourier transform of an image. The concept of "frequency" is often used in interpreting the Fourier transform and arises from the fact that this particular transform is composed of complex sinusoids. Due to extensive processing requirements, frequency-domain methods are not nearly as widely used in robotic vision as are spatial-domain techniques. However, the Fourier transform does play an important role in areas such as the analysis of object motion and object description. In addition, many spatial techniques for enhancement and restoration are founded on concepts whose origins can be traced to a Fourier transform formulation. The material in this section will serve as an introduction to these concepts. A more extensive treatment of the Fourier transform and its properties may be found in Gonzalez and Wintz [1977].

We begin the discussion by considering discrete functions of one variable, $f(x)$, $x = 0, 1, 2, \ldots, N - 1$. The forward Fourier transform of $f(x)$ is defined as

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi ux/N} \tag{7.6-4}$$

for $u = 0, 1, 2, \ldots, N - 1$. In this equation $j = \sqrt{-1}$ and $u$ is the so-called *frequency variable*. The inverse Fourier transform of $F(u)$ yields $f(x)$ back, and is defined as

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{j2\pi ux/N} \tag{7.6-5}$$

for $x = 0, 1, 2, \ldots, N - 1$. The validity of these expressions, called the *Fourier transform pair*, is easily verified by substituting Eq. (7.6-4) for $F(u)$ in Eq. (7.6-5), or vice versa. In either case we would get an identity.

A direct implementation of Eq. (7.6-4) for $u = 0, 1, 2, \ldots, N - 1$ would require on the order of $N^2$ additions and multiplications. Use of a fast Fourier transform (FFT) algorithm significantly reduces this number to $N \log_2 N$, where $N$ is assumed to be an integer power of 2. Similar comments apply to Eq. (7.6-5) for $x = 0, 1, 2, \ldots, N - 1$. A number of FFT algorithms are readily available in a variety of computer languages.

The two-dimensional Fourier transform pair of an $N \times N$ image is defined as

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux + vy)/N} \tag{7.6-6}$$

for $u, v = 0, 1, 2, \ldots, N - 1$, and

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux + vy)/N} \tag{7.6-7}$$

for $x, y = 0, 1, 2, \ldots, N - 1$. It is possible to show through some manipulation that each of these equations can be expressed as separate one-dimensional summations of the form shown in Eq. (7.6-4). This leads to a straightforward procedure for computing the two-dimensional Fourier transform using only a one-dimensional FFT algorithm: We first compute and save the transform of each row of $f(x, y)$, thus producing a two-dimensional array of intermediate results. These results are multiplied by $N$ and the one-dimensional transform of each column is computed. The final result is $F(u, v)$. Similar comments apply for computing $f(x, y)$ given $F(u, v)$. The order of computation from a row-column approach can be reversed to a column-row format without affecting the final result.

The Fourier transform can be used in a number of ways by a vision system, as will be shown in Chap. 8. For example, by treating the boundary of an object as a one-dimensional array of points and computing their Fourier transform, selected values of $F(u)$ can be used as descriptors of boundary shape. The one-dimensional Fourier transform has also been used as a powerful tool for detecting object motion. Applications of the discrete two-dimensional Fourier transform in image reconstruction, enhancement, and restoration are abundant although, as mentioned earlier, the usefulness of this approach in industrial machine vision is still quite restricted due to the extensive computational requirements needed to implement this transform. We point out before leaving this section, however, that the two-dimensional, continuous Fourier transform can be computed (at the speed of light) by optical means. This approach, which requires the use of precisely aligned optical equipment, is used in industrial environments for tasks such as the inspection of finished metal surfaces. Further treatment of this topic is outside the scope of our present discussion, but the interested reader is referred to the book by Goodman [1968] for an excellent introduction to Fourier optics.

### 7.6.2 Smoothing

Smoothing operations are used for reducing noise and other spurious effects that may be present in an image as a result of sampling, quantization, transmission, or disturbances in the environment during image acquisition. In this section we consider several fast smoothing methods that are suitable for implementation in the vision system of a robot.

**Neighborhood Averaging.** Neighborhood averaging is a straightforward spatial-domain technique for image smoothing. Given an image $f(x, y)$, the procedure is to generate a smoothed image $g(x, y)$ whose intensity at every point $(x, y)$ is obtained by averaging the intensity values of the pixels of $f$ contained in a

predefined neighborhood of $(x, y)$. In other words, the smoothed image is obtained by using the relation

$$g(x, y) = \frac{1}{P} \sum_{(n, m) \in S} f(n, m) \tag{7.6-8}$$

for all $x$ and $y$ in $f(x, y)$. $S$ is the set of coordinates of points in the neighborhood of $(x, y)$, including $(x, y)$ itself, and $P$ is the total number of points in the neighborhood. If a $3 \times 3$ neighborhood is used, we note by comparing Eqs. (7.6-8) and (7.6-3) that the former equation is a special case of the latter with $w_i = 1/9$. Of course, we are not limited to square neighborhoods in Eq. (7.6-8) but, as mentioned in Sec. 7.6.1, these are by far the most predominant in robot vision systems.

**Example:** Figure 7.23 illustrates the smoothing effect produced by neighborhood averaging. Figure 7.23*a* shows an image corrupted by noise, and Fig. 7.23*b* is the result of averaging every pixel with its 4-neighbors. Similarly, Figs. 7.23*c* through *f* are the results of using neighborhoods of sizes $3 \times 3$, $5 \times 5$, $7 \times 7$, and $11 \times 11$, respectively. It is noted that the degree of smoothing is strongly proportional to the size of the neighborhood used. As is true with most mask processors, the smoothed value of each pixel is determined before any of the other pixels have been changed. □

**Median Filtering.** One of the principal difficulties of neighborhood averaging is that it blurs edges and other sharp details. This blurring can often be reduced significantly by the use of so-called *median filters*, in which we replace the intensity of each pixel by the median of the intensities in a predefined neighborhood of that pixel, instead of by the average.

Recall that the median $M$ of a set of values is such that half the values in the set are less than $M$ and half the values are greater than $M$. In order to perform median filtering in a neighborhood of a pixel, we first sort the values of the pixel and its neighbors, determine the median, and assign this value to the pixel. For example, in a $3 \times 3$ neighborhood the median is the fifth largest value, in a $5 \times 5$ neighborhood the thirteenth largest value, and so on. When several values in a neighborhood are the same, we group all equal values as follows: Suppose that a $3 \times 3$ neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. A little thought will reveal that the principal function of median filtering is to force points with very distinct intensities to be more like their neighbors, thus actually eliminating intensity spikes that appear isolated in the area of the filter mask.

**Example:** Figure 7.24*a* shows an original image, and Fig. 7.24*b* shows the same image but with approximately 20 percent of the pixels corrupted by "impulse noise." The result of neighborhood averaging over a $5 \times 5$ area is

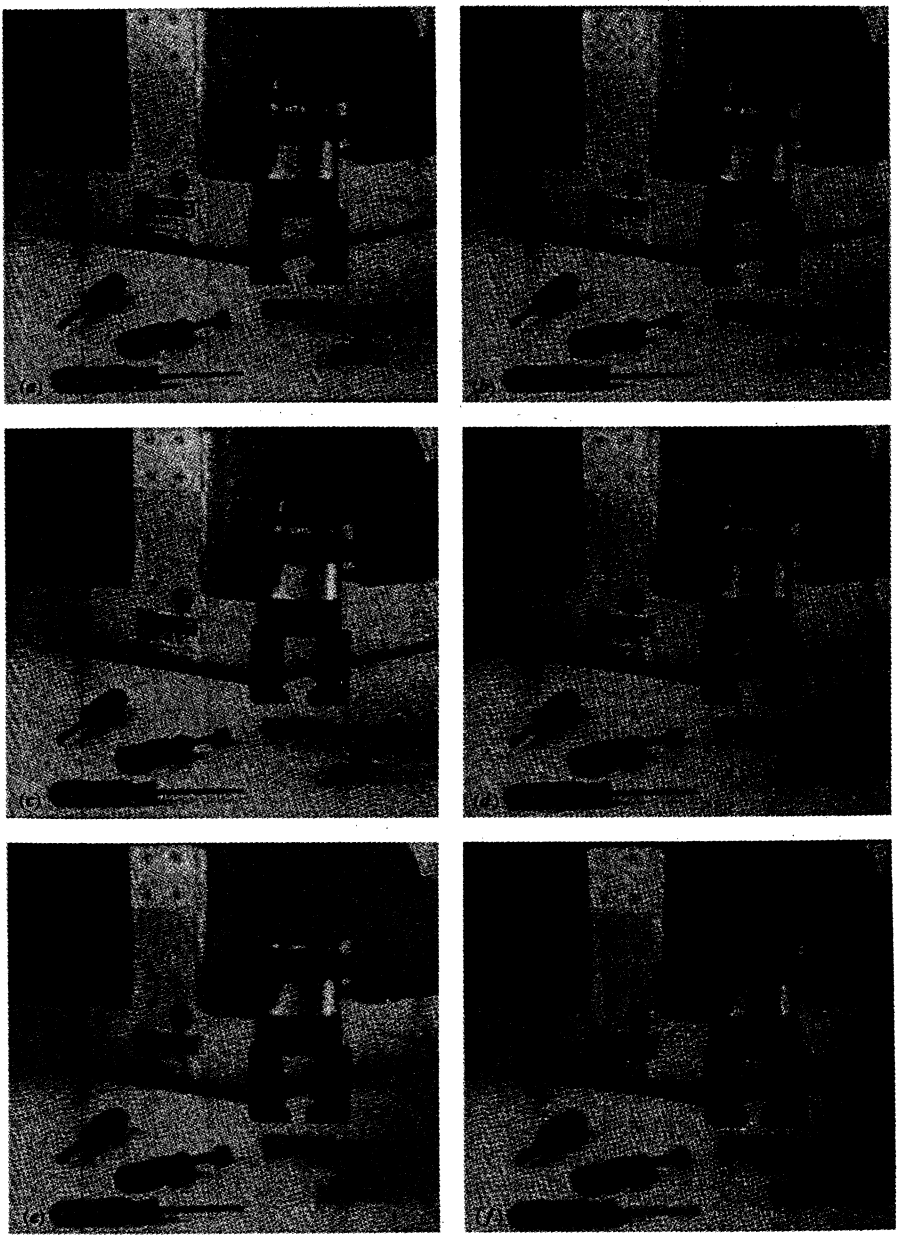

**Figure 7.23** (*a*) Noisy image. (*b*) Result of averaging each pixel along with its 4-neighbors. (*c*) through (*f*) are the results of using neighborhood sizes of 3 × 3, 5 × 5, 7 × 7, and 11 × 11, respectively.

**Figure 7.24** (*a*) Original image. (*b*) Image corrupted by impulse noise. (*c*) Result of 5 × 5 neighborhood averaging. (*d*) Result of 5 × 5 median filtering. (Courtesy of Martin Connor, Texas Instruments, Inc., Lewisville, Texas.)

shown in Fig. 7.24*c* and the result of a 5 × 5 median filter is shown in Fig. 7.24*d*. The superiority of the median filter over neighborhood averaging needs no explanation. The three bright dots remaining in Fig. 7.24*d* resulted from a large concentration of noise at those points, thus biasing the median calculation. Two or more passes with a median filter would eliminate those points. □

**Image Averaging**. Consider a noisy image $g(x, y)$ which if formed by the addition of noise $n(x, y)$ to an uncorrupted image $f(x, y)$; that is,

$$g(x, y) = f(x, y) + n(x, y) \tag{7.6-9}$$

where it is assumed that the noise is uncorrelated and has zero average value. The objective of the following procedure is to obtain a smoothed result by adding a given set of noisy images, $g_i(x, y)$, $i = 1, 2, \ldots, K$.

If the noise satisfies the constraints just stated, it is a simple problem to show (Papoulis [1965]) that if an image $\bar{g}(x, y)$ is formed by averaging $K$ different noisy images,

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y) \tag{7.6-10}$$

then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y) \tag{7.6-11}$$

and

$$\sigma_{\bar{g}}^2(x, y) = \frac{1}{K} \sigma_n^2(x, y) \tag{7.6-12}$$

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

$$\sigma_{\bar{g}}(x, y) = \frac{1}{\sqrt{K}} \sigma_n(x, y) \tag{7.6-13}$$

Equations (7.6-12) and (7.6-13) indicate that, as $K$ increases, the variability of the pixel values decreases. Since $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ will approach the uncorrupted image $f(x, y)$ as the number of noisy images used in the averaging process increases.

It is important to note that the technique just discussed implicitly assumes that all noisy images are registered spatially, with only the pixel intensities varying. In terms of robotic vision, this means that all object in the work space must be at rest with respect to the camera during the averaging process. Many vision systems have the capability of performing an entire image addition in one frame time interval (i.e., one-thirtieth of a second). Thus, the addition of, say, 16 images will take on the order of ½ s. during which no motion can take place.

**Example:** An an illustration of the averaging method, consider the images shown in Fig. 7.25. Part (*a*) of this figure shows a sample noisy image and Fig. 7.25*b* to *f* show the results of averaging 4, 8, 16, 32, and 64 such images, respectively. It is of interest to note that the results are quite acceptable for $K = 32$. □

**Smoothing Binary Images.** Binary images result from using backlighting or structured lighting, as discussed in Sec. 7.3, or from processes such as edge detection or thresholding, as discussed in Secs. 7.6.4 and 7.6.5. We will use the convention of labeling dark points with a 1 and light points with a 0. Thus, since binary

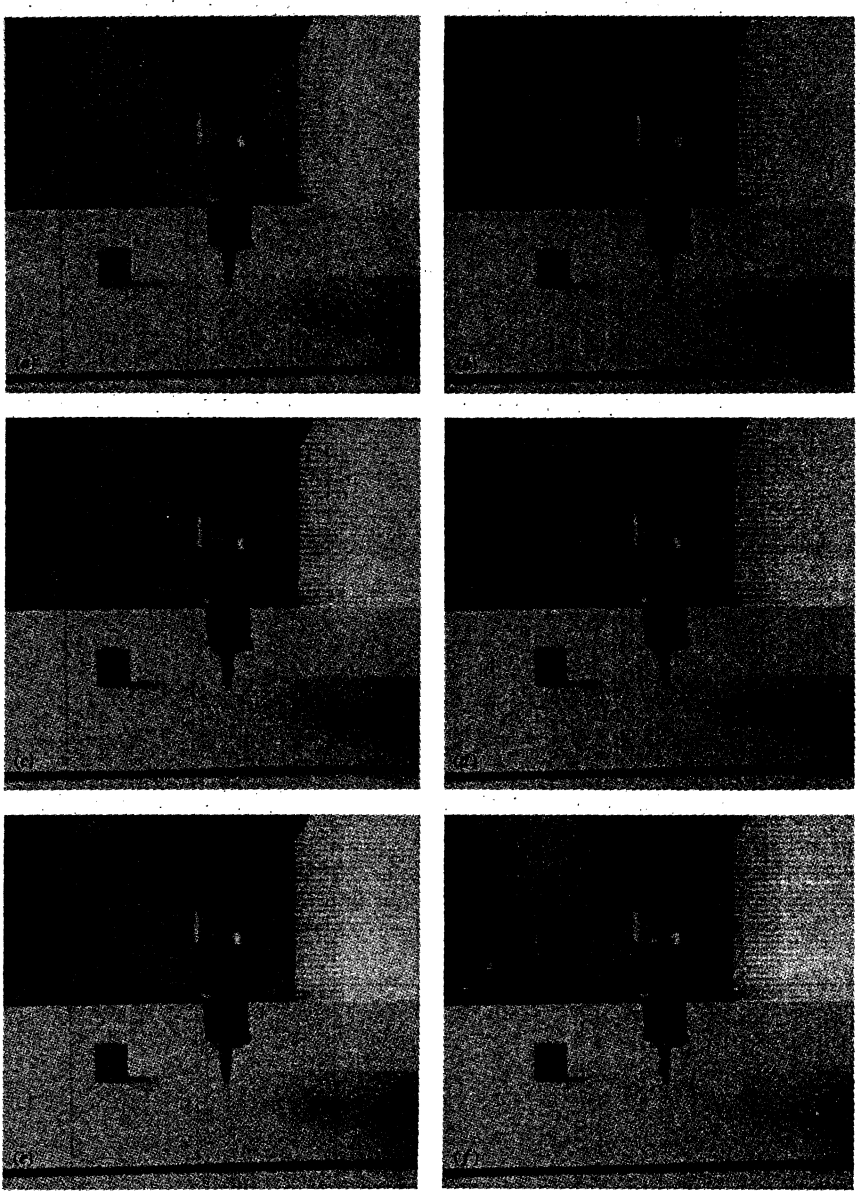

**Figure 7.25** (*a*) Sample noisy image. (*b*) through (*f*) are the results of averaging 4, 8, 16, 32, and 64 such images.

images are two-valued, noise in this case produces effects such as irregular boundaries, small holes, missing corners, and isolated points.

The basic idea underlying the methods discussed in this section is to specify a boolean function evaluated on a neighborhood centered at a pixel $p$, and to assign to $p$ a 1 or 0, depending on the spatial arrangement and binary values of its neighbors. Due to limitations in available processing time for industrial vision tasks, the analysis is typically limited to the 8-neighbors of $p$, which leads us to the $3 \times 3$ mask shown in Fig. 7.26. The smoothing approach (1) fills in small (one pixel) holes in otherwise dark areas, (2) fills in small notches in straightedge segments, (3) eliminates isolated 1's, (4) eliminates small bumps along straightedge segments, and (5) replaces missing corner points.

With reference to Fig. 7.26, the first two smoothing processes just mentioned are accomplished by using the boolean expression

$$B_1 = p + b \cdot g \cdot (d + e) + d \cdot e \cdot (b + g) \tag{7.6-14}$$

where "$\cdot$" and "$+$" denote the logical AND and OR, respectively. Following the convention established above, a dark pixel contained in the mask area is assigned a logical 1 and a light pixel a logical 0. Then, if $B_1 = 1$, we assign a 1 to $p$, otherwise this pixel is assigned a 0. Equation (7.6.14) is applied to all pixels simultaneously, in the sense that the next value of each pixel location is determined before any of the other pixels have been changed.

Steps 3 and 4 in the smoothing process are similarly accomplished by evaluating the boolean expression

$$B_2 = p \cdot [(a + b + d) \cdot (e + g + h) + (b + c + e) \cdot (d + f + g)] \tag{7.6-15}$$

simultaneously for all pixels. As above, we let $p = 1$ if $B_2 = 1$ and zero otherwise.

| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $p$ | $e$ |
| $f$ | $g$ | $h$ |

**Figure 7.26** Neighbors of $p$ used for smoothing binary images. Dark pixels are denoted by 1 and light pixels by 0.

Missing top, right corner points are filled in by means of the expression

$$B_3 = \bar{p} \cdot (d \cdot f \cdot g) \cdot \overline{(a + b + c + e + h)} + p \qquad (7.6\text{-}16)$$

where overbar denotes the logical complement. Similarly, lower right, top left, and lower left missing corner points are filled in by using the expressions

$$B_4 = \bar{p} \cdot (a \cdot b \cdot d) \cdot \overline{(c + e + f + g + h)} + p \qquad (7.6\text{-}17)$$

$$B_5 = \bar{p} \cdot (e \cdot g \cdot h) \cdot \overline{(a + b + c + d + f)} + p \qquad (7.6\text{-}18)$$

and

$$B_6 = \bar{p} \cdot (b \cdot c \cdot e) \cdot \overline{(a + d + f + g + h)} + p \qquad (7.6\text{-}19)$$

These last four expressions implement step 5 of the smoothing procedure.

**Example:** The concepts just discussed are illustrated in Fig. 7.27. Figure 7.27*a* shows a noisy binary image, and Fig. 7.27*b* shows the result of applying $B_1$. Note that the notches along the boundary and the hole in the dark area were filled in. Figure 7.27*c* shows the result of applying $B_2$ to the image in Fig. 7.27*b*. As expected, the bumps along the boundary of the dark area and all isolated points were removed (the image was implicitly extended with 0's for points on the image border). Finally, Fig. 7.27*d* shows the result of applying $B_3$ through $B_6$ to the image in Fig. 7.27*c*. Only $B_4$ had an effect in this particular case. □

### 7.6.3 Enhancement

One of the principal difficulties in many low-level vision tasks is to be able to automatically adapt to changes in illumination. The capability to compensate for effects such as shadows and "hot-spot" reflectances quite often plays a central role in determining the success of subsequent processing algorithms. In this subsection we consider several enhancement techniques which address these and similar problems. The reader is reminded that enhancement is a major area in digital image processing and scene analysis, and that our discussion of this topic is limited to sample techniques that are suitable for robot vision systems. In this context, "suitable" implies having fast computational characteristics and modest hardware requirements.

**Histogram Equalization.** Let the variable $r$ represent the intensity of pixels in an image to be enhanced. It will be assumed initially that $r$ is a normalized, continuous variable lying in the range $0 \leqslant r \leqslant 1$. The discrete case is considered later in this section.

For any $r$ in the interval [0, 1], attention will be focused on transformations of the form

$$s = T(r) \qquad (7.6\text{-}20)$$

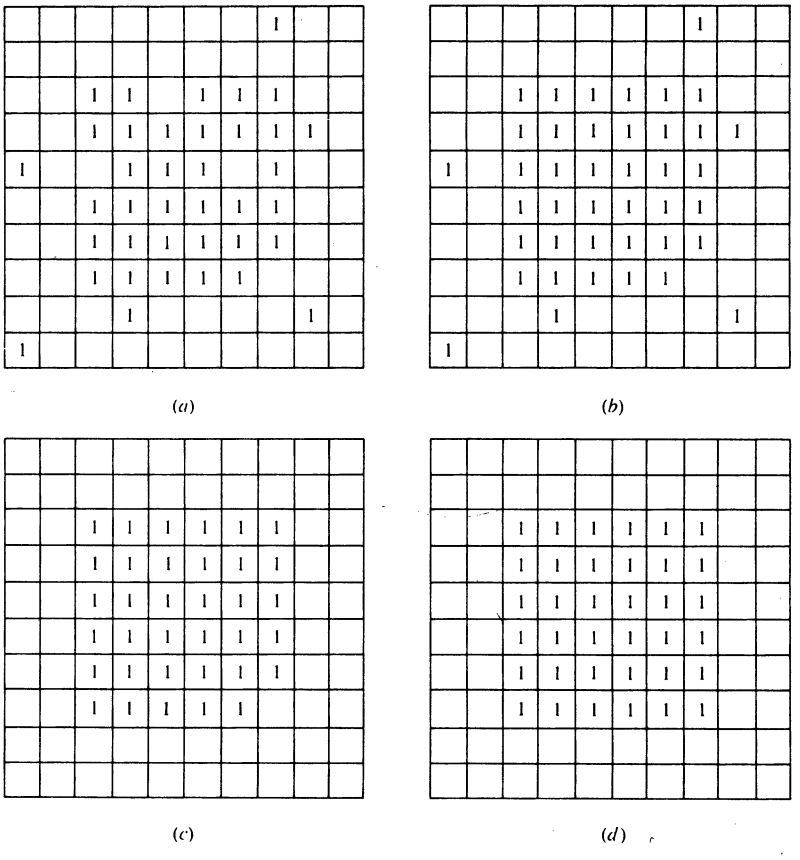

**Figure 7.27** (*a*) Original image. (*b*) Result of applying $B_1$. (*c*) Result of applying $B_2$. (*d*) Final result after application of $B_3$ through $B_6$.

which produce an intensity value $s$ for every pixel value $r$ in the input image. It is assumed that the transformation function $T$ satisfies the conditions:

1. $T(r)$ is single-valued and monotonically increasing in the interval $0 \leqslant T(r) \leqslant 1$.
2. $0 \leqslant T(r) \leqslant 1$ for $0 \leqslant r \leqslant 1$.

Condition 1 preserves the order from black to white in the intensity scale, and condition 2 guarantees a mapping that is consistent with the allowed 0 to 1 range of pixel values. A transformation function satisfying these conditions is illustrated in Fig. 7.28.

The inverse transformation function from $s$ back to $r$ is denoted by

$$r = T^{-1}(s) \tag{7.6-21}$$

where it is assumed that $T^{-1}(s)$ satisfies the two conditions given above.

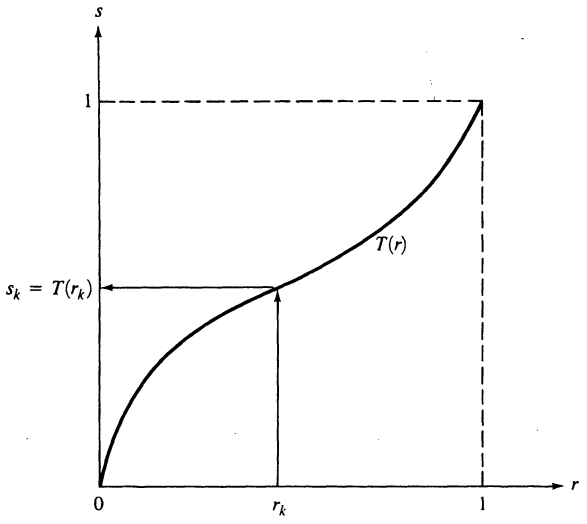


**Figure 7.28** An intensity transformation function.

The intensity variables $r$ and $s$ are random quantities in the interval [0, 1] and, as such, can be characterized by their probability density functions (PDFs) $p_r(r)$ and $p_s(s)$. A great deal can be said about the general appearance of an image from its intensity PDF. For example, an image whose pixels have the PDF shown in Fig. 7.29*a* would have fairly dark characteristics since the majority of pixel values would be concentrated on the dark end of the intensity scale. On the other hand, an image whose pixels have an intensity distribution like the one shown in Fig. 7.29*b* would have predominant light tones.

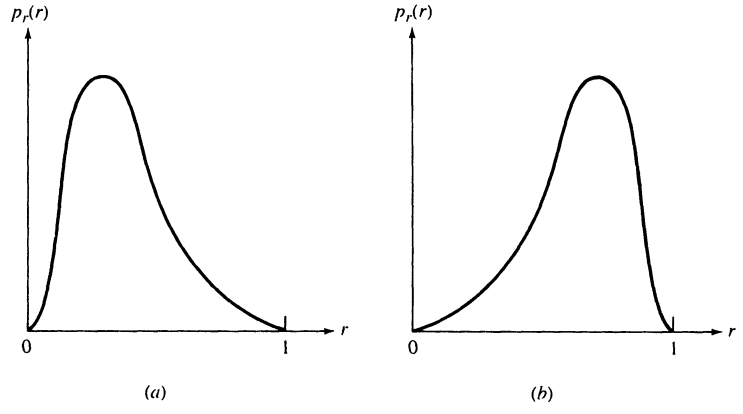


**Figure 7.29** (*a*) Intensity PDF of a "dark" image and (*b*) a "light" image.

It follows from elementary probability theory that if $p_r(r)$ and $T(r)$ are known, and $T^{-1}(s)$ satisfies condition 1, then the PDF of the transformed intensities is given by

$$p_s(s) = \left[ p_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)} \tag{7.6-22}$$

Suppose that we choose a specific transformation function given by

$$s = T(r) = \int_0^r p_r(w)\, dw \qquad 0 \leqslant r \leqslant 1 \tag{7.6-23}$$

where $w$ is a dummy variable of integration. The rightmost side of this equation is recognized as the cumulative distribution function of $p_r(r)$, which is known to satisfy the two conditions stated earlier. The derivative of $s$ with respect to $r$ for this particular transformation function is easily found to be

$$\frac{ds}{dr} = p_r(r) \tag{7.6-24}$$

Substitution of $dr/ds$ into Eq. (7.6-22) yields

$$\begin{aligned} p_s(s) &= \left[ p_r(r) \frac{1}{p_r(r)} \right]_{r=T^{-1}(s)} \\ &= [1]_{r=T^{-1}(s)} \\ &= 1 \qquad 0 \leqslant s \leqslant 1 \end{aligned} \tag{7.6-25}$$

which is a uniform density in the interval of definition of the transformed variable $s$. It is noted that this result is independent of the inverse transformation function. This is important because it is often quite difficult to find $T^{-1}(s)$ analytically. It is also noted that using the transformation function given in Eq. (7.6-23) yields transformed intensities that always have a flat PDF, *independent* of the shape of $p_r(r)$, a property that is ideally suited for automatic enhancement. The net effect of this transformation is to balance the distribution of intensities. As will be seen below, this process can have a rather dramatic effect on the appearance of an image.

In order to be useful for digital processing, the concepts developed above must be formulated in discrete form. For intensities that assume discrete values we deal with probabilities given by the relation

$$p_r(r_k) = \frac{n_k}{n} \qquad 0 \leqslant r_k \leqslant 1 \tag{7.6-26}$$

$$k = 0, 1, 2, \ldots, L - 1$$

where $L$ is the number of discrete intensity levels, $p_r(r_k)$ is an estimate of the probability of intensity $r_k$, $n_k$ is the number of times this intensity appears in the

image, and $n$ is the total number of pixels in the image. A plot of $p_r(r_k)$ versus $r_k$ is usually called a *histogram*, and the technique used for obtaining a uniform histogram is known as *histogram equalization* or *histogram linearization*.

The discrete form of Eq. (7.6-23) is given by

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n}$$

$$= \sum_{j=0}^{k} p_r(r_j) \tag{7.6-27}$$

for $0 \leqslant r_k \leqslant 1$ and $k=0, 1, 2, \ldots, L-1$. It is noted from this equation that in order to obtain the mapped value $s_k$ corresponding to $r_k$, we simply sum the histogram components from 0 to $r_k$.

The inverse discrete transformation is given by

$$r_k = T^{-1}(s_k) \qquad 0 \leqslant s_k \leqslant 1 \tag{7.6-28}$$

where both $T(r_k)$ and $T^{-1}(s_k)$ are assumed to satisfy conditions 1 and 2 stated above. Although $T^{-1}(s_k)$ is not used in histogram equalization, it plays a central role in histogram specification, as discussed below.

**Example:** As an illustration of histogram equalization, consider the image shown in Fig. 7.30*a* and its histogram shown in Fig. 7.30*b*. The result of applying Eq. (7.6-27) to this image is shown in Fig. 7.30*c* and the corresponding equalized histogram is shown in Fig. 7.30*d*. The improvement of details is evident. It is noted that the histogram is not perfectly flat, a condition generally encountered when applying to discrete values a method derived for continuous quantities. □

**Histogram Specification**. Histogram equalization is ideally suited for automatic enhancement since it is based on a transformation function that is uniquely determined by the histogram of the input image. However, the method is limited in the sense that its only function is histogram linearization, a process that is not applicable when a priori information is available regarding a desired output histogram shape. Here we generalize the concept of histogram processing by developing an approach capable of generating an image with a specified intensity histogram. As will be seen below, histogram equalization is a special case of this technique.

Starting again with continuous quantities, let $p_r(r)$ and $p_z(z)$ be the original and desired intensity PDFs. Suppose that a given image is first histogram equalized by using Eq. (7.6-23); that is,

$$s = T(r) = \int_0^r p_r(w)\, dw \tag{7.6-29}$$

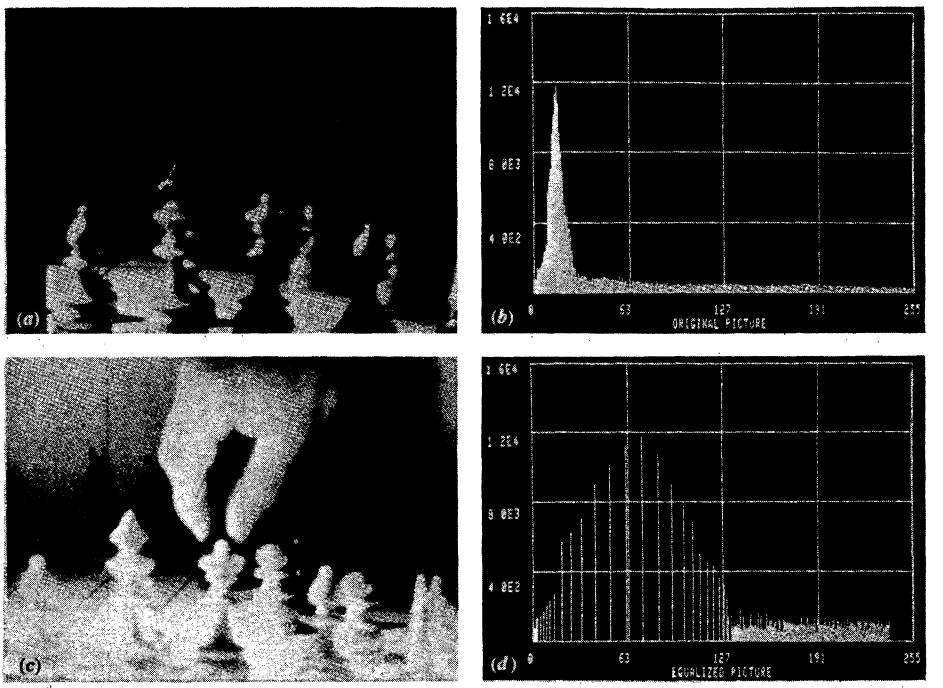


**Figure 7.30** (*a*) Original image and (*b*) its histogram. (*c*) Histogram-equalized image and (*d*) its histogram. (From Woods and Gonzalez [1981], © IEEE.)

If the desired image *were* available, its levels could also be equalized by using the transformation function

$$v = G(z) = \int_0^z p_z(w)\,dw \tag{7.6-30}$$

The inverse process, $z = G^{-1}(v)$ would then yield the desired levels back. This, of course, is a hypothetical formulation since the $z$ levels are precisely what we are trying to obtain. It is noted, however, that $p_s(s)$ and $p_v(v)$ would be identical uniform densities since the use of Eqs. (7.6-29) and (7.6-30) guarantees a uniform density, regardless of the shape of the PDF inside the integral. Thus, if instead of using $v$ in the inverse process, we use the inverse levels $s$ obtained from the original image, the resulting levels $z = G^{-1}(s)$ would have the desired PDF, $p_z(z)$. Assuming that $G^{-1}(s)$ is single-valued, the procedure can be summarized as follows:

1. Equalize the levels of the original image using Eq. (7.6-29).
2. Specify the desired intensity PDF and obtain the transformation function $G(z)$ using Eq. (7.6-30).

3. Apply the inverse transformation $z = G^{-1}(s)$ to the intensity levels of the histogram-equalized image obtained in step 1.

This procedure yields an output image with the specified intensity PDF.

The two transformations required for histogram specification, $T(r)$ and $G^{-1}(s)$, can be combined into a single transformation:

$$z = G^{-1}(s) = G^{-1}[T(r)] \tag{7.6-31}$$

which relates $r$ to $z$. It is noted that, when $G^{-1}[T(r)] = T(r)$, this method reduces to histogram equalization.

Equation (7.6-31) shows that the input image need not be histogram-equalized explicitly in order to perform histogram specification. All that is required is that $T(r)$ be determined and combined with $G^{-1}(s)$ into a single transformation that is applied directly to the input image. The real problem in using the two transformations or their combined representation for continuous variables lies in obtaining the inverse function analytically. In the discrete case this problem is circumvented by the fact that the number of distinct intensity levels is usually relatively small (e.g., 256) and it becomes feasible to calculate and store a mapping for each possible integer pixel value.

The discrete formulation of the foregoing procedure parallels the development in the previous section:

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) \tag{7.6-32}$$

$$G(z_i) = \sum_{j=0}^{i} p_z(z_j) \tag{7.6-33}$$

and

$$z_i = G^{-1}(s_i) \tag{7.6-34}$$

where $p_r(r_j)$ is computed from the input image, and $p_z(z_j)$ is specified.

**Example:** An illustration of the histogram specification method is shown in Fig. 7.31. Part (*a*) of this figure shows the input image and Fig. 7.31*b* is the result of histogram equalization. Figure 7.31*c* shows a specified histogram and Fig. 7.31*d* is the result of using this histogram in the procedure discussed above. It is noted that, in this case, histogram equalization had little effect on the image. □

**Local Enhancement.** The histogram equalization and specification methods discussed above are global, in the sense that pixels are modified by a transformation function which is based on the intensity distribution over an entire image. While this global approach is suitable for overall enhancement, it is often necessary to enhance details over small areas. Since the number of pixels in these areas may have negligible influence on the computation of a global transformation, the use of

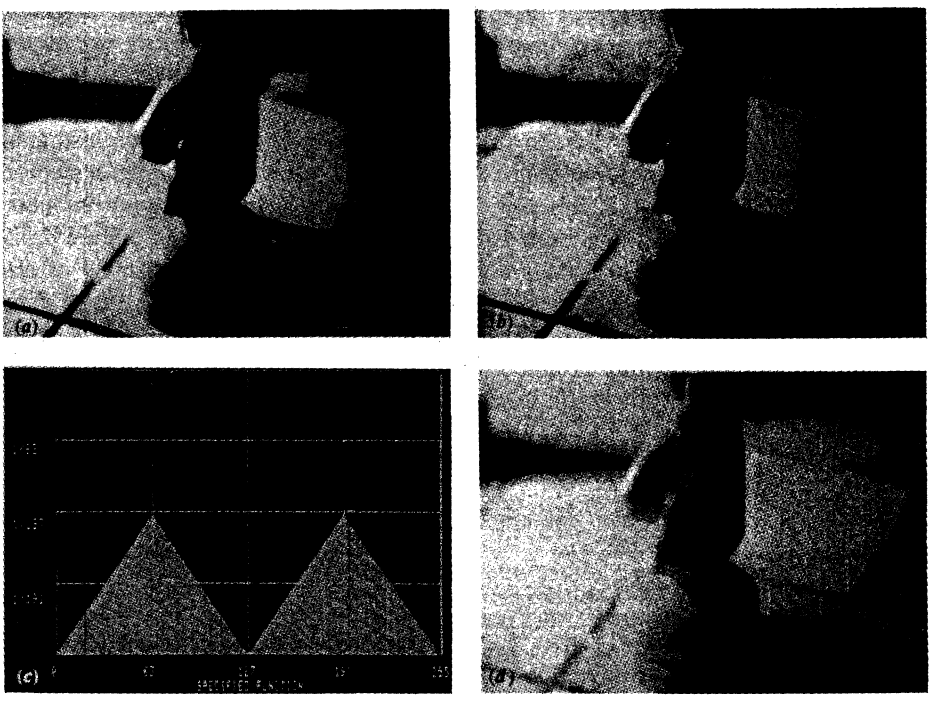

**Figure 7.31** (*a*) Input image. (*b*) Result of histogram equalization. (*c*) A specified histogram. (*d*) Result of enhancement by histogram specification. (From Woods and Gonzalez [1981], © IEEE.)

global techniques seldom yields acceptable local enhancement. The solution is to devise transformation functions that are based on the intensity distribution, or other properties, in the neighborhood of every pixel in a given image.

The histogram-processing techniques developed above are easily adaptable to local enhancement. The procedure is to define an $n \times m$ neighborhood and move the center of this area from pixel to pixel. At each location, we compute the histogram of the $n \times m$ points in the neighborhood and obtain either a histogram equalization or histogram specification transformation function. This function is finally used to map the intensity of the pixel centered in the neighborhood. The center of the $n \times m$ region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighborhood changes during a pixel-to-pixel translation of the region, it is possible to update the histogram obtained in the previous location with the new data introduced at each motion step. This approach has obvious advantages over repeatedly computing the histogram over all $n \times m$ pixels every time the region is moved one pixel location. Another approach often used to reduce computation is to employ nonoverlapping regions, but this often produces an undesirable checkerboard effect.

ment approach. This example clearly demonstrates the necessity for using local enhancement when the details of interest are too small to influence significantly the overall characteristics of a global technique. □

Instead of using histograms, one could base local enhancement on other properties of the pixel intensities in a neighborhood. The intensity mean and variance (or standard deviation) are two such properties which are frequently used because of their relevance to the appearance of an image. That is, the mean is a measure of average brightness and the variance is a measure of contrast. A typical local transformation based on these concepts maps the intensity of an input image $f(x, y)$ into a new image $g(x, y)$ by performing the following transformation at each pixel location $(x, y)$:

$$g(x, y) = A(x, y)[f(x, y) - m(x, y)] + m(x, y) \tag{7.6-35}$$

where

$$A(x, y) = k \frac{M}{\sigma(x, y)} \quad 0 < k < 1 \tag{7.6-36}$$

In this formulation, $m(x, y)$ and $\sigma(x, y)$ are the intensity mean and standard deviation computed in a neighborhood centered at $(x, y)$, $M$ is the global mean of $f(x, y)$, and $k$ is a constant in the range indicated above.

It is important to note that $A$, $m$, and $\sigma$ are variable quantities which depend on a predefined neighborhood of $(x, y)$. Application of the local gain factor $A(x, y)$ to the difference between $f(x, y)$ and the local mean amplifies local variations. Since $A(x, y)$ is inversely proportional to the standard deviation of the intensity, areas with low contrast receive larger gain. The mean is added back in Eq. (7.6-35) to restore the average intensity level of the image in the local region. In practice, it is often desirable to add back a fraction of the local mean and to restrict the variations of $A(x, y)$ between two limits $[A_{\min}, A_{\max}]$ in order to balance out large excursions of intensity in isolated regions.

**Example:** The preceding enhancement approach has been implemented in hardware by Narendra and Fitch [1981], and has the capability of processing images in real time (i.e., at 30 image frames per second). An example of the capabilities of the technique using a local region of size $15 \times 15$ pixels is shown in Fig. 7.33. Note the enhancement of detail at the boundary between two regions of different overall intensities and the rendition of intensity details in each of the regions. □

## 7.6.4 Edge Detection

Edge detection plays a central role in machine vision, serving as the initial preprocessing step for numerous object detection algorithms. In this chapter we are

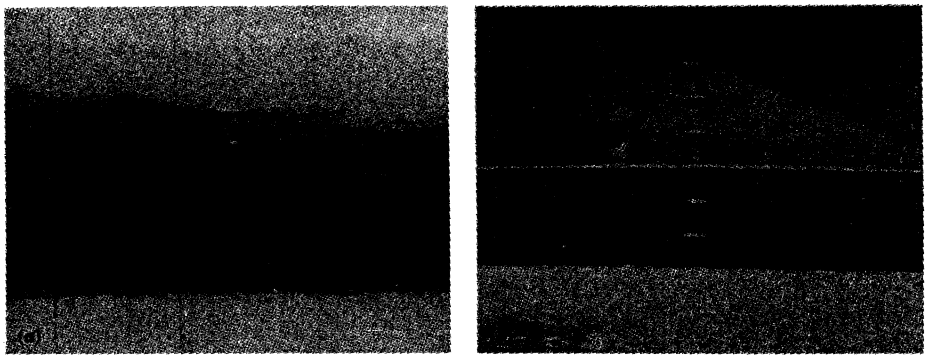

**Figure 7.33** Images before and after local enhancement. (From Narendra and Fitch [1981], © IEEE.)

interested in fundamental techniques for detecting edge points. Subsequent processing of these edge points is discussed in Chap. 8.

**Basic Formulation.** Basically, the idea underlying most edge detection techniques is the computation of a local derivative operator. This concept can be easily illustrated with the aid of Fig. 7.34. Part (*a*) of this figure shows an image of a simple light object on a dark background, the intensity profile along a horizontal scan line of the image, and the first and second derivatives of the profile. It is noted from the profile that an edge (transition from dark to light) is modeled as a ramp, rather than as an abrupt change of intensity. This model is representative of the fact that edges in digital images are generally slightly blurred as a result of sampling.

The first derivative of an edge modeled in this manner is zero in all regions of constant intensity, and assumes a constant value during an intensity transition. The second derivative, on the other hand, is zero in all locations, except at the onset and termination of an intensity transition. Based on these remarks and the concepts illustrated in Fig. 7.34, it is evident that the magnitude of the first derivative can be used to detect the presence of an edge, while the sign of the second derivative can be used to determine whether an edge pixel lies on the dark (background) or light (object) side of an edge. The sign of the second derivative in Fig. 7.34*a*, for example, is positive for pixels lying on the dark side of both the leading and trailing edges of the object, while the sign is negative for pixels on the light side of these edges. Similar comments apply to the case of a dark object on a light background, as shown in Fig. 7.34*b*. It is of interest to note that identically the same interpretation regarding the sign of the second derivative is true for this case.

Although the discussion thus far has been limited to a one-dimensional horizontal profile, a similar argument applies to an edge of any orientation in an image. We simply define a profile perpendicular to the edge direction at any given point and interpret the results as in the preceding discussion. As will be shown below, the first derivative at any point in an image can be obtained by using the magnitude of the gradient at that point, while the second derivative is given by the Laplacian.

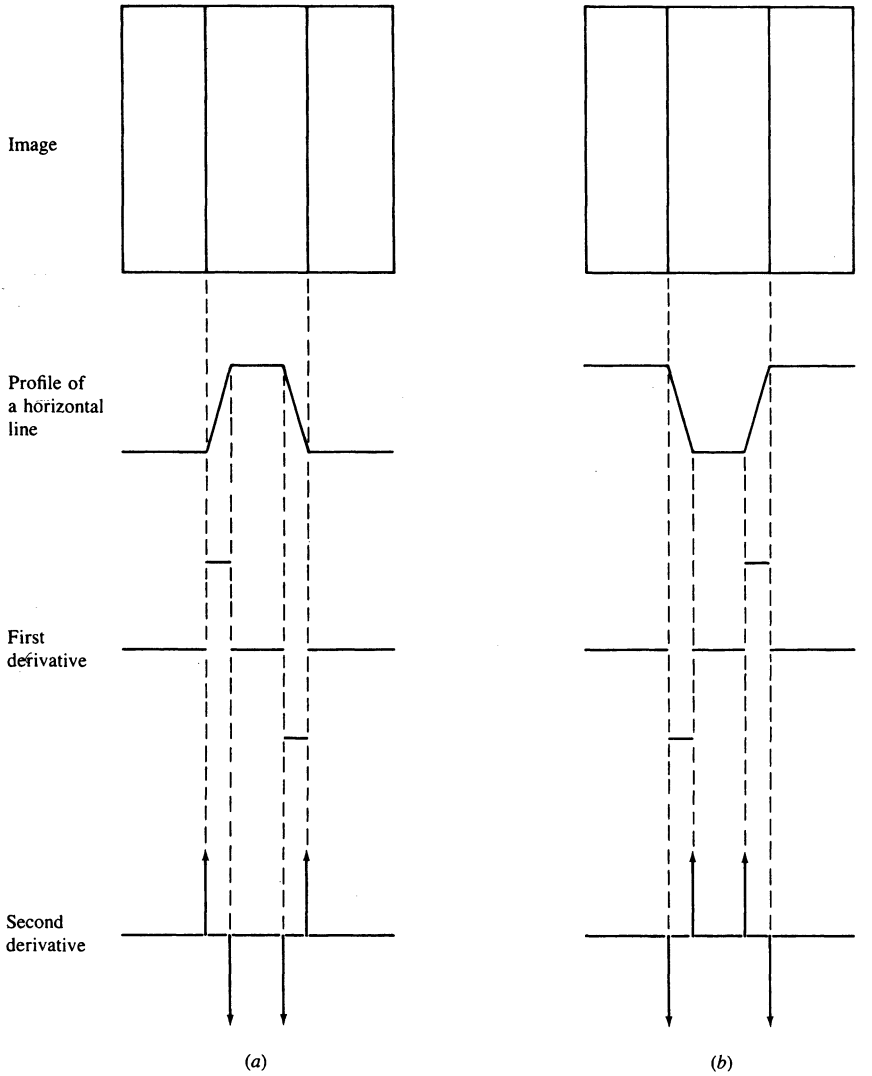


**Figure 7.34** Elements of edge detection by derivative operators. (*a*) Light object on a dark background. (*b*) Dark object on a light background.

**Gradient Operators.** The gradient of an image $f(x, y)$ at location $(x, y)$ is defined as the two-dimensional vector

$$\mathbf{G}[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{7.6-37}$$

It is well known from vector analysis that the vector **G** points in the direction of maximum rate of change of $f$ at location $(x, y)$. For edge detection, however, we are interested in the magnitude of this vector, generally referred to as the *gradient* and denoted by $G[f(x, y)]$, where

$$G[f(x, y)] = [G_x^2 + G_y^2]^{1/2} \tag{7.6-38}$$

$$= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

It is common practice to approximate the gradient by absolute values:

$$G[f(x, y)] \approx |G_x| + |G_y| \tag{7.6-39}$$

This approximation is considerably easier to implement, particularly when dedicated hardware is being employed.

It is noted from Eq. (7.6-38) that computation of the gradient is based on obtaining the first-order derivatives $\partial f/\partial x$ and $\partial f/\partial y$. There are a number of ways for doing this in a digital image. One approach is to use first-order differences between adjacent pixels; that is,

$$G_x = \frac{\partial f}{\partial x} = f(x, y) - f(x - 1, y) \tag{7.6-40}$$

and

$$G_y = \frac{\partial f}{\partial y} = f(x, y) - f(x, y - 1) \tag{7.6-41}$$

A slightly more complicated definition involving pixels in a $3 \times 3$ neighborhood centered at $(x, y)$ is given by

$$G_x = \frac{\partial f}{\partial x} = [f(x + 1, y - 1) + 2f(x + 1, y) + f(x + 1, y + 1)]$$

$$- [f(x - 1, y - 1) + 2f(x - 1, y) + f(x - 1, y + 1)]$$

$$= (g + 2h + i) - (a + 2b + c) \tag{7.6-42}$$

and

$$G_y = \frac{\partial f}{\partial y} = [f(x - 1, y + 1) + 2f(x, y + 1) + f(x + 1, y + 1)]$$

$$- [f(x - 1, y - 1) + 2f(x, y - 1) + f(x + 1, y - 1)]$$

$$= (c + 2e + i) - (a + 2d + g) \tag{7.6-43}$$

where we have used the letters $a$ through $i$ to represent the neighbors of point $(x, y)$. The $3 \times 3$ neighborhood of $(x, y)$ using this simplified notation is shown

in Fig. 7.35*a*. It is noted that the pixels closest to $(x, y)$ are weighted by 2 in these particular definitions of the digital derivative. Computing the gradient over a $3 \times 3$ area rather than using Eqs. (7.6-40) and (7.6-41) has the advantage of increased averaging, thus tending to make the gradient less sensitive to noise. It is possible to define the gradient over larger neighborhoods (Kirsch [1971]), but $3 \times 3$ operators are by far the most popular in industrial computer vision because of their computational speed and modest hardware requirements.

It follows from the discussion in Sec. 7.6.1 that $G_x$, as given in Eq. (7.6-42), can be computed by using the mask shown in Fig. 7.35*b*. Similarly, $G_y$ may be obtained by using the mask shown in Fig. 7.35*c*. These two masks are commonly referred to as the *Sobel operators*. The responses of these two masks at any point $(x, y)$ are combined using Eqs. (7.6-38) or (7.6-39) to obtain an approximation to the gradient at that point. Moving these masks throughout the image $f(x, y)$ yields the gradient at all points in the image.

There are numerous ways by which one can generate an output image, $g(x, y)$, based on gradient computations. The simplest approach is to let the value of $g$ at coordinate $(x, y)$ be equal to the gradient of the input image $f$ at that point; that is,

$$g(x, y) = G[f(x, y)] \tag{7.6-44}$$

An example of using this approach to generate a gradient image is shown in Fig. 7.36.

| | | |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $(x, y)$ | $e$ |
| $g$ | $h$ | $i$ |

(*a*)

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

(*b*)

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

(*c*)

**Figure 7.35** (*a*) $3 \times 3$ neighborhood of point $(x, y)$. (*b*) Mask used to compute $G_x$. (*c*) Mask used to compute $G_y$.

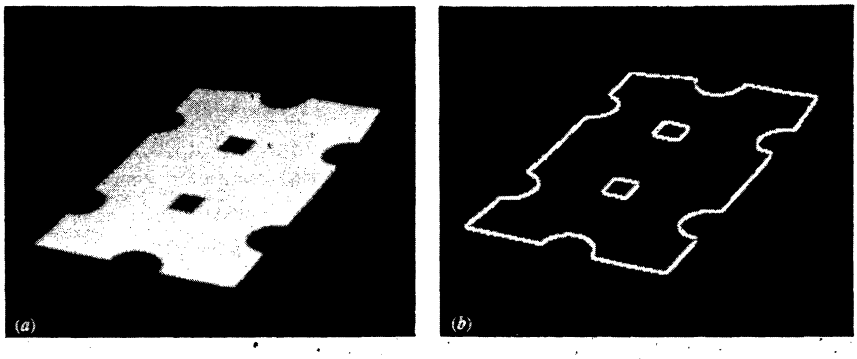


**Figure 7.36** (*a*) Input image. (*b*) Result of using Eq. (7.6-44).

Another approach is to create a binary image using the following relationship:

$$g(x, y) = \begin{cases} 1 & \text{if } G[f(x, y)] > T \\ 0 & \text{if } G[f(x, y)] \leqslant T \end{cases} \tag{7.6-45}$$

where $T$ is a nonnegative threshold. In this case, only edge pixels whose gradients exceed $T$ are considered important. Thus, the use of Eq. (7.6-45) may be viewed as a procedure which extracts only those pixels that are characterized by significant (as determined by $T$) transitions in intensity. Further analysis of the resulting pixels is usually required to delete isolated points and to link pixels along proper boundaries which ultimately determine the objects segmented out of an image. The use of Eq. (7.6-45) in this context is discussed and illustrated in Sec. 8.2.1.

**Laplacian Operator**. The Laplacian is a second-order derivative operator defined as

$$L[f(x, y)] = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{7.6-46}$$

For digital images, the Laplacian is defined as

$$L[f(x, y)] = [f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1)] - 4 f(x, y) \tag{7.6-47}$$

This digital formulation of the Laplacian is zero in constant areas and on the ramp section of an edge, as expected of a second-order derivative. The implementation of Eq. (7.6-47) can be based on the mask shown in Fig. 7.37.

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

**Figure 7.37** Mask used to compute the Laplacian.

Although, as indicated at the beginning of this section, the Laplacian responds to transitions in intensity, it is seldom used by itself for edge detection. The reason is that, being a second-derivative operator, the Laplacian is typically unacceptably sensitive to noise. Thus, this operator is usually delegated the secondary role of serving as a detector for establishing whether a given pixel is on the dark or light side of an edge.

### 7.6.5 Thresholding

Image thresholding is one of the principal techniques used by industrial vision systems for object detection, especially in applications requiring high data throughputs. In this section we are concerned with aspects of thresholding that fall in the category of low-level processing. More sophisticated uses of thresholding techniques are discussed Chap. 8.

Suppose that the intensity histogram shown in Fig. 7.38*a* corresponds to an image, $f(x, y)$, composed of light objects on a dark background, such that object and background pixels have intensities grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold $T$ which separates the intensity modes. Then, any point $(x, y)$ for which $f(x, y) > T$ is called an object point; otherwise, the point is called a background point. A slightly more general case of this approach is shown in Fig. 7.38*b*. In this case the image histogram is characterized by three dominant modes (for example, two types of light objects on a dark background). Here, we can use the same basic approach and classify a point $(x, y)$ as belonging to one object class if $T_1 < f(x, y) \leqslant T_2$, to the other object class if $f(x, y) > T_2$, and to the background if $f(x, y) \leqslant T_1$. This type of *multilevel thresholding* is generally less reliable than its single threshold counterpart because of the difficulty in establishing multiple thresholds that effectively isolate regions of interest, especially when the number of corresponding histogram modes is large. Typically, problems of this nature, if handled by thresholding, are best addressed by a single, variable threshold, as discussed in Chap. 8.

Based on the foregoing concepts, we may view thresholding as an operation that involves tests against a function $T$ of the form

$$T = T[x, y, p(x, y), f(x, y)] \tag{7.6-47}$$

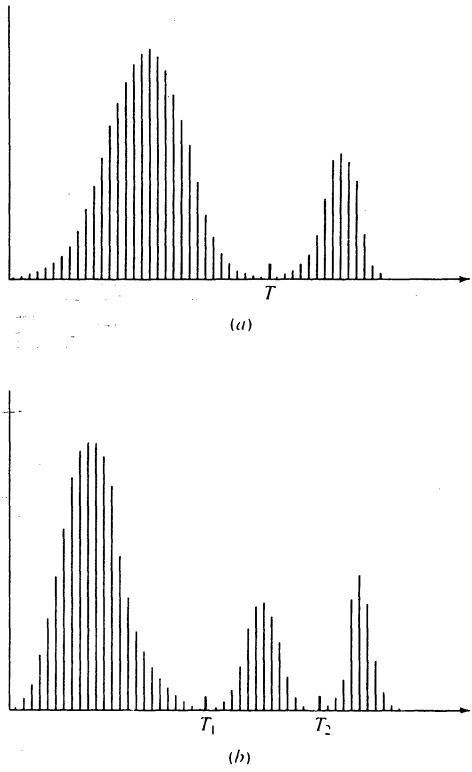


**Figure 7.38** Intensity histograms that can be partioned by (*a*) a single threshold and (*b*) multiple thresholds.

where $f(x, y)$ is the intensity of point $(x, y)$, and $p(x, y)$ denotes some local property of this point, for example, the average intensity of a neighborhood centered at $(x, y)$. We create a thresholded image $g(x, y)$ by defining

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leqslant T \end{cases} \tag{7.6-48}$$

Thus, in examining $g(x, y)$, we find that pixels labeled 1 (or any other convenient intensity level) correspond to objects, while pixels labeled 0 correspond to the background.

When $T$ depends only on $f(x, y)$, the threshold is called *global* (Fig. 7.38*a* shows an example of such a threshold). If $T$ depends on both $f(x, y)$ and $p(x, y)$, then the threshold is called *local*. If, in addition, $T$ depends on the spatial coordinates $x$ and $y$, it is called a *dynamic threshold*. We associate with low-level

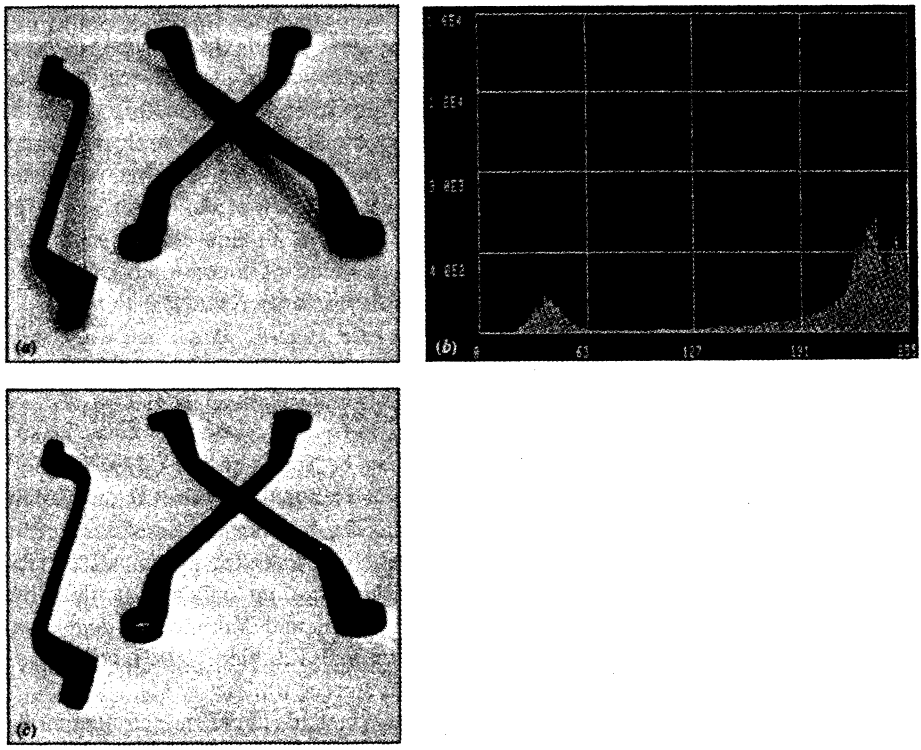

**Figure 7.39** (*a*) Original image. (*b*) Histogram of intensities in the range 0 to 255. (*c*) Image obtained by using Eq. (7.6-48) with a global threshold $T = 90$.

vision those thresholding techniques which are based on a single, global value of $T$. Since thresholding plays a central role in object segmentation, more sophisticated formulations are associated with functions in medium-level vision. A simple example of global thresholding is shown in Fig. 7.39.

## 7.7 CONCLUDING REMARKS

The material presented in this chapter spans a broad range of processing functions normally associated with low-level vision. Although, as indicated in Sec. 7.1, vision is a three-dimensional problem, most machine vision algorithms, especially those used for low-level vision, are based on images of a three-dimensional scene. The range sensing methods discussed in Sec. 7.2, the structured-lighting approaches in Sec. 7.3, and the material in Sec. 7.4 are important techniques for deriving depth from image information.

Our discussion of low-level vision and other relevant topics, such as the nature of imaging devices, has been at an introductory level, and with a very directed focus toward robot vision. It is important to keep in mind that many of the areas we have discussed have a range of application much broader than this. A good example is enhancement, which for years has been an important topic in digital image processing. One of the salient features of industrial applications, however, is the ever-present (and often contradictory) requirements of low cost and high computational speeds. The selection of topics included in this chapter has been influenced by these requirements and also by the value of these topics as fundamental material which would serve as a foundation for further study in this field.

## REFERENCES

Further reading on image acquisition devices may be found in Fink [1957], Herrick [1976], and Fairchild [1983]. The discussion in Sec. 7.3 is based on Mundy [1977], Holland et al. [1979], and Myers [1980]. The transformations discussed in Secs. 7.4.1 and 7.4.2 can be found in most books on computer graphics (see, for example, Newman and Sproull [1979]). Additional reading on camera modeling and calibration can be found in Duda and Hart [1973] and Yakimovsky and Cunningham [1979]. The survey article by Barnard and Fischler [1982] contains a comprehensive set of references on computational stereo.

The material in Sec. 7.5 is based on Toriwaki et al. [1979], and Rosenfeld and Kak [1982]. The discussion in Sec. 7.6.1, is adapted from Gonzalez and Wintz [1977]. For details on implementing median filters see Huang et al. [1979], Wolfe and Mannos [1979], and Chaudhuri [1983]. The concept of smoothing by image averaging is discussed by Kohler and Howell [1963]. The smoothing technique for binary images discussed in Sec. 7.6.2 is based on an early paper by Unger [1959].

The material in Sec. 7.6.3 is based on Gonzalez and Fittes [1977] and Woods and Gonzalez [1981]. For further details on local enhancement see Ketcham [1976], Harris [1977], and Narendra and Fitch [1981]. Early work on edge detection can be found in Roberts [1965]. A survey of techniques used in this area a decade later is given by Davis [1975]. More recent work in this field emphasizes computational speed, as exemplified by Lee [1983] and Chaudhuri [1983]. For an introduction to edge detection see Gonzalez and Wintz [1977]. The book by Rosenfeld and Kak [1982] contains a detailed description of threshold selection techniques. A survey paper by Weska [1978] is also of interest.

## PROBLEMS

**7.1** How many bits would it take to store a $512 \times 512$ image in which each pixel can have 256 possible intensity values?

**7.2** Propose a technique that uses a single light sheet to determine the diameter of cylindrical objects. Assume a linear array camera with a resolution of $N$ pixels and also that the distance between the camera and the center of the cylinders is fixed.

**7.3** (*a*) Discuss the accuracy of your solution to Prob. 7.2 in terms of camera resolution ($N$ points on a line) and maximum expected cylinder diameter, $D_{\max}$. (*b*) What is the maximum error if $N = 2048$ pixels and $D_{\max} = 1$ m?

**7.4** Determine if the world point with coordinates $(1/2, 1/2, \sqrt{2}/2)$ is on the optical axis of a camera located at $(0, 0, \sqrt{2})$, panned 135° and tilted 135°. Assume a 50-mm lens and let $r_1 = r_2 = r_3 = 0$.

**7.5** Start with Eq. (7.4-41) and derive Eqs. (7.4-42) and (7.4-43).

**7.6** Show that the $D_4$ distance between two points $p$ and $q$ is equal to the shortest 4-path between these points. Is this path unique?

**7.7** Show that a Fourier transform algorithm that computes $F(u)$ can be used without modification to compute the inverse transform. (*Hint*: The answer lies on using complex conjugates).

**7.8** Verify that substitution of Eq. (7.6-4) into Eq. (7.6-5) yields an identity.

**7.9** Give the boolean expression equivalent to Eq. (7.6-16) for a $5 \times 5$ window.

**7.10** Develop a procedure for computing the median in an $n \times n$ neighborhood.

**7.11** Explain why the discrete histogram equalization technique will not, in general, yield a flat histogram.

**7.12** Propose a method for updating the local histogram for use in the enhancement technique discussed in Sec. 7.6.3.

**7.13** The results obtained by a single pass through an image of some two-dimensional masks can also be achieved by two passes of a one-dimensional mask. For example, the result of using a $3 \times 3$ smoothing mask with coefficients $\frac{1}{9}$ (see Sec. 7.6.2) can also be obtained by first passing through an image the mask $[1 \quad 1 \quad 1]$. The result of this pass is then followed by a pass of the mask $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. The final result is then scaled by $\frac{1}{9}$. Show that the Sobel masks (Fig. 7.35) can be implemented by one pass of a *differencing* mask of the form $[-1 \quad 0 \quad 1]$ (or its vertical counterpart) followed by a *smoothing* mask of the form $[1 \quad 2 \quad 1]$ (or its vertical counterpart).

**7.14** Show that the digital Laplacian given in Eq. (7.6-47) is proportional (by the factor $-\frac{1}{4}$) to subtracting from $f(x, y)$ an average of the 4-neighbors of $(x, y)$. [The process of subtracting a blurred version of $f(x, y)$ from itself is called *unsharp masking*.]